Adaptive User Interface for Web Applications

Hazem M. El-Bakry, Alaa M. Riad, Mohamed Abu-Elsoud, Samaa Mohamed,
Faculty of Computer Science & Information Systems,
Mansoura University, EGYPT
E-mail: helbakry20@yahoo.com

Ahmed E. Hassan, Mahmoud S. Kandel,
Faculty of Engineering, Mansoura University, EGYPT

Nikos Mastorakis
Technical University of Sofia, BULGARIA

Abstract

User interface has becoming an essential part in the design of software applications. In order to make it easy to deal with user's current task, it is required to simplify and optimize the user interface. Therefore, in this paper, a new design for adaptive user interface is presented. Furthermore, several issues of user interface design for web applications are described. In addition, the guidelines and architecture for the design of web applications as well as the nature of the web medium are discussed. The basic web application technologies are reviewed. Moreover, the languages and frameworks used in building user interface of web applications are described. The concept that the future of web2 could bridge the gab between desktop and web applications user interface is achieved. The web as a platform and building a web operating system could enhance the user interface for web applications.

Keywords: Adaptive User Interface, User Interface Design, User Interface Types, Web User Interface, Web Applications, Web Frameworks.

I. Introduction

The user interface (also known as human computer interface or man-machine interface (MMI)) is the aggregate of means by which people—the users—interact with the system—a particular machine, device, computer program or other complex tool. The user interface provides means of:
1. Input, allowing the users to manipulate a system
2. Output, allowing the system to indicate the effects of the users' manipulation.

To work with a system, users have to be able to control the computer and assess the state of the system. For example, when driving an automobile, the driver uses the steering wheel to control the direction of the vehicle, and the accelerator pedal, brake pedal and gearstick to control the speed of the vehicle. The driver perceives the position of the vehicle by looking through the windshield and exact speed of the vehicle by reading the speedometer. The user interface of the automobile is on the whole composed of the instruments the driver can use to accomplish the tasks of driving and maintaining the automobile [82-115].

The term user interface is often used in the context of computer systems and electronic devices where a network of equipment or computers is interlinked through an MES (Manufacturing Execution System--or Host. There is a distinct difference between User Interface or Operator Interface and HMI--Human Machine Interface. An HMI is typically local to one machine or piece of equipment, and is the interface method between the human and the equipment/machine. An Operator interace is the interface method by which multiple equipment that are linked by a host control system is accessed or controlled. The user interface of a mechanical system, a vehicle or an industrial installation is sometimes referred to as the human-machine interface (HMI). HMI is a modification of the original term MMI (man-machine interface). In practice, the abbreviation MMI is still frequently used although some may claim that MMI stands for something different now. Another abbreviation is HCI, but is more commonly used for human-computer interaction than human-computer interface. Other terms used are operator interface console (OIC) and operator interface terminal (OIT). However it is abbreviated, the terms refer to the 'layer' that separates a human that is operating a machine from the machine itself [43-45].

User interfaces are considered by some authors to be a prime ingredient of Computer user satisfaction.

In science fiction, HMI is sometimes used to refer to what is better described as direct neural interface. However, this latter usage is seeing increasing application in the real-life use of

(medical) prostheses—the artificial extension that replaces a missing body part (e.g., cochlear implants).
The system may expose several user interfaces to serve different kinds of users. For example, a computerized library database might provide two user interfaces, one for library patrons (limited set of functions, optimized for ease of use) and the other for library personnel (wide set of functions, optimized for efficiency).
In some circumstance computers might observe the user, and react according to their actions without specific commands. A means of tracking parts of the body is required, and sensors noting the position of the head, direction of gaze and so on have been used experimentally. This is particularly relevant to immersive interfaces [43-45].

II. Characteristics of User Interface Design

The design of a user interface affects the amount of effort the user must expend to provide input for the system and to interpret the output of the system, and how much effort it takes to learn how to do this. The user interface has the following characteristics:

1. Usability: is the degree to which the design of a particular user interface takes into account the human psychology and physiology of the users, and makes the process of using the system effective, efficient and satisfying. Usability is mainly a characteristic of the user interface, but is also associated with the functionalities of the product and the process to design it. It describes how well a product can be used for its intended purpose by its target users with efficiency, effectiveness, and satisfaction, also taking into account the requirements from its context of use.
2. Modalities and modes: A modality is a path of communication employed by the user interface to carry input and output. Examples of modalities:
Input — computer keyboard allows the user to enter typed text, digitizing tablet allows the user to create free-form drawing
Output — computer monitor allows the system to display text and graphics (vision modality), loudspeaker allows the system to produce sound (auditory modality)
The user interface may employ several redundant input modalities and output modalities, allowing the user to choose which ones to use for interaction.
A mode is a distinct method of operation within a computer program, in which the same input can produce different perceived results depending of the state of the computer program. Heavy use of modes often reduces the usability of a user interface, as the user must expend effort to remember current mode states, and switch between mode states as necessary [43-45].

