Web Engineering: Creating a Discipline among Disciplines

Yogesh Deshpande and Steve Hansen
University of Western Sydney, Australia

Web engineering is a discipline among disciplines, cutting across computer science, information systems, and software engineering, as well as benefitting from several non-IT specializations. Intertwining so many disciplines presents a unique problem for organization and development. Here we discuss Web engineering's classification, define its characteristics, and contrast its present issues with previous problems in information technology.

While the World Wide Web may be "just another application of distributed computing" to some computer scientists, it is now widely acknowledged as a medium to deploy and develop applications. At the same time, there's a certain déjà vu about the development of Web-based applications, reminiscent of the 1960s, before computing professionals acknowledged that computer applications involved much more than expertise in programming and general intelligence.

Today's Web-based application developers generally seem young, full of energy, conversant in new technologies, and as though they're enjoying themselves (analogous to the programming fraternity of the 1960s). In contrast, those in charge of corporate applications and systems generally seem old, tired, and weighed down by legacy systems (the same perceptions we had of people using unit record systems).

The similarities of this scenario are actually much more in-depth than simply discerning outward appearances. There are real dangers that the new Web technology and the pace at which it's changing may divide the information technology community, and the Web-based application developers may repeat the mistakes of their predecessors. The 1960s programmers generally disregarded the older systems and systems personnel to build applications anew—reflecting their youthful exuberance and technological superiority—with insufficient attention to the users' needs and without sound methods of building and testing the applications. These themes have dominated software engineering and information systems research and conferences for more than 30 years.

The information technology community is accustomed to continuous change and has responded to change in the past by taking stock and creating new fields of study. Here, then, we take a close look at Web-based application development and argue that there's more to such work than computer science, software engineering, and information systems encompass. Our community proposed the term "Web engineering" in 1998 with the first international workshop on Web engineering,[1] and this has since been used by many. However, the term Web engineering is still unfamiliar to many and not fully understood. We argue that the information technology community should view Web engineering as a new, emerging discipline in its own right, rather than subsuming it mainly under software engineering.[2]

Perceptions of the Web and Web-based applications

The Web has reached a level of public consciousness and a level of hype whereby almost everyone encountering the Web for the first time comes to it with some preconceived notions about what it might do. These perceptions directly affect the way Web developers may work within and outside organizations.

Compared to the general public, our first major experience of the Web was relatively more innocent and significantly instructive. Even after more than 20 years of information technology background for each of us, we found the Web an amazing experience. On the other hand, the centralized information technology services within the university were profoundly upset by it. With hindsight, we now understand that they saw it as a thoroughly unsettling experience—beyond their current competence, budget, and corporate concerns—whereas we tried to grapple with the implications of transcending physical boundaries in information dissemination.

Soon after that, we put up the faculty Web site as an experiment and realized the Web can unsettle people in other ways, too. We were half-prepared to face the somewhat expected criticisms about its looks and organization, but there were

1070-986X/01/$10.00 © 2001 IEEE

other issues that, taken together, made us aware of the possible extra dimensions (such as graphic design, information ownership, network performance) that impinge on Web-based (application) development. Additionally, we then encountered indifference and even hostility from some of our colleagues within academe and industry who regarded the Web as frivolous, full of pretty pictures, and of no consequence to serious information technology—including legacy systems.

As we continued our work—developing sites and simple database applications—it became obvious that the Web-based application development didn't fit into any neat category, such as computer science, software engineering, or information systems, even though it drew from them. It also became increasingly clear that the information technology community wasn't concerned with information dissemination (and hence with Web site design and management, information structuring, and document and link management) nor did it see much potential in the Web as a medium for application deployment and development.

At this stage, the major contribution from information technology came only from those who built network infrastructures, created protocols and tools for the Web, and carried out research in the computer security area. The software industry seemed dominated by its corporate bias and was slow to realize the enormous reach of the Web and its potential to change the nature of applications. The 1997 Australasian Web (AusWeb97) conference provided an early indication of this apathy. Organized by a noninformation technology department of an Australian university, it drew 200 participants, only 15 of whom were from recognized information technology departments, including three from our own group. Most of the delegates were Web (site) designers, part of the World Wide Web Consortium (W3C), or from the public relations, education, library, and hypertext communities. This pattern repeated at the following year's conference, which was merged with the Seventh World Wide Web conference (WWW7) in Brisbane in April 1998.

