An Application-Centered Course on Data-Driven Web Sites

David P. Jacobs and Brian A. Malloy
Dept. of Computer Science
Clemson University
Clemson, SC 29634–0974
{dpj,malloy}@clemson.edu

Abstract- In this paper, we describe our experience with an *application-centered* approach in teaching a course on web technology. The purpose of the course was to teach the principles of data-base driven web sites, their design and the integration of the underlying components. The class was partitioned into small teams, where each team had their own dedicated server. Each team was responsible for achieving certain milestones that culminated in the implementation of a data-base driven web site. Lectures and demonstrations were used to provide information on a need-to-know basis. Our experience was highly positive. Synergy and student motivation grew as the course progressed. By the end of the semester, teams had developed near-professional quality web sites. We describe course details including syllabus, lecture topics, and milestone objectives.

1 Introduction

In computer science, current approaches to course instruction are typically *topic-centered*, where the major focus of the course is a discipline such as data structures or algorithms. A fundamental issue that must be addressed by the instructor of such a course is how to motivate the topics in the course. Some courses, such as computer graphics, are highly self motivating for students and require less effort by the instructor. Other courses, such as programming languages, require a great deal of planning and innovation to avoid losing student interest. In the topic-centered approach, programming projects tend to be unrelated so that they become "dixie-cup" programs that are used once to acquire a grade and then are thrown away. Even interest in a self-motivating course can be diminished by the assignment of unrelated projects. Furthermore, it is more difficult to incorporate software engineering topics into the programming projects when they are not developed incrementally.

In this paper, we describe our experience with an *application-centered* approach in teaching a course on web technology. The purpose of the course was to teach the principles of data-base driven web sites, their design and the integration of the underlying components. The class of upper-level undergraduates was partitioned into small teams, where each team had their own dedicated server. Each team was responsible for achieving certain milestones that culminated in the implementation of a data-base driven web site. Lectures and demonstrations were used to provide information on a need-to-know basis.

Our experience was highly positive. Synergy and student motivation grew as the course progressed. Initial interest in the course was strong because of the current popularity of the web. However, as the semester progressed and students saw their projects develop into complete web sites, interest in the course rose tremendously. By the end of the semester, teams had developed near-professional quality web sites that were well tested across several platforms.

In the next section we provide background information about web technology, including definitions of terms that are used throughout the paper. In Section 3 we describe the general approach to application-centered instruction. In Section 4,

we apply the application-centered approach to our course on web technology, including details about milestones in the course, phases of development of the web site, and our approach to supplying information on a need-to-know basis. Finally, we conclude in Section 5 with a summary of our experiences with the course.

2 Background

Although a web page may involve beautiful text, animated graphics, and even sound, it may really be just a *static page* that involves no interaction with the user. On the other hand, most sophisticated web site applications must read information from a user, process the information, and create new content resulting from that process. Examples of such sites are e-stores, proposal processing sites, and travel reservation systems. Typically such web sites are built upon a database. In a sense, the web site becomes a sophisticated interface between the user and database.

Often, the pages of such sites must be constructed dynamically. For example, the page is built in real-time, utilizing data (prices, seat availabilities, etc.) that might be changing. Such web sites must inherently involve programs or *scripts*. Although web-programming often utilizes *client-side* scripts, that is, programs that are interpreted or executed by the user's machine, typically most of the processing is *server-side*.

A server-side program must conform to the client-server protocol known as Common Gateway Interface (CGI). However, various technologies have been designed for providing a higher-level view, and for writing scripts that can construct dynamic web pages. Two such technologies are PHP on Unix servers, and Active Server Pages (ASP) on Windows servers. Even more sophisticated tools exist, such as Macromedia UltraDev, that can generate scripts automatically.

Active Server Pages allow the developer to create scripts in several languages, although the most commonly used are VBScript and JScript. The main advantages of this technology are that it hides details, and allows the programmer to mix HTML and script. Two of the most common tasks are making database queries and processing forms. Because HTTP is a stateless protocol, a common problem is how to represent the state of a session. Although this problem can be solved with Cookies, ASP provides a more integrated approach by allowing the programmer to declare Session Variables whose values persist during an entire session.

3 Application-Centered

In this section we overview, in general terms, the application-centered approach to computer-science instruction [6, 7]. We first discuss the important goals that guide the instructor in the application centered approach, and we then present the phases involved in implementing the approach.

Goals of the approach

Since the application centered approach requires more effort by both the instructor and the student, it is essential that the instructor review the goals of the approach with the students so that the effort is not wasted.

