The Development of Web-based Services

N. Partarakis[1], C. Doulgeraki[1], M. Antona[1], C. Stephanidis[1,2]

[1]Foundation for Research and Technology – Hellas (FORTH)
Institute of Computer Science
Heraklion, Crete, GR-70013, Greece
cs@ics.forth.gr
[2] University of Crete, Department of Computer Science

Abstract: This Chapter presents an overview of widely available methods and tools for developing web-based services. In this context, modern programming languages, popular user interface (UI) toolkits and Integrated Development Environments (IDEs) are briefly reviewed, taking into account the facilities offered to software developers for producing more advanced web-based services. The main drawback of these development approaches comes out to be their inability to cope with the diversity of the target user population. A potential solution is proposed through a novel approach to the development of inclusive Web-based interfaces (web content), capable to adapt to multiple and significantly different profiles of users and contexts of use. To this purpose, an advanced toolkit called *EAGER* is proposed. EAGER allows *Microsoft® .NET* developers to create interfaces that conform to the World Wide Web Consortium (W3C) accessibility guidelines, and which are able to adapt to the interaction modalities, metaphors and user interface elements most appropriate to each individual user, according to profile information (user and context). The process of employing EAGER is significantly less demanding in terms of time, experience and skills required from the developer than the typical process of developing Web interfaces for the "average" user. Ultimately, EAGER offers significant benefits to developers, and ensures the delivery of widely accessible, usable and satisfying Web-based interfaces.

1. Introduction

The World Wide Web serves as an unprecedented resource for knowledge, communication, and data and services acquisition, and plays a key role in an increasing number of aspects of everyday life, including commerce, information, education and training, job searching and remote collaboration, entertainment, social participation, and interaction with public administrations. The Web, thanks to its universality and the evolving usefulness (if not necessity) of its content, holds an unprecedented potential of reaching an enormous number of individuals; a population of potential users significantly characterized by diverse interaction skills, abilities, preferences, and access equipment (personal computers, mobile phones and other small display devices, web-TV, kiosks, assistive technology, etc.).

Admittedly, development of Web applications and services that meet the needs and requirements of as many diverse users as possible is a difficult and demanding task. The development frameworks available to modern software developers (such as ASP.NET, JavaServer Faces, Ajax based frameworks, PHP toolkits and the traditional HTML syntax) are designed to offer complex artifacts for building advanced interaction scenarios. Unfortunately, little has been done for incorporating into these development frameworks knowledge regarding the user and the context of use, in order to support not only usability, but also access to anyone and in particular people at risk of exclusion. The task of embedding these features in modern web based applications is put in the hands of developers that are supported with powerful development environments for achieving their goal (Microsoft Visual Studio .NET, NetBeans IDE, Eclipse, etc). The vast majority of them, by "tradition" (if not as a compromise), design and develop their artifacts for the typical or so-called "average" user, trusting this as the best solution to cater the needs of the broadest possible population. Unfortunately, this approach leads to excluding numerous categories of users, such as non-expert IT users, the very young or the elderly, people with disability, etc. [21]. However, specialized designs for one user group often constrain the capabilities of another still important group. As a result, developers, eventually pushed by social or market needs towards broadening their user base, are often required to further "improve" their artifacts so that these adhere to generalized (i.e., average - again) usability and accessibility guidelines. Ultimately, this way of

practice, usually also accompanied by limited user testing, often leads to end-products that fail to justify their underlying effort investments.

Contemporary users increasingly desire and expect the delivery of interfaces that are highly tailored to their own needs, and hardly compromise on rigid solutions for some imaginary "average" users. To this end, the main challenge faced today by designers and researchers of Web User Interface (WUI) is to elaborate and deploy approaches that can meet effectively, in various contexts, as many diverse needs and requirements as possible. An indicative list of dimensions of diversity to be considered is presented in Table 1. As a result, design - or better saying design needs to be equally targeted towards all potential users [21].

Table 1. Dimensions of diversity – some examples

Target population	User tasks	Context of use	Means of access
Language Age Background Skills Preferences Disability (blind, motor impaired, deaf, cognitive impaired) …	Work Socialisation Entertainment Education Surfing Commerce Government …	Home Office School Car Internet café … (light, noise, privacy, security, etc.)	Platform (Public terminals, PCs and Laptops, PDA and smart phones) Assistive Technology (screen readers, scanning, Browser (Explorer, Netscape, Firefox) …

Recent approaches to *Universal Access* and *Design for All* emphasize the central role of user interface adaptation towards satisfying, equally, the needs and requirements of diverse target user groups, including of people with disability [20]. So far however, adaptation has been explored mainly in the context of independent applications. In the Web environment, adaptation techniques have been applied mainly at the level of user agents (e.g., the AVANTI browser [22]). However, such approaches are limited by the fact that the user must have the actual product installed on the computer used to gain access to Web content. On the other hand, intermediary agents acting as filtering and transformation tools have been proposed and used to build alternative and often called "more accessible" versions of web pages taking into account a collection of accessibility guidelines that can be checked against a web page automatically. The deployment of this later concept (e.g., [2]) has highlighted a number of practical issues putting the universality of the approach under question, as a posteriori developments of specialized intermediary agents are required almost from scratch for each website.

2. Development methods and tools

This section provides an overview of facilities currently available to developers for addressing the increasing need for more versatile and powerful applications that can cope not only to the transition from the traditional computer aided work environment to the trend of collaborative web-based work places but also to the need of providing seamless access to this facilities regardless of age disability and context of use. To this end, development methods in terms of programming languages and UI toolkits are presented together with the modern Integrated Development Environments (IDEs) that are envisioned to support the majority of development activities.

2.1 Programming languages

Traditional web based development was carried out through the development of simple HTML pages [5] that were later enriched by scripts for allowing a richer interaction with a web page. This approach was quickly found not sufficient, since more aspects of everyday work needed to be transferred on the web, such as mail browsing, communication, collaboration and project administration and management. The need for these applications leaded to the need of employing general purpose programming languages (such as C# and java) and advanced scripting languages (such as PHP) for supporting their development.

2.1.1 Microsoft C#

C# is a simple, modern, object-oriented, and type-safe programming language [3]. C# has its roots in the C family of languages and will be immediately familiar to C, C++, and Java programmers. C# is standardized by ECMA International as the ECMA-334 standard and by ISO/IEC as the ISO/IEC 23270 standard. Microsoft's C# compiler for the .NET Framework is a conforming implementation of both of these standards [3]. C# together with the available powerful web development toolkits (such as ASP.NET [12] and AJAX.NET [1]) can be used to build advanced web-based services.

2.1.2 SUN java

The Java programming language is a general-purpose concurrent class-based object-oriented programming language, specifically designed to have as few implementation dependencies as possible. It allows application developers to write a program once

and then be able to run it everywhere on the Internet [6].

2.1.3 php

PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML [18]. What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client. The client receives the results of running that script and not the script itself. The main advantage of using PHP is that it is extremely simple for a newcomer, but offers many advanced features for a professional programmer. Although PHP development is focused on server-side scripting, it can also be used for other purposes [19].

2.2 UI toolkits and web application development libraries

In the previous section an overview of the programming languages offered to developers for developing modern web application were highlighted. Unfortunately, the existence of these languages alone cannot solve all the problems faced, mainly in relation to the limited functionality of elements provided by traditional HTML syntax. The main drawback introduced by HTML was the inability to deal with HTML elements in an object oriented manner, as well as the lack of facilities for creating more rigid and powerful HTML that would ensure code reusability, reduced programming time and flexibility. These drawbacks made clear that the development of special purpose UI toolkits, which could cooperate in an object oriented manner with general purpose languages, was essential. This section focuses on these UI toolkits, mainly presenting the innovations they introduced.

2.2.1 Microsoft ASP.NET

Microsoft ASP.NET is a free technology that allows programmers to create dynamic web applications. ASP.NET can be used to create anything from small, personal websites through to large, enterprise-class web applications [12]. More specifically, ASP.NET is a unified Web platform that provides all the services necessary for building enterprise-class applications. ASP.NET is built on the .NET Framework, so all .NET Framework features are available to ASP.NET applications. Applications can be written in any language that is compatible with the common language runtime (CLR), including Visual Basic and C# [13]. ASP.NET includes [14]:

- A page and controls framework
- The ASP.NET compiler
- Security infrastructure
- State-management facilities
- Application configuration
- Health monitoring and performance features
- Debugging support
- An XML Web services framework
- Extensible hosting environment and application life cycle management
- An extensible designer environment

2.2.2 Java server faces

JavaServer Faces technology is a server-side user interface component framework for Java technology-based web applications. The main components of JavaServer Faces technology are as follows [7]:

- An API for representing UI components and managing their state; handling events, server-side validation, and data conversion; defining page navigation; supporting internationalization and accessibility; and providing extensibility for all these features.
- Two JavaServer Pages (JSP) custom tag libraries for expressing UI components within a JSP page and for wiring components to server-side objects.