We became concerned at the divide between the Web community and the mainstream information technology professionals and researchers, because it reminded us of the early PC and mainframe communities in the 1980s. We also began to identify more dimensions to the Web-based developmental work that fell outside the main boundaries of information technology, so we started to discuss our concerns and perceptions with both information technology and noninformation technology professionals and lay people. As a result, we organized the first international workshop on Web engineering at WWW7, followed by a similar workshop at the International Conference on Software Engineering (ICSE99), and have continued with these two conferences (WWW8, WWW9, and ICSE2000), interacting with slightly different audiences.

What follows is the result of these deliberations and the work carried out so far. It seems to us that Web-based application development has now progressed sufficiently far—and is expanding sufficiently fast—to justify proposing it as a new field of study that draws from computer science, software engineering, and information systems but also as distinct from them. Moreover, the influences don't simply stop at these disciplines—they're derived from many others. This further underlines the need to recognize Web engineering as a discipline in its own right.

Characteristics and development

There are many important and distinguishing characteristics of the Web-based applications. They include a relatively standard interface across applications and platforms, applications which disseminate information, the underlying principles of graphic design, issues of security, legal, social, and ethical ramifications, attention to site, document and link management, influences of hypertext and hypermedia, network and Web performance, and evolving standards, protocols, and tools. These give rise to a potential world-wide user base, possibilities in end user computing, evolutionary systems development, and completely new types of applications.

Application development and user orientation

In developing applications and systems—assuming a stable hardware and operating system environment—developers divide specifications into three segments:

- Process logic
- Data management
- User interface

Partly as a consequence of practices dating back to the 1960s—and partly because of the time constraints under which such systems are developed—developers perfect the process logic and

data management first. They deal with the user interface later and generally in a hurry. It therefore isn't surprising that the users of information systems have complained for decades about the poor quality of the interface.

Users find the Web appealing because of its consistent user interface (dictated by the use of HTML). Additionally, the Web's effective platform independence also frees the user from many headaches inherent in learning different interfaces. These two characteristics of the Web environment contrast strongly with the need to customize the user interfaces for each non-Web-based application depending on the hardware and software packages in use. Thus, Web-based applications benefit from spending less time in creating interfaces. By freeing the application developers from designing interfaces every time, the Web lets developers concentrate on back-end processing, such as process logic and data management.

Categories

We can divide Web-based applications into two broad categories—informational applications (dissemination/presentation) and software applications (in the usual sense).

Information technology professionals didn't consider the first category as part of their domain and consequently didn't take much notice of the Web in the initial stages. They also considered Web technology development as inadequate and unstable to support back-end applications as well as hampered by the slow speed of networks. However, the advent of faster networks, new standards and improved database connectivity, better understanding of security issues, and commercial imperatives of e-commerce have now persuaded many information technology departments to take the Web environment more seriously.

Meanwhile, other noninformation technology departments and both information technology and noninformation technology enthusiasts went on to create Web-based applications—without any reference to the old information technology professionals—replicating the 1960s experience of not including or learning from the experience of the previous generation of application builders. Thus, possibly two types of Web-based application developers exist: those who come from the traditional applications and who need to update their technical knowledge, and those who need to learn from the lessons of the last four decades to develop good methods and practices.

However, our experience suggests that even experienced application developers will find that Web-based development requires newer methods and wider awareness than the current software engineering practices engender.

Wider user base and graphic design

Web-based applications, almost by definition, are meant for a more inclusive user base, which goes beyond the previous confines of departments, divisions, or organizations. One consequence is that application developers may not know who the users are. These applications could be within an organization (intranets), across a number of organizations (extranets), or over the Internet. The users may come from specifically identified or targeted groups or remain unidentified even when they use the systems. Systems analysis and design methodologies have until now advocated fairly explicit identification of the users. It's a new challenge to devise methods to deal effectively with unknown users.

User interfaces for applications must consider unknown users (current and future) and compete—indirectly—with the interfaces that competitors and even collaborators create. Now that Web surfers have seen alternatives, they won't be satisfied with the merely functional interfaces they've had to deal with before. So, creating aesthetically pleasing Web sites and pages has become part of the application development. Accordingly, graphic designers become a necessary part of the development team. Although the Web is supposedly platform independent, in reality the range of monitors and the proliferation of browser versions can create problems when developing a standard interface for an application. This adds a new dimension to testing applications as well.

Security considerations

The Web opens applications and servers to the world. Developers must be aware then of the possible security issues and preferably include—as necessary—experts from the Web security area.

Legal, social, and ethical issues

We've observed scores of Web site/page developers (with and without an information technology background) and we've been surprised by how much and how easily everyone tends to downplay the legal, social, and ethical issues. Simply put, the Web adds an element of publishing to whatever else the developers may do and this brings with it the responsibilities of the publishing industry. The Web makes it easy to copy

elements from other sites, to publish information about users without their permission, and even to steal identities. Therefore, privacy for clients must be guaranteed. Even an interface's design and construction can interfere with copyright and intellectual property rights.