III. Types of User Interface

In computer science and human-computer interaction, the user interface (of a computer program) refers to the graphical, textual and auditory information the program presents to the user, and the control sequences (such as keystrokes with the computer keyboard, movements of the computer mouse, and selections with the touchscreen) the user employs to control the program.
Currently (as of 2009) the following types of user interface are the most common:
1. Graphical user interfaces (GUI) accept input via devices such as computer keyboard and mouse and provide articulated graphical output on the computer monitor. There are at least two different principles widely used in GUI design: Object-oriented user interfaces (OOUIs) and application oriented interfaces.
2. Web-based user interfaces or web user interfaces (WUI) accept input and provide output by generating web pages which are transmitted via the Internet and viewed by the user using a web browser program. Newer implementations utilize Java, AJAX, Adobe Flex, Microsoft .NET, or similar technologies to provide real-time control in a separate program, eliminating the need to refresh a traditional HTML based web browser. Administrative web interfaces for web-servers, servers and networked computers are often called Control panels.
3. Command line interfaces, it is one of the sser interfaces that are common in various fields outside desktop computing , where the user provides the input by typing a command string with the computer keyboard and the system provides output by printing text on the computer monitor. Used by programmers and system administrators, in engineering and scientific environments, and by technically advanced personal computer users.
4. Tactile interfaces supplement or replace other forms of output with haptic feedback methods. Used in computerized simulators etc.
5. Touch user interface are graphical user interfaces using a touchscreen display as a combined input and output device. Used in many types of point of sale, industrial processes and machines, self-service machines etc.
6. Attentive user interfaces manage the user attention deciding when to interrupt the user, the kind of warnings, and the level of detail of the messages presented to the user.
7. Batch interfaces are non-interactive user interfaces, where the user specifies all the details

of the batch job in advance to batch processing, and receives the output when all the processing is done. The computer does not prompt for further input after the processing has started.

8. Conversational Interface Agents attempt to personify the computer interface in the form of an animated person, robot, or other character (such as Microsoft's Clippy the paperclip), and present interactions in a conversational form.

9. Crossing-based interfaces are graphical user interfaces in which the primary task consists in crossing boundaries instead of pointing.

10. Gesture interface are graphical user interfaces which accept input in a form of hand gestures, or mouse gestures sketched with a computer mouse or a stylus.

11. Intelligent user interfaces are human-machine interfaces that aim to improve the efficiency, effectiveness, and naturalness of human-machine interaction by representing, reasoning, and acting on models of the user, domain, task, discourse, and media (e.g., graphics, natural language, gesture).

12. Motion tracking interfaces monitor the user's body motions and translate them into commands, currently being developed by Apple.

13. Multi-screen interfaces, employ multiple displays to provide a more flexible interaction. This is often employed in computer game interaction in both the commercial arcades and more recently the handheld markets.

14. Noncommand user interfaces, which observe the user to infer his / her needs and intentions, without requiring that he / she formulate explicit commands.

15. Object-oriented user interface (OOUI)

Computing an OOUI is a type of user interface based on an object-oriented programming metaphor. In an OOUI, the user interacts explicitly with objects that represent entities in the domain that the application is concerned with. Many vector drawing applications, for example, have an OOUI - the objects being lines, circles and canvases. The user may explicitly select an object, alter its properties (such as size or colour), or invoke other actions upon it (such as to move, copy, or re-align it). If a business application has any OOUI, the user may be selecting and/or invoking actions on objects representing entities in the business domain such as customers, products or orders. Jakob Nielsen defines the OOUI in contrast to function-oriented interfaces: "Object-oriented interfaces are sometimes described as turning the application inside-out as compared to function-oriented interfaces. The main focus of the interaction changes to become the users' data and other information objects that are typically represented graphically on the screen as icons or in windows. Dave Collins defines an OOUI as demonstrating three characteristics:

- Users perceive and act on objects
- Users can classify objects based on how they behave
- In the context of what users are trying to do, all the user interface objects fit together into a coherent overall representation.

Jef Raskin suggests that the most important characteristic of an OOUI is that it adopts a 'noun-verb', rather than a 'verb-noun' style of interaction, and that this has several advantages in terms of usability [67-77].

There is a great deal of potential synergy between the OOUI concept and other important ideas in user interface design including:

a. graphical user interface (GUI).

b. direct manipulation interface.

c. interface metaphor.

However there are many examples of user interfaces that implement one or more of those other ideas, but which are not in fact OOUIs - though they are often wrongly labeled as OOUIs. Conversely, there are examples of OOUIs that are neither graphical, nor employ direct manipulation techniques, nor employ strong metaphors. For example, the earliest versions of the Smalltalk programming language had a command line interface that was nonetheless also clearly an OOUI, though it subsequently became better known for its pioneering role in the development of GUIs, direct manipulation and visual metaphors.

Although there are many conceptual parallels between OOUIs and object-oriented programming, it does not follow that an OOUI has to be implemented using an object-oriented programming language.

The guidelines for IBM's Common User Access (CUA), (possibly the most comprehensive attempt at defining a standard for OOUI design) stated that 'while object-oriented programming can facilitate the development of an object-oriented user interface, it is not a pre-requisite. An object-oriented user interface can be developed with more traditional programming languages and tools.

However, there are strong synergies. Larry Tesler, who left Xerox PARC in 1980 to join Apple underlined the relationship: 'Many observers have hypothesized that [the] Smalltalk user interface and the Smalltalk language are separable innovations. Consequently, most systems influenced by the Smalltalk user interface have been engineered without resorting to Smalltalk’s implementation approach. At Apple, after using Pascal to implement six initial applications for Lisa, we discovered compelling reasons to change our programming language to incorporate more ideas from Smalltalk. Lisa applications are now written in the language Clascal, an extension of Pascal featuring objects, classes, subclasses, and procedure invocation by message-passing.

16. Reflexive user interfaces where the users control and redefine the entire system via the user interface alone, for instance to change its command verbs. Typically this is only possible with very rich graphic user interfaces.

17. Tangible user interfaces, which place a greater emphasis on touch and physical environment or its element.

18. Task-Focused Interfaces are user interfaces which address the information overload problem of the desktop metaphor by making tasks, not files, the primary unit of interaction

19. Text user interfaces are user interfaces which output text, but accept other form of input in addition to or in place of typed command strings.