The first goal is to provide an environment where the students can become excited about the course. We have found that students are highly motivated about the topics in a course, even topics that would otherwise be uninteresting, when we used the application centered approach. For example, it was more relevant for students to study data base technology when they knew that they would soon use the technique in their implementation of an ongoing project that they might take with them when the course completes.

Figure 1 illustrates some differences in the topic centered approach as compared to the application centered approach. In the topic centered approach, a programming project is assigned to illustrate a topic; when completed, the students submit the project to the instructor, receive a grade, and the project is discarded. In the application centered approach, an ongoing project threads the course and the project is developed incrementally. When a stage of the project is completed, the students submit the partially completed project to the instructor, receive a grade, and then continue developing the project. An initial stage of the project,

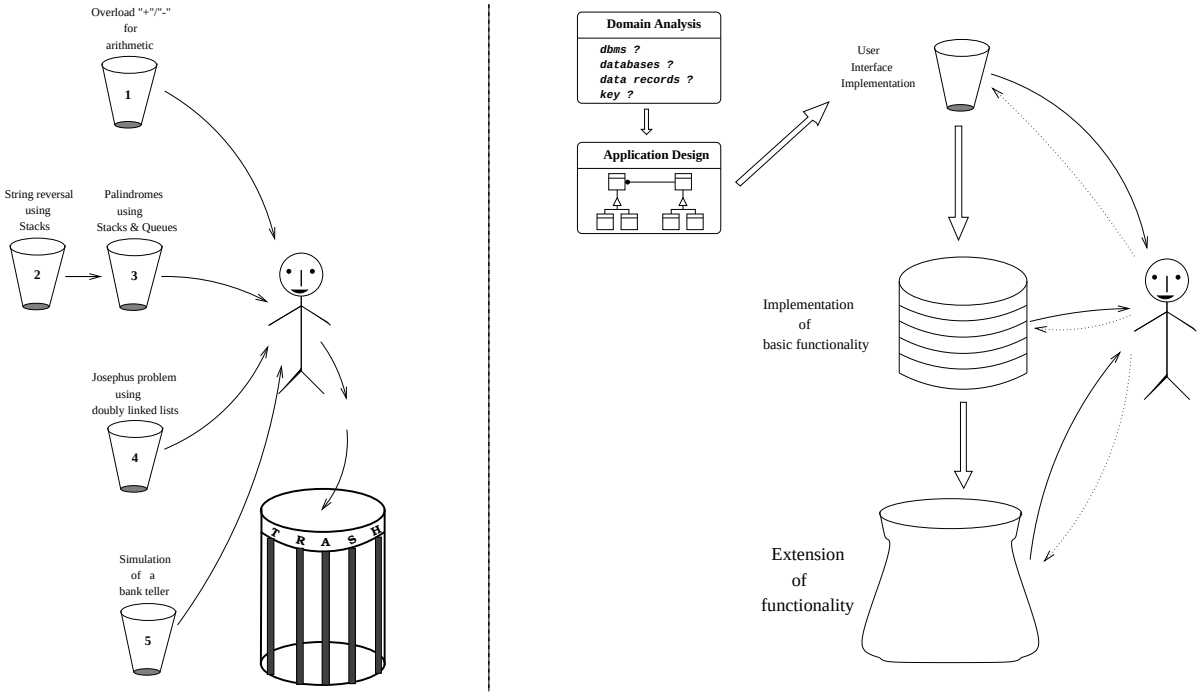

Figure 1: Comparing the dixie-cup approach to the application-centered approach. In the dixie-cup approach, various problems are assigned to the students such that each assignment helps the student to become familiar with a topic, such as how to use stacks or queues or how to use linked lists. A characteristic of this approach is that an assignment is unrelated to the previous or subsequent assignment. In the application-centered approach, each assignment represents a thread in the completed fabric.

---

as illustrated in the right side of Figure 1, is to implement a user interface; the second stage is to incorporate basic functions into the interface and the final stage is to extend the functionality. The user interface could be command-line driven or it may be a graphical user interface (GUI). In our case the user interface design required the students to describe actual web pages and links. These stages are discussed further in Section 4

The second goal of the application centered approach is to choose an application and corresponding course project that will provide a proper forum for topics that should be covered at this point in the student's development. A key idea in our approach is that the instructor is presenting the course topics in a context that gives the topics meaning. A good approach to ensuring student interest in the application is to allow the students to participate in the choice of application. Students are frequently more motivated to make the project work when they have shared in the responsibility for its choice. When the instructor chooses the topic, the students might not share the instructor's enthusiasm for the topic.

The application centered approach is effective even though some topics or techniques do not apply to the chosen application. In this case, the application provides a basis of comparison that frequently provides understanding of the technique and its applicability. For example, we presented CGI programming even though the students were not going to exploit this technology in their web based implementation. Nevertheless, the study of CGI provided insights into the engineering that might be occurring "under the hood" of ASP, a technology that they did use.

