Evolving the Browser Towards a Standard User Interface Architecture

Michael J. Rees

School of Information Technology, Bond University
Gold Coast, Qld 4229, Australia

mrees@bond.edu.au

Abstract

If current trends continue, it is likely that the web browser will become the only widely used user interface. Web applications will become the predominant software. Should this happen, user interface design, implementation and evaluation skills can become more focussed and effective. Some of the benefits current browser user interfaces provide are discussed in the context of web application tools produced by the author and supported by examples. The software architecture of the Web brings special HCI demands, and the user design experts of the future will require training in this architecture. This evolution is scrutinised in terms of the new web services that will become available. Recent trends in this direction are presented, and future trends explored, with supporting evidence taken from a range of applications. The influence of the Web, with now a long history of user experience, can bring benefits to user interface design in the future. [1]

Keywords: browser user interface, user interface standards, XML, user interface markup language

1 Introduction

It is over a decade since the first web browser was written by Tim Berners-Lee. As described in [1] and [2] this browser was written in Objective-C running on NeXTStep, and was a complete browser/editor using the powerful built-in text editing classes. Thus all users could publish as well as read web pages. The need to port the browser to other platforms without built-on text editing forced the browser to become display-only.

Over the intervening 10 years there have been various add-on technologies introduced to make the browser user interface fully interactive once more. Java Applets, ActiveX controls and a plethora of add-ins now provide interactive facilities at all levels. However, the web page author can never rely on the necessary facilities being present in all browsers. A standard set of features is needed in all browsers—a universal browser user interface for interactive web applications. The remainder of this paper reviews whether the browsers of today match up to this requirement, and if and how they might evolve towards such a standard.

2 Prior Work

That the web browser user interface has become a prime candidate to be nominated as a universal user interface has already occurred to the Netscape software engineers. Their Gecko architecture [3] is designed for user interface expression and implementation. Not only has Netscape used Gecko in version 6 of Netscape Navigator for displaying the web page content, they have adopted it for the whole user interface, menus, toolbars and dialogue box contents, and so on. This has been repeated with all the other applications in the Netscape Communicator suite.

There can be general agreement that the architectures, notations and models that modern browsers must support make the needs of their user interface design quite generic. Content notations such as XHTML [4] and XML [6] are becoming widely-adopted standards. The HTTP protocol is becoming endemic, primarily because it passes through most firewalls easily. Upon this protocol are based SOAP (Simple Object Access Protocol) and UDDI (Unified Distributed Data Interchange). The World Wide Web Consortium (W3C) dynamic HTML DOM (Document Object Model) [7] must be supported by all conforming web browsers, manipulated by standard ECMAScript [8] executable components (scripts). Dynamic hyperlinks form the glue between different user interface components and different views of the web application.

Set against this impressive list of advantages there is a downside. The built-in user interface controls in dynamic HTML are small in number, and certainly do not match the sophistication of the proprietary user interface libraries in Windows, Mac OSX and Unix/Linux platforms. User interfaces built with the DOM and ECMAScript are not easily transportable between pages (applications), and the user interface behaviours are very difficult to replicate across the various platforms.

Another effort at standardising the user interface specification and rendering in a browser environment is User Interface Markup Language (UIML) [4] from Harmonica Inc. UIML is a useful start to definition of device-independent user interfaces, and employs a three-step approach to user interface definition. The three stages in UIML are:

1. Device independent user interface elements
2. Whole-of-interface device-dependent style definitions for user interface classes

3. A content database for specific user interface elements which allow for internationalisation and specific rendering environments

XML is used at all levels to describe the elements of each stage. The specifications for UIML are at version 2.0 and a commercial tool, LiquidUI™, supports UIML for a number of rendering environments. Definitions of UIML vocabularies for HTML, Java, VoiceXML, and WML are now available. Unfortunately, many of the attribute values for user interface elements are taken from the Java user interface classes AWT and Swing. Nevertheless, the mechanisms for XML expression and the device-independence of the first stage show the way for universal user interface definition in general.

3 Web browser number one

The very first web browser was developed in Objective-C running on NeXTStep by Tim Berners-Lee in 1990, [1] Using the powerful built-in text editing classes, he effectively built a word processor for web pages. This application could create pages, browse them, and allow any user viewing the page to edit a new version. Thus from the start Web pages were fully collaborative. These capabilities are described further at length by Gillies and Cailliau in [2]

As the browser was ported to the prevalent operating systems platforms of the day, there was no easy-to-use equivalent of the NeXTStep editing classes, and so, to our detriment, the vast majority of browsers became display-only. The collaborative editing facilities were lost.

