TRENDS IN TEACHING WEB-BASED DEVELOPMENT
A Survey of Pedagogy in Web Development Courses

Ralph F. Grove
Department of Computer Science, James Madison University, Harrisonburg, VA, 22807, USA

Keywords: Web Development, Pedagogy, Computer Science Education.

Abstract: Many universities now offer a computer science course covering the development of information systems for the World Wide Web. Over the last ten years much has been written about experiences gained in developing and teaching such a course. This paper summarizes these experiences, including curriculum, common problems, and lessons learned. It also provides a suggested model curriculum and argues for more emphasis on fundamental concepts in addition to the standard core set of Web development technologies.

1 INTRODUCTION

Since its invention 15 years ago, the World Wide Web has become an important medium for commerce, entertainment, education, and communication, and the Web now affects virtually every part of our lives in some way. To computer scientists, the Web is an exciting phenomenon because of the revolution in software technology and computer applications that it has spawned. In the past decade, many computer science educators have come to regard the Web as a unique environment with respect to teaching software development, and courses focusing on developing applications for the Web have appeared at many universities. The experiences of those educators include several common problems related to teaching this course and some common solutions as well.

This paper presents a retrospective on 10 years of teaching Web development in colleges and universities, primarily in computer science programs. It summarizes curriculum design, experiences, problems, and concerns described in ten papers published primarily in CS education conferences and journals from 1998 to 2004. In chronological order, the papers describe these courses. Each course is one semester in length unless otherwise noted.

- *Web Development Technologies* (Lim 1998)
- *Internet Administration* (3 weeks, 24 total contact hours) (Walker and Browne 1999)
- *Webware: Computational Technbology for Network Information Systems* (Finkel and Cruz 1999)
- *Internet Application Design: Theory and Practice* (Klassner 2000)
- *Client/Server Programming* (Stiller and LeBlanc 2001)
- *Client/Server Programming for Internet Applications* (Chung and McLane 2002)
- *Web-Based Application Design* (Treu 2002)
- *Web-Based Development* (Yang and Grove 2002)
- *Web Programming 1 and 2* (a sequence of two semester-long courses) (Phillips, Tan, Phillips and Andre 2003)
- *Internet Application Development* (Yue and Ding 2004)

This paper also presents a model curriculum for a one semester course in Web application development. The proposed curriculum is based upon the experiences described in the above papers and the Computing Curricula 2001 guidelines (Computing Curricula 2001).

To assess the prevalance of Web-based development courses, an informal survey of universities with computer science or engineering departments was conducted. The computer science/engineering program course listings for 100 highly respected American and European schools with computer science/engineering programs were reviewed for courses that described primarily topics related to Web or Internet programming. Of those schools, 25% offered one or more such courses. Of the remaining 75%, it's possible that Web programming is integrated into other courses or that the course listsings were not complete, so 25% should be considered a lower limit on the percentage of schools offering a Web development course.

2 CURRICULUM ISSUES

The courses described in the papers were undergraduate courses of standard one quarter (10 weeks) or one semester (15 weeks) length, with one exception, which was a short course offered on weekends (Walker and Browne 1999). All of the courses assumed introductory-level programming experience, though the actual diversity of experience in classes was often found to be problematic (discussed in Section 3). Other prerequisites described in the papers included:

- Software Engineering (2 papers)
- Operating Systems (2 papers)

Not all of the papers described course prerequisites, however.

Topics covered in the courses tended to emphasize a core set of development technologies, including client and server programming languages (more than one of each in some cases), database programming, and server configuration. This set provided background for project development, which was an essential curriculum component, though the specific contents of the set varied, e.g. Java vs. Perl. Beyond this essential programming knowledge other topics covered in these courses included network protocols, Web application architecture, security, encryption, media and compression, an overview of other (than the core set) technologies, ethics, and history. None of these were universal, however.

