A Need-Oriented Assessment of Technological Trends in Web Engineering

Damith C. Rajapakse and Stan Jarzabek

Department of Computer Science, School of Computing
National University of Singapore
{damithch,stan}@comp.nus.edu.sg

Abstract. As Web technologies change and multiply fast, their comprehension, assessment, selection and adoption are likely to be increasingly difficult, accidental and sub-optimal. Most often, needs are both important elements in technology assessment/selection and drivers of technology proliferation and evolution. We believe a need-oriented organization of Web technologies, as presented in this paper, is a useful starting point for comprehending the multitude of existing and emerging Web technologies from an essential and stable perspective. We identify important technological needs in relation to a reference architecture for Web Applications, and show how different technological trends address each need. We hope the paper will be of interest to those who want to get a grasp of the Web technology landscape and understand major trends.

1 Introduction

Web technologies change and multiply fast. For the practitioner and the researcher alike, a single summary of the state of the art in Web technologies could be invaluable in quickly grasping the current state of the art. To be useful, such a summary needs to be concrete enough to give sufficient details about the technologies, yet abstract enough to withstand rapid changes to concrete details. In this paper we attempt to present such a summary, organized around "Technology needs". Technology needs are both important elements in technology assessment/selection and drivers of technology proliferation and evolution. Hence, we believe that such an organization provides a perspective that is more user-oriented, fundamental and stable than the technologies themselves. Starting with a reference architecture for WAs, we identify important technology needs of the tiers and workflows of a typical WA, and then organize the technologies into different trends that has emerged to serve these needs.

Ours is not the first attempt to ease the difficulty of comprehending Web Engineering Resources (WER). Christodoulou et al [4][5], for example, proposed a reference model [4] for organizing knowledge about WERs, with a framework [5] for comparative evaluation of WERs. While the goal of Christodoulou's work and ours is the same, the methods are different, and the results – complementary to each other. Christodoulou's framework is more abstract; it does not concentrate on needs or specific technologies. Our framework is specific about concrete details of technologies, and their relation to needs and trends. It does not require the reader to discover and assemble concrete details on their own, as is the case with [5]. Therefore, we believe our paper could be of immediate benefit to those seeking a quick overview of the Web technology landscape. We consider integrating concepts from Christodoulou's framework into ours, as an interesting future project.

D. Lowe and M. Gaedke (Eds.): ICWE 2005, LNCS 3579, pp. 30–35, 2005.

2 Needs, Trends and Technologies

As shown in Fig. 1, WAs typically follow a multi-tier client-server architecture. The client-side of a WA consists of users accessing the WA using a *User agent* (E.g., Web browser) running on a *User device* (e.g., PC). In this paper, we focus on the most common User agent configuration: Web browser running on a PC. The server-side of the WA may be organized into multiple tiers and run on a Web server, possibly augmented by Application servers, Transaction monitors or Message servers. In this section, we discuss most important *WA-specific* needs of the tiers and workflows of a WA, and trends in Web technologies that address those needs. For each trend, we briefly mention the *implementation related* technologies (languages, standards, protocols, tools and techniques) that typify each trend.