The well-defined programming model and tag libraries of JSP significantly ease the burden of building and maintaining web applications with server-side UIs, making it possible with minimal effort to wire client-generated events to server-side application code, bind UI components on a page to server-side data, construct a UI with reusable and extensible components, and save and restore UI state beyond the life of server requests.

2.3 Integrated Development Environments (IDEs)

An Integrated Development Environment is a set of tools that aids application development. Most IDEs have tools that allow developers to [9]:

- Write and edit source code

- See errors while typing
- See highlighted code syntax
- Automate repetitive tasks
- Compile code
- Browse class structures
- View Documentation
- Use drag-and-drop utilities for easy building of features, such as graphic objects or creating database connections

In addition, some advanced IDEs [9]:

- Provide templates for quick creation of web components
- Provide code-completion while typing
- Automatically create classes, methods, and properties
- Integrate with source code repositories
- Integrate with web application servers
- Integrate with build utilities
- HTTP monitoring for debugging web applications
- Unified UI for debugging code
- Macros and abbreviations
- Refactor code
- Provide UML support.

This sections provides an overview of the specific features offered by the most popular IDEs (such as Microsoft Visual studio, Net Beans IDE and Eclipse)

2.3.1 Microsoft Visual studio

Visual Studio is the standard Development Environment provided by Microsoft for building applications using .Net technologies. Visual Studio provides a developer friendly environment together with a vast number of facilities such as syntax highlighting, advanced debugging facilities, etc. Microsoft® Visual Studio delivers on Microsoft's vision of smart client applications by enabling developers to rapidly create connected applications that deliver the highest quality, rich user experiences. According to Microsoft Visual Studio offers facilities for [10]:

- Developing Smart Client Applications.
- Creating Microsoft Office Applications.

- Building Windows Vista Applications.
- Handling Data More Productively.
- Enabling New Web Experiences
- Gaining an Improved Overall Developer Experience.
- Improving Application Lifecycle Management (ALM)

More specifically, in the context of web applications through Microsoft® Visual Studio, Microsoft developers are offered with [11]:

- a robust, end-to-end platform for building, hosting, and exposing applications over the Web.
- easy creation of Web applications with more interactive, responsive, and efficient client-side execution
- creation of new Web experiences by empowering Web developers through simplifying Web development
- effective collaboration and faster results by integrating the advanced designers and editors of
- the tools required to create compelling, expressive, Web applications with "AJAX-style" interactive Web user interfaces.

2.3.2 Net Beans IDE

NetBeans is a free, open-source Integrated Development Environment for software developers [15]. It provides all the tools needed to create professional desktop, enterprise, web, and mobile applications with the Java language, C/C++, and Ruby. The NetBeans IDE is easy to install and use straight out of the box and runs on many platforms including Windows, Linux, Mac OS X and Solaris [16]. Latter releases provide several new features and enhancements, such as rich JavaScript editing features, support for using the Spring web framework, and tighter MySQL integration. Some of the key features introduced by NetBeans include:

- Easy-To-Use Java GUI Builder
- Visual Web and Java EE Development
- Visual Mobile Development
- Visual UML Modeling
- Ruby and Rails Support
- C and C++ Development.

2.3.3 Eclipse

Eclipse Foundation's Eclipse IDE [4], originally designed and implemented by IBM, aims to offer a comprehensive service platform for integrating development and deployment tools for a variety of programming languages. The Eclipse platform, however, mainly constitutes a complete IDE for the language it is written in – Java.

Eclipse employs a component framework based on the OSGi [17] specification in order to provide all of its functionality on top of its platform. Through that mechanism, Eclipse can be fully extended in the Java language, as it essentially allows programmers to access the platform's components and replace them by implementing their Java abstract interfaces.

Lastly, Eclipse's editor for the Java programming language utilizes the compiler to validate the edited program's syntax. By using the compiler's internal representation of the program, the editor provides refactoring tools and automatic symbol completion for Java objects.

2.4 Case Study: Developing a simple web based service using Microsoft C#, the ASP.NET UI toolkit and Visual Studio 2003 IDE

The aim of this section is to elucidate the benefits of employing a subset of the facilities presented in the previous sections for developing a simple web based service in terms of the developer's performance and efficiency. To this end, the technologies used are Microsoft C# programming language ASP.NET UI toolkit and Visual Studio 2003 IDE and the case study is a simple web-based service for posting messages. The facilities that are highlighted include the usage of a modern IDE with design time support together with a general purpose programming language and a specialized web base UI toolkit.

In order to develop a Web application using the ASP.NET, the following are considered as minimum prerequisites:

- An operating system supporting ASP .NET: Windows 2000 (Professional, Server, and Advanced Server) or Windows XP Professional or Windows Server 2003
- Internet Information Services
- .NET Framework version 1.1

- Microsoft Visual studio 2003
- Microsoft SQL Server.

For providing an overview of the tasks involved in developing a web application using the aforementioned facilities, this sections addresses how to create, build and run a simple web page, for example a page that supports posting a topic on a message board. The development language used to build the aforementioned example is C# along with the standard ASP .NET Web UI controls Library.

The first step introduced in the development process is the creation of a new web application. This is achieved by selecting File → New → Project in the Visual Studio 2003 IDE. This brings up the 'New project' dialog that is presented in Figure 1, where the developer clicks on the 'Visual C#' node in the tree-view on the left hand side of the dialog box and choose the 'ASP.NET Web Application' icon. Next, the developer types the name of the project and hits the button OK.

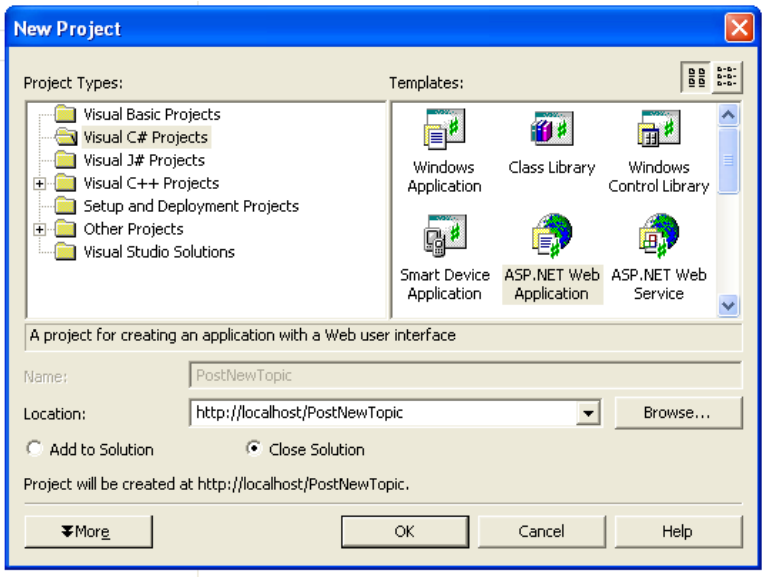


Figure 1: Visual studio: The New Project dialog

Visual Studio will then create and open a new web project within the solution explorer. By default, it will have a single page (WebForm1.aspx), an AssemblyInfo.cs file, a Global.asax file, as well as a Web.config file (see Figure 2). All project file-meta-data is stored within an MSBuild based project file.

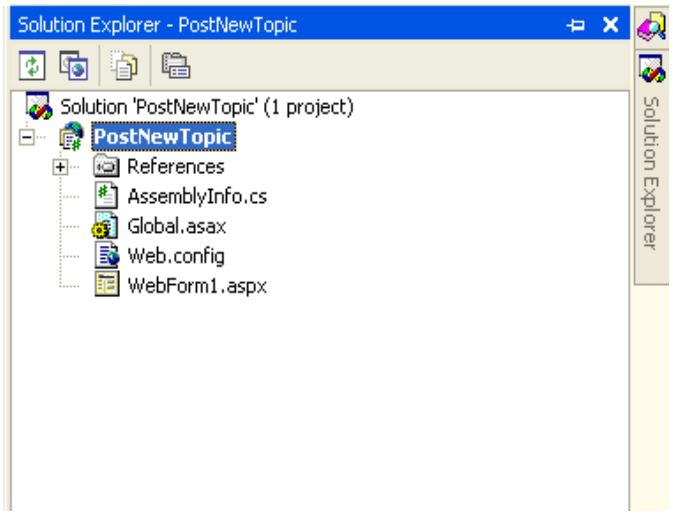


Figure 2: Visual studio: New project solution explorer

2.4.1 Using the Designer

Step 1. Design the User Interface

Firstly, a mockup of the page to be developed is designed. Figure 3 presents a mockup that includes a title and a description field along with the date entry module and a module for attaching files to topic. Files are attached using the browse button in order to locate the file, and the attach button to upload the located file. The field-set topic files are used to present the uploaded files. If a file was uploaded by mistake, the user can delete it by checking it and then by pressing the delete button.