The choice of which core set of development tools to use was driven by a number of factors, including pedagogy, program curriculum (primarily introductory programming language), laboratory facilities, budget, and instructor preference. The most popular set of tools was Apache/Tomcat, Java/J2EE and MySQL on a Linux host. This choice was driven primarily by cost and availability, since all of these components are open-source and can run on low-end computers. Using open-source software also eliminated any licensing problems for students who wanted to build the same environment on personal computers. The fact that most students learn C++ or Java as their introductory programming language was also mentioned as a factor in choosing Java. CGI/Perl was also used frequently as a server-side programming language, in addition to Java in some cases. Windows and the ASP programming environment were used in one case and were mentioned as part of an earlier prototype for the course in two other cases. PHP was used in one case as the server-side programming language.

The courses used a variety of standard assessment methods. All of the courses included a project development assignment in addition to standard activities and assessment such as homework and exams. Projects required design and development of a Web application with active server components. Most of the projects were team-based though individual projects were assigned as well, or exclusively in some courses. Some of the projects also required installation and configuration of an operating system and/or Web server. Most projects were student-conceived though in a few courses actual Web sites were developed for local clients. Many of the courses also included a student presentation assignment requiring students to research and present information on a particular Web technology. This assignment has the added benefit of exposing the entire class to a wide variety of technologies beyond those in the core set.

Laboratory needs were unusual in that students required an inordinate level of access to a server in order to complete class projects. One solution was to dedicate to the class a lab room with enough computers to provide two computers (client and server) to each student team. These systems were isolated from the Internet in order to forestall security problems. In other courses each student was provided with an individual server connected to the Internet. Another solution was to provide a common server platform connected to the Internet for the entire class, with a unique instance of Apache for each student, bound to a unique port number. Some authors felt it was important for students to go through the process of building the entire server, including installation of the operating system, in order to appreciate the task of system administration, while in other courses students installed only application components.

3 COMMON PROBLEMS AND CONCERNS

The number and diversity of protocols and programming languages required for Web development presented a problem. Students typically learned several Internet and Web protocols (e.g., TCP/IP, HTTP. SSL/TLS) as well as markup languages (HTML, XML), a scripting language (e.g., Javascript, ECMAScript), one or two server programming languages (e.g., Perl, Java, JSP, C#), and a database manipulation language (SQL). Students were also required to learn to configure and control operating system and server components (e.g., Linux, Apache, Tomcat, ASP/.Net). Mastery of all of these is an unreasonable objective but some practical level of understanding of each is necessary for completing a Web development project.

Many of the authors reported difficulty in keeping the course current with respect to the development tools. Given that the course touches on a Web client, client scripting language, operating system, Web server, server programming language, and database, there are at least six software products to keep up with. This course is typically taught once per year, which is the same approximate frequency for major upgrades to all six products, and so the instructor must complete a significant research and revision project every time the course is taught. If additional tools and languages beyond a core set are taught the problem becomes more severe. The problem can be addressed in part by simplifying the set of topics. For example, Tomcat can serve as both a Web server and servlet container; the languages taught can be reduced to Javascript, Java or Perl, and SQL.

Several authors reported having a problem with the diversity of student backgrounds with respect to programming experience and knowledge of Web technologies. For example, a course might include lower level students who have completed only introductory programming courses as well as experienced upper level students and possibly part-time students with professional experience. Teaching HTML, for example, is at the same time necessary for the lower level students and boring for the others. One solution is to provide tutorials and/or labs covering introductory material and require inexperienced students to learn them independently. This seems to work well for topics such as HTML or basic Linux commands, which are not very complex and not totally foreign. Another practical solution is to group students into teams such that experience is complementary and allow students to get help from team-mates as needed.

Management of the laboratory development environment took an excessive amount of time for some instructors, as much as 50 hours for one instance of the course. This problem is likely attributable to immaturity of the software components being used and inexperience with using the software to support this course. As the software improves and experience is gained maintenance should be less of a burden.

Most of the authors reported being unable to find a suitable textbook for the course. The available books failed to provide adequate and complete coverage of the course material individually. In some cases, more than one book was assigned. This may be a reflection of the fact that this course is implemented in very different ways by different instructors, making it difficult for authors to produce a generally useful text. It is also likely that textbook production has required some time to catch up with curriculum development, and that there are now more good textbook candidates available than when these papers were written. For example *Programming the World Wide Web* (Sebesta 2005) and Java Web Development Illuminated (Qian, Allen, Gan and Brown 2007) are relatively new books targeting this topic.