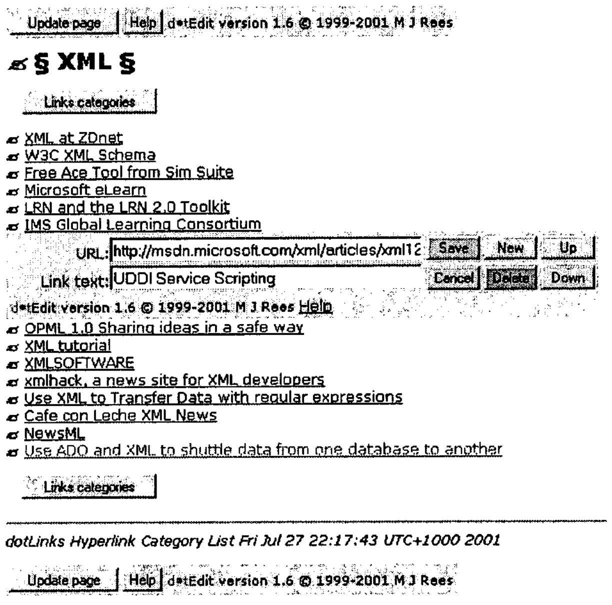
4 Browser user interfaces of today

For over half a decade, many efforts have expended to bring back the collaboration and editing facilities into web pages. The working group in this area [9] at the W3C have coordinated many research, commercial and shareware browser add-ns and other software mechanisms. All unfortunately use differing approaches to user interface within a web page.

One of the earliest papers to foresee the emergence of a powerful interactive user interface within a web page is that by Rice et al [10] . They addressed deep issues such support for naïve and power users, multiple browsers and hyperlink click minimisation. The implementation, however, like all in that time, was restricted by the CGI scripting mechanism and the need for server round-trips on nearly every user interaction with their user interface within the web page.

Around this time, the work of the author was instigated by the insightful paper by Chang [11] at the WWW7 conference in 1998, where he described the Sparrow project from Xerox PARC. The Pardalote project [12] has produced a lightweight editing tool that is easy to install, responsive and effective in allowing a team of collaborators to evolve the content of web pages in a controlled manner. Pardalote is a significant improvement on the PARC work. Instead of traditional CGI scripts, Pardalote uses the DHTML DOM to achieve immediate in-memory editing. At a point chosen by the user, a final web page update triggers a server round-trip.

The editor tool from the Pardalote project has been renamed dotEdit [13] and is available for distribution [14] . dotEdit allows the users sharing the web page to edit nominated sections directly in the browser. Moreover, dotEdit has been designed to allow the original page author to use favourite, standard web publishing tools, and does not need special HTML tags to be inserted like other competitive software.



d•tEdit version 1.6 © 1999-2001 M J Rees

§ XML §

- XML at ZDnet
- W3C XML Schema
- Free Ace Tool from Sim Suite
- Microsoft eLearn
- LRN and the LRN 2.0 Toolkit
- IMS Global Learning Consortium

URL: http://msdn.microsoft.com/xml/articles/xml12
Link text: UDDI Service Scripting

d•tEdit version 1.6 © 1999-2001 M J Rees Help

- OPML 1.0 Sharing ideas in a safe way
- XML tutorial
- XMLSOFTWARE
- xmlhack, a news site for XML developers
- Use XML to Transfer Data with regular expressions
- Cafe con Leche XML News
- NewsML
- Use ADO and XML to shuttle data from one database to another

dotLinks Hyperlink Category List Fri Jul 27 22:17:43 UTC+1000 2001

d•tEdit version 1.6 © 1999-2001 M J Rees

Figure 1. dotLinks Hyperlinks Page with dotEdit.

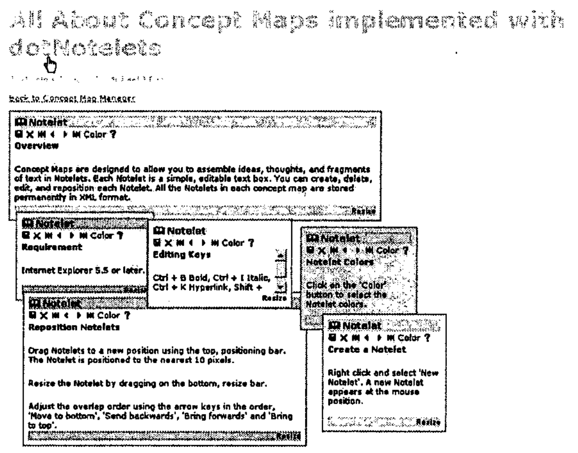
The versatility in user interface design comes to the fore when each web page can be treated as a separate user interface element. A recent tool built by the author called dotLinks [14] exploits this mechanism by generating web pages containing the I-grain editable components provided by dotEdit, the hyperlink I-grain in this case.

dotLinks provides a simple hyperlink repository accessible and updateable anywhere on the Web. A simple list element in the dotLinks home page gives access to categories of hyperlinks. Each category consists of a dotEdit page containing the list of hyperlink I-grains. All hyperlink I-grains are editable and movable within the lists. Figure 1 shows one of the dotLinks generated pages with the user editing one of the dotEdit hyperlink I-grains. Once the page has been created, users in the team sharing the hyperlink repository can continue to add, modify, delete and move the hyperlinks.

