D.B.E.

Digital Business Ecosystem

Contract n° 507953

WP20: User Interface

20.1 – Specification and design of user interfaces

Project funded by the European Community under the "Information Society Technology" Programme.

Contract Number: 507953

Project Acronym: DBE

Title: Digital Business Ecosystem

Deliverable N°:20.1

Due date:

Delivery Date:

Short Description: Discussion of the design and implementation of user interfaces for the services in the DBE

Partners owning:

Partners contributed:

Made available to:

Versioning

Version	Date	Name, organisation
0,1	20041014	Juanjo Aparicio, Sun Microsystems

Quality check

1st Internal Reviewer: First name, Name - Organisation

2nd Internal Reviewer: First name, Name - Organisation

Table of Contents

1 Introduction

In order to allow human users to consume computer-based services an interface is needed. Graphical user interfaces (GUIs) date from the 60s[1], and are nowadays the norm for computer users, ranging from PC desktops to workstations to mobile phones.

Design and implementation of GUIs is a time and resource consuming activity, with a small return of inversion. The main reason why this is so is because of GUI complexity. Even a simple-looking GUI is composed of lots of small abstractions: windows, buttons, text fields, menus. The GUI computer model has to represent all these components, plus the relationships among them. Usually this is accomplished by writing programs in one of the available imperative languages, with the aid of some toolkit and supporting libraries. But the level of abstraction of these toolkits is usually too fine grained, and their use requires a great deal of boilerplate.

Coarser grained UI toolkits would allow the programmer to concentrate on the business logic and divert time and thought from the UI design and implementation. This usually comes with an associated cost in UI performance. One size doesn't fit all, and the UI abstractions that fit one UI style or application are poor for other styles or applications. For example, the graphical elements available to HTML authors allow them to represent data in an efficient way, but allow little or no interaction between the user and the application.

[1] http://en.wikipedia.org/wiki/History_of_the_graphical_user_interface

Furthermore, UI design and implementation is a multidisciplinar activity: it requires a designer with notions of ergonomics and graphical design to make the UI be intuitive and easy to use, and to look appealing; and it requires a programmer to implement the required functionality in an efficient way, a process that must be driven by the requirements of the application the UI is designed for.

But good programmers usually make bad designers, and vice-versa. A plausible solution to this problem is to decouple the UI design from the UI interactions with the applications computational model. This has been a design pattern since the days of SmallTalk 80: the MVC[2] pattern (short for Model-View-Controller). The intent of the MVC patter is to facilitate reuse of GUI modules as well as separating the three main parts of a GUI application in order to allow easier maintenance with minimum side effects on modifications.

A second reason not to have UIs implemented in imperative languages is that by doing so usually the platform of choice of the implementor is enforced onto the customers of the UI. For example, a programmer may provide a UI application implemented in Visual Basic. By doing so, any customer not running a Windows operating system on a compatible machine can not execute the application.

By adding an additional layer of abstraction the UI execution environment is decoupled from the specific computing platform. Consider how the development of the UNIX operating system was sped up by the development of the C programming language, which served in the

[2]http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

beginning as a macro language over heavily repeated constructs, that is, a reduction in boilerplate to be explicitly typed in, and an abstraction over the machine code of the platform the OS was being developed for. Ultimately, with the promotion of C to ANSI standard, a program developed in the C programming language could be more easily ported to platforms different from the original supporting platform, that is, where the program where initially developed and expected to run on.

The same notion of a simplified syntax expressing lots of things has been done with regard to UIs in the field of the world wide web, where a few constructs suffice to provide a good-looking interface to an application. A web page looks essentially the same (or at least it should) on different client machines. Most notably, the author of the web page never took care of what machine the page would be viewed from. This was possible because of the additional level of abstraction: the web page describes in a declarative language the elements of the page. The actual rendering of the page takes place on the client side machine, by the virtue of a client program called the web browser. This client program provides the coupling from the abstraction layer that the web page is to the actual implementation details of the client machine architecture.

