A User Adaptable User Interface Model to Support Ubiquitous User Access to EIS Style Applications

Jon Davis, Andrew Tierney, Elizabeth Chang
Curtin University of Technology, Curtin Business School, Bentley, 6102, Australia
Jon.Davis@cbs.curtin.edu.au, Andrew.Tierney@cbs.curtin.edu.au,
Elizabeth.Chang@cbs.curtin.edu.au

Abstract

The user interface for computer applications is typically hard coded and offers minimal flexibility for user customisation. In this paper we list and discuss several requirements for application user interfaces that offer; widespread application accessibility, access across international boundaries, and that support user interface individualisation by normal users (without the need for specific software customization by developers). We review related works in XML based user interface deployment and identify shortcomings, and present a solution that we name the Web service based Adaptable User Interface (WAUI). The major features of the WAUI are; the decoupling of the traditional user interface from both the execution platform and the rest of the application layers by the use of web services, and providing for an automated user interface adaptation capability based on; the user's selected cultural and internationalization options, the user's preferences for alternative visual display objects, and the preferences that a user defines for modifying the function and layout of application forms and display objects.

1. Introduction

User interaction within computer applications is largely a one-way process where each application is coded with a fixed user interface (UI) structure requiring a correspondingly fixed sequence of actions and behaviour from the users. Modifying these user workflows in applications remains a complex, time-consuming and laborious task that typically involves an instantiation of a popular software development life cycle [1] by the software developer in order to effect the required change in functionality.

An alternative strategy to increase application availability is the modelling of the user interfaces, with separation of the user interface layer from the other layers of the application. While this concept is not novel and has been receiving increasing attention in recent years, technologies such as XML and Web Services have increased the opportunity to realize concrete benefits of such strategies which are being actively pursued by endeavours such as XAML [2], XUL [3], Xforms [4] and UIML [5]. These efforts provide a sound basis for providing technical solutions for user interface separation but they do not address other efficiency and accessibility issues that need to be considered for widespread ubiquitous access to application user interfaces.

We list the following general requirements that we believe must be achieved in order to provide for application user interfaces that support a high level of accessibility to all users, and support a high degree of group individualization plus individualized user interaction without the need for specific software customization by software developers:

A. Define a standard language structure for the definition, update and manipulation of the visual user interface elements and other displayable content.

Without a progressive merging towards a standard UI language, the majority of applications will not be available to all users - via the use of a UI processor that accepts a universal UI language, all such applications can be accessed by all available users and user groups.

B. Utilise a common and standard communications protocol between the user interface execution platform and the application logic platform.

The separation of these application layers provides maximum availability of applications to users. Compatibility and processing requirements of the entire application in terms of user interface, application logic and database layers often preclude application deployment to many platforms, of which the popularity yet limitations of the modern Personal Digital

Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05)
0730-3157/05 $20.00 © 2005 **IEEE**

Assistant (PDA) and mobile phone so readily demonstrate. By allowing a richly featured user interface (richer than HTML) to operate on all computing platforms, yet communicate to other platforms (if and as required) for other components of the application provides full application distribution capabilities. The use of XML and Web Services is an obvious candidate to provide the communication services.

C. Fully support the nominated cultural and language variations of international users.

The vast majority of application software has been hard-coded for specific languages and cultures which severely limits the adoption of applications for use in and by other nations and cultures. The utilisation of universal character encoding, automated translation and multi-language conversion increases accessibility to all user groups.

D. Allow the user to customise the common "look and feel" of the standard user interface objects, and to specify preferred or alternate compatible user interface objects.

The design of a user interface is typically determined and hard-coded by the application developer and cannot usually be changed. Many of the common user interface objects and design metaphors that are implemented are functionally synonymous and thus could permit a user or user group to determine their own UI display preferences to suit individual requirements. This can result in improved UI interaction performance efficiencies by matching the user interaction method to personal behaviours and working environments.

E. Allow applications to identify user interface elements that are permitted to be modified by the user within the user interface to provide local customisation and user individualization of the application – many applications, particularly large and complex Enterprise Information System style applications aim "to be all things to all users". These applications can become very confusing to users and user groups that do not require particular functions in their work role or organisation. By defining non-core application objects and providing a mechanism whereby users can either modify or disable non-core UI features, users and user groups can streamline and optimise the application for their localised requirements.