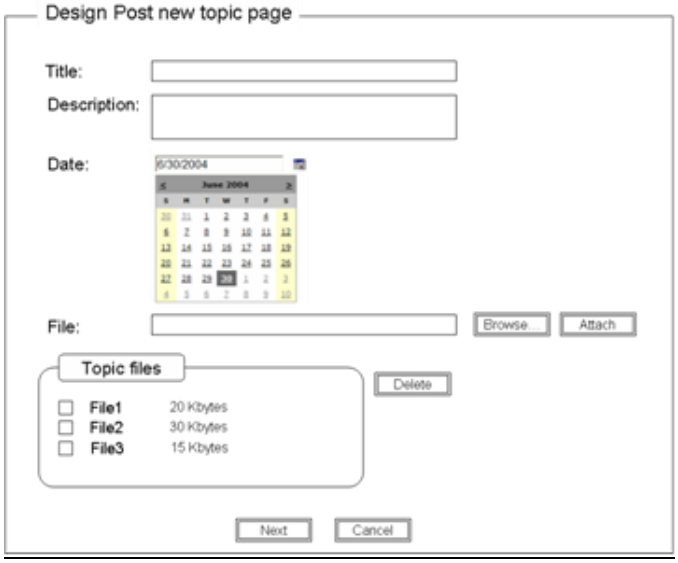


Figure 3: Mock-up of the page

Step 2. Writing ASP. Net Markup

When the web page has been designed, the next step is the process of writing ASP .Net markup that renders the controls that have been designed. Figure 4 includes the mockup fields for title, description, file uploader and date along with their ASP .Net representation. Figure 5 presents the ASP .Net markup, that is required in order to render the field 'Topic files', the list of uploaded files and the buttons 'Next' and 'Cancel'.

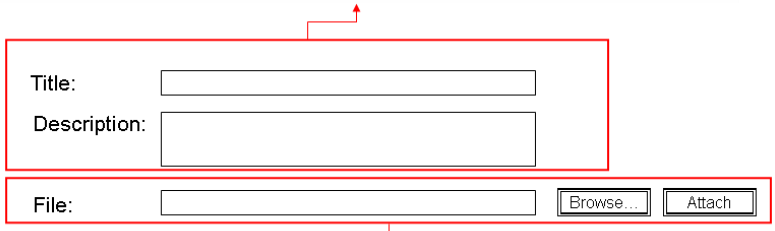
Step 3. Adding functionality to the controls

Subsequently, functionality has to be added to the controls in order for them to function properly. This functionality may vary and includes assigning text to labels, setting visibility to controls, filling repeater of files, deleting files, storing data to database, etc. In Figure 6 an example of the functionality that is required in order to upload files (topic attachments) is presented. Figure 7 includes the necessary code in order to fill in the repeater representing the uploaded files, and Figure 8 includes the code to delete undesired files.

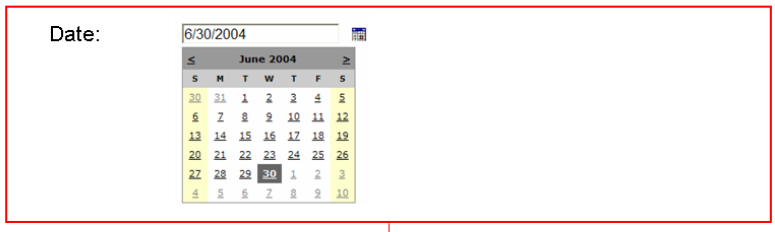
Step 4. Building and running the project

When the developer has incorporated all the appropriate functionality, the web application can be run by hitting the button 'F5', and the results are viewed in the browser.

```
<DIV class="tr-div">
    <LABEL id="lblTopicTitle" Runat="server" class="td-20l-b"></LABEL>
    <SPAN class="td-80r">
        <asp:textbox id="txtTopicTitle" Runat="server"></asp:textbox>
    </SPAN>
</DIV>
<DIV class="tr-div">
    <LABEL id="lblTopicDescription" Runat="server" class="td-20l-b"></LABEL>
    <SPAN class="td-80r">
        <asp:textbox id="txtTopicDescription" Runat="server"></asp:textbox>
    </SPAN>
</DIV>
```

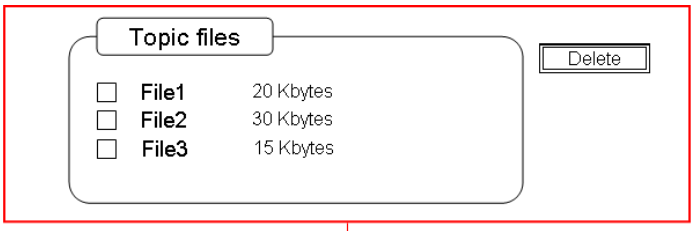



```
<DIV class="tr-div">
    <asp:label id="lblAttachments" Runat="server" CssClass="td-20l-b"></asp:label>
    <SPAN class="td-80r">
        <INPUT id="txtBrowseAttachments" type="file" size="50" runat="server" class="btn-browse" >
        <asp:button id="btnAddAttachment" Runat="server" CssClass="button_2"></asp:button>
    </SPAN>
</DIV>
```

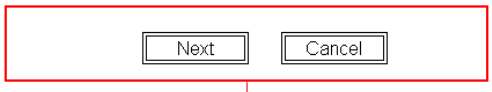



```
<div class="tr-div">
    <asp:label id="lblDate" Runat="server" CssClass="td-20l-b"></asp:label>
    <SPAN class="td-80r">
        <asp:Calendar id="dateEntry" runat="server"></asp:Calendar>
    </span>
</div>
```

Figure 4: Writing HTML (1)




```
<DIV class="box6_container_1" id="flAttachments" runat="server"><!-- box 6 head -->
    <DIV class="box6_head_container">
        <DIV class="box6_head_bg_b">
            <DIV class="box6_head_bg_a">
                <DIV class="box6_head_bg_c"><!-- box 6 title -->
                    <DIV class="box6_title_container">
                        <DIV class="box6_title_bg_a_2">
                            <DIV class="box6_title_bg_a_1">
                                <DIV class="box6_title_bg_a_3"></DIV>
                            </DIV>
                        </DIV>
                        <DIV class="spacer"></DIV>
                        <DIV class="box6_title_bg_b_2">
                            <DIV class="box6_title_bg_b_1">
                                <DIV class="box6_title_bg_b_3">
                                    <DIV class="box6_title"><LABEL class="lg" id="lgAttachments" runat="server"></LABEL></DIV>
                                </DIV>
                            </DIV>
                        </DIV>
                        <DIV class="spacer"></DIV>
                        <DIV class="box6_title_bg_c_2">
                            <DIV class="box6_title_bg_c_1">
                                <DIV class="box6_title_bg_c_3"></DIV>
                            </DIV>
                        </DIV>
                    </DIV>
                    <DIV class="spacer"></DIV> <!-- end box 6 title --></DIV>
            </DIV>
        </DIV>
    </DIV> <!-- end box 6 head -->
    <DIV class="spacer"></DIV> <!-- box 6 body -->
    <DIV class="box6_body_container">
        <DIV class="box6_body_bg_b">
            <DIV class="box6_body_bg_a">
                <DIV class="box6_body_bg_c">
                    <DIV class="box6_body">
                        <asp:Repeater id="rptFiles" Runat="server" OnItemDataBound="rptFiles_DataBound">
                            <ItemTemplate>
                                <div class="tr-div">
                                    <span class="td-51">
                                        <asp:CheckBox ID="chkFiles" Runat="server"></asp:CheckBox>
                                    </span><span class="td-95r">
                                        <asp:Button ID="btnFiles" Runat="server" CssClass="transparentButton"></asp:Button>
                                    </span>
                                </div>
                            </ItemTemplate>
                        </asp:Repeater></DIV>
                </DIV>
            </DIV>
        </DIV>
    </DIV> <!-- end box 6 body -->
    <DIV class="spacer"></DIV> <!-- box 6 footer -->
    <DIV class="box6_footer_container">
        <DIV class="box6_footer"></DIV>
    </DIV> <!-- end box 6 footer -->
```




```
<DIV class="tr-div">
    <asp:button id="btnNext" Runat="server" CssClass="button_2"></asp:button>
    <asp:button id="btnCancel" Runat="server" CssClass="button_2"></asp:button>
</DIV>
```

Figure 5: Writing HTML (2)

```
    private void btnAddAttachment_Click(object sender, System.EventArgs e)
    {
        string tmpFileName= txtBrowseAttachments.PostedFile.FileName.ToString();
        if(!tmpFileName.Equals(""))
        {
            string fileName = tmpFileName.Substring( tmpFileName.LastIndexOf( "\\" ) + 1 );
            string path = "./MessageBoard/Files/";
            string filePath = Server.MapPath( path + fileName );

            System.IO.DirectoryInfo newFolder = new System.IO.DirectoryInfo( Server.MapPath(path));
            if ( !newFolder.Exists )
            {
                newFolder.Create();
            }

            if(!txtBrowseAttachments.Value.Equals(""))
            {
                txtBrowseAttachments.PostedFile.SaveAs( filePath );
                string[] files = attached_files.Value.Split(',');
                string[] sizes = string_size.Value.Split(',');

                for (int i=0;i<files.Length;i++)
                {
                    repeaterItems.Add(files[i] + "(" + sizes[i] + ")");
                    ItemsSizes.Add(sizes[i]);
                    ItemsNames.Add(files[i]);
                }
            }
            else
            {
                lblErrorTopic.Text = interfaceText[13].ToString();
                pErrorTopic.Visible = true;
            }
        }
        else
        {
            divAtError.Visible = true;
            lblAtError.Text = interfaceText[22].ToString();
        }
    }
```