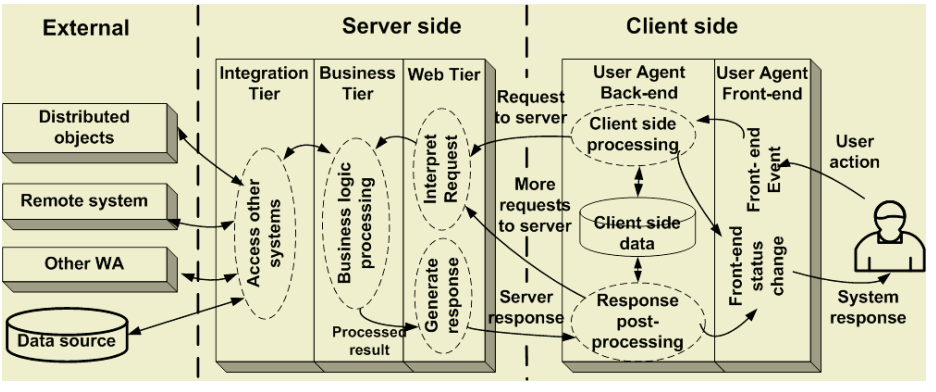

Fig. 1. Web Application Reference Architecture

The Need for Better Front-End Languages. Client-side of a WAs is primarily driven by HTML, a non-proprietary language standardized by World Wide Web Consortium (W3C) [18]. However, HTML syntax lacks the strictness of a programming language. The resulting difficulties in validating and processing HTML documents have led to a trend towards XML syntax. Extensible HTML (XHTML), the successor of HTML, is a family of document types and modules that reproduce, subset, and extend HTML, reformulated in XML [18]. Reduced authoring costs, an improved match to database and workflow applications, and clean integration with other XML applications are some of the cited benefits of XHTML [18]. Furthermore, HTML's lack of support for specialized contents has led to a number of specialized markup languages (e.g., MathML [18] – for mathematical content).

The Need to Separate Content, Structure, and Presentation. A typical HTML document is a mixture of content, structure, and presentational information. Keeping these three aspects as separate as possible is beneficial for development, maintenance (as different experts could develop/maintain each separately), and reuse (as each could be reused separately). *Styles* [18] were added to HTML as a way to separate out presentational information. Styles describe how documents are presented on a User agent. Cascading Style Sheets (CSS) [18] is one such style mechanism that is gaining wide use. Another related technology is XSL (Extensible Style Sheets) [18], a family

of recommendations for defining XML document transformation and presentation. Included in XSL is XSL Transformations (XSLT). An XSL style sheet can change the presentational as well as structural information of a document. It can be used on any XML document. XSL and CSS can be used together in a complementary manner.

The Need for a Better UI. Pure HTML UIs are static, and limited in functionality. The need to make WA UIs as sophisticated as traditional GUI applications has resulted in several trends. The first trend is to embed client-side scripts in HTML pages. JavaScript and VBScript are two languages commonly used for client-side scripting. Jscript [1] (succeeded by Jscript.NET [1]) is the Microsoft variant of JavaScript. ECMAScript [7] is a public domain specification that attempts to standardize client-side scripting. The second trend is embedding lightweight applications/components in HTML pages. Java applets and ActiveX controls are two technologies used for this purpose. A Java applet is a Java program that can be downloaded and executed by a browser. ActiveX controls can be run by a COM (Component Object Model) [1] aware browser and can be written in a variety of languages. The third trend is the use of plug-ins to enable using different objects inside the browser (e.g., Adobe Acrobat plug-in allows viewing PDF documents from within browsers).

The Need for Client-Side Processing. Although WAs follow "thin client" paradigm (minimal functionality client, more processing on server), performing some processing on the client-side (e.g., input validation on forms) can significantly reduce network traffic and improve response time. The trends for client-side processing are similar to that of the previous section, i.e., embedded client-side scripts (JavaScript, VBScript, etc.), embedded small applications (Applets, ActiveX), and plug-ins.

The Need to Use Mainstream Languages for Business Logic Processing. The bulk of the business logic processing of a typical WA happens on the server-side. Common Gateway Interface (CGI) is one standard for using mainstream programming languages to implement business logic. CGI defines how data is passed from a server to a CGI-compliant program. Two popular CGI programming languages are Perl and Python. Java is another popular language used for developing WAs. For example, Java Servlets [11] are modules of Java code that run in a server application and respond to client requests by interpreting the request, doing business logic processing, and generating dynamic content. Component technologies such as Enterprise Java Beans (EJB [11]) can further simplify server-side programming. They facilitate reuse of common services, allowing a developer to focus on the business logic of a WA, rather than on the "plumbing" code.