However, web forms are mostly constrained to a client-server model, in which all the processing power is taken over by the server side. In order to free the server side of having to compute everything the solution is to implement rich clients. In fact, web browsers are rich clients themselves, only constrained to understand an abstraction layer mainly directed towards producing beautiful output, but very limited user interaction: the HyperText Markup Language.

The same notion of an abstraction layer that decouples the user interaction (both output and input) from the application logic can be successfully applied to the design and implementation of DBE user interfaces. To accomplish this, several commercial and non-commercial alternatives exist.

2 UI Frameworks

There exist several UI frameworks at different levels of standardization, and which provide different levels of abstraction.

In the case of the DBE, where widespread adoption is a main goal, standardization as provided by standards bodies is not a guarantee of ease of deployment. More important than using standards is to use standards that are highly available.

For example, SVG has been a W3C recommendation since January 2003. However, not may browsers usually support SVG graphics.

PNG is also a W3C recommendation, this time since October 1996. But the most widely deployed web browser[3] is notorious for its flawed support of PNG image files[4].

The selected framework/s must comply with the following criteria:

- Portability: Be easily redistributable, in terms of redistribution policies, installed size and OS/library dependencies.

- Versatility: Be powerful enough to allow extensibility to build interactive applications, as well as simple client/server forms.

[3] http://www.pcworld.com/news/article/0,aid,116848,00.asp

[4] http://www.libpng.org/pub/png/pngapbr.html#msie-win-unix

- Freedom: Not suffer from vendor lock-in
- i18n: Easy support for internationalization

2.1 XForms

XForms is a W3C recommendation since October 14, 2003. It provides a layer of separation between the form presentation and the form model.

The W3C specifies (http://www.w3.org/MarkUp/Forms/) the key goals of XForms:

- Support for handheld, television, and desktop browsers, plus printers and scanners
- Richer user interface to meet the needs of business, consumer and device control applications
- Decoupled data, logic and presentation
- Improved internationalization
- Support for structured form data
- Advanced forms logic
- Multiple forms per page, and pages per form
- Suspend and Resume support
- Seamless integration with other XML tag sets

In spite of all this, Xforms is still aimed at providing rich interfaces to client-server applications. That is, the processing is still done by the server side. The DBE needs user interfaces for client-server applications, but also for

rich client applications, where as much processing as possible is being carried on by the client side.

There are implementations of XForms clients in Java, for example, XSmiles[5].

Although it can be argued that Java is a platform in itself and, therefore, choosing Java as the programming platform equals ruling out other alternatives and risking a vendor lock-in in the long run, the facts are that any vendor can provide a Java implementation, and that the language and the platform themselves are governed by a community process, although it is true that Sun Microsystems has the ultimate word on it. This, together with the existance of Java implementations for most (if not all) of the DBE consumers/providers computing platforms, make it an at least defendable choice.

2.1.1 Evaluation

- Portability: being a descriptive language, portability is assured. The problem is that it requires a client application that understands the descriptive language. The W3C has listed three fully compliant implementations[6] as of July 8th, 2003, as well as another list of known implementations[7] as of September 1st, 2004.

- Versatility: Provides a conceptual separation between the

[5]http://www.xsmiles.org/

[6]http://www.w3.org/MarkUp/Forms/Test/ImplementationReport.html

[7]http://www.w3.org/MarkUp/Forms/#implementations

view of the form and the data model behind it. Apart from that, the framework is still aimed at providing input/output forms for client/server applications. Little computation is performed on the client side[8]. Interactions that involve complex computations are constrained by the server response time plus two roundtrip times. A possible performance optimization could be to have the server side as a client side application which can receive XForms submissions. This diminishes roundtrip times to local network connection times.

- Freedom: Being a W3C standard, any vendor can implement it. See the paragraph on “Portability” for discussions on available implementations.

