Awakening Rip Van Winkle: Modernizing the Computer Science Web Curriculum

Randy Connolly
Dept. Computer Science & Information Systems
Mount Royal University
4825 Mount Royal Gate SW, Calgary, AB, Canada T3E 6K6
403-440-6061
rconnolly@mtroyal.ca

ABSTRACT
The world of web development has experienced a great deal of change over the past decade. The importance and complexity of web development is currently not adequately reflected in the ACM Computer Science 2008 Curriculum, nor in most reported computer science programs. This paper examines published literature on teaching the web since 2001 and argues that the computer science curriculum needs to be woken up and modernized in regards to the importance of web development. The paper critiques the approach of teaching web development topics within a single course. It articulates a wide variety of web development topics that need to be covered in any contemporary computer science program and which are often absent in other published accounts of this course. The paper concludes by arguing that a multi-course stream in web development can help the students integrate the discrete pieces of knowledge garnered during their undergraduate education.

Categories and Subject Descriptors
K.3.2 [Computers and Education]: Computer & Information Science Education – computer science education.

General Terms
Design, Experimentation.

Keywords

1. INTRODUCTION
One of the earliest and most well-known characters in American literature is that of Rip Van Winkle, who appeared in Washington’s Irving’s Sketch Book of 1819. In the story, Rip, a bit of a lazy vagabond, drinks some dubious home brew up in the hills, falls asleep, awakens twenty years later, and is utterly astounded by the transformation of his colonial village of Sleepy Hollow into a bustling commerce-driven republican town. If like Rip, we were to have slept, but only for the ten years since 2000, we also might be amazed at the transformations in how people experience the web. Back in 1999 people used the web for occasional online shopping at home and examining stock prices or breaking news on their desktop computers. While this kind of usage is still important, the type of web sites being accessed, the user experience while using these sites, and the devices we use to do it has changed quite a bit in the intervening ten years. In this decade, the web has transformed into an ever-present information retrieval mechanism as well as the principal platform for hosting software applications [25]. Somewhat inexplicably, however, while the web has experienced this remarkable transformation, there has been a certain dormancy to the way that the web is being taught in the computer science curriculum.

While web topics are certainly a recommended component of a variety of ACM curricula reports, it still very much appears as a peripheral topic within computer science. In the CS 2008 curriculum, for instance, almost all of the topics recognizable as pertaining to the field of web development are marked as elective topics [1]. Similarly, there is a relative dearth of research in the computer education literature which is somewhat surprising given the ostensive importance of web technology in the real world of software development. Each year SIGCSE and ITiCSE have multiple papers on teaching beginning programming, databases, and other curricula areas, but only fairly rarely will a paper on teaching web development appear.

In fact, looking at all the papers presented at SIGCSE and ITiCSE in the last decade, only about 1.5% of them have pertained at all to the teaching of web topics. Since 2005 the percentage is even lower; there have been only 17 papers during that time about web topics – and many of those were only peripherally focused on teaching web development. As a point of comparison, in the same five-year time span, there were 65 papers on game-related topics in those two conferences. While clearly this represents the discipline’s interest in trying to increase enrollments in computer science via the appeal of games, it certainly is not at all indicative of the job market computer science graduates will face, one in which arguably a majority of software development jobs are broadly within the web context [21]. One can verify this by examining an employment website such as monster.com or simplyhired.com. For instance, on two different three-day time periods in October and November 2010, there were roughly twice as many web-related development job postings as non-web ones on those two job sites.
Now it is quite possible that the reason why there have been so few papers on teaching web topics is because the professors teaching these courses are too busy adapting to new web technologies to publish their experiences. This is certainly the impression one gets from reading many of these papers (for instance, [2], [19], and [40]). These papers invariably agree with Treu’s designation of the web application course as the “Unteachable Class” [34].

This paper is going to argue that the computer science curricula needs to be woken up and modernized in regards to the importance of web development, and will argue that a two- or a three-course stream in web development can acquaint students with the real world of web application development in a significantly less “unteachable” manner. The paper will describe a variety of web development topics that should be covered in a web stream, and yet which are almost always absent in published accounts of web courses. Its perspective is informed by the author’s experience teaching these topics since 1998, by his ongoing experiences as a professional web developer for a wide variety of international clients, and by his experiences writing a book in this field. The paper will conclude by arguing that such a series of courses can also address one of the key problems in the education of our students: namely, how to integrate the discrete pieces of knowledge that they have gathered during their progress through their computer science education.