Figure 6: Code for uploading file

```
protected void rptAttachments_DataBound(object sender, System.Web.UI.WebControls.RepeaterItemEventArgs e)
{
    string attachments = (string)e.Item.DataItem;
    string[] sizes = string_size.Value.Split(',');
    int i;

    Button btnAttachments = (Button)e.Item.FindControl( "btnAttachments" );
    HyperLink lnkAttachment = (HyperLink)e.Item.FindControl("lnkAttachment");

    ListItemType itemType = e.Item.ItemType;
    if ( itemType == ListItemType.Item ||
        itemType == ListItemType.AlternatingItem)
    {
        btnAttachments.CommandName = "File";
        btnAttachments.CommandArgument = attachments;
        btnAttachments.Text = attachments + "(" + sizes[Convert.ToInt32(hdSizeValue.Value)] + ")";
        i = Convert.ToInt32(hdSizeValue.Value) + 1;
        hdSizeValue.Value = i.ToString();
    }
}

private void rptAttachments_ItemCommand(object source, System.Web.UI.WebControls.RepeaterCommandEventArgs e)
{
    string path = "MessageBoard/Files/" + e.CommandArgument.ToString();
    UploadDownload.DownloadFile(Server.MapPath(path),Response);
}
```

Figure 7: Code for presenting uploaded files

```
private void btnRemoveFiles_Click(object sender, System.EventArgs e)
 {
     ArrayList filenames = new ArrayList();
     ArrayList sizes = new ArrayList();
     ArrayList names = new ArrayList();

     foreach( RepeaterItem ri in rptFiles.Items)
     {
         CheckBox chkFiles = (CheckBox)ri.FindControl("chkFiles");
         Button btnFiles = (Button)ri.FindControl( "btnFiles" );
         if (chkFiles.Checked)
         {
             filenames.Remove(btnFiles.Text);
             string path = "./MessageBoard/Files/";
             string filePath = Server.MapPath( path + temp1 );
             System.IO.DirectoryInfo folder = new System.IO.DirectoryInfo( Server.MapPath(path));
             if (folder.Exists)
             {
                 System.IO.FileInfo[] files = folder.GetFiles();

                 for (int k=0; k<files.Length;k++)
                 {
                     if(files[k].Name.Equals(temp1))
                     {
                         files[k].Delete();
                     }
                 }
             }
             names.Remove(temp1);
             sizes.Remove(temp2);
         }
     }
     rptFiles.DataSource = filenames;
     rptFiles.DataBind();
 }
```

Figure 8: Code to remove uploaded files

2.5 Discussion

The previous sections presented an overview of the widely available development facilities and tools used today for creating the majority of web-based services, including the programming languages used, the UI toolkits offered and the IDEs employed during development. Additionally, a case study of the process followed for building a web-based service was presented, illustrating how the aforementioned facilities can be used. Unfortunately, however, little has been done so far for incorporating in these development frameworks knowledge regarding the user and the context of use for supporting not only usability but also access to anyone, and in particular people at risk of exclusion. The task of embedding these features in modern web based applications is put in the hands of web developers. The vast majority of web-based applications and services today, however, are developed for the so-called "average" users, trusting this as the best solution to cater the needs of the broadest possible population. Unfortunately, this approach leads to excluding numerous categories of users, such as non-expert IT users, the very young or the elderly, people with disability, etc.

3. EAGER: A development toolkit for supporting user interface adaptation of electronic services

This section presents a development toolkit that aims at overcoming the limitations discussed above. The approach followed by this toolkit builds on the facilities offered by modern programming languages and UI toolkits, and furthermore employees features offered by modern IDEs (such as design time support), and at the same time provides output that can be adapted to various user profiles and contexts of use.

3.1 Adaptive and Adaptable behavior

The support for adaptive and adaptable behavior has been recognized as a trustful medium for supporting the needs of the broadest possible user population in various contexts. In the context of web applications providing seamless access to web-based services demands solutions able to cope with a wide range of user and context requirements for facilitating the interaction needs of potentially all citizens. In this context, web user interface adaptations must take into account a wider collection of parameters, such as context and user specific attributes (e.g., input-out devices, disabilities, user attitude towards technology etc). The adaptation of an application can occur in different ways and can cover a number of aspects of the application or its environment.

The adaptation of an application can be classified according to which aspects of the application are adapted. Generally, Web application model described in [8] distinguishes five orthogonal aspects of an application:

(1) Content: Adaptation of content affects the content such as text, graphics or any other media type or data used or displayed by the application. This type of adaptation is most common on the Web. EAGER supports adaptations which automatically modify the presentation and conceived behavioral attributes of interactive elements. As an example images can be transformed as normal images, as simple text containing image's alternative text and as a hyperlink that downloads the image and has as text the image's alternative text.

(2) Navigation: Adaptation of navigation adapts the navigational structure of a web application hiding or modifying links. EAGER supports navigation adaptations. Some examples include the linearization of the whole navigation of the portal in

a top navigation bar in order to facilitate blind users or the step by step navigation which reduces the number of links that motor-impaired user has to scan.

(3) Layout: Adaptation of layout changes the way information is presented to a user visually. This can be done to accommodate different types of displays or to satisfy preferences of aesthetic, cultural or other nature a user may have. The proposed framework supports layout adaptations, as long as it offers alternative templates layouts depending on screen resolution, disability etc.

(4) User-interaction: Adaptation of user-interaction changes the way the user interacts with the application. An application might adapt offering a wizard based interface to less experienced users and a single page form to other users. EAGER supports conditional activation and deactivation of multiple interaction modalities based on the user profile including alternative task structures, alternative syntactic paradigms, task simplification and adaptable and adaptive help facilities - runtime task guidance.

(5) **Processing:** Adaptation of processing changes the way user-input is processed. For example, a product request of a person that has placed many large orders in the past might be processed differently than that of a previously unknown person. EAGER doesn't support such adaptations. These kinds of adaptations cannot be address by a generic framework and rely solely on the implementation of each web application.

3.2 The EAGER toolkit

EAGER[1] is a development toolkit for supporting user interface adaptation of electronic services. EAGER is targeted to support the application of the Unified Web Interfaces methodology which in turn is derived from the architectural structure proposed for enabling the development of Unified User Interfaces [20]. EAGER supports the development of web user interfaces which can adapt to interaction and accessibility requirement of the broadest possible end user population (see also the Chapter on "Designing web-based services" of this book) taking into account user profile information such as:

1 EAGER stands for "toolkit for **e**mbedding **a**ccessibility, **g**raceful transformation and **e**ase of use in Web–based products".

- language
- input device used (mouse, switches, game pad, etc)
- disability (e.g., blind, motor impaired, etc)
- device used for accessing the web (e.g., pc, pda, tablet pc, etc)
- assistive technology used (e.g., screen reader, screen magnifier, etc)
- web familiarity (e.g., novice, expert user, etc)

Using these basic user profile parameter EAGER decides upon the conditional activation of the available interaction elements and accessibility characteristics allowing at the same time end users to manually override the default decision making process by making specific selections to fine tune their interface. In this context some of the basic adaptations supported by means of EAGER in various contexts include:

- Accessibility:
    - Alternative table linearization styles
    - Alternative chart presentation schemes
    - Alternative image presentation schemes
    - Alternative font sizes
    - Alternative color setting schemes
    - Alternative text entry styles
    - Alternative links and buttons presentation styles
    - Alternative field sets presentations
    - Alternative text editing styles
    - Rendering of quick access links
    - Rendering of section breaks
    - Enabling template linearization
    - Enabling support for dynamic adaptation (adapting to context change)
    - Enabling text to speech
- Interaction:
    - Alternative file uploading styles
    - Alternative file displaying styles
    - Alternative paging styles
    - Alternative date selection styles
    - Alternative image uploading styles
    - Alternative image displaying styles

- Alternative styles for module functions
- Alternative styles for module options
- Alternative tab presentation styles
- Alternative Font Families
- Alternative search styles
- Alternative styles for presenting commonly used options
- Alternative navigation styles
- Alternative skins

From a technical point of view, EAGER is a prototype development toolkit of the core UWI architecture components: User Information, Context Information, Decision Making, Dialogue Controls (activation/deactivation), of the primitive UI elements with enriched attributes (e.g., buttons, links, radios, etc.), of the structural page elements (e.g., page templates, headers, footers, containers, etc.), and of the fundamental abstract interaction dialogues in multiple alternative styles (e.g., navigation, file uploaders, paging styles, text entry). The EAGER toolkit has been developed in Microsoft® Visual C# .NET and according to the UWI framework. The technologies that were used for the development of the EAGER toolkit include:

- Microsoft Visual C# .NET for the implementation of the UI modules.
- Microsoft Visual C# .NET and XML for Business Logic and Web Services.
- Microsoft SQL server 2000 for the database implementation.