While a useful tool, dotEdit provides only raw text editing of the content of the I-grains. Styles may be applied to I-grains to affect font family, font size, colour, and so on, to make the I-grain presentation acceptable. Many more features are needed if the browser is to support user interfaces in general.

5 Direct manipulation in the browser

While the DHTML DOM makes real-time changes to the web page possible, one of the most urgent requirements is to support drag-and-drop direct manipulation. The set of event types provided by the DOM is rich enough to support this requirement.


All About Concept Maps implemented with dotNotelets

Back to Concept Map Manager

Notelet
Color ?
Overview

Concept Maps are designed to allow you to assemble ideas, thoughts, and fragments of text in Notelets. Each Notelet is a simple, editable text box. You can create, delete, edit, and reposition each Notelet. All the Notelets in each concept map are stored permanently in XML format.

Resize

Notelet
Color ?
Requirement

Internet Explorer 5.5 or later.

Notelet
Color ?
Editing Keys

Ctrl + B Bold, Ctrl + I Italic, Ctrl + K Hyperlink, Shift +

Resize

Notelet
Color ?
Notelet Colors

Click on the 'Color' button to select the Notelet colors.

Notelet
Color ?
Reposition Notelets

Drag Notelets to a new position using the top, positioning bar. The Notelet is positioned to the nearest 10 pixels.

Resize the Notelet by dragging on the bottom, resize bar.

Adjust the overlap order using the arrow keys in the order, 'Move to bottom', 'Send backwards', 'Bring forwards' and 'Bring to top'.

Resize

Notelet
Color ?
Create a Notelet

Right click and select 'New Notelet'. A new Notelet appears at the mouse position.

Resize

Figure 2. Concept Map in the dotNotelets Tool.

The versatility of browser drag-and-drop was investigated in a simple concept map tool called dotNotelets [14] . Rectangular notelets similar in appearance to Post-IT Notes™ can be created, resized and dragged around the browser window. The contents of each notelet can be edited directly, and the relative z-order of the notelets changed. Each notelet represents an idea, remark, comment or a small fact in text form. A collection of sample notelets is shown in Figure 2. Using the top Notelet bar in each notelet, the user can drag the notelet to a new position. Dragging the bottom Resize bar of a notelet allows it to be resized within sensible limits. Clicking the save icon in the toolbar just beneath the Notelet bar saves any changes made to the notelet size, position and contents.

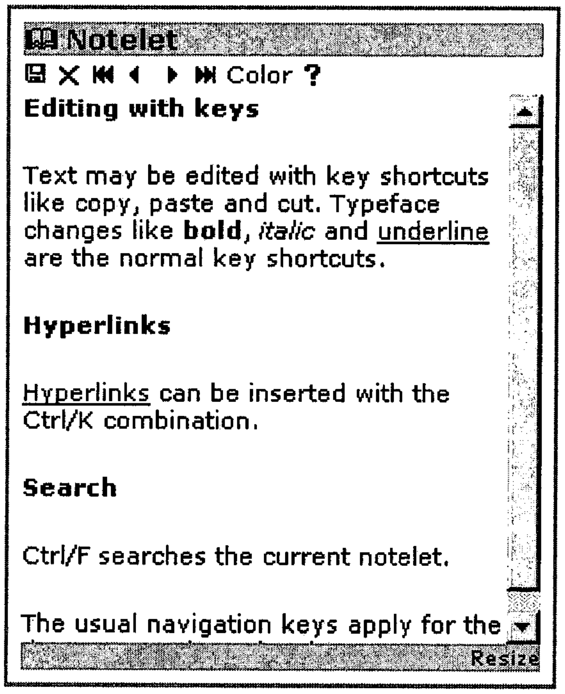
The top left position of each notelet is significant. When the text is serialised for export to other tools the notelet text is output in vertical then horizontal order. All these features fit naturally into the event handling of the DHTML DOM and can be realised using simple scripting. The browser interface takes one further step forward to becoming a generally useful user interface standard.

6 Rich Editing

Next the ability to offer more than simple text editing must be addressed. Largely unheralded, there has been a significant DHTML editing capability incorporated into Internet Explorer 4 and beyond. Initially, access to this rich editing was, and still can be, via an ActiveX control. In Internet Explorer 5, every displayable tag in the web page can be edited simply by setting the contentEditable attribute value to 'true'.

Figure 3 shows a notelet is more detail. Note that the text within the notelet contains simple typeface changes. These are all accomplished without the aid of a toolbar using simple and Windows standard short keys. There is even a mechanism to create hyperlinks using the Ctrl/K keyboard shortcut.