2. HOW THE WEB IS BEING TAUGHT

Most computer science educators would no doubt agree that there is a lot of material that needs to be shoe-horned into a typical four-year computer science program. The ACM curriculum for computer science [1] articulates the very broad body of knowledge in computer science; it consists of recommended mandatory learning outcomes as well as additional optional outcomes. The web topics, subsumed under the banner of Net-centric computing, include only a very small number of mandatory core topic outcomes, mainly of the awareness variety. There are also a number of elective topics that conceivably could be covered by a single optional course on web development. Tellingly, despite a decade of transformation in the real world of web development, the Net-centric topic list in the CS 2008 report is essentially unchanged from the 2001 version [17].

Yet it appears that despite the (limited) mandate of the ACM curriculum and despite the importance of web technology to the student’s own lives, to their job prospects, and also to the research efforts of many computer science professors, not every computer science program includes a course in these topics. In 2003, Lee noted that “there is a surprisingly small number of universities that offer a serious web programming course for the advanced computer science student” [19]. Six years later, Stepp in 2009 noted that most universities still don’t cover web development in their computer science programs, but do so in their IS or IT ones. [33].

Given the unruly and shifting number of web standards and practices, it is not surprising that “there does not seem to be a consensus about where in the curriculum, and at what detail, to introduce this material” [33]. Surveying the literature that does exist on teaching web topics, one can see the most common teaching approaches in Table 1. As is apparent from the number of references, by far the most common way of teaching the web course in the reported literature is to try to fit all the material within a single upper-level course. Not coincidentally, it is in these papers that one finds most of the complaints about the difficulty of teaching the web course.

### Table 1. Summary of CS web education papers

<table>
<thead>
<tr>
<th>Course Approach</th>
<th>#</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teach the material as a CS0 course or as non-major elective.</td>
<td>4</td>
<td>12, 16, 26, 27</td>
</tr>
<tr>
<td>Use Javascript in the CS1/CS2 course or teach web topics in a CS 1.5 course.</td>
<td>3</td>
<td>16, 29, 33</td>
</tr>
<tr>
<td>An intermediate to advanced single web course which sometimes serves as a capstone course.</td>
<td>14</td>
<td>2, 3, 4, 10, 14, 19, 22, 23, 29, 32, 34, 36, 37, 40</td>
</tr>
<tr>
<td>A two (or more) course stream on web development.</td>
<td>3</td>
<td>13, 16, 24</td>
</tr>
</tbody>
</table>

3. WEB TOPICS TO BE TAUGHT

Given the oft-stated worries about the breadth of material needing to be taught in a web course, this author strongly believes that the All-the-Web-in-One-Course (AWOC) approach very much needs to be retired. Back in the late 1960s/early 1970s, a math program might have had a single course in programming in Fortran or a business program might have a single course on data processing in Cobol, but eventually it was recognized that a body of knowledge as complex as programming requires multiple courses to teach the material properly. Web development should by now be in a similar state.

The principal reason why the AWOC approach is no longer appropriate is due to the peculiar combination of change and persistence that characterizes the web development world. As noted by many of the authors cited in this paper, the web environment is one very much characterized by flux. It is this author’s belief that the web environment experiences something analogous to paradigm shifts, in that web development experiences cycles of short periods of substantial change followed by periods of relative stability (see Table 2). The key point is that these are not exactly paradigm shifts in the Kuhnean sense since a previous web technology isn’t always being displaced and replaced (though, for instance, with CGI and ASP it was); more often the next set of important technologies or approaches are being used “on top” of, or in addition to, the previous ones, meaning that knowledge and expertise in the previous paradigm is still required.

### Table 2. Web development paradigms

<table>
<thead>
<tr>
<th>Principal technology context/layer</th>
<th>Approx. Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML + CGI</td>
<td>1995-1998</td>
</tr>
<tr>
<td>CSS (simple) + Javascript (simple) + ASP/JSP</td>
<td>1999-2002</td>
</tr>
<tr>
<td>Semantic Web Standards + PHP/ASP.NET</td>
<td>2003-2006</td>
</tr>
<tr>
<td>AJAX + REST/JSON services</td>
<td>2006-2009</td>
</tr>
<tr>
<td>Frameworks + Platforms (WordPress, JQuery, Sharepoint, etc)</td>
<td>2009-</td>
</tr>
</tbody>
</table>
There are indications that the web development environment is currently at the start of a new cycle of change and, as such, new topics need to be integrated into how we teach the web. For those stuck in an AWOC model, it will become increasingly difficult to keep those courses comprehensive and relevant. As the title of another paper with a similar theme suggests, those courses will increasingly seem like they are “partying like it’s 1999” [5].

So what needs to continue to be taught in any web course and what new topics need to be added in? The remainder of this section describes those web topics divided into two subcategories: material that is prescribed, however briefly, in the ACM CS 2008, and material that is not mentioned in the ACM report, but which, nonetheless, needs to be covered in contemporary and future web courses.