For the development of EAGER, a number of UI elements were designed and implemented in various forms (polymorphic task hierarchies) according to specific user and context parameters values. This phase provided input to the actual development process of EAGER, which involved the implementation of the alternative interaction elements and of the mechanisms for facilitating the dynamic activation - deactivation of interaction elements and modalities based on individual user interaction and accessibility preferences.

In brief, EAGER is an advanced library of: (a) the core UWI architectural components; (b) primitive UI elements with enriched attributes, e.g., buttons, links and radios; (c) structural page elements, e.g., page templates, headers, footers and containers; and (d) fundamental abstract interaction dialogues in multiple alternative styles, e.g., navigation, file up-loaders, paging styles and text entry.

3.3 EAGER vs. traditional development approaches

In section 2.4, the process followed for developing a simple web application using the Microsoft ASP.NET development platform together with the Microsoft Visual Studio 2003 environment was presented. The aim of this section is to elucidate the benefits of employing the EAGER toolkit together with Microsoft's Visual Studio, in terms of the developer's performance and efficiency. Toward this end, an example of how to redevelop the simple webpage for posting messages presented in section 2.4 using the EAGER toolkit is presented, focusing on the expected empowerment of the developer, together with the support provided for developing web user interfaces capable of automatic adaptation behavior.

3.3.1 Using EAGER in combination with .Net

As discussed in the previous sections, the EAGER framework supports the development of web applications that adapt their UI elements to meet the requirements set by user and context specific attributes. In this section, the methods and techniques used for building a post topic page using the EAGER framework are presented.

Step 1. Design the User Interface

The design of a web page for posting topic demands a different logic when it is intended to be developed with the EAGER toolkit. The design of such a web page is task oriented, since the DCC discussed in section 3.2 includes several alternative User Interface components which encapsulate functionality for several tasks. Figure 9 includes two text entry components, a date entry component, multiple files entry component and a component for functions to be applied.

Step 2. Writing EAGER Markup

Following the design phase of the web page, the subsequent step includes the process of writing EAGER markup that renders the components that have been designed. Figure 10 includes all the components included in the mockup, along with their EAGER markup code. As an example, 'ics:icsDatePicker' defines a date entry control which may be transformed to alternative user interfaces such as dropdowns which contain year, month, day or textboxes for filling in year, month and day or a graphical calendar for clicking on date.

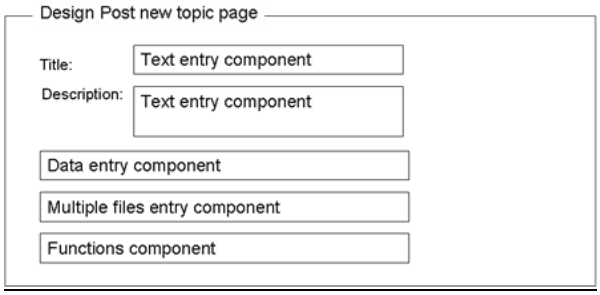


Figure 9: Task oriented mock-up of the page

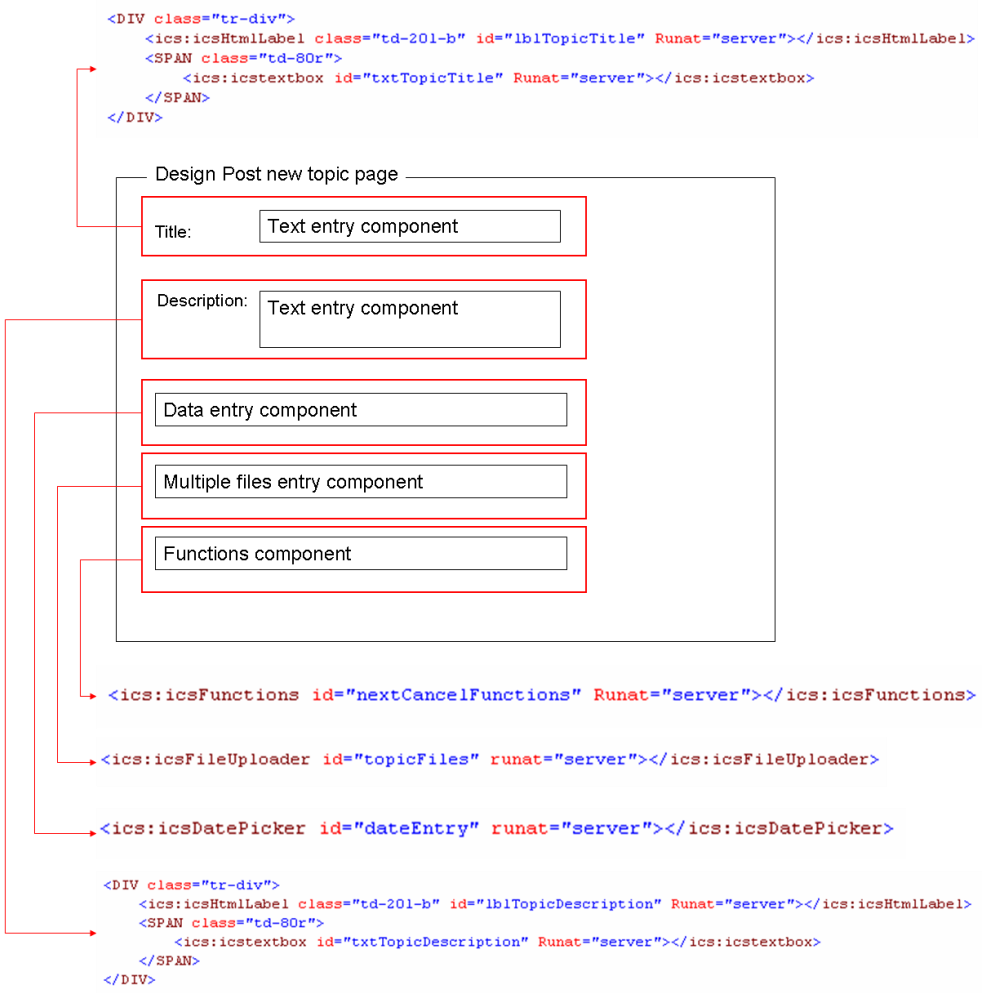


Figure 10: Writing EAGER Markup

Step 3. Adding functionality to the controls

At this stage, functionality has to be added to the controls in order for them to function properly. The functionality that is needed to be added to EAGER controls is radically reduced compared to the code that is required for the ASP .Net controls. Figure 11 represents all the code that is required in order for the module to support uploading - presenting files and deleting files. To configure the component for file uploading four variables are used, 'sourcePath' is used along with the 'sourceID' to combine the temporary location where the files will be saved and the 'fPath' along with the 'destID' to combine the final location where the files have to be saved. The source path and the final path can be the same.

```
topicFiles.sourcePath = "~/MessageBoard/Files/temp";
topicFiles.fpath = "~/MessageBoard/Files/Message";
topicFiles.sourceID = userId;
topicFiles.destID = topicId;
```

Figure 11: Code for uploading file

Step 4. Building and running the project

When the developer has incorporated all the appropriate functionality, the web application can be run by hitting the button 'F5' and the results are viewed in the browser. In Figure 12, some of the alternative representations that may appear when the web application runs, depending on end user characteristics and the context of use, are presented.

In the first alternative representation, simple textboxes appear along with a graphical calendar and field-set and a simple file up loader. In the second option, the textbox changes color on focus and is offered along with a virtual keyboard. Date entry uses three dropdowns for year, month and day respectively. Finally, the functions 'Next' and 'Cancel' are provided with a small description next to each of them.