The dotNotelets tool stores each notelet in an XML file with DHTML embedded within it. Every notelet carries its own GUID so that it can be uniquely identified. Further development of dotNotelets will extend the notelet XML repository with features for searching, importing and exporting notelet XML files. Extensions into mind maps and topic maps are easily possible.


Notelet
Color ?
Editing with keys

Text may be edited with key shortcuts like copy, paste and cut. Typeface changes like **bold**, *italic* and underline are the normal key shortcuts.

Hyperlinks

Hyperlinks can be inserted with the Ctrl/K combination.

Search

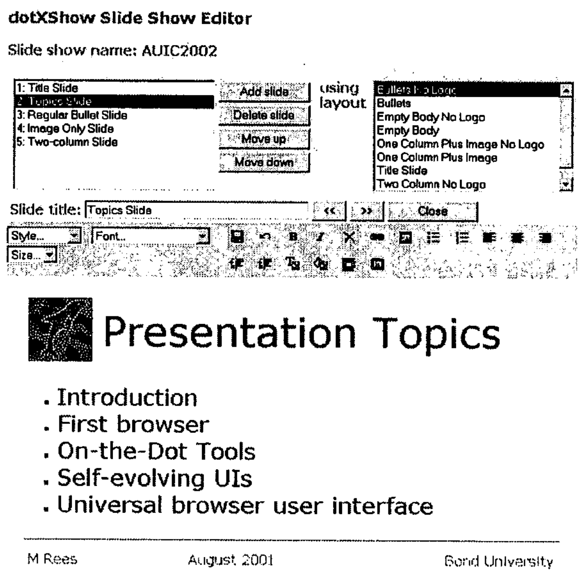
Ctrl/F searches the current notelet.

The usual navigation keys apply for the

Resize

Figure 3. A notelet uses the DHTML Editor.

To make full use of the DHMTL editor in Internet Explorer additional toolbars must be added to the user interface in the page. Exploration of these sophisticated editing capabilities was undertaken within the dotXShow tool shown in Figure 4.


dotXShow Slide Show Editor

Slide show name: AUIC2002

1: Title Slide
2: Topics Slide
3: Regular Bullet Slide
4: Image Only Slide
5: Two-column Slide

Bullets No Logo
Bullets
Empty Body No Logo
Empty Body
One Column Plus Image No Logo
One Column Plus Image
Title Slide
Two Column No Logo

Slide title: Topics Slide

Presentation Topics

. Introduction
. First browser
. On-the-Dot Tools
. Self-evolving UIs
. Universal browser user interface

M Rees August 2001 Bond University

Figure 4. DHTML Editor used in dotXShow Tool.

The dotXShow tool is a browser-based slide presentation manager. It generates slide presentations in XML files containing a series of slides. Apart from a small amount of fixed information like a slide title, each slide's contents are completely arbitrary. In preparation mode the slide contents are represented in a single <div> tag in the web page. This <div> has the contentEditable attribute set to allow full DHTML editing. Special Cascading Style Sheets with larger font sizes are used to make the slide contents more suitable for projection.

As can be seen in Figure 4 the slide author is presented with a toolbar not unlike that seen in Microsoft Word or FrontPage. There is therefore no limit to the content of each slide in terms of text layout, fonts, foreground and background colours, hyperlinks, and so on. Even tables are supported although not shown in the example. A series of pre-built slide layout templates are available as shown in the list in the top right of Figure 4.

DHTML generated by the editor component within the web page is embedded in the XML file representing the whole slide show. A simple slide show presentation page allows the presenter to show individual slides in sequence or to jump to any slide in the show. Listings of all slides in a show are easily implemented by applying an XLST style sheet to generate all slides on one HTML page in a suitable layout. The browser print facility then provides a printed listing. dotXShow is a good example of the types of simple-to-implement web applications using XML and the standard browser user interface.

The dot* tools are just part of a large international effort in providing live editing in web pages. A good summary of US efforts is presented by Jon Udell in [15] . In a follow-up article he also mentions this author's tools and several others. Many of these tools make use of the DHTML editing component in Internet Explorer.

7 Browser User Interface Usability

Much research into user interface usability has been performed over the life of the HCI discipline. Jakob Nielsen in [15] specialises in applying standard graphical user interface usability guidelines to the design of web pages. This approach is particularly sensible where the page is acting as an interactive graphical user interface in its own right. A similar approach is adopted by the author and his co-authors in [18] where the well-understood principles of HCI are introduced, then applied to user interfaces including web pages.

Nevertheless there are some significant points of difference to be aware of when designing web pages as user interfaces:

- Knowing the exact URL of a page allows a user to jump straight to it, circumventing introductory material that the user may need to see; this means the user interface of each page must stand alone.
- There is a huge diversity in display size; make the page user interface easily resizable, where possible, or use a strategy of information reduction.
- Older browsers lack interface features like scripting, Iframes and so on; build rendering forgiveness into the user interface design.