- i18n: The XForms standard has explicit support for internationalized applications[9].

[8] http://www.idealliance.org/papers/dx_xml03/papers/03-05-04/03-05-04.html#s2.2.3

[9] http://www.idealliance.org/papers/dx_xml03/papers/03-05-04/03-05-04.html#s2.1.5

2.2 XUL – XML User interface Language

XUL is an XML based framework for the declarative description of GUIs. Full-fledged applications can be constructed with the aid of XUL. XUL plays a role in the development of Mozilla applications similar to the role of the C programming language in the development of the UNIX operating system[10]. For an example of the power of XUL, see any of the Mozilla applications[11].

However, XUL is not a standard[12], but a standards based product of mozilla.org, built for their own necessities. Note, though, that the FireFox web browser and the ThunderBird mail user agent are both XUL applications, which shows what XUL is capable of. The XUL Alliance[13] project contains a list of XUL implementations for several platforms.

Another problem of XUL is that the platform needed to execute XUL applications amounts to several megabytes to be installed in each client. Mozilla is in the process of releasing their Gecko Runtime Environment to allow the easy installation and deployment of XUL-based applications[14]. In the meantime, if you want to use XUL, you

[10] http://cm.bell-labs.com/cm/cs/who/dmr/chist.html

[11] http://www.mozilla.org

[12] http://www.xml.com/pub/a/2001/09/05/xforms.html?page=last

[13] http://xul.sourceforge.net

[14] http://www.mozilla.org/projects/embedding/GRE.html#mozTocId334331

either have to have installed one of the mozilla.org applications, or select one of the open source implementations.

2.2.1 Evaluation

- Portability: The XUL specification seems to be fragmented among implementations with different goals. The originator of the XUL specification is the Mozilla project. Other implementations provide different syntax and different goals.

- Versatility: The Mozilla suite (browser, email client, chat application, ...) is a proof of the versatility of the language regarding the definition of user interfaces. The Mozilla suite is composed of the XUL engine plus the supporting libraries for HTML rendering, JavaScript engine, communications with email backends, etc.

- Freedom: The Gecko Runtime Environment is a product of Mozilla.org. It is the main implementation that supports the full XUL. Luxor is an open source project that tries to "add more value to XUL through a built-in web server, portal engine or template engine, for example.[15]" This is a clear departure from the original standard, and can be seen as a threat to XUL itself in the long run.

- i18n: Mozilla XUL supports internationalization of

[15] http://luxor-xul.sourceforge.net/, paragraph "How does Luxor XUL differ from Mozilla XUL?"

applications through the use of XML entities[16].

[16]http://xulplanet.com/tutorials/xultu/locale.html

2.2.2 Thinlet

It has been shown that the XUL is not consistently implemented among vendors, and that several subsets exist. One of the most attractive is Thinlet[17]. Thinlet is an implementation of XUL in 38 KBytes of Java classes. The choice of Java as the implementation language forces a choice on the execution environment, but fortunately the Java platform has been ported to all the major hardware platforms, and some of the minor, as well (PDAs, mobile phones).

The advantages of Thinlet are:

- Implementation of XUL in less than 40 KBytes compressed.
- Inherits the good traits of XUL, that is, the UI is expressed declaratively in an XML document.
- The implementation of the actions of the UI is done through Java classes, allowing applications to be run on any platform Java has been ported to.
- Thinlet runs on Java Runtime Environment 1.1 or above.
- Thinlet applications can be embedded as applets.
- Thinlet is licensed under the Lesser GNU Public License[18].

The disadvantages of Thinlet are:

[17] http://xul.sourceforge.net/thinlet.html

[18] http://www.gnu.org/licenses/lgpl.html

- The XUL documents Thinlet understands are not the mozilla.org XUL documents. That means that you can not use Thinlet to execute mozilla.org XUL applications, nor mozilla browsers to execute Thinlet applications.