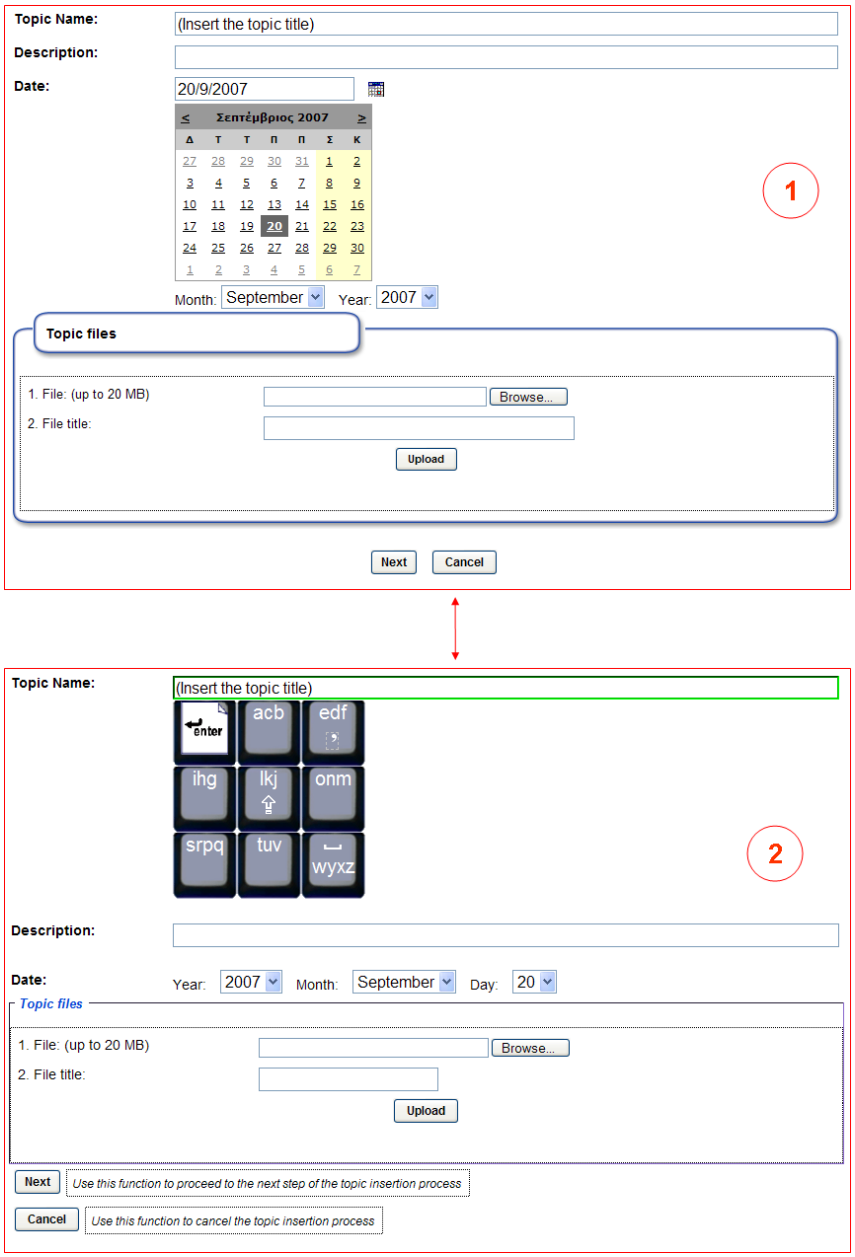


Figure 12: Post topic the alternative representations

3.4 Porting EAGER into an existing web-based service

The EAGER toolkit is not only a versatile tool for developing UWI from scratch. In the case of a portal developed by means of Microsoft's Visual Studio, EAGER can be ported in order to incorporate adaptation and improve the user-experience of end-users. This transformation can bring great benefits in a number of directions including accessibility, usability and the ability to serve diverse user requirements. It is worthy noticing that when the EAGER toolkit was ported into an existing web-based module

which was originally developed using Visual Studio, the resulting total number of code lines was significantly reduced by 50% (see Figure 13).

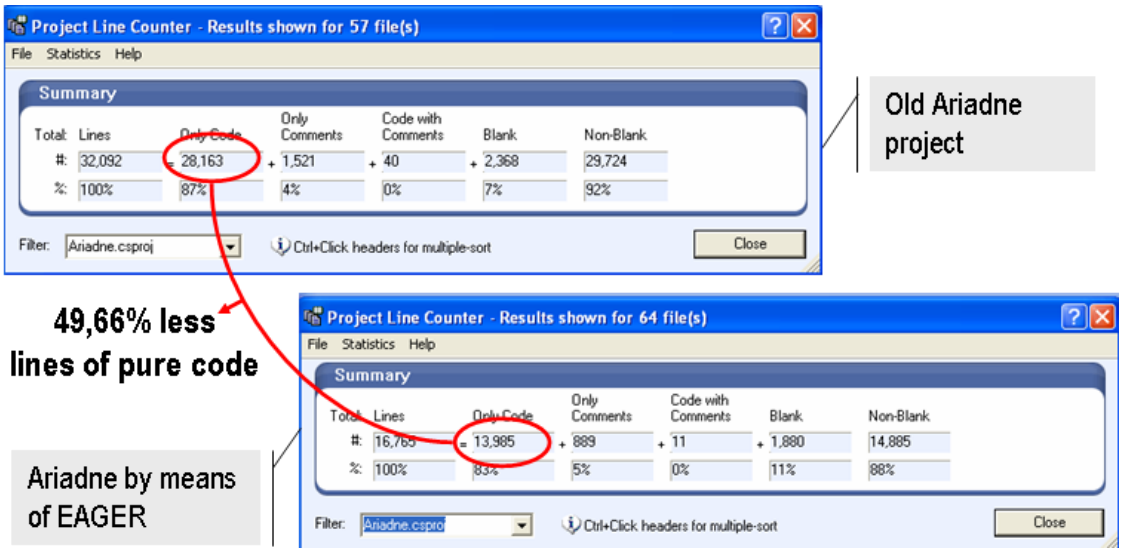


Figure 13: Comparison between developing with Visual Studio alone and with the EAGER toolkit

The process of porting EAGER into existing Web portals typically requires the following steps:

- EAGER setup: Involves the process of setting up the database schemes – Web services used by the EAGER toolkit for storing and retrieving user and context specific parameters.
- Import EAGER UI toolkit: Add a new reference to the EAGER toolkit to each project contained in the old portal.
- Import EAGER administrative facilities: Add the EAGER Profile Selection UI Module, Profiles Administration UI module and Statistics UI Module projects to the solution.
- Analyze existing application interfaces and identify functionality that can be abstracted and therefore replaced by the EAGER equivalents (such as file uploading code snippets, paging facilities, etc).
- Use the EAGER toolkit primitive controls instead of the build in ASP.NET controls (for example use the EAGER label instead of the ASP.NET label).

3.5 Benefits of using EAGER

The benefits gained by using the EAGER toolkit lie on a number of dimensions, including:

- The time required for designing a web application and the detail of design information needed.
- The time required for designing the front end of the application to be used by end users.
- The developer effort for setting up the application.

As discussed above, in the process of designing a simple form for posting topics, the complexity of the UI design effort is radically reduced due to the flexibility provided by the EAGER toolkit for designing interfaces at an abstract task-oriented level. Using EAGER, designers are not required to be aware of the low level details introduced in representing interaction elements, but only of the high level structural representation of a task and its appropriate decomposition into sub tasks, each of which represents a basic UI and system function.

On the other hand, the process of designing the actual front end of the application using a mark-up language is radically decreased in terms of time, due to the fact that developers initially have to select among a number of interface components each of which represents a far more complex facility. Additionally, developers do not have to spend time for editing the presentation characteristics of the high level interaction element, due to the internal styling behavior.

The actual process of transforming the initial design into the final Web application using traditional UI controls introduces a lot of coding. On the contrary when using EAGER the amount of code required is significantly reduced due to the fact that developer has the option to use a number of plug and play controls each of which represent a complex user task. These controls are contained in the advance UI library of EAGER consisting of a total number of 55K pure code lines (see Figure 14). Furthermore, the incorporation of EAGER's higher level elements make a portal's code more usable, more readable and especially safe, due to the fact that each interaction component introduced is designed separately, developed and tested introducing a high level of code reuse, efficiency and safety.

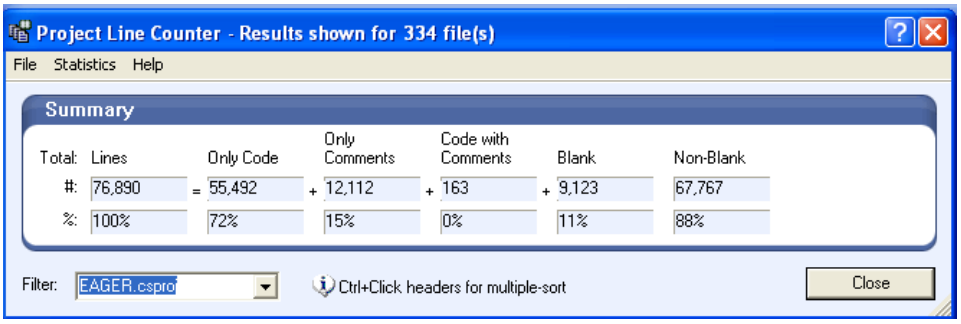


Figure 14: Total number of lines of pure code constituting the EAGER toolkit

Finally, the UIs developed with the EAGER toolkit can adapt according to specific user and context parameters, and therefore are rendered in a number of variations. It is therefore clear that using a standard UI toolkit a monolithic interface is created, whereas using the EAGER toolkit dynamically adaptable interfaces are generated.

4. Case Study: A prototype web-based service developed by means of EAGER

4.1 The portal of the European Design for All e-Accessibility Network (EDeAN)

As a proof-of-concept, a prototype portal was developed by means of the EAGER toolkit. In order to elucidate the benefits of EAGER, an already existing portal was selected and redeveloped from scratch. In this way, it was possible to identify and compare the advantages of using EAGER, both at the developer's site, in terms of developer's performance, as well as at the end-user site, in terms of user-experience improvement. In particular, the original portal of the *European Design for All e-Accessibility Network* (EDeAN) was redesigned and re-implemented using the EAGER development framework.