In summary, these requirements for ubiquitous access can be stated as:

- Define the UI in a standard language to allow UI execution on the majority of platforms,
- Communicate with the application logic layer using standard methods to permit UI access from the majority of platforms,
- Generate UI displays that operate in any orientation and language,
- Allow users to choose their own look and feel of the UI objects,
- Allow users to modify or discard non-core UI components to suit their local requirements.

This paper reviews various user interface distribution methods currently in use, identifies their suitability in satisfying the above requirements for adaptable user interfaces and presents a model for providing an adaptable user interface capable of providing a significant improvement for the ubiquitous accessibility of applications.

2. Related Works

In this section we provide a brief summary of the primary user interface distribution methods that are in use or development as a review of how well the proposed requirements (from the introduction) are satisfied by existing methods.

2.1. HTML, XML and the World Wide Web

The emergence and acceptance of HTML since 1989 provides an obvious candidate for a standard UI language specification [6]. Indeed, it would be difficult to find a modern computing platform that does not support a version of the HTML specification. Accordingly, great efforts have been expended by application developers to provide HTML based user interfaces to their applications which has thus provided for a higher level of accessibility to their applications. However, it has also become widely recognised that the HTML standard does not provide a suitably "rich" user interface experience for users without the use of additional elements such as ActiveX and Java components that then re-introduce platform specific compatibility requirements and does not satisfy the standard language requirements (item A from the introduction).

In 1991, the Uni-Code Consortium was formed to promote a worldwide character encoding standard that "provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language" [7]. Support for Uni-Code adoption and usage has been fairly slow although it is now provided for in most platforms and development tools. However, Uni-Code is merely a provider of the correctly identified character – it has no support for the character context i.e. how the character is initially identified or used. What is required to provide full internationalisation of text in applications

are translation mechanisms (either lookup or automated) that are used in conjunction with the correct orientation of the user interface display.

Apart from HTML, one of the most important technologies to have emerged from the web is eXtensible Mark-up Language (XML) which is a text based language for the classification of data although XML can also define how the data should be processed. Originally designed in 1996 XML is rapidly becoming the standard for data representation and portability between systems [8].

With a conception emerging from 1999 as a solution to enabling the transfer of XML data, Web Services [9] is also rapidly progressing as a standard for facilitating interoperation between applications regardless of their location on the connected network or the host platform [10].

The utilisation of an XML based language structure for UI definition has been adopted for several high profile industry initiatives such as; Microsoft in their recently prepared specification for XAML [2], a mark-up language to define their next generation of user interface; the open source community has been working on XUL (Xml User interface Language) [3]; the World Wide Web Consortium is working on Xforms as the next generation of web forms [4]; and Harmonia Inc which has been working on the definition and implementation of UIML (User Interface Mark-up Language) [5].

These initiatives utilise XML-based user interface specifications but to only limited platforms and require currently proprietary support software or direct interfacing which does not fully satisfy the standard language requirements (item A from the introduction) but does represent significant progress towards the stated goals. They do seek to and succeed in providing a better technical solution for presenting user interfaces on their compatible platforms but they do not address other stated issues B to E to provide higher accessibility.

Microsoft's XAML [2] is targeted towards Windows based systems only but its specification does provide for a rich UI feature set. While it is intended for both local and remote UI deployment, as a potential merging of the current competing strategies of thick and thin client solutions, XAML relies on explicit object bindings to application logic objects and thus does not seek to achieve nor provide platform independence. While initially targeted for release with the next version of Microsoft's Windows as "Longhorn" in 2006, XAML support may be released as an earlier standalone enhancement. A number of open source and commercial pre-releases of XAML-based products such as MyXAML [11] and Xamlon [12] have already been released for use by the development community.

The creation of XUL [3] by the open source community is another example of the high quality of open source software that is currently being produced in some sectors. Following the tenets of the popularist open source movement, XUL is supported on a wider range of platforms than XAML – extending from Windows to include Apple Macintosh and various popular strains of UNIX and Linux operating systems. The platform specific runtime support for XUL is provided as part of the Mozilla project in terms of installed versions of their HTML browsers Mozilla or Firefox. Many of the available extensions to these open source browsers are developed in XUL which also utilises direct object bindings to local application objects and is thus platform dependent and does not directly support separation of the UI layer from the application.

W3C's Xforms is a specification for the next generation of web forms. It seeks to extend current web browser form functionality by providing richer forms content, although it does not have to be limited to only browser usage. Xforms does nicely decouple the presentation logic from the application logic, and uses an extensible XML structure, although the communication stream is dependent on the implementation. Improved internationalisation is a stated goal but has been limited to allowing external opportunity to make use of the XML structure [4]. Xforms provides another very good example of defining user interfaces but lacks the capabilities of user and user interface customisation and readily useful internationalisation.