20. Voice user interfaces, which accept input and provide output by generating voice prompts. The user input is made by pressing keys or buttons, or responding verbally to the interface.

21. Natural-Language interfaces - Used for search engines and on webpages. User types in a question and waits for a response.

22. Zero-Input interfaces get inputs from a set of sensors instead of querying the user with input dialogs.

23. Zooming user interfaces are graphical user interfaces in which information objects are represented at different levels of scale and detail, and where the user can change the scale of the viewed area in order to show more detail.

24. Brain–Computer Interface (BCI), sometimes called a direct neural interface or a brain–machine interface, is a direct communication pathway between a brain and an external device. BCIs are often aimed at assisting, augmenting or repairing human cognitive or sensory-motor functions. Research on BCIs began in the 1970s at the University of California Los Angeles (UCLA) under a grant from the National Science Foundation, followed by a contract from DARPA. The papers published after this research also mark the first appearance of the expression brain–computer interface in scientific literature. The field of BCI has since blossomed spectacularly, mostly toward neuroprosthetics applications that aim at restoring damaged hearing, sight and movement. Thanks to the remarkable cortical plasticity of the brain, signals from implanted prostheses can, after adaptation, be handled by the brain like natural sensor or effector channels. Following years of animal experimentation, the first neuroprosthetic devices implanted in humans appeared in the mid-nineties.

BCI versus neuroprosthetics:

Neuroprosthetics is an area of neuroscience concerned with neural prostheses—using artificial devices to replace the function of impaired nervous systems or sensory organs. The most widely used neuroprosthetic device is the cochlear implant, which, as of 2006, has been implanted in approximately 100,000 people worldwide. There are also several neuroprosthetic devices that aim to restore vision, including retinal implants.

The differences between BCIs and neuroprosthetics are mostly in the ways the terms are used: neuroprosthetics typically connect the nervous system to a device, whereas BCIs usually connect the brain (or nervous system) with a computer system. Practical neuroprosthetics can be linked to any part of the nervous system—for example, peripheral nerves—while the term "BCI" usually designates a narrower class of systems which interface with the central nervous system.

The terms are sometimes used interchangeably, and for good reason. Neuroprosthetics and BCIs seek to achieve the same aims, such as restoring sight, hearing, movement, ability to communicate, and even cognitive function. Both use similar experimental methods and surgical techniques [46-67].

25. Organic user interface

An organic user interface (OUI) is a user interface "with non-planar displays that actively or passively change shape via analog physical inputs." OUIs are characterized by displays that can change or take on any shape and their ability to use the display as an input device. The folding camera is an early example of an OUI.

Holman and Vertegaal present three design principles for OUIs:

a. Input Equals Output: In the GUI there is a clear division of input and output. The mouse and keyboard input actions from the user. Based on those actions, output is generated graphically on the screen. A key feature of OUI is that a piece of OLED paper, or any potentially non-planar object for that matter, is meant to input actions from the user and also output them onto the same object.

b. Function Equals Form: The form of an object clearly determines its ability to be used as an input. The statement Function Equals Form emphasizes this dependency on one another. Holman and Vertegaal argue that these two are in fact inseparable and that it is a mistake to try to deny this in any way.

c. Form Follows Flow: This principle states that it is of utmost necessity for OUIs to negotiate user actions based on context. e.g. the ubiquitous 'clamshell' phone, where incoming calls alter the phone's function when opening the phone during an incoming call [78-81].

IV. Design of Adaptive User Interface

User interface design or user interface engineering is the design of computers, appliances, machines, mobile communication devices, software applications, and websites with the focus on the user's experience and interaction. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals—what is often called user-centered design. Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to itself. Graphic design may be utilized to apply a theme or style to the interface without compromising its usability. The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interaction yet also require some unique skills and knowledge. As a result, designers tend to specialize in certain types of projects and have skills centered around their expertise, whether that be software design, user research, web design, or industrial design.

There are several phases and processes in the user interface design, some of which are more demanded upon than others, depending on the project.

1. Functionality requirements gathering – assembling a list of the functionality required of the system to accomplish the goals of the project and the potential needs of the users.
2. User analysis – analysis of the potential users of the system either through discussion with people who work with the users and/or the potential users themselves. Typical questions involve:
   - What would the user want the system to do?
   - How would the system fit in with the user's normal workflow or daily activities?
   - How technically savvy is the user and what similar systems does the user already use?
   - What interface look & feel styles appeal to the user?
3. Information architecture – development of the process and/or information flow of the system (i.e. for phone tree systems, this would be an option tree flowchart and for web sites this would be a site flow that shows the hierarchy of the pages).
4. Prototyping – development of wireframes, either in the form of paper prototypes or simple interactive screens. These prototypes are stripped of all look & feel elements and most content in order to concentrate on the interface.
5. Usability testing – testing of the prototypes on an actual user—often using a technique called talk aloud protocol where you ask the user to talk about their thoughts during the experience.
6. Graphic Interface design – actual look & feel design of the final graphical user interface (GUI). It may be based on the findings developed during the usability testing if usability is unpredictable, or based on communication objectives and styles that would appeal to the user. In rare cases, the graphics may drive the prototyping, depending on the importance of visual form versus function. If the interface requires multiple skins, there may be multiple interface designs for one control panel, functional feature or widget. This phase is often a collaborative effort between a graphic designer and a user interface designer, or handled by one who is proficient in both disciplines.

User interface design requires a good understanding of user needs.

The adaptive user interface can be achieved through a combination of BCI, OOUI, GUI, Intelligent user interface, natural language user interface, voice user interface, text user interface, motion tracking interface, gesture interface, noncommand user interfaces and information fusion.

