A Suite of WWW-based Tools for Advanced Course Management

Kurt Nørmark
Department of Computer Science
Aalborg University
Denmark
normark@cs.auc.dk

Abstract

A collection of tools for creation of advanced and comprehensive course home pages is presented. The tools cover the spectrum from course overview pages and hypertext teaching materials to interactive services that support the teaching activities during the course. From the teacher's perspective the tools allow for abstraction from details and automation of routine work in the authoring process. Seen from a student's perspective the comprehensive linking of course plans, teaching material, and interactive services provides for a valuable organization of a large body of information.

1 Introduction

A course home page may in one extreme cover a single page with a course overview. In the other extreme, a course home page is a network of pages with a complete set of courses resources, such as plans (including a calendar and overviews in various details) teaching materials (book or course notes, possibly in several editions), exercises, solutions to exercises, and interactive services, such as synchronous and asynchronous tools which mediate a dialogue between the students and the teacher. The plans, teaching materials, and the interactive course services are often integrated via mutual links.

In this paper we will describe an integrated suite of WWW-based tools which, taken together, provide extensive and advanced support of a university course in computer science.All the tools are built on the same technical platform, using a technology called LAML [3] which we have developed as part of this work.

In this paper we will concentrate on the course impact of the tools, hereby emphasizing organizational and pedagogical issues. The technical issues involved are addressed in [4, 5].

In the following sections we will discuss and give an overview of the tools we have made during the last two years for the support of a course in object-oriented programming (in Java). In section 2 we describe the overall course plan system, which generates the 'home page' as such together with individual lecture pages. In section 3 we go on with an overview of the lecture note system via which it is possible to create a multi-view, hypertext-based teaching material. In section 4 and 5 we discuss the interactive services used on the course. Together with the conclusions we briefly point out some related work.

2 The Course Plan System

As a fundamental premise, we have decided to base the entire suite of tool on documents in pure HTML. This is a contrast to using specialized formats (calling for browser plugins), Java applets, or dynamic HTML such JavaScript. By this decision our materials can seen from any machine with a modern Internet browser without having to download or install any additional software.

A typical course home page covers a network of WWW pages which presents an overview of the course together with detailed descriptions and presentations of individual lectures. A number of details may occur redundantly on several pages. A course home page embodies knowledge about the course constituents (such as plenum sessions and exercises), course content (topics), reading materials, training plans (such as exercises), overall sequencing, and spatial/temporal details (schedule and room allocation).

It is not efficient to deal with this body of information in a collection of WWW pages, where also details about layout, typography, and decoration are addressed. We have made a *course plan system* which as input takes a complete and clean *course model*, and outputs a set of

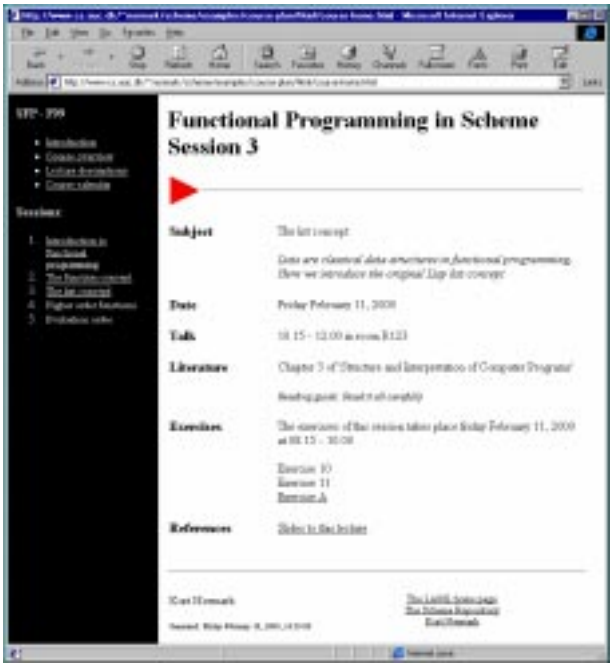

Figure 1: A snapshot of a demo course plan WWW page.

pages representing the 'course home page'. The course model separates relatively stable aspects of a course (such as topics covered) from the more fluctuating aspects (such as room and time information). In that way it becomes much easier to revise the course home page from one year to the next. Figure 1 shows a snapshot of a home page generated by the course plan system.

As one of the novel aspects of the course plan system, all information about time stems from a single list of 'lecture start times' together with additional information about the course model (involving details about the mutual timing of exercises and the plenum lectures). The system uses a central lecture description table which relates particular lectures to topics, times, and rooms. In addition, the table defines the sequencing among the lectures. Several different overviews can be generated, including listings of the course contents and calendar presentations, which plot the lectures into a semester overview calendar (with a possibility of merging calendar entries of other courses, attended by the same students). The generated pages share a common layout, and they include a number of useful standard links to course relevant resources. The lecture pages, which deal with details such as literature and exercises, can be processed individually. Alternatively, the complete set of pages can be regenerated by executing a single command. In the current system the course model is represented and edited in a textual 'programmatic' format. We are considering a more user friendly (but probably less powerful) supplementary interface based on WWW forms.