The Need to Separate Response from Response Generation Code. Generating the response involves generating text of one language (e.g., HTML) using another language (e.g., Perl or Java). The simplest solution is to write the server response directly to the output stream (e.g. using print() function). Java Servlets follow this method. However, this approach requires encoding each piece of the server response as a string literal, obviously a cumbersome task. Embedding scripts to represent dynamic content in otherwise static text files, commonly called "Server pages", tries to separate server response from the code generating that response (scripts). The web server processes the server page and sends the generated text output to the client-side. In Server-side Includes (SSI) technique – a limited form of server pages – scripting

commands embedded within a web page are parsed by the web server to generate dynamic content. SSI functionality is limited to adding small pieces of dynamic information (e.g, common footer). PHP (Hypertext Preprocessor), ASP (Active Server Pages – succeeded by ASP.NET), and JSP (Java Server Pages) are Server page technologies that are more capable than SSI. Several extensions with similar capabilities exist for Perl (e.g. Mason [9]) and Python (e.g., Spyce [15]). A further improvement is to separate the server response and scripts into separate files. Java Beans (in conjunction with JSP) and ASP.NET's *Code-behind* feature are some technologies that push in this direction. A successful separation of server response from code gives us *Templates* – representative documents one can create and edit using ordinary Web authoring tools while preserving the hooks to scripts. Freemarker [8] and Velocity [17] for Java, HTML::Template and Text::Template for Perl, Smarty [14] for PHP, DTML for Python, are examples of templating mechanisms. Macromedia's CFML (Cold Fusion Markup Language) [3] is another proprietary templating language.

The Need for Rapid UI Building. Unlike a traditional application where UI and the event handling code form one cohesive unit, UI of a WA needs to run on a diverse set of thin clients while communicating with the server-based event handling logic via the stateless HTTP protocol. Server-side UI component technologies are an effort to hide this complexities from the developer. They include a set of APIs for representing UI components against which it is easy to write code for managing their state, handling events, input validation etc. ASP.NET Web Forms [1] and JSF (Java Server Faces) [12] are two such server-side UI component technologies.

The Need for Integration. There are three types of integration that we can think of: intra-WA integration, inter-WA integration, and integration between WA and other external systems. The trend in intra-WA integration (integration of the remotely located parts of a WA) is to use general purpose distributed application technologies (e.g., CORBA [6], DCOM [1], .NET remoting technology [1], and Java RMI [11]) In inter-WA integration we can also use WA-specific technologies. For example, JSR-168 [12] Portlet specification defines a common API for Portlets in Web Portals. Even more sophisticated integration could be achieved using *Web services* [18] – programmatic interfaces made available by a WA for communication with other WAs. Web services could be combined to create WAs, regardless of where they reside or how they were implemented. When WAs need to integrate with external non-WAs (e.g. Mail servers) the integration method depends on the mutual availability of an integration technology and a communication protocol.

The Need for End-to-End Solutions. The need for end-to-end technology solutions is based on two desires: the desire to start with a set of compatible technologies, to avoid interoperability issues, and the desire to have much of the common infrastructure ready-made and well integrated, to minimize the development effort. Platforms (underlying technological environments or architectures) and frameworks (collections of software containing specialized APIs, services, and tools) serve this need. The J2EE (Java 2 Platform, Enterprise Edition) [11] defines the standard for developing multi-tier enterprise applications (not limited to WAs) using Java. It provides containers for client applications, web components based on Servlets and JSP technologies, and EJB components. The J2EE Connector Architecture defines a standard architec-

ture for connecting the J2EE platform to heterogeneous Enterprise Information Systems (EIS). From the Microsoft camp, the .NET [1] umbrella includes a similar set of WA building technologies. It is integrated with Windows platform and has a heavy emphasis on web services. A major part of .NET is the .NET framework, which consists of the Common Language Runtime (CLR) and the .NET Framework class library. CLR provides common services for .NET Framework applications written in a variety of languages, including C, C++, C#, and Visual Basic. The NET Framework class library includes ASP.NET, ADO.NET, and support for Web services. Microsoft Host Integration Server and Microsoft BizTalk Server aid in integration of .NET WAs and other EIS. In addition, numerous other less sophisticated frameworks exist (e.g., Seagull [13] for PHP, Mason [9] for Perl, Albatross [2] for Python, Jakarta Struts [10] and Turbine [16] for Java).