4 MODEL CURRICULUM

The primary impetus for increased interest in teaching Web-based development has been the continuing growth in economic and social significance of the Web. The Web has become a significant medium in commerce, communication, education, etc., and has spawned entirely new business and information sharing models, such as Google and Wikipedia.

This growth has been reflected in Computing Curriculum 2001, which is a widely accepted standard model for computer science education that was developed jointly by the IEEE Computer Society and ACM. The CC2001 report recognizes the significance of Web-based systems to the computing curriculum in Chapter 3, "Changes in the Computer Science Discipline".

Today, networking and the web have become the underpinning for much of our economy. They have become critical foundations of computer science, and it is impossible to imagine that undergraduate programs would not devote significantly more time to this topic.

Fifteen hours of the CC2001 core curriculum are related to Net-Centric Computing, including three hours focused directly on Web-based systems (*NC4. The web as an example of client-server computing*). Web-based systems also appear as an elective topic within the Net-Centric Computing group (*NC5. Building web applications*). In the five years since

CC2001 was published the significance of the Web as a development environment has grown, giving this topic increased significance in the computing curriculum.

Several common principles are suggested by the experience of teaching Web-based development.

- *Technology is secondary to concepts.* Whether students learn to program server-side components in CGI/Perl, J2EE, .Net, Ruby, etc., is relatively unimportant. A student who thoroughly understands the structure and operation of an application built with any of these frameworks can adapt that experience to the others as needed.
- *Practical experience is necessary.* Teaching Web development abstractly is not likely to produce essential learning. Students must be actively involved in the development of a realistic application in order to thoroughly understand how Web systems function.
- *Architecture is fundamental.* Learning to use various programming languages does not constitute an understanding of Web applications. Proper architectural design is essential to successful development of an application with real scope. Both networking architectures (client-server models) and software architecture such as the Model-View-Controller pattern for server-side systems are essential to a thorough understanding of Web systems.
- *Security is essential.* Web systems do not operate in a vacuum. An awareness of common threats for Web applications and their countermeasures is essential for developers.

The model curriculum for a Web-Based Development course suggested here (Table 1) is derived from the collective experience described above as well as first-hand experiences of the author in teaching this course. This curriculum is designed for a standard semester-long upper-level course in a computer science curriculum. The only absolute prerequisite for the course is programming experience, either in Java or in another Web-compatible language. Software Engineering, Database Systems, and Operating Systems are possible prerequisites, but aren't essential. Orthogonal topics such as Software Engineering and UML are not included but may be integrated across the listed topics.

This model curriculum focuses on the development of application functionality (primarily server-side) and does not include topics related to user interface design. The basics of programming a simple but functional user interface (XHTML Forms, Javascript, Ajax) are included, but presentation technologies (e.g., CSS, Flash) and the entire topic of interface design and testing are not. These topics are extensive and complex enough to require separate courses, even separate majors! In fact, the design of interfaces for Web applications is generally handled by professionals other than those who develop the server-side software, so this approach is consistent with industry practice. It is reasonable, however, to explain to students how to map the flow of an application interface, using a UML state chart, for example.

Table 1: Model Curriculum Components.

Topic	***Weeks***
Internet & Web Protocols	1
Web architecture • 2,3,N-tier client/server systems	1
Web server operation and configuration • {Apache, IIS, Tomcat}	1
Client-side Programming • XHTML Forms, Javascript, Ajax	2
Server-side Programming • {CGI/Perl, J2EE, .NET}	2
Database access • {MySQL, JDBC, ODBC}	1
Web Application Design • Model-view-controller pattern	1
Security & Encryption • Threat model for Web app's • SSL, HTTPS	1
Efficiency & Reliability	1
Media & Compression	1
Internationalization	1