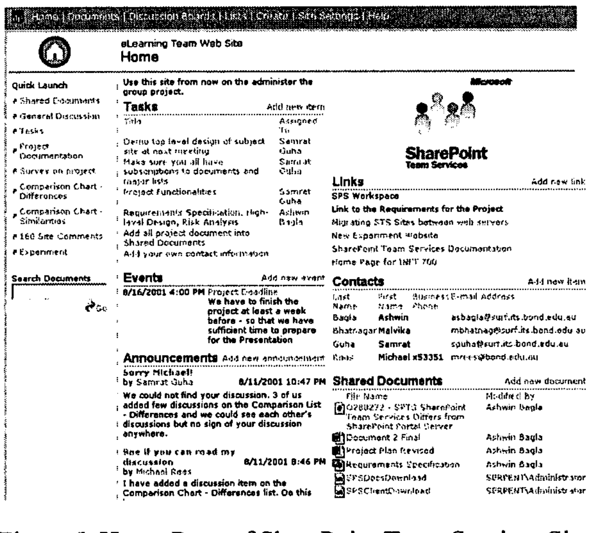
Examples of web page user interfaces abound; the Web as a whole acts as a global, ever-available usability laboratory. This is an aspect of the Web which is extremely valuable and under-rated.

8 Users controlling content and layout

Mention has already been made of the ease with which browser user interfaces can be self-generating by the simple expedient of creating or modifying the HTML within web pages. Several tools now exist that allow web page readers, to generate related page collections and to specify content for these pages in a more or less constrained manner. This allows a group of users to evolve a set of web pages to meet their needs without resorting to traditional web publishing packages that require substantial training.

A recent example of such a tool is the Microsoft SharePoint Team Services (SPTS) [19] feature that runs in the Internet Information Server environment. SPTS offers:

- An authenticated site with a self-defining user group
- Team-managed web pages without any knowledge of HTML
- A web site user interface that constantly changes to meet current user needs
- Information content in the form of standard and customisable lists (database tables) tailored at will
- Auto-notification of changes to document and list content
- Any number of shared document libraries with discussions at the document level
- Information-specific discussions created by the users



eLearning Team Web Site
Home

Use this site from now on the administer the group project.

Tasks

Title	Assigned To
Demo top level design of subject site at next meeting	Samrat Guha
Make sure you all have subscriptions to documents and major lists	Samrat Guha
Project Functionalities	Samrat Guha
Requirements Specification, High-level Design, Risk Analyses	Ashwin Bagla
Add all project document into Shared Documents	
Add your own contact information	

Events

8/16/2001 4:00 PM Project Deadline
We have to finish the project at least a week before - so that we have sufficient time to prepare for the Presentation

Announcements

Sorry Michael!
by Samrat Guha 8/11/2001 10:47 PM
We could not find your discussion. 3 of us added few discussions on the Comparison List - Differences and we could see each other's discussions but no sign of your discussion anywhere.

See if you can read my discussion 8/11/2001 8:46 PM
by Michael Rees
I have added a discussion item on the Comparison Chart - Differences list. Do this

Links

SPS Workspace
Link to the Requirements for the Project
Migrating STS Sites between web servers
New Experiment Website
SharePoint Team Services Documentation
Home Page for INFT 700

Contacts

Last Name	First Name	Business Phone	E-mail Address
Bagla	Ashwin		asbagla@surf.its.bond.edu.au
Bhatnagar	Malvika		mbhatnag@surf.its.bond.edu.au
Guha	Samrat		sguha@surf.its.bond.edu.au
Rees	Michael x53351		mrees@bond.edu.au

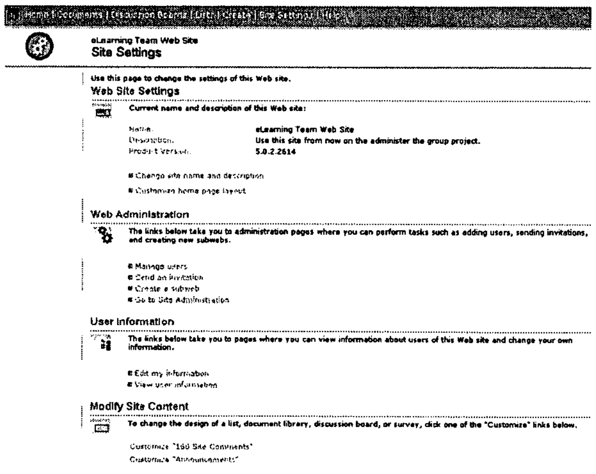
Shared Documents

File Name	Modified By
Q280272 - SPTS SharePoint Team Services Differs from SharePoint Portal Server	Ashwin Bagla
Document 2 Final	Ashwin Bagla
Project Plan Revised	Ashwin Bagla
Requirements Specification	Ashwin Bagla
SPSDocsDownload	SERPENT\Administrator
SPSClientDownload	SERPENT\Administrator

Figure 5. Home Page of SharePoint Team Services Site.