2.2.3 Evaluation

- Portability: Being a Java application the portability is guaranteed to all those platforms for which a Java platform exists. Thinlet requires a modest Java Runtime Environment version 1.1.

- Versatility: Though much more limited in power than full-fledged XUL implementations like the Gecko Runtime Environment, Thinlet still provides GUIs rich enough for most applications. The main site offers samples[19] in the form of applets. The code to develop for a Thinlet applet is the same as for a Thinlet application, only the packaging details differ. Behavioural extensions are done in the Java programming language.

- Freedom: Thinlet is a product licensed under the LGPL. This means that you can embed it in your own application without the need to change the whole application license, which may be closed source, save by the Thinlet engine itself.

- i18n: Thinlet supports internationalized applications by the means of resource files with the proper messages.

[19] http://thinlet.sourceforge.net/demo.html

2.3 Eclipse platform

The Eclipse platform is “an open extensible IDE for anything and nothing in particular”[20]. Sophisticated GUIs can be constructed with it. The Eclipse Java Development Tools[21] is a Java IDE built around the Eclipse platform. The minimum installable bundle for Linux is 24.4 MBytes in size. The minimum installable bundle for Windows is 24.6 MBytes in size. There are versions for Windows 98, Windows ME, Windows 2000, Windows XP, Linux on x86 with Motif, Linux on x86 with GTK 2, Linux on AMD 64 with GTK 2, Linux on Intel 64 with GTK 2, Solaris 8 on SPARC with Motif, AIX on PowerPC with Motif, HP-UX on HP9000 with Motif and MacOS X. All of them need a Java Runtime Environment 1.4.1 or higher installed to run.

2.3.1 Evaluation

- Portability: The Eclipse platform is available for Windows, Linux on Intel and AMD, Solaris on SPARC, AIX on PowerPC, HP-UX on HP9000 and MacOS X.

- Versatility: The Eclipse platform is an abstraction of the GUI related software of the Eclipse Java IDE. The versatility is out of doubt, as can be seen on the eclipse site[22].

- Freedom: The Eclipse Platform is a product of eclipse.org.

[20] http://www.eclipse.org/

[21] http://www.eclipse.org/jdt/index.html

[22] http://www.eclipse.org/eclipse/presentation/eclipse-slides_files/v3_document.htm

No other implementations are available. The Eclipse platform is licensed under the Eclipse Public License[23].

- i18n: The Eclipse Platform provides facilities to allow the internationalization of extensions[24].

[23] http://www.eclipse.org/legal/epl-v10.html

[24] http://www.eclipse.org/articles/Article-Internationalization/how2I18n.html

2.4 NetBeans platform

The NetBeans platform is a “generic desktop application” which is meant to be extended by the means of platform independent modules[25]. The platform independent bundle weighs 3.7 MBytes compressed. The modules created for the platform are completely platform independent, once it is granted that the NetBeans platform is installed on the target machine.

2.4.1 Evaluation

- Portability: The NetBeans platform requires a Java Runtime Environment 1.3 or greater installed to run. The NetBeans platform itself is otherwise platform independent.

- Versatility: The NetBeans IDE is built on top of the NetBeans platform. As the Eclipse platform, it is versatile enough to build an IDE around it.

- Freedom: The NetBeans platform is a derivative of a product from Sun Microsystems, later licensed as open source software[26]. It is licensed under the Sun Public License[27].

[25]This platform independence is achieved by choosing the Java platform as the execution environment common to different hardware platforms.

[26]http://www.netbeans.org/about/history.html

[27]http://www.netbeans.org/about/legal/spl.html

- i18n: Modules for the NetBeans platform are internationalized in the same way as Java applications are internationalized: by the means of resource files[28].

[28] http://www.netbeans.org/download/dev/javadoc/OpenAPIs/org/openide/doc-files/i18n-branding.html

3 B9 – Use cases

(Content to be provided by FZI)

4 C16 – Interface specification

(Depends on data from task B9 – this is the critical path)