Harmonia's UIML [5] has been under development since 1995 and is in limited commercial use. UIML has progressed to become a proprietary XML based mark-up language with a relatively simple syntax structure which is specifically designed for user interface specification. UIML code is not a generic model of the user interface requirements but rather is a specific rendering of the user interface requirements for a pre-determined specific device. UIML's simplicity is a benefit and it is supported on a large number of platforms although it does not have an option for decoupling via web services, nor does it support specific user customisation or internationalisation options.

While all of these initiatives utilise XML based language specifications and collectively provide rich UI functionality, only Xforms comes close to utilising a communications standard and fully supporting separation of the UI layer from the application logic.

The authors propose the use of Web Services as the most suitable means of providing a common and standard communications protocol between the user interface platform and the application logic execution platform which would therefore make the application user interfaces accessible from any device. The combination of XML and Web Services has finally yielded a suitable candidate technology that would satisfy the communications standards requirements (item B from the introduction).

2.2. PDA's, Smart Devices and Mobile Computing

Emerging in the mid-1990's the main contenders in the PDA market are based on either the Palm Operating System (PalmOS) or on the later Microsoft's Windows CE which is usually found as Pocket PC's (although Linux is also available for some systems). The necessarily smaller processing and storage capability, low screen resolution and proprietary operating systems are inhibitors to "real world" application usage on these devices.

Modern mobile phones are also powerful digital computers approaching the processing power of PDA processors and are starting to utilise operating systems based on Sun's Java, Microsoft's Mobile Windows and even Linux, and can also offer many of the applications and other benefits of PDA's plus the inclusion of wireless Internet and data access. The typical screen of a mobile phone is even smaller than a PDA and user input is usually provided by the numeric keypad, issues which are severe inhibitors to efficient user interface operation [13].

The desire to develop applications once and deploy them either directly or with minor modification to any computing device is a goal of many computer scientists and commercial organizations [14], [15]. There are many limitations to these goals, not the least of these is the great disparity between the different user interfaces and the availability of processing power and supporting operating systems for these mobile devices. Many efforts are aimed at resolving the issue for the user interface tier of multi-tier applications, and relying on the availability of wireless communications to the application logic and database server layers to complete the full application architecture [5], [16].

2.3. Current Problems that are Restricting User Accessibility

The use of XML to define user interface requirements is rapidly becoming a standard method. However, primary attention is being given to solutions that employ direct binding of application objects rather than pursuing full decoupling between the user interface and application logic layers. There is a much higher focus on the technical deployment of user interfaces to many device types rather than contextual deployment as required for true internationalisation and user individualisation. There is much work proceeding on automating user interface generation [17], [18], [19], [20] rather than on user customisation of the interface.

These advances in technology have progressively contributed to more efficient user interfaces and processes to develop the user interfaces but collectively they fall short of adequately addressing the requirements listed in the introduction:

A. *Standard UI language:* there is no current standard for a user interface language specification although the use of XML or a variation is the thrust of most convergence efforts. The development of applications for access by different platforms and devices is largely a re-development effort (notwithstanding the opportunities that cross-platform applications such as Java can produce).

B. *Common communications between application layers:* there is no current standard and the majority of application development either does not separate the user interface or relies on the use of proprietary and / or direct object interface protocols and methods.

C. *Internationalisation support:* Uni-Code is an available standard and is supported by the major operating systems and development software vendors but has only been used minimally. Typically, developers will re-develop major portions of an application for language specific versions or simply not bother with multi-language versions.

D. *Look and feel:* operating system support provides for only very basic customization of the user interface although Linux and Java offers "themes" [21] designed to simulate the "look and feel" of other operating systems.

E. *Customisation:* configuration and minor customization options are entirely dependent on whether a developer has provided any options for the user but are typically not available to any significant level.

3. Introducing the Web Service Based Adaptable User Interface

To address the problems detailed in Section 2.3 we have developed a solution named the Web service based Adaptable User Interface (WAUI) that

decouples the traditional user interface from both the execution platform and from the rest of the application layers and provides for an automated user interface adaptation capability based on; the user's selected cultural and internationalization options, the user's preferences for alternative visual display elements, and the preferences that a user defines for modifying the function and layout of application forms and display objects.