In essence, SPTS allows users of a collection of web pages to compose their own user interfaces that reference their information (documents), discussions, announcements, and list of all kinds. Figure 5 shows the basic tiled nature of the user interface. Apart from the basic navigation structure at the top of the page, most page content is composed of lists of various kinds. Most list contents contain hyperlinks to take the user to the detail of each list entry. Shared document libraries are list of documents that can be accessed and opened directly from the web page.

The SPTS software is remarkable in that the administration of the team site is incorporated into the same user interface. A set of five prebuilt roles is included with the ability to define new ones. Site users are managed with Windows 2000 local accounts for fully-authenticated access control. An SPTS site administrator requires no knowledge of Windows 2000 administration or of HTML. A typical administration page is shown in Figure 6.



eLearning Team Web Site
Site Settings

Use this page to change the settings of this Web site.

Web Site Settings

Current name and description of this Web site:

Name:	eLearning Team Web Site
Description:	Use this site from now on the administer the group project.
Product Version:	5.0.2.2614

- Change site name and description
- Customize home page layout

Web Administration

The links below take you to administration pages where you can perform tasks such as adding users, sending invitations, and creating new subwebs.

- Manage users
- Send an invitation
- Create a subweb
- Go to Site Administration

User Information

The links below take you to pages where you can view information about users of this Web site and change your own information.

- Edit my information
- View user information

Modify Site Content

To change the design of a list, document library, discussion board, or survey, click one of the "Customize" links below.

Customize "160 Site Comments"
Customize "Announcements"

Figure 6. SharePoint Team Services Site Settings Page.

From this page, a user assigned the Administrator role can customise settings for the whole team site such as site name and layout of the home page. Other users can be added, deleted and their roles changed. Subwebs of the main site web can be created, where each subweb becomes a self-contained team site with the same features as the top-level site. Finally, lists of information can be created, modified and deleted, including the ability to define complex customised lists. All list content is stored in a database automatically created for each SPTS site. One of the most useful built-in lists is the survey. Each team site member can complete the survey, and survey results pages can be accessible to all users or just the survey author. Survey results can be downloaded into Excel 2002 for analysis with pivot tables.

Users with Office XP installed on their machines gain additional facilities, but only a standard Internet Explorer or Netscape browser is needed to access over 95% of the SPTS features.

An even more sophisticated tool for building powerful browser user interfaces is the Digital Dashboard technology [20] from Microsoft. A digital dashboard takes user interface definition to a more formal architecture employing well-defined interface building blocks, each block being a web page in its own right.

These blocks are referred to as web parts, and are represented in a series of XML files. Each dashboard is composed of a number of web parts displayed in a customisable tiled grid. Again the user interface layout is massaged into a grid, not unlike many user interface layout manager libraries over several platforms. The original Java AWT gridBag layout manager is infamous, because of the vast number of parameters needed to give a flexible layout. User interface designers and implementers will be better served when the grid layout is given over for a more flexible architecture. One can gain insight into what this architecture might be when considering the <iframe> tag in HTML 4.0. with banners that occupy the full page width at the top and bottom of the page. Nevertheless, quite complicated user interface designs can be built. The web parts themselves are intended to contain sophisticated components—indeed any object that can be presented in a web page. These components act as middleware and allow access to a wide variety of network services and services on the local machine.

Example web parts provided by Microsoft include access to email inbox, address book, task list and instant messaging communications. A third-party web part production industry is already in existence. This mechanism starts to point the way to a universal user interface design and building capability. The sample dashboard in Figure 7 shows the grid-like nature of the layout. The inbox web part appears at the centre left side with a calendar web part beneath it. In the centre and lower right appear web parts containing the dotNotelets tool and dotLinks tool pages respectively.

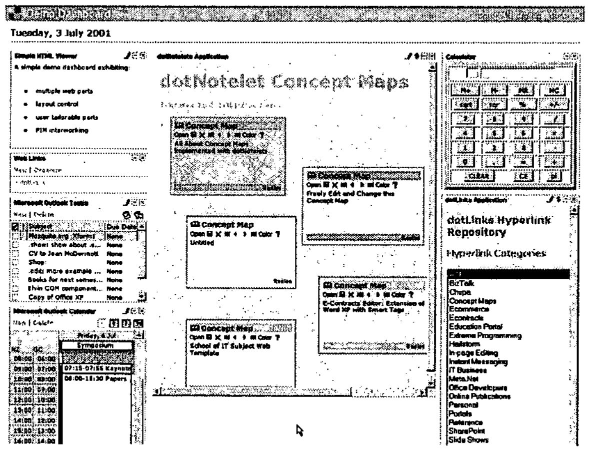


Figure 7. Sample Digital Dashboard.