A third goal is to integrate software engineering principles into the course with other computer science concepts. In particular, we incorporate the engineering process of analyzing the problem explicitly, designing a solution based on the analysis and

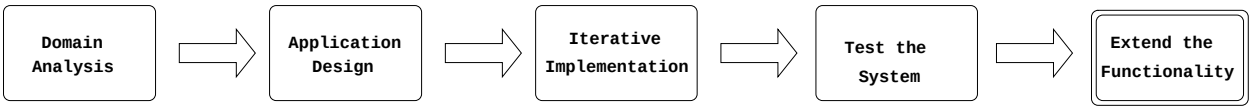

Figure 2: The five phases of development in the application centered approach.

following through with an implementation and test of the system. This approach is in contrast with many courses where no effort is spent on the design or testing of the system.

Implementation of the approach

We have partitioned the implementation of the application centered approach into five phases, as illustrated in Figure 2. We discuss each of these phases in this section.

In the domain analysis, the first phase in our approach, students research the types of systems that currently exist, and determine a niche that they might fill. The instructor reserves the right to veto the students' choice, to ensure that all of the required concepts can be supported. Having chosen the domain, the students must then research the domain topic to assure complete understanding of the domain concepts and to have a high-level knowledge of the kinds of tools that might be required in the chosen domain. Once the domain is understood, the students can move to the second phase of development.

The *design* of the project is the second phase of the process. The project is chosen so that a number of alternatives can be selected both for the functionality that they might add to the target application and the underlying technology that the student will master by extending the application. The project is also chosen for its ability to engage the students. The application must not intimidate by being too complex, nor should it be so simple that it bores the student. The scope of the problem should allow students to do real work and to build most of the application themselves. Students receive a sense of accomplishment from completing a project, so the problem should be sufficiently constrained to guarantee that most students complete the work. Once the scope of the project is established, the design can proceed. This design might exploit a modeling language, such as the Unified Modeling Language, UML, to actually model classes and their interaction [9, 2]. In our project, the design included descriptions of each web page and the corresponding links.

The third phase of the project requires that the students proceed with the implementation in a progressive fashion. We require students to begin by implementing the core of the system, that is, the fundamental part of the system required to "get off the ground". The core of the system represents an early deliverable and, once approved, the students may proceed iteratively to implement additional functionality. The fourth phase of the project, module and system test, should be initiated at this point so that even the core of the system has been tested. These test cases will represent part of the final deliverable, and can be reused for regression testing of later phases of the implementation.

The final phase of the project is to extend the completed project in some meaningful way. The purpose of this phase it to provide an opportunity for the student to experience the maintenance phase of application development. In "real world systems", maintenance represents an important and costly phase. For our project on web technology, we did not have time at the end of the semester to extend the project in a meaningful manner. In future semesters, with more experience with web topics and technology, we hope to incorporate this phase into the curriculum.

4 Applying the Approach

The purpose of this course was to explore tools, technologies, scripting languages, and design issues relevant to database driven web site construction. The prerequisites for the course were an undergraduate database course and some experience with HTML. The major activity of the course was a sin-

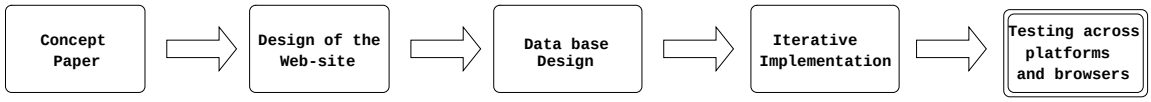

Figure 3: *Milestones.* This figure illustrates the milestones identifying the phases of student development of a web-site. The first milestone was a *concept paper* describing the major focus, goals and functionality of the web-site. The second milestone was a *design of the web site* that described the appearance of each page in the proposed site, and links between pages. The third milestone was a *design of the data base* including tables and fields. The fourth milestone was an iterative implementation broken into three phases. The final milestone was a *project portfolio*, submitted to the instructor, detailing the test procedure and included an invaluable users guide to the web-site.

---

gle project. Students were assigned to a team consisting of four or five students. While the instructor provided lectures, students were asked to be self-motivated enough to discover and explore independently. In order to tie ideas together and make them concrete, a case study web site was used throughout the course. Lecture material was often made available on a need-to-know basis.

Each team was given a dedicated server running the Windows2000 Server operating system, and IIS, the Microsoft web server. This environment provided students with Active Server Pages (ASP) technology. Each machine also had Access 2000, Microsoft's low-end relational data base. Although we might have used different technologies, we feel it was essential that all teams worked in identical environments. Our reason is that problems encountered by one team are often encountered, and successfully solved by another team.