3 The Lecture Note System

Many teachers use transparencies and slides during a plenum lecture session. Such teaching material is often produced using presentation programs, for instance Powerpoint. Most presentation programs were invented before the advent of the World Wide Web and the Internet. The presentation programs can produce WWW presentable on-line documents, either via common document formats (such as postscript or pdf), via vendor supplied 'browser plug ins', or as bitmapped graphics. However, none of these *secondary formats* provides for a smooth integration with the rest of the resources on the Internet (for instance the course plan pages discussed above) nor full utilization of the power of hypertext.

We have made an alternative to these presentation programs, which produces slides in HTML as the *primary format*. The system, called LENO, processes a textual input file in a particular format, and produces a set of HTML pages. The practical processing procedure is similar to old fashioned text formatting, using LaTeX for instance. The input format, which is called LAML (Lisp Abstracted Markup Language) [5] is similar to XML, although different from XML when considering the details. From a technical perspective, the LENO input is a program written in the functional programming language Scheme using a particular library of Scheme functions. As a consequence of this, we can provide programmatic solutions to many routine tasks, because a full-fledged programming language is available anywhere in a document, and any time during the authoring process. For authors with a computer science background this programmatic approach turns out to be an interesting alternative to more conventional approaches, where abstraction and automation facilities are rather limited.

LENO organizes a teaching material as hypertext, centered around annotated slides. The material can be presented at three different levels of abstractions, one of which uses a large font to ease readability using a projector in an auditorium. The other views show the annotations in a separate column of the screen, and in an aggregated view including all slides in a lecture. LENO supports direct inclusion of external, textual material such as program source files. Partial inclusion of a text file as well as superposition of colors on the included text is also supported. The external material is taken from the input file when the HTML files are generated. This facility prevents a proliferation of different versions of the material. A *trail* facility makes it possible to make special blends of slides from different lectures without any replication of material. The trails are represented as HTML frame pages which refer to already

generated pages. Exercise pages, exercise solutions and overviews of exercises are also integrated with LENO. LENO makes it possible to open up for the exercise solutions at appropriate points of time. As an advanced feature, we can integrate the exercise part of the system with synchronous and asynchronous tools for management of the student's work with exercises (see section 4). Finally, LENO supports flexible keyboard navigation from page to page in selected browsers (realized via a simple JavaScript program). This turns out to be an essential facility when LENO is used for presentation of many slides during a lecture (but it also slightly compromises our ideas of using plain and simple HTML).

On the negative side, the graphical illustrations in LENO are rather primitive, because they are based on the image tag in HTML (which can present bitmapped graphics in the gif or jpg formats). It is hard to avoid scrolling of pages containing a single slide because of variations among browsers, browser setup, and screens. This is often a source of frustration. Similarly, paper hardcopies of the material (printed from a browser) are problematic. As a consequence of this we plan to make a special printed edition via a bridge from LENO to a conventional text formatting system such as LaTeX.

Based on our own experience with LENO we find that the virtues from above outweigh the deficiencies.

4 Synchronous Exercise Management

Following each lecture, the students carry out some concrete and practical exercises. During the exercises the students get help and advice from the teacher of the course and a number of teaching assistants. Due to the problem-oriented and project-organized teaching model at Aalborg University [2] the students are located in many small group rooms (with 6-7 students in each room). In this particular course, there were 25 groups, three teaching assistants, and the typical exercises involved practical programming in Java.

It is a major challenge to manage an exercises session which follows the setup described above. We do not know which of the groups work actively on the exercises, and it not trivial to ensure that a group of students get help when it is needed. Furthermore, it is difficult to evaluate the outcome of an exercise session.

In order to deal with these problems we have created a WWW-based interactive *exercise manager*. The manager presents itself as a small frame, which sticks to the window in which the exercise formulation is shown. Figure 2 shows an example. LENO (see section 3) can be set up to generate the underlying frame set. The exercise manager tool allows the students to send brief, one-line messages of particular types to the teacher via the WWW server.

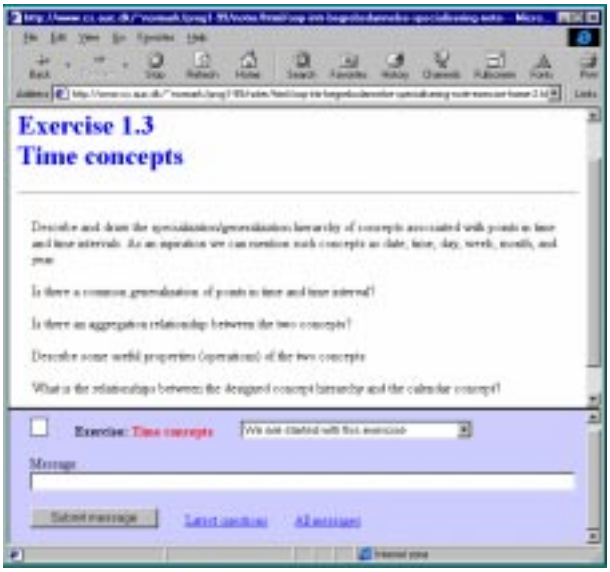

Figure 2: A snapshot of the exercise manager as attached to the bottom of a LENO generated exercise formulation.