3 Concluding Remarks

We hope our need-oriented perspective helps one to grasp essential trends in Web technology landscape, independently of the many specific technological solutions that have emerged in response to various needs. Space limitations prevented us from a detailed discussion of a number of needs (e.g., the need for device independence, the need to make WAs secure, the need to "internationalize", need for "accessibility", the need for server-side/client-side data persistence). In the future work, we plan to extend our Web technology assessment framework with concepts introduced by others [4]. We also plan to continuously refine our need/trend/technology taxonomy. For the ease of reference, given next is a tabulated summary of the needs, trends, and technologies discussed in this paper.

Need	Trends → Technologies
Better front-end languages	Incorporate XML → XHTML
	Markup for specialized contents → MathML, SVG, etc.
Separate content, structure, presentation	Styles → CSS, XSL
	Transformations → XSLT (part of XSL)
Better UI, Client-side processing	Embed client-side scripts → JavaScript, Jscript, VBScript
	Embed light weight applications → Java Applets, ActiveX
	User agent plug-ins → e.g., Adobe plug-in for pdf
Use mainstream languages	Standards (e.g., CGI with Perl, Python, etc.)
	Components → E.g, Java Servlets, EJB, COM+
Separate response from response generation code	Write to output stream → Java Servlets
	Server pages → SSI, ASP/ASP.NET, JSP, PHP, Mason, Spyce
	Server pages (with hooks) → JSP+Java Beans, ASP.NET Code behind
	Templates → Freemarker, Velocity, Smarty, HTML::Template, DTML, CFML
Rapid UI building	Server-side UI components → ASP.NET Web forms, JSF
Integration	Regular → CORBA, RMI, DCOM, .NET Remoting
	Web specific → Portlets, Web services
End-to-end solutions	Platforms/ Frameworks → J2EE, .NET Struts, Turbine, Seagull, Mason, Albatross

References

1. ASP, ASP.NET, COM, JScript, VBScript and .NET at, http://www.microsoft.com/
2. Albatross home page, http://www.object-craft.com.au/projects/albatross/
3. CFML home page, http://www.macromedia.com/devnet/mx/coldfusion/cfml.html
4. Christodoulou, S. P., and Papatheodorou, T. S., "WEP: A Reference Model and the Portal of Web Engineering Resources", *Proc. Intl Workshop on Web Engineering*, 2004
5. Christodoulou, S. P., Tzimou D. G., and Papatheodorou, T. S., "An Evaluation Support Framework for Internet Technologies and Tools", *Proc. IASTED conference on Communications, Internet and Information Technology*, 2003
6. CORBA home page, http://www.corba.org/
7. ECMA home page, http://www.ecma-international.org
8. Freemarker home page, http://freemarker.sourceforge.net/
9. HTML::Mason home page, http://www.masonhq.com
10. Jakarta Struts project home page, http://struts.apache.org/
11. Java Servlets, JSP, J2EE, RMI home pages at http://java.sun.com/
12. JSR documentation at http://www.jcp.org
13. Seagull home page, http://seagull.phpkitchen.com/
14. Smarty home page, http://smarty.php.net
15. Spyce home page, http://spyce.sourceforge.net/
16. Turbine project home page, http://jakarta.apache.org/turbine/
17. Velocity home page, http://jakarta.apache.org/velocity/
18. World Wide Web Consortium Web site (containing home pages for CSS, HTML, XHTML, MathML, Styles, WebServices, XForms, XSL), http://www.w3.org