The main text book for this course, and the only book required, was [11]. Some lecture material was taken from the book, but it was also used as a reference book. Every student was also asked to obtain some HTML reference; one recommendation was reference [3]. We also made available reference texts on ASP [4, 8, 10], a reference book on Access 2000 [1], and a text on Internet programming with VBScript and JavaScript [5].

Teams were told they would build a nontrivial application (web site) of commercial quality. It could be a retail store, a site related to art or music, or an original idea of their own. Teams and individuals were free to implement in any scripting languages (e.g. JScript, VBScript). Good software engineering principles and documentation were required.

The application was to be ASP-based and involve some nontrivial use of the database. Text and graphics were also required. The application was to use *session variables*, the ASP mechanism for data persistence throughout a session. The application needed to incorporate some use of client-side scripting (for example, form validation). Cascading style sheets (CSS) were also required to give the pages a consistent look. The application needed to have at least two views: the user's view and the administrator's view. Thus, password-protected administrative back end pages for updating and accessing the database were a requirement.

Optionally, the application could make use of many other features such as sound, video, animation, streaming media, a secure socket layer, email generation, file transfer, server side include files, development tools (eg. Front Page), or any other appropriate technologies.

During the semester there were several *milestones*. The first was the *concept paper*, due two weeks into the course. This document was to be about three to five pages, describing the functionality of a database web site. It was to describe the site from the point of view of both the user and the administrator. Data was to be described in general terms, but a description of the database or the specific appearance of the individual web pages, and their relationships, were premature at this point.

The next milestone was the *design*, to be completed at the end of four weeks. This document described the appearance of each page in the proposed

site, relationships (links) between pages, and details of the database including tables and fields. The design also included an implementation plan indicating how tasks (data base, programming, graphic design, etc.) would be parceled among team members.

The implementation was broken into three phases. By the end of week seven the database was to have been completed and populated with data. For example, the database for a retail store might contain products, prices, vendor information, and customer information. By the end of week eleven, the so-called *front-end* of the web site was to be completed. By the end of week fourteen the administrative back end was to be finished.

During the last week, each team submitted a *project portfolio*, a professional-quality guide to the project that might serve as an invaluable aid to a future developer or investor. In addition to the earlier documents, the portfolio also included the scripts (code), a discussion about known problems, evidence that the system had been tested using multiple platforms and multiple browsers. Teams were also asked to provide a six-month plan for the project in the event that capital was available to support continued development. The plan would include goals and needed tools. During the last week of class, each team also gave a thirty minute *presentation* of their project.

5 Experience and Conclusions

Our experience with this course was, on balance, highly positive. Almost without exception, every student expressed gratitude for having been able to participate in this course. Generally, students regarded the course as relevant and fun.

The projects were very high quality. One team built a retail store called MP3-World which allowed customers to purchase songs in MP3 format and download them. Another project was a restaurant site in which customers could order food for take-out, post and read restaurant reviews. A third project was a delivery service in which customers could order items from local stores and groceries. The fourth project was an outdoor adventure site, which in addition to having a store, provided information for many parks and outdoor recreational places in a three state area.

Given the limited time frame and level of student expertise, it was impractical to implement some aspects of the projects. For example, one thorny area involves credit card processing. Teams were encouraged to utilize off-line processing through the administrative back-end rather than real-time credit card processing. This drawback in the implementation did not detract from the learning experience.

References

[1] Andersen. The complete reference to access 2000. *Osborne/McGraw Hill*, 1999.

[2] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Object Technology Series. Addison-Wesley, 1999.

[3] Castro. HTML 4 for the world wide web. *Peachpit Press*, 1998.

[4] Hatfield. Active server pages for dummies. *IDG Books*, 1998.

[5] Kalata. Internet programming with vbscript and javascript. *Thomson Learning*, 2001.

[6] B. A. Malloy, D. Gupta, A. Kare, and J. D. McGregor. Incorporating reusability and extensibility into the cs 2 curriculum. *Proceedings of OOPSLA'95 Educators Symposium*, October 1995.

[7] B. A. Malloy, J. D. McGregor, and D. Gupta. An approach to blending analysis, design and implementation. *Proceedings of OOPSLA'96 Educators Symposium*, October 1996.

[8] Morneau. Active server pages. *Thomson Learning*, 2001.

[9] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison Wesley Longman, Inc, 1999.

[10] Weissinger. ASP in a nutshell. *O'Reilly*, 1999.

[11] Willie and Koller. Active server pages in 24 hours. *SAMS*, 1999.