Also, we must afford access to handicapped people. The New South Wales government in Australia, for example, requires that its sites be accessible to visually impaired people. Until now, application developers operated within the confines of organizations and haven't developed sufficient awareness of the consequences of their actions when they are exposed to the rest of the world. Even where criminal or mischievous element is absent, acts of omission and commission can have consequences both for developers and their organizations, as the Napster case shows. We know of several examples first hand where copyright and privacy violations have taken place and our colleagues and students weren't even aware of their transgressions. One such case in fact led to publicity in the local media and had to be cleared up before more misunderstandings arose. Fortunately, the editor responsible in the latter case exonerated the concerned person and apologized for any offense caused. The media and experienced Web designers highlight some of these issues, but we tend to dismiss them rather easily.

Site, document, and link management

A Web site quickly grows to thousands of computer files with hyperlinks criss-crossing throughout. Therefore, any site or application developer must not only cater to the application's functionality (process logic and data management) but also to the structure and constituting elements of its design at a more detailed level (such as documents and links). Links may be broken or missing for a variety of reasons, anything from simply changing the file's name—an oversight—to restructuring any portion of the site. Published information easily becomes outdated; even when it isn't, the visitors need reassurance that what was last updated six months before is still valid. Web developers might not undertake some of this routine maintenance, leaving it to the administrative—as opposed to technical—personnel. Thus, there's a need for new methods from computer science and software engineering disciplines for site management. We also need new policies for management and human resources for their effective implementation. For an early effort in defining new jobs, see Hansen et al.[3]

Information structuring, hypertext, multimedia, and hypermedia

Information structuring isn't a new topic for application developers because database design strategies pay explicit attention to information structuring. However, the introduction of hyperlinks, multimedia, and hypermedia now complicates the scene, going beyond the traditional numbers and text that developers have dealt with previously. Structuring this information for efficient and reliable management is an area where a lot of research occurs and new methods are being devised.

End user computing

The era of end user computing started with the mass arrival of PCs in the mid 1980s. Information technology professionals have justifiably criticized it for its deficiencies—its lack of formal methods, insufficient understanding of theory, and poor maintenance, for example. The Web greatly amplifies the end users' reach—beyond the desktop PC to the Internet—both in accessing and publishing information. The end users will harness this power to solve their problems, regardless of whether information technology professionals help. It's imperative that we devise methods and processes to assist end users, develop their applications, and take the message of systematic development, testing, and maintenance to them along with the responsibilities of deploying the applications on the Web.

Software engineering is meant to address problems of large, team-based projects. End user projects are unlikely to fall into this category but must be addressed now, to increase the reliability of applications and at the same time release the creative power of people in general.

New types of applications

Until now, application developers were confined to specific places and organizations and rarely combined resources from diverse origins. The Web can link widely dispersed sites, organizations, and resources. The Environmental Defense Fund, for example, links detailed US maps with public domain information on chemical dumping to create an information system accessible across the Web. Problems and issues of distributed, collaborative, and cooperative applications will arise in this environment and it's up to us to find solutions.

Evolutionary systems development

Many systems development methodologies are popular now and the waterfall method[2] of appli-

cation development dominates discussions. A specific problem identified in systems development is the effort spent on working to a specification. We arrive at the specifications after consulting with a set of users identified at a project's start. However, users' requirements continue to change even before the final implementation of a system.

Web-based applications frequently deal with completely unidentified users, and their expectations (requirements) and behavior patterns decide whether the application is successful. An additional complicating factor is that, as mentioned before, such applications are likely to compete against similar offers from other organizations. This introduces a new variable of popularity—and hence, durability—of the application. Understanding users' requirements in these circumstances becomes much more complicated than it was before. Again, new methods of user analysis are required for Web-based applications.

Network and Web performance

Until now, we've implemented network applications in environments with known network capabilities. We've regarded network application tuning as a technical issue for the experts. However, Web-based application developers can't afford such luxuries. Their decisions on where and what to include in Web pages (for example, text, images, other multimedia objects, and database connections) affect the time to download information and the subsequent use of the application. They also have to make assumptions about the kind of networks their anonymous users access. Thus, Web-based applications acquire another dimension of complexity.

Evolving standards, protocols, and tools

Web technology is changing fast, and developers are continually adding new standards, protocols, and tools. There's also the imperative—driven by competition—to adapt to and adopt changing technology almost as soon as it's available. There are applications for DOS 3.1 still in use in various organizations around the world. It's almost impossible to think of any Web-based applications based on HTML 2.0—or for that matter, HTML 3.2—and these developments are only a few years old. These changes in Web technology raise questions about maintenance of applications and training of personnel. For an individual application developer, keeping track of new developments and the competition introduces extra dimensions with an urgency previously unencountered.