The new EDeAN portal[2] disseminates information about the scope, objectives and outcomes of the EDeAN networking activities. Through the portal public area (Figure 15) a number of facilities can be accessed, such as information about EDeAN, resources from a dedicated resource centre, news and announcements, frequently Asked Questions, statistics regarding the networking activities and surveys for

[2] http:www.edean.org

collecting user feedback. The portal area for subscribed users is intended to support the actual networking activities, and therefore provides a number of communication and collaboration facilities.

Figure 15: The EDeAN portal (various skins)

4.2 Adaptation based on user and context specific parameters

The users of the portal have the option to access the portal settings and alter them in order to match their personal characteristics and the characteristics of the context of use. A number of parameters can be set, such as Language, Device & display resolution, Assistive technology, Input Device, Disability and Web familiarity. Additionally, in order to allow users to quickly alter their settings, the quick settings option can be used, offering a number of predefined user profiles (see Figure 16).

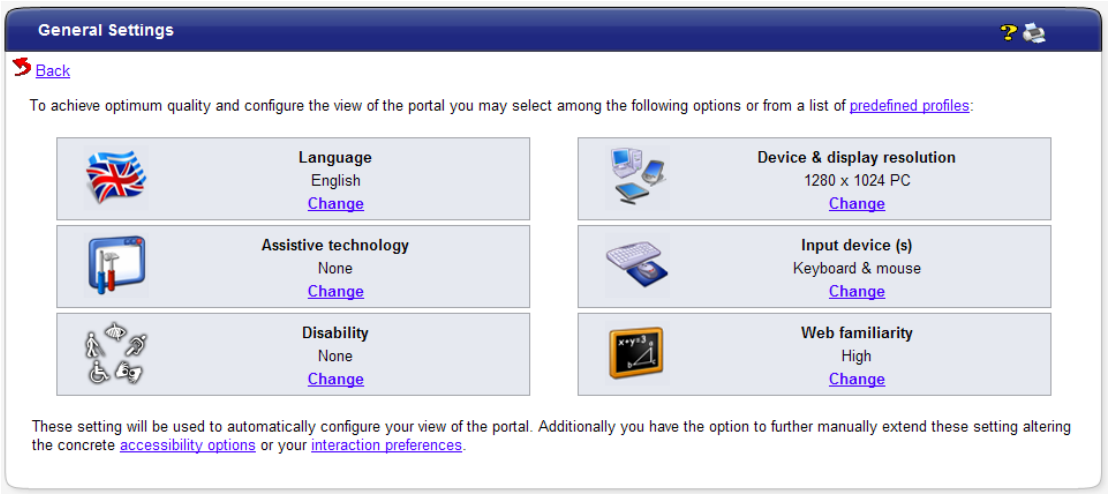


Figure 16: General Settings

4.3 Adaptation based on specific interaction preferences

Interaction preferences represent settings that affect the way the user interacts with the portal. More specifically, these settings can alter the interaction elements used for performing fundamental operations, such as browsing content and images or uploading files. The changes made to these settings are propagated to all portal modules. By manually altering these setting the default adaptation logic that occurs based on the user basic setting is enriched. The administration interface provided to portal users for altering these settings is presented in Figure 17.

Figure 17: The interaction preferences administration interface

4.4 Adaptation based on specific accessibility preferences

Custom Accessibility includes all the settings that can be altered to enhance the accessibility characteristics of the final user interface. This is very important for offering personalized experience from an accessibility perspective. Although each user interface is already compliant with the W3C accessibility guidelines, theses settings can further enhance the actual system accessibility and the perceived quality of interaction. The main administration interface provided to end users for editing these setting is illustrated in Figure 18. Each setting is presented graphically together

with the currently selected value, and the option to alter the selected values is provided.

Figure 18: The Accessibility preferences administration interface

4.5 Adaptation examples

This section presents some examples of the resulting portal UIs using a number of alternative predefined profiles, in order to provide a quick overview of the possible transformation at the user's end. As a first example, activating the "Blind with no Assistive Technology and High Expertise" profile results in the interface presented in Figure 19, which highlights the following adaptations:

(1) Text to Speech output is enabled for coping with the lack of assistive technologies. This adaptation is therefore used to mimic the functionality offered by screen readers.

(2) Quick access links are presented on the top right and bottom right section of the page allowing blind users to quickly accesses the most important areas without the need to repeatedly scan the whole page

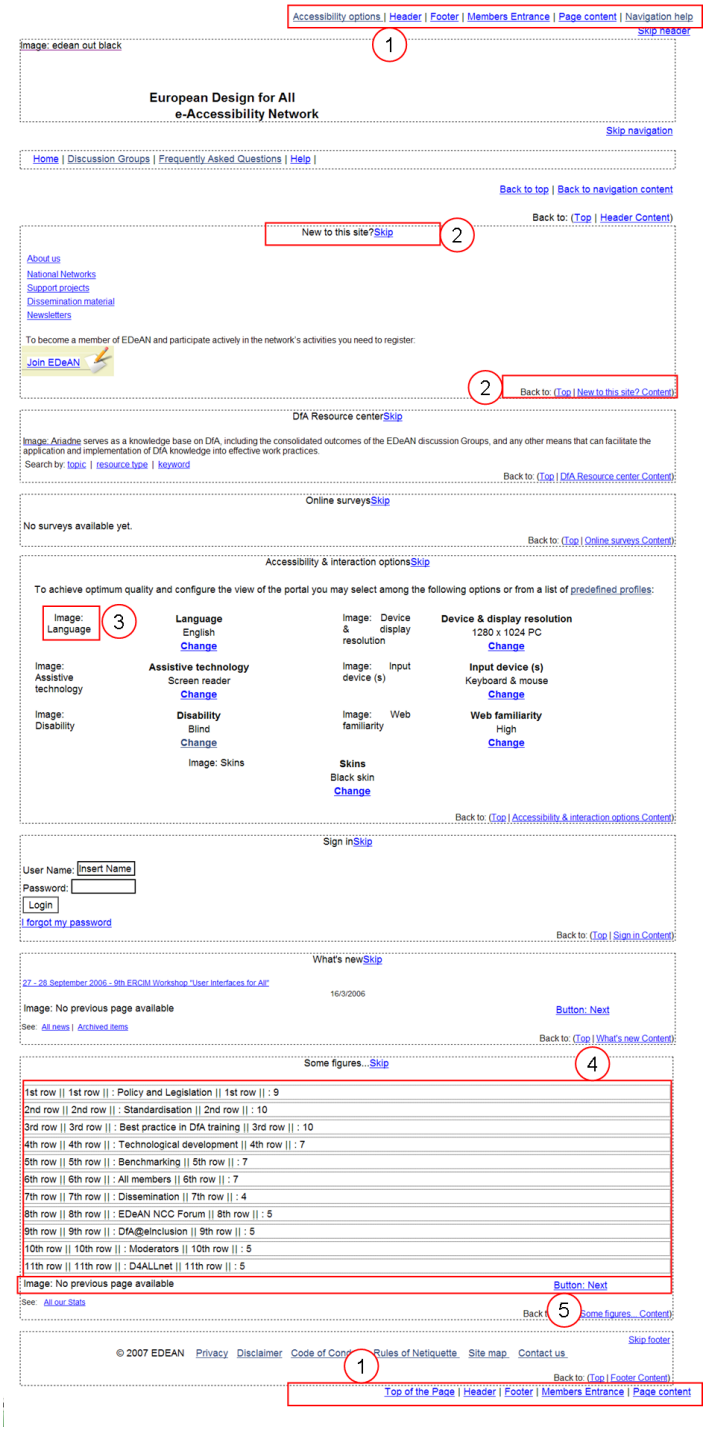


Figure 19: Profile of blind user with no Assistive Technology and High Expertise

(3) Section breaks are displayed on each page region allowing users with high expertise to skip page sections while navigating resulting to reduced navigation time

(4) Images are displayed as text enabling blind users to access their alternative descriptions and furthermore reduce the portal loading time

(5) Tables are linearized in order to provide meaningful information to blind users following the appropriate scheme for representing table data together with row and column information.

(6) Image Buttons are transformed to links enabling blind users to access links with their alternative image descriptions and furthermore reduce the portal loading time

General adaptations that affect the overall look and feel of the page include the linearization of templates, the absence of graphics, and the color scheme introduced (white background and black foreground).

Activating the "Motor Impaired, two Switches, Low Expertise" profile results in the layout presented in Figure 20.