3.1 Web Topics in CS 2008
The obvious beginning point of any web stream is with the way the web works (e.g., the various protocols, the hardware infrastructure, etc) and with HTML itself. While HTML is relatively trivial to learn, it is important to cover it in a way that is consonant with contemporary best practice usage. For reasons of accessibility [27], maintainability, content management system (CMS) integration, and search-engine optimization (SEO), HTML markup today is semantically-structured so as to separate content from presentation [41].

For these same reasons, CSS is now an essential part of web development. CSS coverage in reported AWOC courses is often quite minimal [2,22,36]. While basic text formatting in CSS is indeed quite straightforward, real world CSS, which is commonly used as well for positioning and layout, is notoriously difficult to master due to browser bugs, incompatibilities, and non-obvious CSS box model interactions [41]. To complicate matters further, the CSS landscape is undergoing a period of transition, with both CSS3 and CSS frameworks like 960 adding to the layers of CSS knowledge necessary for contemporary and future practitioners.

Javascript is another key web development technology. The type of Javascript that can be covered in just a few lectures (rollovers, form data validation, browser sniffing) was reasonably close to what was needed professionally in the late 1990s. Since the “discovery” of XmlHttpRequest and the subsequent flourishing of new user interface coding and asynchronous communication with web services, Javascript coding has become simultaneously crucial to contemporary web development and significantly more complicated. Analogous to the case with CSS, this type of Javascript programming is very difficult to learn due to browser differences, the untyped nature of the language, the lack of a cross-browser debugging environment, and the general conceptual complexity of working with callback functions.

Another key part of learning real-world web development is the server-side environment. Potentially this is a very large topic and has its own difficulties from a teaching perspective. There are a number of different competing technologies (PHP, ASP.NET, JSP, Ruby on Rails) which all accomplish the same thing: interacting with server resources and programatically generating HTML, CSS, and Javascript that is returned to the browser. Server-side development also has a number of substantial additional topics, such as the HTTP protocol, SQL and database-access APIs, replicating database changes across data servers, local and distributed transactions, maintaining state (via cookies, sessions, querystrings, and form elements) across requests (and, in a web farm, across processes), internationalization, architecting web infrastructures for scalability, and enterprise design patterns, which are almost impossible to fit into a AWOC course.

Digital media and information architecture and usability is another vital area of real-world web practice. This area is often under-represented in most reported web courses. Usability in general is a very large topic and will likely be covered in a computer science program’s HCI courses. Nonetheless, usability in the web context does require its own unique topics which may not make it into the typical HCI course. These topics include: the unique factors affecting web site usability, the different ways of organizing and structuring web content, the development and articulation of web conventions, designing web navigation systems, and an overview of visual design principles. These usability topics rarely make it into an AWOC course. Of all the papers examined for this article, only [27], [31], and [39] included usability or digital media in its reported topic lists.

Web vulnerabilities/security is another vital web topic that can be difficult to comprehensively cover using the AWOC approach (see [22] for an exception). Security in general is a very large topic, and many of the more important general issues and concepts are likely to be covered in a dedicated networking course. Web security does have its own unique problems and concerns. For instance, cross-site scripting, SQL injection attacks and other common web vulnerabilities are unique to the web context and as a consequence are often not covered in a standard network security course. Interestingly, none of the current web textbooks [7,18,30] examined for this paper contained any substantial material on web security.

3.2 Web Topics Not in CS 2008
While most of the reported web courses cover some portion of the preceding topics, the remaining topics that need to be taught in any future web courses are almost never covered. The first of these topics is that of web frameworks and APIs. One of the key features of the contemporary web landscape is that many organizations are no longer creating their web infrastructure from scratch but are using single (or integrating multiple) already existing open-source and/or proprietary web frameworks [11, 35]. Complex content management systems such as Drupal, Joomla, DotNetNuke, or Microsoft SharePoint, blogging systems such as WordPress or Blogger, web forums such as phpBB or vBulletin, e-commerce systems such osCommerce or Magento, and common business practice portals and services such as SugarCRM, Salesforce, or Exact are often used as the main framework for an organization’s public or private web presence.

In other words, the platform for current and future web development is expanding significantly beyond LAMP (Linux, Apache, MySQL, and PHP) or WISA (Windows, IIS, SQL Server, and ASP.NET). Contemporary web development is increasingly more about building new solutions that involve customizing or leveraging APIs and/or extension frameworks (which may themselves be built on LAMP or WISA, so knowledge of them is still required), such as the Facebook API, Sharepoint, Drupal, WordPress, Zend, JQuery, and ASP.NET MVC. The scale of something like the Facebook API usage (over a million developers and over half a million applications [9]) or...
the number of sites built on WordPress (perhaps as many as 10% of all significant sites on the entire web [11]) is often invisible to the end user, but is something that needs to be made completely visible to our computer science students. For the students, it is important to learn about the existence of these existing frameworks because the future of web systems will increasingly lie in their use, integration, and customization.