One of the benefits of a formal architecture where user interfaces are built with web pages is that the refresh times of individual pages can be specified. The collection of dot* tools mentioned in this paper are collaborative, and an individual user must refresh the page at intervals to see the changes made by others in the collaborative group. Digital dashboard web parts have a refresh setting defined in one of the XML files. This refresh setting can be tailored to the expected activity on the shared web page. The refresh time for the dotNotelets tool in the centre of Figure 7 is set at about 5 minutes.

Although it slows performance somewhat, another benefit of digital dashboards is that the web parts are assembled into the layout dynamically each time. This allows components within the web parts to make changes to the layout information so that the look and feel of the dashboard can alter over time.

9 Web Services and Universality

At a more fundamental software architecture level, the recently-introduced web services model promises to bring the concept of a universal browser user interface a step closer. Microsoft's web services [21] address the issues of application integration that must of necessity include user interfaces. The goal is to produce applications running on different operating systems built with different object models using different programming languages and turning them into easy-to-build and easy-to-use web applications. The web services use open Web standards such as HTTP, XML, XMLDOM, SOAP and UDDI mentioned in Section 2 of this paper. Figure 8 shows the architecture with the Internet at its heart. Applications on the left using web services on right are connected to the users via the devices and/or browsers at the top of the figure. Notice that service contracts are a major part of this architecture.

Web services promise to revolutionise the methodology for building web applications. Both document-centric and remote procedure call-centric models are supported by the SOAP protocol. Thus web services can act like distributed applications or pass whole XML documents around in a data storage paradigm. Web pages become the natural glue in this architecture, and hence the browser user interface will take on greater importance.

One can easily imagine each user interface control being represented as a web service, with design-time and run-time service members. This will support both the user interface design and implementation phases. The user interface designs can be represented in a UIML-like XML notation, but most of the less elegant device-dependencies of UIML can be eliminated. The final rendering will be

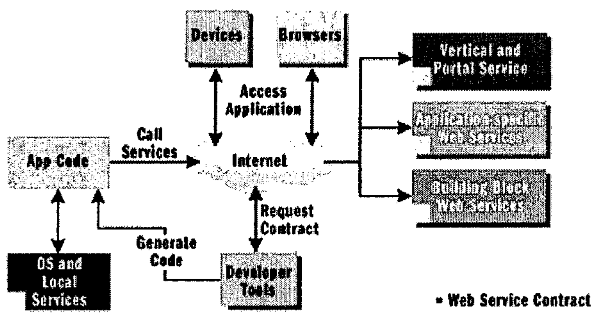


Figure 8. XML Web Services.

supported by the DHTML DOM and maybe other extensions like the XForms specification. This is the combination of technologies that the author perceives as leading to the universal browser user interface.

The foregoing discussion has demonstrated that tools already exist that allow the browser to support most of the business and organisational user interfaces. The challenge is to evolve the browser user interface using web services to cover a much wider range of user interface types such as:

- Scientific data display and analysis
- Multi-dimensional
- Multimedia
- Real-time control

A good test of whether a browser user interface can support the Greenberg Notification Collage [22] that uses a wide array of static and real-time components. The Notification Collage, shown in Figure 9, is a groupware system where team members post media elements onto a

Figure 9. Greenberg at al Notification Collage.

shared, real-time collaborative surface. This surface must contain many different, refreshable user interfaces, and will be a good test of the capabilities of the universal browser user interface.

Another good example of a web-related user interface design requirement is the trail marker mentioned by Hochheiser and Shneiderman in [23] . This idea lays the foundation for universal usability by incorporating markers into the user interface for all classes of users of that application (web page). Once again we see the influence of web page design make a major change to traditional user interface design.

Given that the delivery mechanisms for a universal user interface are in place, the designers and implementers need a model to guide them. At the Australasian User Interface Conference 2001 held at Bond University, the keynote speaker, Harold Thimbleby, was reminiscing with the author about the Apple HyperCard package [24] . Both in the conversation had used HyperCard extensively in the past. Speculation began on how to implement a HyperCard-like technology using today's web browsers. The author has tentatively named this WebCard and suggested that this technology:

- uses the HyperCard architecture from Apple
- treats web pages as a stack of index cards
- employs ECMAScript instead of HyperScript
- utilises the built-in event handling
- provides a user interface control tool palette that is extensible with third-party user interface controls
- Use design-time user interface controls supported by web services

The WebCard page model, shown in outline in Figure 10, would treat each page as a card, with each card having a background page, potentially shared between cards. Card pages and backgrounds would be grouped into page stacks, following the original ideas of the HyperCard authors. Within the DHTML DOM, cards and backgrounds could be implemented using overlapping <div> tags with the stack master user interface items being contained in an <iframe> tag.