V. Web User Interface

Here, we are interested in the design of web interface. Nowadays, anyone can use the internet applications very easily. The design of user interface has a great effect on facilitating the interaction of Internet user with web applications. The Internet today has a unique reach—right into the homes of a vast audience worldwide. Some organizations (and individuals) see this medium as a good chance for extending the reach of their computer systems. One popular approach used for such activities is to run an application on a server, using web technology for displaying its user interface remotely. Developing such a web-based user interface can be quite tedious—it is a concurrent, distributed program which has to run in a hostile environment .Furthermore, the platform on which it is implemented (the web) was not originally intended for such usage. User interface is the only way of communication between the computer user & the computer system, a lot of

researchers and developers gave a huge interest to this. User interface is classified as desktop application, web application. The web application user interface is the way through which the web user communicates to the web system. Web applications are essentially client-server applications - there is always a web client. The term "Web application" is now widely used to distinguish a certain type of web site [41].
Several definitions exist—in this paper, a web application is taken to be a possibly complex server application that utilizes web technology in order to deliver its user interface. (See Conallen [1999] for an example of another definition.) [40]. The end-to-end technical architecture of a web application needs to take into account a fairly impressive range of problems (some of which are catered for by the web standards). To name a few: since a web application is run in an unfriendly environment, it typically needs security—both for the server and its clients. Often scalability is of importance, since the potential audience is vast. Of course, the user interface should be of acceptable quality and flexibility. Even in the most collapse case a web application is a concurrent, distributed application, which, to make matters worse, depends on a completely stateless protocol (HTTP). Typically, web applications need to be integrated seamlessly with several back-end systems (or at least present a seemingly integrated front to them). Often database transactions and security realms need to span several such system boundaries. While solutions for each one of these problems exist, the particular implementation of a chosen solutions impact one another. Building user interface for them in this particular context would not seem to be a simple task at all.
The web was conceived as a hyper linked network of information which is embedded in documents using a simple document format that is easily editable and viewable on many different platforms [36]. Since then, the web has enjoyed overpowering success and plays host to many unanticipated uses (and, it could be argued, abuses). The basic architecture supporting this massively distributed system has matured with experience, guided by the Internet Engineering Task Force (IETF) [38] and the World Wide Web Consortium (W3C) [5]. This architecture is enabled by three primary standards: Hypertext Transfer Protocol (HTTP), Uniform Resource Identifier (URI), and Hypertext Markup Language (HTML). Some uses of the web require more than what is provided by traditional web standards.
Indeed, for many, the web has merely become the presentation tier of a multi-tiered corporation of systems [37]. These applications are opportunistic attempts to exploit the large, standardized installed base of the web in our heterogeneous world for the purpose of delivering the UI of a system to a wide audience .The specification of such "web applications" is considered in this study. Also web applications and web frameworks are reviewed. Web frameworks are executable architectural frameworks for implementing web applications—thus closely related to the structure and requirements of a web application. Before digging in the detailed techniques used for web application user interface, the authors briefly discuss the technical issues related to the kinds of web application and how it differs from classical client server application .Web applications, however, are different from traditional client-server applications in a few crucial ways.

VI. User Interface Markup Languages

A user interface markup language is a markup language that renders and describes graphical user interfaces [2]. It is preferred to combine all the user interface Markup Languages [UIMLs] in a simple comparison. This comparison in Table 2 compares the general properties of the different [UIMLs] according the following six criteria's:
Short Description: A short description of what each language is**.**
Publication date: The date when the language published .
Development(Creator): The creator of the language.
License : The License type for the language(ie,open source ,commercial..etc).
Development environment: What tool the developer use to write the language code.
Runtime environment : The runtime environment for the language.
The authors notice from *Table 2* that : Most of UIMLs are based on XML (eXtensible MarkupLanguage),that was designed to be self descriptive, totally extensible, language independent, open standard, platform independent, textual information, supports shareable structure (using DTDs),enables interoperability, XML documents are easily committed to a persistence layer[19]. We can note also that, around 90% of these languages were created after 1998. And About 28% need only an internet browser in the runtime, and othes need specific environment such as ZK framework, Java, PHP, GTK. Around 80% of the UIMLs don't need specific editors to write their code, it could be written in the text editor. Around 45% of them are GPL, LGPL license. Another web application UIML were not

classified at table 2 Are briefly discussed as followed:

XBL: The XML Binding Language (XBL) describes the ability to associate elements in a document with script, event handlers, CSS, and more complex content models, which can be stored in another document. This can be used to re-order and wrap content so that, for instance, simple HTML or XHTML markup can have complex CSS styles applied without requiring that the markup be polluted with multiple semantically neutral div elements [5].

AAIML: A key part in the Universal Remote Console specification is the definition of an XML-based language to convey a UI description from a service or device (target) to the URC. This 'Alternate Abstract Interface Markup Language' (AAIML) must be sufficiently abstract (in terms of modality independence), so that a particular URC device can render the provided UI in its own way, taking advantage of the specific interaction techniques the URC device is capable of. For example, a PDA could render the UI description by using GUI elements (visual) for output, and pointing with a stylus, as well as hand writing recognition for input; a car radio would render the same user interface description auditorially with sound and synthetic speech for output, and speech recognition for input; and a braille note-taker would use its braille output and input capabilities in order to render the very same user interface description tactily [20,22].