3.1. Features of the Web Service Based Adaptable User Interface

Standard UI language: By utilizing XML as the structure of the WAUI language specification, interaction with local or remote application logic layers and other applications can be reduced to a simple user interface command syntax mapping issue. In particular, the complete separation of the user interface from the rest of the application provides a ready made user interface solution for any applications based on the Model Based Architecture paradigm [22], [23], [24].

Common communications between application layers: The WAUI exists as a standalone UI layer and utilizes standard Web Services [9] (with an XML [8] based command structure) as the communication protocol to the application logic layer(s). Application logic access from the WAUI is then greatly simplified and the web services technology used is already available for most platforms that support Microsoft's .NET [25], Novell's Mono [26] and Sun's Java which covers the vast majority of computing platforms and is progressively including devices such as digital mobile phones and PDA's, pending full operating system incorporation.

Internationalisation: Access to the full international character set in the WAUI is provided by adherence to the Uni-Code standards. The user interface objects respond to the identified cultural requirements of the visual display such as horizontal and vertical character ordering. The availability of multiple language sets for the text attributes of user interface objects for the application should normally be a provision of the application logic layer – similarly, where a required language set is not available, the use of automated translation services should also be provided by the application logic layer. As an interim measure the WAUI should support the capability to define alternate language sets and interact with defined translation services as an alternate source of text attributes for unknown or unsupported languages.

Look and Feel: The WAUI allows users to define their own "look and feel" in two main ways; by over-riding the default visual appearance of standard objects such as forms layout, buttons, edit boxes etc. This feature should normally be a provision of the operating system and should be supported by the WAUI as an interim feature; and by specifying alternative visual object representation for each visual object type e.g. a user may choose that all radio button objects that are specified by the application logic layer are replaced with an object of their choice that provides similar functionality, say a single selection drop-down list, when it is requested in the user interface. The automated conversion of user interface objects to more space efficient objects can also aid in the deployment of user interfaces to devices with smaller display areas.

Customisation: The WAUI allows for all user interface objects to be tagged with one of the following attributes specifying the level of manipulation that the user is permitted for that object within the WAUI user interface:

"Primary" - where the function or operation of the object cannot be modified. These objects have been identified by the application logic layer as a mandatory requirement hence their function cannot be changed by the user although basic visual attributes such as location on the form can be modified. Objects marked as "Primary" would typically be; database primary keys or mandatory fields, navigation controls that provide access to other forms, and function controls that access important application functionality.

"Optional" - where any aspect of the object can be locally customized with the exception of any schema definition, validation constraints, and the actions of navigation or function controls. The application logic layer does not necessarily require this element to be activated so it can be disabled or discarded by the user, relocated on the form, or its appearance can be redefined in terms of text labels and displayed or edited values and styles if appropriate. Objects marked as "Optional" would typically be; non-mandatory database fields, static display objects such as text and bitmaps, and function controls that access less important application functionality.

"Local" - allows for unlimited local customization of the object. This object can be modified in any way that the user requires. Objects marked as "Local" would typically be elements that had been defined by the user or another local user that extended the base functionality of the application and could be any type of object.

In all of the above cases where the WAUI changes or is requested to change the layout of any object from the original as requested by the application logic layer,

the layout of the current object and any subsequently affected objects (in terms of resizing and positioning) can be performed automatically by the WAUI.

3.2. Operation of the Web Service Based Adaptable User Interface

Any application that interoperates with the WAUI user interface layer requires a supporting local API (Application Programming Interface) or function set that aids in the creation of the XML based WAUI commands (optional but desired) and for issuing the WAUI command via local Web Services (mandatory but typically already available). The authors have developed the WAUI as the front-end to their work on a model driven architecture implementation for Enterprise Information Systems style applications [24].

Local interpretation of the received XML based WAUI commands requires the installation of a local but single WAUI application on the client computer system to receive, process and action the WAUI commands as the application user interface for any number of WAUI based applications. Transport support via standard Web Services is a trivial exercise for most platforms using existing functionality.

The local WAUI application is used as both the rendering and interaction engine for presenting and operating the user interface, plus it acts as the profile editor for users to specify their requirements for their preferred "look and feel" and for specifying customizations to the user interface.