Virtual organizations

Globalization, internetworking, and the Web have led some experts to speculate about creating virtual organizations where people could belong to an organization but live and work anywhere in the world. The ubiquity of networks would overcome geography and time differences. The kinds of applications enabling such organizations and their development are mainly in the realm of research for now.

Web engineering

The foregoing analysis and discussion identifies the new elements of Web-based applications that aren't covered by parts of computer science, software engineering, or information systems. The analysis also establishes the need for systematic approaches, and development strategies for Web-based application development. In 1998, we proposed calling Web-based application development "Web engineering."

Computer science, software engineering, information systems, and Web engineering

In a nutshell, computer science concentrates on hardware and (systems) software to get optimal and reliable performance. Software engineering addresses large-scale, team-based projects while information systems, originating from data processing, concerns itself with information systems within organizational units.

Engineering itself is a term that people understand fairly well even though individual disciplines within engineering debate from time to time what constitutes their own specializations. When discussing the term in the context of software engineering, Berry[4] noted that, "Engineering is about the systematic application of scientific knowledge in creating and building cost-effective solutions to practical problems."

With Berry's definition in mind, Web engineering can be defined as the application of a systematic, disciplined, quantifiable approach to development, operation, and maintenance of the Web-based applications or the application of engineering to Web-based software.[5]

Web engineering is thus an early identification of a rapidly growing field with a much more forward looking approach than just implying a collection of preexisting and proven development practices. It draws not only from computer science, software engineering, and information systems, but also from other disciplines. As such, it identifies the next stage in information technology evolution.

Conclusions

During the last 50 years, computing has changed the way the world works and the Web is changing it even faster. Even with the mass participation and influence implicit in the Web, the information technology community has a great responsibility to actively guide the development of Web-based applications. Information technology has seen many very significant changes in the past and has responded by creating new fields of study. Web-based application development now has reached such a stage. Since so many fields influence Web-based applications outside computer science, software engineering, and information systems, it's time to acknowledge this as Web engineering—a discipline in its own right. **MM**

References

1. *Proc. 1st Int' Workshop on Web Eng.* (WebE 98), 14 Apr. 1998, http://fistserv.macarthur.uws.edu.au/san/webe98/.
2. R.S. Pressman, *Software Engineering: A Practitioner's Perspective*, 5th ed., McGraw-Hill, New York, 2000.
3. S. Hansen and Y. Deshpande, "A Skills Hierarchy for Web Information System Development," *Proc. Australasian Web Conf.* (AusWeb 97), Southern Cross Univ. Press, Lismore, Australia, 1997, pp. 114-121.
4. D. Berry, *Academic Legitimacy of the Software Engineering Discipline,* SEI tech. report CMU/SEI-92-TR-34, Carnegie Mellon Univ., 1992.
5. S. Murugesan et al., "Web Engineering: A New Discipline for Web-Based System Development," *Online Proc. 1st Int'l Conf. Software Eng.,* Workshop on Web Eng., http://fistserv.macarthur.uws.edu.au/san/icse99-WebE/ICSE99-WebE-Proc/San.doc.

Yogesh Deshpande is a senior lecturer in computing at the School of Computing and Information Technology and a founding member of the Web-based Information Systems and Methodologies (WebISM) Research Group at the University of Western Sydney. He has BStat and MStat degrees from the Indian Statistical Institute and MSc and PhD degrees from the London School of Economics. His research interests include Web engineering and design, information systems, end user computing, and simulation modelling. Deshpande is a member of the Operational Research Society, the ACM, and the IEEE Computer Society. He's also a member of the WebNet 2001 Conference program committee.

Steve Hansen is an associate professor in the Department of Computing and Information Systems and the director of the PlatformWeb project team at the University of Western Sydney. Hansen has an MSc degree from Sydney University. He has won the University service award for the PlatformWeb project designed to help in Web-based teaching delivery and student administration. PlatformWeb received an honorable mention at the Australasian Web, AusWeb 99 conference. He actively works for and advocates the integration of Web engineering with institutional operations. His interests are in the areas of Web engineering, electronic communications, technology management, and in the diffusion and adoption of innovations.

Readers may contact Deshpande and Hansen at the WebISM Research Group, University of Western Sydney, Campbelltown Campus, Locked bag 1797, Penrith South DC, NSW 1797, Australia, email {y.deshpande, s.hansen}@uws.edu.au.