The students use the exercise manager to send a start message when they start working on a particular exercise. If they encounter a problem they send an exercise question message together with a brief explanation of the problem. When a group of students get help they send a message of the type 'got help'. This makes the previous questions disappear from the overviews, which are regularly consulated by the teachers. Finally, the students send an exercise finish message when they are done with the exercise. As a variant, the students can send a 'gave up' message. The students are asked to explain the outcome of the exercise in the message field.

The students are encouraged to use the exercise manager via three different incentives. First, the students cannot ask for help via the system before they have sent a start message; Second, when the students send a question via the system, a teaching assistant will strive to visit the students within a few minutes; And third, the students get access to the teacher's solution (via 'opening of a link') when they claim to have solved the exercise successfully.

From a teacher perspective the exercise manager provides various useful overviews. As an example, the teacher can get access to a list of groups that are active at a given point of time. Similarly, the teacher, who services particular groups, can get a concise overview of the groups who need help. Only the latest, non-serviced questions are shown in the overview, and only messages from the particular groups, to which the teacher is allocated, are shown. As of now, the teachers use the student's browser to access the list of groups that have

asked for help. One could alternatively imagine that the teachers would use mobile equipment. The exercise manager can also generate a status report, in which the questions and feedback from the groups can be surveyed. We can also evaluate the waiting times for help, and the level of activity in the individual groups.

The successful use of the exercise manager requires discipline from both the students and teachers. One should also be aware that there may be elements of 'big brother watches you' using the system. However, the experiences until now is quite positive and encouraging. The students experience that they more often get timely help when they need it. Using the messages submitted to the system, it is possible to plan future modifications of the exercise programme, and to spot recurring problems that need to be addressed in a future plenum session on the course.

Until now, the messages from the exercise manager are one-way, from the students to the teacher. As an obvious generalization one could imagine a two-way communication using the system. The exercise manager was designed to alleviate some concrete problems with exercises in many small rooms. We believe, however, that some of ideas of synchronous exercise support via WWW tools can be used in other settings, for instance in more classical lab sessions in a distributed environment.

5 Asynchronous Activity Management

The exercise manager, described in the previous section, is a synchronous tool designed to be used by on-campus students. We are also supporting an asynchronous tool, IDAFUS [6], for *activity management*, which is targeted at open university students. The typical activities supported by IDAFUS are exercises and discussion forums. The dialogue mediated by IDAFUS may be public, limited to a group of students, or it may be private between a teacher and a student. IDAFUS allows 'just in time' solutions to exercises in terms of automatically released contributions from the teacher when the student submits his or her solution.

Contributions in IDAFUS are presented together with a photo of the contributing student or teacher. This has turned out to be very important in relation to seminars or lectures. Students who have been using the system a lot are easily recognized during subsequent classes or seminars.

6 Conclusions and Related Work

We will here briefly compare our suite of tools with similar tools. First it should be noticed that our tool set is less coherent and less complete than commercial counterparts such as WebCT [1] and Luvit (www.luvit.com). Second, our approach is radically different from the systems that feature quizzes, tests, and multiple choice questions. Examples of such systems are Hot Potatoes from Half Baked Software (web.uvic.ca/hrd/hotpot/), QuizSite from Indiana University (www.best.indiana.edu/qs2000/), and Question Mark's Perception (www.qmark.com). In our educational system we do not have traditions for 'black and white' testing techniques, as promoted by these three systems.

In summary, we have in this paper presented a suite of tools for management of WWW support of a university course in computer science. Taken as a whole, the tools make it possible to produce and maintain a large body of material which it would be almost impossible to manage without tool support. Our material for the object-oriented programming course consists of more than 1000 HTML pages, each with a number of internal links, and a considerable amount of links to external targets. In addition we have presented a couple of tools for synchronous exercise management and asynchronous activity management.

The course plan tool and the LENO system are available as free software from the LAML home page [3].

References

[1] Goldberg, M. W., and Salari, S. An update on WebCT - a tool for the creation of sophisticated web-based learning environments. In *NAUWeb'97 - Current Practices in Web-Based Course Development* (June 1997).

[2] Kjersdam, F., and Enemark, S. The Aalborg experiment – project innovation in university education. Aalborg University Press, Niels Jernesvej, DK-9220 Aalborg, Denmark, 1994.

[3] Nørmark, K. The LAML home page. http://www.-cs.auc.dk/~normark/laml/, 1999.

[4] Nørmark, K. Programming World Wide Web Pages in scheme. *Sigplan Notices 34*, 12 (December 1999). Also available via [3].

[5] Nørmark, K. Using Lisp as a markup language—the LAML approach. Presented at the European Lisp User Group Meeting, Amsterdam. Available via [3].

[6] Nørmark, K. The IDAFUS home page. http:-//www.cs.auc.dk/~normark/idafus/, 2000.