AUIML: (Abstract User Interface Markup Language) is 'an XML vocabulary which has been designed to allow the intent of an interaction with a user to be defined.' This clearly contrasts with the conventional approach to user interface design, which focuses on appearance. With an intent based approach, designers are able to 'concentrate on the semantics of the interactions without having to concern themselves with which particular device type(s) need to be supported.' Being an XML vocabulary, AUIML allows device independent encoding of information [20,23].

XIML: (Extensible Interface Markup Language)XIML is an XML-based interface representation language for universal support of functionality across the entire lifecycle of a user interface: design, development, operation, management, organization, and evaluation [20,24].

UsiXML: (USer Interface eXtensible Markup Language) is a XML-compliant markup language that describes the user interface for multiple contexts of use such as Character User Interfaces (CUIs), Graphical User Interfaces (GUIs), Auditory User Interfaces, and Multimodal User Interfaces. In other words, interactive applications with different types of interaction techniques, modalities of use, and computing platforms can be described in a way that preserves the design independently from peculiar characteristics of physical computing platform [20,25].

WSXL: (The Web Services Experience Language) released by IBM, is a Web services centric component model for interactiveWeb applications. It is intended for applications providing a user experience across the Internet [28]. The two goals of WSXL are firstly to give a way to build web applications to a wide variety of channels and secondly to create web applications from other ones. WSXL is built on widely established and emerging open standards, and is designed to be independent of execution platform, browser, and presentation markup languages [21].

XISL: (Extensible Interaction Sheets Language) is a multi-modal interaction description language. It is designed for describing interaction using multi-modal inputs and outputs[21,29].

TADEUS-XML: In a TADEUS-XML description, a UI is made up of two parts: a model component (abstract interaction model), that describes the feature of the UI on a high level of abstraction, and a presentation component [21].

Seescoa XML : (Software Engineering for Embedded Systems using a Component Oriented Approach) is a project that started in October 1999 and has to be finished 382 N. Souchon and J. Vanderdonckt in September 2003, involving a research consortium of four Belgian university partners. The main objective of Seescoa project is to adapt the software engineering technologies to the needs of embedded software [21,26].

Teresa XML: Teresa XML is the XML-compliant language that was developed inside the Teresa project, which is intended to be a transformation-based environment designed and developed at the HCI Group of ISTI-C.N.R (http://giove.cnuce.cnr.it) It provides an environment that supports the design and the generation of a concrete user interface for a specific type of platform [21,27].

VoiceXML: is a markup language for creating voice-user interfaces. It uses speech and telephone touchtone recognition for input and prerecorded audio and text-to-speech synthesis (TTS) for output [34].

VRML:Virtual Reality Markup Language. A scripting language that allows for the creation of three-dimensional "worlds" that the user can explore [5].

MathML: (Mathematical Markup Language), is an XML application for describing mathematical notation and capturing both its structure and content. The goal of MathML is to enable mathematics to be served, received, and processed on the Web, just as HTML has enabled this functionality for text [5].

WebRB: WebRB, an implementation of RBlocks for a Web application environment. It's deployed as a software service; developers run the WebRB visual editor in a standard Firefox Web browser and store their page designs and data on the WebRB server. WebRB is a different way to write Web applications because it uses visual page designs, lacks imperative code, and uses relational semantics [7], that facilitate the user interaction.

WebML: (Web Modeling Language) for designing traditional data-intensive

Web applications suggested us to extend the visual formalism to model relevant interaction and navigation operations typical of Web GIS. The proposed extension consists of a set of content units specifically tailored for GIS concepts and tasks [17].

Harel: A web application user interface specification language [18]

The following section describes the frameworks that could be used in building web applications user interface.

VII. Web Application Frameworks

A web framework is a collection of software components which provides its users with support for developing and executing web-based user interface. In part, web frameworks can be seen as being analogous to interpreters: given a specification of a user interface using a specification technique dictated by the framework, server components of the framework can present the user interface using web technology [30].

Topics related to web frameworks are limited in the academic literature, but be plentiful in industry and open discussion forums. Similarly, the designers of web frameworks not often found their work on existing theory in the literature. This study is an attempt to bridge this gap. It is focused on two aspects of web frameworks:

The specification technique a framework mandates, and how such a specification can subsequently be used to present a user interface via web technology. As part of this study, a survey was conducted of 80 open source web frameworks. Based on the survey, a partial overview of the domain of web frameworks is given, covering what is seen as being typically required of a web framework and covering specification techniques that are used by existing frameworks. Using the web as platform implies adherence to certain (intended) architectural constraints. Web framework designers often strain against these constraints. However, another point of view is to recognize that the success of the web platform is made possible precisely because of its intended architecture. (And the success of the web is surely the principal motivation for using it for remote UI in the first place.)

The design of a web framework for presenting a UI so specified is also proposed.

The purpose of this section is to make a general comparison of all the web frameworks. Table 3, Table 4[appendix A], compares the general properties of the different common web frameworks according the following twelve criteria's:

Language/Technology : The language that the framework is based on .There are about 12 language and technology .About 28% based on PHP ,about 28% based on Java about 14% based on Python, 7% are based on Coldfusion SW, 5% based on .NET , also 5% are based on Ruby , 4% are based on JavaScript ,and about 9 % are based on another languages (smalltalk,scala,lua,MXML,Perl) from these measurements we can note that Java and PHP are widely used in building web frameworks.

AJAX [35]: Whether the framework supports AJAX technique or not. About 86% of them supports AJAX this yielded in building highly interactive web UI, as Asynchronous Web pages, which can change some of their content without reloading the whole page**.**

MVC [31]: the MVC is a design pattern that aims to modularize an application into 3 parts. The Model represents the data for the application; the View represents the presentation of the user interface such as text, checkbox items; and the Controller ties these two together and deals with user input such as keystrokes and mouse movements. MVC is traditionally associated with GUI development**.** About 81% of these frameworks supports the MVC.