The topic list presented in Table 1 is meant to be roughly sequential. The first few weeks are given to discussion of how the Web works in general, including its basic protocols (IP, HTTP, XML), architecture, and operation. The "server operation and configuration" section should include the operation of servers in general (request handling, multithreading, etc.) as well as the configuration of the specific server that students will use in their projects. The next six weeks present specific programming technologies for client, server, and data management layers of Web applications. The specific set of tools to be covered should depend upon the background of the students, experience of the instructor, and available laboratory facilities for students. The specific set that is covered is unimportant as long as it is consistent and adequate. The study of Web application design should help students to place into context the set of development technologies that they learned previously. The Model-View-Controller design is especially appropriate since it is an excellent model for structuring a Web application. It also provides a basis for discussion of Web applications from a software engineering perspective. Alternative designs or frameworks may be presented as well. The study of security and encryption should include a comprehensive threat model that gives students an understanding of various ways in which a Web application can be penetrated and the countermeasures that can be incorporated during design and development. Basic security tools such as SSL should also be presented and required in student projects. The final three topics are highly important to successful Web applications but may be omitted from student projects in order to manage their scope. Students might practice what is learned in these weeks with lab exercises instead.

5 SUMMARY

Several courses in Web-based development have been described in papers published over the last eight years. Instructors have described these courses as successful and feel encouraged to continue offering them despite difficulties involved. The descriptions share several themes, including:

- the difficulty of keeping up with changing technologies
- the complexity of managing laboratory facilities
- the difficulty of dealing with diverse student backgrounds and experience levels
- the lack of an appropriate textbook

Most of these problems will persist as courses in Web-based development continue to evolve. Though development technologies will become more stable and reliable (reducing lab administration headaches), they will also become more complex as their capabilities and features are extended. The availability of textbooks for this course is improving, however. Several newer books offer instructors good textbook options.

The courses surveyed here included a variety of practical experiences for learning and assessment, the most common of which was a team project requiring development of a working application. Leveraging diverse student experience by requiring them to complete a presentation on leading Web technologies was another common practice.

Newer architectural developments such as the Service Oriented Architecture and mobile computing will challenge instructors to present a complete picture of Web development in a single semester. One of the papers surveyed describes a two-semester sequence for students wishing to study Web development, which might become the norm as Web technologies continue to grow in complexity and diversity.

REFERENCES

Computing Curricula 2001. (2001). Joint Task Force, IEEE Computer Society and ACM.

Chung, W. S., & McLane, D. (2002). Developing and enhancing a client/server programming for internet applications course. Journal of Computing Sciences in Colleges, 18(2), 79-91.

Finkel, D., & Cruz, I. F. (1999). Webware: A course about the web. ACM SIGCSE Bulletin, 31(3).

Klassner, F. (2000). Can web development courses avoid obsolescence? ACM SIGCSE Bulletin, 32(3).

Lim, B. B. L. (1998). Teaching web development technologies in CS/IS curricula. *Technical Symposium on Computer Science Education Archive*, 107 - 111.

Qian, K., Allen, R., Gan, M. & Brown, R. (2007). *Java Web Development Illuminated*. Jones and Bartlett.

Phillips, J., Tan, J., Phillips, M., & Andre, N. (2003). Design of a two-course sequence in web programming and E-commerce. Journal of Computing Sciences in Colleges, 19(2).

Sebesta, R.W. (2006). *Programming the World Wide Web*. 3rd Edition, Addison-Wesley.

Stiller, E., & LeBlanc, C. (2001). Teaching Client/Server programming in the context of computing curricula 2001. Journal of Computing in Small Colleges, 16(4), 79-91.

Treu, K. (2002). To teach the unteachable class: An experimental course in web-based application design. Proceedings of ACM SIGCSE 2002, 201-205.

Walker, E. L., & Browne, L. (1999). Teaching web development with limited resources. ACM SIGCSE Bulletin, 31(1), 22-26.

Yang, T. A., & Grove, R. F. (2002). A web-based application development course for the computing curricula 2001/NC3 track, Procedings of WWW 2002.

Yue, K., & Ding, W. (2004). Design and evolution of an undergraduate course on web application development. ACM SIGCSE Bulletin, 36(3), 22-26.