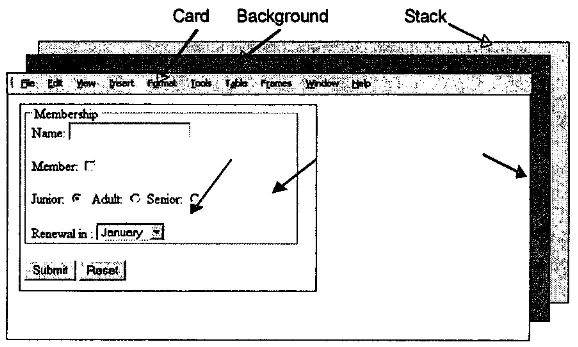


Figure 10. WebCard User Interface Architecture.

10 Summary

This paper attempted to demonstrate the evolution of the browser user interface, and extrapolate the trend towards a future user interface standard. There is much evidence for the rise of the browser, with the constraint of supporting international standards, to the pinnacle of the universal user interface. It is asserted this will be a thin-client, browser-independent application supporting the DHTML DOM, XML, XMLDOM, XSLT, SOAP and UDDI. The user interface controls will be served at both design time and run time using distributed XML web services. A fundamental user interface architecture will be needed, and the yet-to-be-implemented WebCard model offers simplicity and a proven track record in HyperCard. Such a combination offers a way out from the plethora of user interface architectures of today.

11 References

[1] Berners-Lee, T, *Weaving the Web*, Orion Business Books, 1999, ISBN: 0-75282-090-7.

~~[2] Gillies, J. & Cailliau, R., *How the Web was Born*, Oxford University Press, 2000, ISBN: 0-19-286207-3.~~

[3] DevEdge Online – Gecko Developer Central. [Online] Available http://developer.netscape.com/tech/gecko.

[4] UIML.org. [Online] Available http://www.uiml.org.

[5] HTML Home Page. [Online] Available http://www.w3.org/MarkUp/.

[6] Extensible Markup Language (XML). [Online] Available http://www.w3.org/XML/.

[7] World Wide World Consortium Document Object Model DOM. [Online] Available http://www.w3.org/DOM.

[8] Standard ECMA-262 ECMAScript Language Specification. [Online] Available http://www.ecma.ch/ecma1/stand/ecma-262.htm.

~~[9] Collaboration, Knowledge Representation and Automatability Working Group. [Online] Available http://www.w3.org/Collaboration/.~~

~~[10] Rice, J., Farquar, A. Piernot, P. & Gruber, T., "Using the Web Instead of a Window System", Proceedings of CHI'96, Vancouver, Canada, 1996.~~

[11] Chang, B-W, 'In-Place Editing of Web Pages: Sparrow Community-Shared Documents', WWW7 Conference, Brisbane, 1998, http://www7.conf.au/programme/fullpapers/1929/com1929.htm.

[12] Rees, M J, "Implementing Responsive Lightweight In-page Editing", Proceedings of AusWeb 2000, Cairns, June, 2000, pp 134-142.

[13] Rees, M J, "Implementing Shared Document Preparation with Lightweight Editing", Proceedings Fifth Australasian Document Computing Symposium (ADCS), December 2000, Twin Waters Resort, Sunshine Coast, pp 25-33.

[14] Rees, M. J. On-The-Dot Software. [Online] Available http://comet.it.bond.edu.au/dot/.

[15] Udell, J. The Universal Canvas. [Online] Available http://www.byte.com/documents/s=705/BYT20010608S0001/index.htm.

[16] Udell, J. The Universal Canvas Revisited: Approaches To Live Editing of Web Pages. [Online] Available http://www.byte.com/documents/s=1113/byt20010806s0004/20010806_udell.html.

[17] Nielsen, J., *Designing Web Usability*, New Riders, 2000.

[18] Rees, M. J., White, B. & White A., *Designing Web Interfaces*, Prentice-Hall, 2001.

[19] SharePoint Team Services. [Online] Available http://www.microsoft.com/frontpage/sharepoint.

[20] Digital Dashboard on Microsoft Business. [Online] Available http://www.microsoft.com/business/dd.

[21] Mary Kirtland, "The Programmable Web: Web Services Provides Building Blocks for the Microsoft .NET Framework", MSDN Magazine, September 2000. [Online] Available http://msdn.microsoft.com/msdnmag/issues/0900/WebPlatform/WebPlatform.asp.

[22] Greenberg, S. & Rounding, M, "The Notification Collage: Posting Information to Public and Personal Displays", Proceedings of SIGCHI'01, Seattle, 2001, pp 514-521.

[23] Hochheiser, H. & Shneiderman, B, "Universal Usability Statements: Marking the Trail for All Users", ACM Interactions, March + April 2001, pp 16-18.

[24] Colouris, G. & Thimbleby, H., HyperProgramming: Building Interactive Programs with HyperCard, Addison-Wesley, 1993.