MVC Push/Pull [39]: Most MVC frameworks follow a Push-based architecture. These frameworks use actions that do the required processing, and then *"push"* the data to the view layer to render the results, about 45% supports "push "technique. An alternative to this is pull-based architecture, sometimes also called "component-based". These frameworks start with the view layer, which can then *"pull"* results from multiple controllers as needed. In this architecture, multiple controllers can be involved with a single view,about 15% supports "pull" technique and about 20% of them supports both "Push & pull ".

i18n & l10n [32]: means whether the framework supports internationalization and localization or not. Internationalization and localization means of adapting computer software for non-native environments, especially other nations and cultures. Internationalization is the process of designing a software application so that it can be adapted to various languages and regions without engineering changes. Localization is the process of adapting software for a specific region or language by adding locale-specific components and translating text**.** The authors found that about 91% of the frameworks do support "-il8n" (18 letters between the i and the n) , L10N(10 letters between the "l" and the "n in localization about 9%).

ORM [33]: ORM is a programming technique for converting data between incompatible type systems in relational databases and object-oriented programming languages. This creates, in effect, a "virtual object database" which can be used from within the programming language. There are both free and commercial packages available that perform object-relational mapping, although some programmers opt to create their own ORM tools. This shows whether the framework supports Objec-Relational-mapping or not. Authors found that about 90%of them, 10% does not support ORM.

Testing framework: this shows whether the framework perform a test that validates that units of source code are working properly, findings were about 83% do make testing, 17% don't make testing.

DB migration framework : Shows whether the framework contains tools for migrating tables and data from/to all popular databases. Supports Oracle, SQL Server, MySQL, DB2, PostgreSQL and Sybase. Findings were about 31% support DB migration, 69% does not support DB migration.

Security Framework: Shows the level of security does the framework support security or not. About 69% of them support different security levels and 31% don't support security .

Template Framework: If the framework supports a design template such as CherryTemplate , PHPTemplate, Smarty, XTemplate (:about 83% support) ,or not(about 17% don't support.

Caching Framework: Shows whether the framework support caching technique that reduce time and overhead to access the data sources –relational databases of file systems. About 77% do, 23% don't do.

Form Validation Framework(s): It shows whether framework allows to construct HTML forms from data models, and handle the inputted information from the forms seamlessly when interacting with the data-store. About 88% support form validation, and 12% do not support.

VIII. Conclusion

A new design for adaptive user interface has been presented. The new adaptive user interface has been achieved through combination of different types for user interface and information fusion. The new design can be applied for any software including web applications. The impact of web applications force researcher and industries to search for web applications user interface languages and tools. After defining web applications, authors surveys the language and frameworks used to implement we applications user interface. Findings shows that most languages are based on XML. Also web frameworks could play important role in building web application user interface. from these measurements we can note that Java and PHP are widely used in building web frameworks. Based on that survey the web platform is made possible. The design of a web framework for presenting a user interface so specified is also proposed. The future work will be to build an adaptive web user interface components with XUL and compare it with another user interface libraries and investigate the performance.

References:

[1] Open update,"Wasabi Developer Manual", [On-Line]http://docs.wasabidev.org/wasabi_developer_manual/using_xml_reference.php(2003)

[2] Wikipedia The free Encyclopedia, "User-interface-markup-languages", [On-Line] http://en.wikipedia.org/wiki/ user-interface-markup-languages (Accessed in May2008)

[3] Backbase B.V ,"BXML Reference", http://docs.backbase.com/docs/3_3_1/client_3_3_1/BXML%20Reference.pdf (Mar 2007)

[4] w3schools team , [On-Line] http://www.w3schools.com/(2005)

[5]W3C team, [On-Line]http://www.w3.org (2008)

[6] SourceForge, "Comparison-user-interface-markup-languages", [On-Line] http://www.xul.fr/comparison-user-interface-markup-languages.html(2008)

[7] Avraham Leff and James T. Rayfield IBM T.J. Watson Research Center, "WebRB: A Different Way to Write Web Applications", IEEE Computer Society (2008), 1089-780

[8]Wikipedia The free Encyclopedia ,"Web Applications", [On-Line]http://en.wikipedia.org/wiki/Web_application (Accessed in May2008)

[9]Websydian, v5.7 [On-Line Documentation],http://www.websydian.de/(2007)

[10]Adobe Systems Romania ,"How does AJAX WORK", [On-Line] http://www.interaktonline.com/Support/Articles/Details/AJAX (November 10, 2005)

[11] Telerik team, "How does AJAX work" , [On-Line]http://www.telerik.com/products/ajax/how-does-ajax-work.aspx (2007)

[12] San Murugesan, "Understanding Web 2.0 ", IEEE computer society (July 2007), 1520-9202

[13] Farata Systems,AJAX, , [On-Line]http://flexblog.faratasystems.com/?p=106 (October 28, 2006)

[14] Wikipedia The free Encyclopedia,"Web application frameworks", [On-Line]http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks (Accessed in May2008)

[15] Matt Raible ,"Comparing Web Frameworks",Virtuas Open Source Solutions(2006)

[16] Robert H. Halstead, Jr., Mat Hostetter, David A. Kranz," Curl®: A Content Language ", the International Lisp Conference (June 2005)

[17] S. Di Martino, F. Ferrucci, L. Paolino, M. Sebillo, G. Vitiello, G. Avagliano, "A WebML-based Visual Language for the Development of Web GIS Applications", IEEE Computer society (2007)