Look and Feel: the local WAUI application allows users to modify the attributes of UI components for both the default attributes or for each specific object by entering a local edit mode. e.g. specify the location, size and style of the selector for radio button objects. WAUI also allows users to set similar preferences for UI component mapping where users prefer to operate with functionally similar components. e.g. choose to specify the use of drop down list box in place of radio button controls for all or specific objects. For that user, the user interface objects from the calling application that have been re-mapped by the user, will be displayed according to their modified definitions. The most common usage of these capabilities is for setting user interfaces on devices that have much smaller visual displays, and for users who have a strong bias to efficient operation with particular user interface objects.

WAUI utilises a local layout manager to automatically re-arrange the display objects after user modification.

Customisation: the local WAUI application also allows users enter an edit mode to select user interface objects that are identified as optional or local and modify these objects in order to develop their own customized user interfaces that can thus remove all unused objects, reorganize display screens to more accurately reflect their own work behaviour and environment, and in general produce a more visually pleasing and thus more efficient work space.

On a corporate scale, these customization options would benefit the simplistic customization of Enterprise Information Systems style applications whereby each corporate customer typically utilizes only a smaller proportion of the available functionality. By choosing to mask non-required features and re-organise the remaining user interfaces to suit local corporate requirements, and at very minimal cost as it does not require a software customization process with the vendor, their users benefit from simpler training and operation.

On a more personalized scale, individual users will often have varying local requirements where they could receive further benefits of customizing their user interfaces to suit their daily, repetitive or other operational workflows.

3.3. Construction of the Web Service Based Adaptable User Interface

There are two pathways to creating the WAUI generation of tools – designing and creating a new WAUI from scratch or by modifying an existing (open source) XML based user interface tool to act as the WAUI system.

To modify a suitable XML user interface system to become a WAUI system requires the following steps; establish a Web Service interface to the XML user interface system to process the WAUI command structure; develop WAUI command mappings to existing XML based user interface system methods, plus integrate new functionality as required; and over-ride any direct binding requirements within the existing XML user interface system methods if they exist and replace with the appropriate WAUI command.

The authors chose the former approach to facilitate the progressive completion of functionality and interface to other model based components under development. Their project is .NET based providing operation of the WAUI on Pocket PC style PDAs via the .NET Compact Framework, with Linux and Apple Macintosh compatibility to be progressively provided by Novell's Mono Project [26].

4. Recapitulation, Conclusions and Future Works

The Web Service Based Adaptable User Interface satisfies each of the requirements as stated in the introduction and provides a solution to the identified problems from section 2:

A. Define a standard language structure for the definition, update and manipulation of the visual user interface elements and other displayable content - by utilising a formally defined and open XML command structure.

B. Utilise a common and standard communications protocol between the user interface execution platform and the application logic platform – by utilising Web Services as the standard transport for the XML based WAUI commands which also allows distributed applications from non-compatible platforms to execute the user interface on any platform.

C. Fully support the nominated cultural and language variations of international users – by the use of Uni-Code to ensure the reproduction of genuine internationalized character sets, by identifying and responding to the required cultural orientation order of the display, and by the choice of enabling language sets for all visual objects and the ability to access automated translation services for unknown or non-specified languages for both data and component object text.

D. Allow the user to customise the common "look and feel" of the standard user interface objects, and to specify preferred or alternate compatible user interface objects – by allowing for the appearance and operation of the operating system provided user interface objects to be modified, and by allowing user specification of compatible alternative user interface objects to customize the users' preferred choice of user interface objects as managed by a local layout manager.

E. Allow applications to identify user interface elements that are permitted to be modified by the user within the user interface to provide local customisation and user individualization of the application – by identifying user interface objects as either "Primary", "Optional" or "Local" and by providing local profile editing and management for the selective local customisation of the user interface objects for the user.

In addition to the benefits listed above, the WAUI also provides:

- A potential new standard for user interface access and separation of the user interface layer from the application logic layer for all platforms, of particular benefit to embedded systems, mobile computers and other smart devices with remote access that typically have little or no local application availability.
- Greatly reduced user workstation deployment effort achieved by removing the requirement for proprietary vendor protocols and client applications - only need to install the WAUI client application.
- Far richer user interface experience than achieved with standard HTML.
- High level support for software toolsets that support aspects of model based computing as per proponents of the Model Driven Architecture (MDA) where the user interface commands of the WAUI can be created and managed directly and automatically by the application logic layer based on its stored model.

Applications based on the WAUI user interface offer widespread accessibility, support individualized user interaction without the need for software customization, and greatly assist and enhance the ability to develop applications once and deploy them either directly or with minor modification to any computing device which is a goal of many computer scientists including the authors.