Another important part of the web systems knowledge area that is generally missing from computer science education is that of hosting and deployment. For practical reasons, students generally learn web development using their own computer or a lab computer as if it was a web server. Ultimately, however, a web site needs to be deployed on a public web server. Students thus need to learn about the advantages and disadvantages of the main web server platforms as well as third-party hosting environments. Hosting topics such as web gardens, web farms, load balancing, data center redundancy and replication, as well as server configurations for scalability also need to be covered if a computer science graduate is going to have a realistic sense of the contemporary web world.

Finally, it is important for students to learn about web service consumption and integration, especially RESTful services (though SOAP services should be taught as well). With the broad interest in the asynchronous consumption of server data at the browser using Javascript (generally referred to as AJAX) and due to the relatively easy availability of a wide-range of RESTful services, a new style of web development known as the mashup has become increasingly common [10,14]. Consuming these services typically involves learning XML parsing, XPath search expressions, as well as more conceptual issues such as mediation techniques between heterogeneous data sources [6].

4. SAMPLE WEB COURSES

The previous section described a range of topics that should be covered in our web courses. Below is one sample way to split this material across three hypothetical semester courses. It should be mentioned that this list is not completely hypothetical in that it is the way this author’s department teaches the web. It may be possible to reduce some of the coverage in order to fit it into just two courses or to fit it into three quarter-style courses.

4.1 Web 1

This course covers the concepts and practice necessary for creating an effective web site at the introductory level. Topics include: how the web works, XHTML (with an emphasis on web standards, accessibility, and semantic markup), CSS (not just styling but also positioning and layout), digital media, information architecture and usability, a very brief introduction to Javascript, and a brief introduction to server-side development.

4.2 Web 2

This course covers the concepts and technologies needed to design and develop server-side applications. Its main focus would be on how server-side technology works and on developing with server-side technology. Complicated web development environments such as JSP or ASP.NET have a substantial learning curve. A fair percentage of this type of course must be devoted to teaching the practicalities of developing within it. Other topics would include accessing databases in web applications, software design in web applications, designing for scalability and reliability, mechanisms for maintaining state, consuming REST and SOAP web services, and designing and implementing web security (though these last two could move to the Web 3 course).

4.3 Web 3

This course covers: intermediate Javascript development, asynchronous consumption of web services in Javascript, using Javascript frameworks and web APIs, web application deployment and hosting, and issues in adapting web sites for differing locales and cultures.

5. CONCLUSION

It is this author’s strong belief that teaching web topics can play a vital role in the development of a computer science student. Adams has noted that one of the key limitations with computer science education and the many specialized knowledge competencies it tries to engender is that “students struggle to put all the pieces together” [2]. Humphrey similarly noted this lacuna not just in undergraduate but in graduate computer science students as well; even graduate students “usually have little or no experiences in designing, implementing, and evaluating, large-scale software systems for complex, dynamic, and heterogeneous environments” [14]. That is, it is not enough to teach each technology and concept in isolation; “students must understand how these technologies relate to each other” [2]. This theme has also been raised by a number of other authors in their reflections on what needs to be improved in the way computer science is taught [8,28].

One of the key benefits to teaching web development in two or three serious web courses is that it can provide the students with a taste of that needed integration and complexity. Kazmerik [15] reported in an undergraduate student’s perspective on just such a series of two web courses that the experience was akin to finally getting to play the guitar. In Kazmerik’s analogy, all the prior computer courses were about understanding or making some small component pieces of the guitar, such as the strings, the pickups, or the frets. This, according to Kazmerik, frustrates the students because they decided to take a computer degree because of their desire to the guitar – that is, many took computer science because of their desire to create a finished, complex and inspiring software system such as a real game or a real web application. A multi-course web stream can provide that much-desired integration for the upper-level students. It can provide an experience that White and Weinberg have called for: core integration at the end of the CS curriculum that helps “clarify the manner in which the core areas [of the computer science curriculum] are interdependent” [38].

6. ACKNOWLEDGMENTS

The author would like to thank Amber Settle for her very diligent reading and insightful feedback on earlier drafts of this paper.

7. REFERENCES


[34] Treu, K. (2002). To teach the unteachable class: an experimental course in web-based application design. In *SIGCSE '02*.