[18]Lwan Vosloo,"A web application user interface specification language based on the statecharts",University Of Pretoria etd,(2006)

[19] Nazmul Idris ,"Benefits of using XML",http://developerlife.com/tutorials/?p=31 (1999)

[20] Robin Cover,"XML Markup Languages for User Interface Definition", http://xml.coverpages.org/userInterfaceXML.html (Last modified: October 03, 2005)

[21] Nathalie Souchon and Jean Vanderdonckt ,"A Review of XML-compliant User Interface Description Languages", 10th International Conference on Design, Specification, and Verification of Interactive Systems DSV-IS 2003 (June 2003)

[22] Gottfried Zimmermann, Gregg Vanderheiden, and Al Gilman," Universal Remote Console Prototyping of an Emerging XML Based Alternate User Interface Access Standard", The Eleventh International World Wide Web Conference(May 2002)

[23] Roland A. Merrick,"AUIML: An XML Vocabulary for Describing User Interfaces. [Device Independent User Interfaces in XML]",IBM(2001)

[24] Angel Puerta and Jacob Eisenstein,XIML: A Common Representation for Interaction Data. Presented at the Sixth Intelligent User Interfaces Conference (Jan 2002)

[25] Université catholique de Louvain, Belgium , "UsiXML :USer Interface eXtensible Markup Language", [On-Line] http://www.usixml.org/(2007)

[26] K. Luyten, C. Vandervelpen, and K. Coninx. "Adaptable user interfaces in component based development for embedded systems". In Proceedings of the 9th Int.Workshop on Design, Specification, and Verification of Interactive Systems DSVIS'(2002)

[27] Patern`o. F and Santoro. In Ch Kolski and J. Vanderdonckt (Eds.), editors, C. One model, "many interfaces pages 143–154.",Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces CADUI'2002, Dordrecht, 2002. Kluwer Academics Publishers (Valenciennes, 15-17 May 2002)

[28] A. Arsanjani, D. Chamberlain, and et al. "(WSXL) web service experience language Version", [On-Line] http://www-106.ibm.com/developerworks/library/ws-wsxl2/.(2002)

[29]T. Ball, Ch. Colby, P. Danielsen, L.J. Jagadeesan, R. Jagadeesan, K. L¨aufer, P. Matag, and K. Rehor. "SISL: Several interfaces, single logic. Technical report", Loyola University, Chicago(January 6th, 2000)

[30] Wikipedia The free Encyclopedia ,"Web Applications Frameworks", [On-Line]http://en.wikipedia.org/wiki/Web_application_framework (Accessed in May2008)

[31] Primitivetype Team, [On-Line]http://www.primitivetype.com/glossary/mvc.php(2008)

[32] Wikipedia The free Encyclopedia ,"Web Applications Frameworks", [On-Line]http://en.wikipedia.org/wiki Internationalization_and_localization(Accessed in May2008)

[33] Wikipedia The free Encyclopedia ,"Web Applications Frameworks", [On-Line]http://en.wikipedia.org/ Object-relational_mapping (Accessed in May2008)

[34] Peiya Liu and John R. Smith, "VoiceXML and the W3C Speech Interface Framework", IEEE computer society(2003) 1070-986X

[35] Nicolás Serrano and Juan Pablo Aroztegi ,Ajax Frameworks for Interactive Web Apps, IEEE computer scociety(2007) 0740-7459

[36] Berners-Lee,T,"The World Wide Web: Past, Present and future" ,www.w3.org/People/Berners-Lee/1996/ppf.html(1996)

[37] A. Ginige and S. Murugesan, "Web Engineering: An. Introduction," IEEE MultiMedia, (Jan.–Mar. 2001)

[38] IETF teamwork, [On-Line] http://www.ietf.org/(1992)

[39] Kris Thompson, [On-Line forum] http://www.theserverside.com/patterns/thread.tss?thread_id=22143 (2007)

[40] Jim Conallen,Building Web Applications with UML;1 edition [Text book], Wesley Professional Publisher (Dec 1999)

[41] Ahmed E. Hassan, M. Abu-Elsoud, M. S. Kandil, and Samaa Mohamed, " Web Application User Interface, a Survey, "

Mansoura Journal for Computer Science and Information Systems, EGYPT, Vol. 5, No.5, July 2008.

[42] Hazem M. El-bakry, and Nikos Mastorakis “User Interface for Internet Applications,” Proc. of 9th WSEAS International Conference on Applied Informatics and Communications (AIC'09), Moscow, Russia, August 26-28, 2009, pp. 383-392.

[43] User interface design - Wikipedia, the free encyclopedia.

[44] User interface - Wikipedia, the free encyclopedia.

[45] Adaptive user interfaces - Wikipedia, the free encyclopedia.

[46] J. Vidal, "Toward Direct Brain–Computer Communication", in Annual Review of Biophysics and Bioengineering, L.J. Mullins, Ed., Annual Reviews, Inc., Palo Alto, Vol. 2, 1973, pp. 157-180.

[47] J. Vidal, "Real-Time Detection of Brain Events in EEG", in IEEE Proceedings, May 1977, 65-5:633-641.

[48] S. P. Levine, J. E. Huggins, S. L. BeMent, R. K. Kushwaha, L. A. Schuh, M. M. Rohde, E. A. Passaro, D. A. Ross, K. V. Elisevich, and B. J. Smith, "A direct brain interface based on event-related potentials," IEEE Trans Rehabil Eng, vol. 8, pp. 180-5, 2000.

[49] Laura Bailey. "University of Michigan News Service". http://www.umich.edu/news/index.html?Releases/2006/Feb06/r020606a. Retrieved February 6, 2006.