5. References

[1] Sommerville, I., "Software Engineering", 1st Ed-4th Ed. Addison Wesley, 1989-1998.

[2] Microsoft, "Microsoft Windows Code-Named Longhorn Development Centre", Microsoft Corporation, http://msdn.microsoft.com/longhorn/, 2004.

[3] Mozilla, "XML User Interface Language (XUL)", Mozilla Foundation, http://www.mozilla.org/projects/xul/, 2004.

[4] W3C, "XForms - The Next Generation of Web Forms", World Wide Web Consortium, http://www.w3.org/MarkUp /Forms/, 2004.

[5] Ali, M. F., and Abrams, M., "Simplifying Construction of Multi-Platform User Interfaces Using UIML", UIML Europe 2001 Conference, 2001.

[6] Gribble, C., "History of the Web Beginning at CERN", http://www.hitmill.com/internet/web_history.asp, 2004.

[7] Unicode, "What is Unicode?", Unicode Inc, http://www.unicode.org/standard/WhatIsUnicode.html, 2004.

[8] W3C, “Extensible Markup Language (XML)”, World Wide Web Consortium, http://www.w3.org/TR/WD-xml-961114.html, 1996.

[9] W3C, “Web Services Activity”, World Wide Web Consortium, http://www.w3.org/2002/ws/, 2004.

[10] Florescu, D., Grunhagen, A. and Kossmann, D.: XL, “An XML programming language for web service specification and composition”, Eleventh international conference on World Wide Web, ACM Press Honolulu, Hawaii, USA, 2002, pp. 65—76.

[11] MyXAML, "Welcome to MyXaml!" MyXaml, http://www.myxaml.com/, 2004.

[12] Xamlon, "xamlon: accelarating development", Xamlon Inc, http://www.xamlon.com/, 2004.

[13] Pilioura, T., Tsalgatidou, A. and Hadjiefthymiades, S., “Scenarios of using web services in M-commerce”, SIGecom Exch. 3, 2003, 28—36.

[14] Baar, D. J., Foley, J. D. and Mullet, K. E., “Coupling application design and user interface design”, ACM Press, Monterey, California, United States, 1992, pp. 259—266.

[15] Ciancarini, P., Tolksdorf, R. and Zambonelli, F., “Coordination middleware for XML-centric applications”, 2002 ACM symposium on Applied computing. ACM Press, Madrid, Spain, 2002, pp. 336—343.

[16] Repo, P., “Facilitating user interface adaptation to mobile devices”, Third Nordic conference on Human-computer interaction, ACM Press, Tampere, Finland, 2004, pp. 433—436.

[17] Dewan, P. and Solomon, M., "An approach to support automatic generation of user interfaces." *ACM Trans. Program. Lang. Syst.* 12, 566—609, 1990.

[18] Gajos, K. and Weld, D. S., "SUPPLE: automatically generating user interfaces", in, *Proceedings of the 9th international conference on Intelligent user interface*, ACM Press, Funchal, Madeira, Portugal, pp. 93-100, 2004.

[19] Repo, P., "Facilitating user interface adaptation to mobile devices", in, *Third Nordic conference on Human-computer interaction*, ACM Press, Tampere, Finland, pp. 433—436, 2004.

[20] Stocq, J. & Vanderdonckt, J., "WOLD: a mixed-initiative wizard for producing multi-platform user interfaces", in, ACM Press, Funchal, Madeira, Portugal, pp. 331—333, 2004.

[21] ArtGnomeOrg, “Gnome - Art and Themes”, art.gnome.org team, http://art.gnome.org/faq.php, 2003.

[22] OMG, “About the Object Management Group™ (OMG™)”, Object Management Group, http://www.omg.org /gettingstarted/gettingstartedindex.htm, 2004.

[23] Janssen, C., Weisbecker, A. and Ziegler, J., “Generating user interfaces from data models and dialogue net specifications”, ACM Press, Amsterdam, 1993, pp. 418—423.

[24] Davis, J., Tierney, A. and Chang, E., “Meta Data Framework for Enterprise Information Systems Specification - Aiming to Reduce or Remove the Development Phase for EIS Systems”, 6th International Conference Enterprise Information Systems, Porto, Portugal, 2004, pp. 451-456.

[25] Microsoft, “Microsoft Developer Network”, Microsoft, http://msdn.microsoft.com/, 2004.

[26] Novell, “What is Mono?™”, Novell Inc, http://www.mono-project.com/about/index.html, 2004.