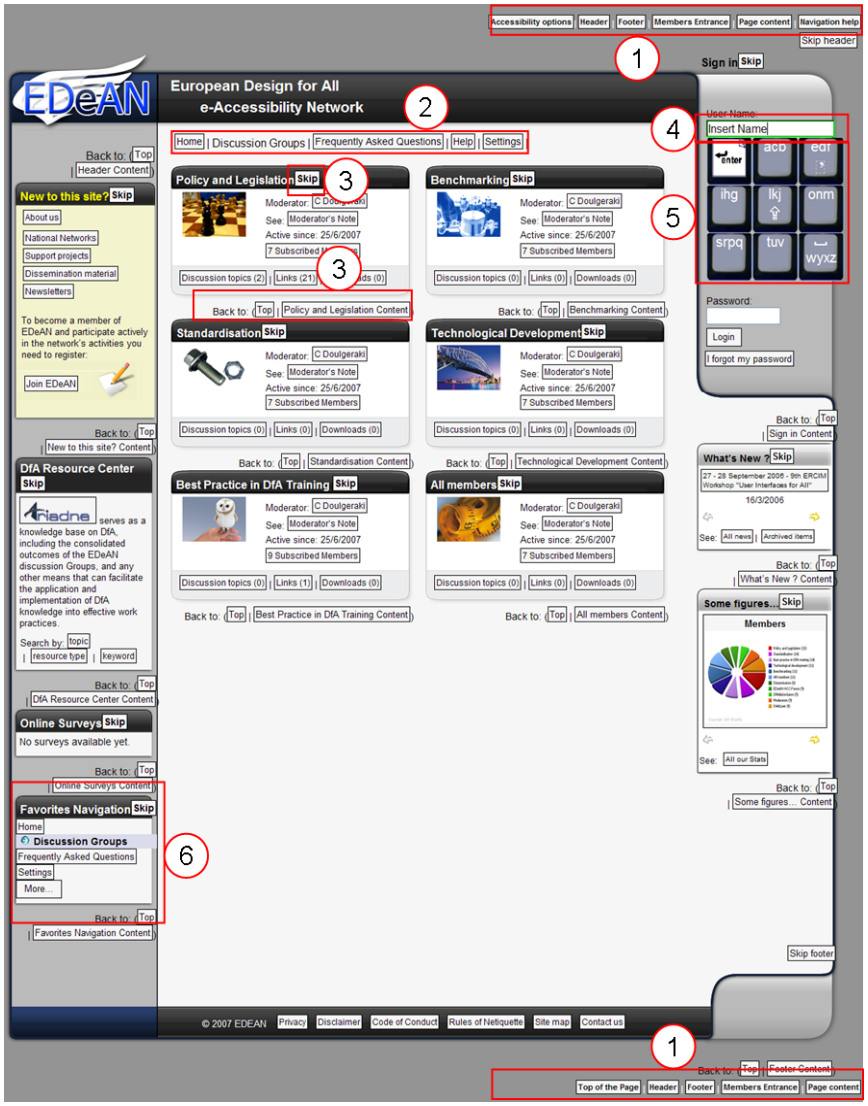


Figure 20: Profile of Motor Impaired user with two Switches and Low Expertise

In this figure, the following adaptations are highlighted:

(1) Various quick access links are presented at the top and bottom of the page allowing in the case of motor impaired users to quickly access various parts of a page reducing the overall scanning effort

(2) Links are displayed as buttons for providing visual clues about the currently focused item

(3) Section breaks are displayed on each page region allowing users with high expertise quickly skip through page sections

(4) Text boxes provide feedback on focus enabling users to quickly identify whether text insertion elements are focused

(5) A software keyboard is provided for text entry for improving the pour text insertion rates resulting from the use of traditional QWERTY based virtual keyboards (reference AUK)

(6) A window with the favorite navigation options is displayed providing access to novice users to their most commonly used navigation options.

Activating the "Colour Blind (Protanope) with Low Expertise" profile results in the interface presented Figure 21.

Figure 21: Colour Blind (Protanope) with Low Expertise

In the figure above, the following adaptations are highlighted:

(1) Links are displayed with pink color while the page background is set to black. These transformations are made for supporting the appropriate

background/foreground scheme (the one that maximizes contrast) for the selected color blindness.

(2) Buttons use yellow color for background, red for border and black for text. This transformation is also employed for maximizing contrast and therefore making buttons easy to spot on the screen

(3) Charts are rendered using an appropriate color palette in order for color blind users to be able to distinguish chart data mainly because their separation is based on color coding.

5. Conclusions and Future Work

This Chapter has presented an overview of popular development methods and tools that support the development of web-based services. In this context, modern programming languages, popular UI toolkits and IDEs were presented, focusing on highlighting their significance in developing more robust and versatile web-based services. At the same time, the main drawbacks of these approaches were also discussed, highlighting the rapidly increasing need to enable seamless access to web-based applications and services to numerous categories of users who face today the risk of exclusion, such as non-expert IT users, the very young or the elderly, people with disability, etc. In this context, **t**his Chapter has proposed a novel approach to the development of web user interfaces that are able to automatically adapt to various user profiles and contexts of use. The proposed approach, intended as an alternative to traditional design for the 'average' user aims to ensure accessibility and usability for users with diverse characteristics. The EAGER toolkit further facilitates Web developers in effectively following the proposed approach in practice. In this context, a number of alternative UI elements were designed and developed addressing specific user- / context- parameter values.

Another key feature of the EAGER toolkit is its ability to be extended and include an unlimited number of alternative interaction modalities and elements. This process mainly entails the design and coding of the alternative interactions styles. Then, they can be easily incorporated in the existing toolkit, simply by modifying the logic for supporting the dynamic instantiation and use. Additionally, existing Web applications or parts of applications implemented with .NET can be easily altered to encapsulate the EAGER toolkit attributes and, thereby, rendered accessible and usable for various

user categories, including novice users, users of Assistive Technologies or portable devices, etc. Notably, it has been estimated that applications implemented from scratch using the EAGER toolkit may contain up to 50% less lines of pure code in total when compared to traditionally developed applications, showing that EAGER generates shorter, more robust and comprehensive source code.

Concerning additional enhancements of the EAGER toolkit, several advanced and intelligent techniques have been identified which can improve its effectiveness and efficiency such as:

- facilities that allow the developer to easily alter the presentation characteristics of a selected UI module (e.g. module skinning);
- incorporation of components written using other programming languages in the context of a task hierarchy (e.g. introduce module styles written in action script)
- extension of the UI library to support new abstract tasks as identified in the context of future case studies
- extension of UI library to address the needs set by new web standards

Overall, the EAGER toolkit is considered as a significant contribution towards supporting the development of WUI that can support the diversity of the target user population in the context of the information society.

References

[1] Ajax.NET - The free library for .NET (C#): http://ajax.schwarz-interactive.de/CSharpSample/

[2] Alexandraki, C., Paramythis, A., Maou, N., & Stephanidis, C. (2004). Web accessibility through adaptation. In Proceedings of the 9th International Conference on Computers Helping People with Special Needs (ICCHP 2004), Paris, France, 7-9 July (pp. 302-309). Berlin Heidelberg: Springer-Verlag.

[3] C# Language Specification Version 3.0

[4] Eclipse Foundation: Eclipse Project. http://www.eclipse.org http://www.eclipse.org/

[5] HTML 4.01 Specification: http://www.w3.org/TR/REC-html40/

[6] The Java™ Language Specification Third Edition

[7] JavaServer Faces Technology: http://java.sun.com/javaee/javaserverfaces/

[8] Gaedke, M., Schempf, D., Gellersen, H. 1999. WCML: An enabling technology for the reuse in object-oriented Web Engineering, in: Poster-Proceedings of the 8th International World Wide Web Conference (WWW8), Toronto, Ontario, Canada.

[9] Getting Started with an Integrated Development Environment (IDE):http://java.sun.com/developer/technicalArticles/tools/intro.html

[10] Microsoft (2007), An Overview of Microsoft® Visual Studio® 2008, White Paper

[11] Microsoft Visual Studio on MSDN: http://msdn.microsoft.com/en-us/vstudio/default.aspx

[12] Microsoft ASP.NET Site: http://www.asp.net/get-started/

[13] Microsoft ASP.NET Web Applications: http://msdn.microsoft.com/en-us/library/ms644563.aspx

[14] Microsoft ASP.NET Overview: http://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx

[15] NetBeans IDE 6.1: http://www.netbeans.org/

[16] NetBeans IDE – Features: http://www.netbeans.org/features/index.html

[17] OSGi Alliance: OSGi Service Platform – Release 4. (2005)

[18] PHP Introduction – Manual: http://www.php.net/manual/en/introduction.php

[19] PHP Hypertext Preprocessor: http://www.php.net/

[20] Savidis, A., & Stephanidis, C. (2004). Unified User Interface Development: Software Engineering of Universally Accessible Interactions. Universal Access in the Information Society, 3 (3), 165-193. (Managing Editor: Alfred Kobsa, University of California, Irvine, USA).

[21] Stephanidis, C. (Ed.), Salvendy, G., Akoumianakis, D., Bevan, N., Brewer, J., Emiliani, P.L., Galetsas, A., Haataja, S., Iakovidis, I., Jacko, J., Jenkins, P., Karshmer, A., Korn, P., Marcus, A., Murphy, H., Stary, C., Vanderheiden, G., Weber, G., & Ziegler, J. (1998) Toward an Information Society for All: An International R&D Agenda. International Journal of Human-Computer Interaction, 10 (2), 107-134.

[22] Stephanidis, C., Paramythis, A., Sfyrakis, M., Savidis, A. (2001). A Case Study in Unified User Interface Development: The AVANTI Web Browser. In User Interfaces for All, Stephanidis, C. (Ed), Lawrence Erlbaum, NJ, 525-568.