[50] Miguel Nicolelis *et al.* (2001) Duke neurobiologist has developed system that allows monkeys to control robot arms via brain signals.

[51] Baum, Michele (2008=09-06). "Monkey Uses Brain Power to Feed Itself With Robotic Arm". Pitt Chronicle. http://www.chronicle.pitt.edu/?p=1478. Retrieved 2009-07-06.

[52] Fetz E E 1969 Operant conditioning of cortical unit activity Science 163: 955-958.

[53] Schmidt E M *et al.* 1978 Fine control of operantly conditioned firing patterns of cortical neurons Exp. Neurol. 61 349–69.

[54] Georgopoulos AP, Lurito JT, Petrides M, Schwartz AB, Massey JT (1989) Mental rotation of the neuronal population vector. Science 243: 234-236.

[55] Lebedev MA, Nicolelis MA (2006),Brain–machine interfaces: past, present and future. Trends Neurosci 29: 536-546 Loaded 18 October 2006.

[56] G. B. Stanley, F. F. Li, and Y. Dan. Reconstruction of natural scenes from ensemble responses in the LGN, J. Neurosci., 19(18):8036-8042, 1999.

[57] Wessberg J, Stambaugh CR, Kralik JD, Beck PD, Laubach M, Chapin JK, Kim J, Biggs SJ, Srinivasan MA, Nicolelis MA. (2000) Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. Nature 16: 361-365.

[58] Carmena, J.M., Lebedev, M.A., Crist, R.E., O’Doherty, J.E., Santucci, D.M., Dimitrov, D.F., Patil, P.G., Henriquez, C.S., Nicolelis, M.A.L. (2003) Learning to control a brain–machine interface for reaching and grasping by primates. PLoS Biology, 1: 193-208.

[59] Lebedev, M.A., Carmena, J.M., O’Doherty, J.E., Zacksenhouse, M., Henriquez, C.S., Principe, J.C., Nicolelis, M.A.L. (2005) Cortical ensemble adaptation to represent actuators controlled by a brain machine interface. J. Neurosci. 25: 4681-4693.

[60] Serruya M.D., Hatsopoulos, N.G., Paninski, L., Fellows, M.R., Donoghue, J.P., (2002) Instant neural control of a movement signal. Nature 416: 141-142.

[61] Taylor DM, Tillery SI, Schwartz AB (2002) Direct cortical control of 3D neuroprosthetic devices. Science 296: 1829-1832.

[62] Pitt team to build on brain-controlled arm, *Pittsburgh Tribune Review*, 5 September 2006.

[63] YouTube - Monkey controls a robotic arm

[64] Musallam S, Corneil BD, Greger B, Scherberger H, Andersen RA (2004) Cognitive control signals for neural prosthetics. Science 305: 258-262.

[65] Santucci, D.M., Kralik, J.D., Lebedev, M.A., Nicolelis, M.A.L. (2005) Frontal and parietal cortical ensembles predict single-trial muscle activity during reaching movements. Eur. J. Neurosci., 22: 1529-1540.

[66] Huber et al. (2008) Sparse optical microstimulation in barrel cortex drives learned behaviour in freely moving mice. Nature 2008, 451(7174):61-4.

[67] Brain–computer interface - Wikipedia, the free encyclopedia

[68] Nielsen, J., Usability Engineering. 1993, San Francisco: Morgan Kaufmann / Academic Press

[69] Collins, D., Designing Object-oriented User interfaces. 1995, Redwood City, CA: Benjamin/Cummings

[70] Raskin, J., The Humane Interface. 2000, Reading, MA: Addison-Wesley / ACM Press

[71] Constantine, L. and L. Lockwood, Software for use. 1999: Addison-Wesley

[72] Kay, A., The early history of Smalltalk, in History of Programming Languages, T. Bergin and R. Gibson, Editors. 1996, Addison-Wesley / ACM Press: Reading, MA. p. 511-.

[73] IBM, Common User Access - Guide to User Interface Design. 1991, IBM: Cary, North Carolina.

[74] Tesler, L. Object Oriented User Interfaces and Object Oriented Languages. in ACM Conference n Personal and Small Computers. 1983. New York: ACM.

[75] Dave Roberts, Dick Berry, Scott Isensee & John Mullaly, Designing for the User with

OVID: Bridging User Interface Design and Software Engineering MacMillan, 1998.

[76] van Harmelen, M., ed. Object Modelling and User Interface Design. 2001, Addison-Wesley: Reading, MA.

[77] Pawson, R., Naked Objects, Ph.D Thesis, 2004, Trinity College, Dublin, Ireland.

[78] Roel Vertegaal and Ivan Poupyrev, Organic User Interfaces: Introduction, Communications of the ACM 51(6), 26-30, June 2008.

[79] David Holman and Roel Vertegaal, Organic user interfaces: designing computers in any way, shape, or form, Communications of the ACM 51(6), 26-30, June 2008.

[80] Todd Bishop, Here comes Sphere: Microsoft debuts computing in round, July 29, 2008.

[81] Organic User Interfaces- Wikipedia, the free encyclopedia

[82] Soren Lauesen: *User Interface Design, A Software Engineering Perspective*, Addison-Wesley, ISBN 0-321-18143-3

[83] Ben Shneiderman, Catherine Plaisant: *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Addison-Wesley, ISBN 0-321-26978-0

[84] Jef Raskin: *Humane Interface - The New Directions for Designing Interactive Systems*, Addison-Wesley, ISBN 0-201-37937-6

[85] Jennifer Preece, Yvonne Rogers, Helen Sharp: *Interaction Design*, John Wiley & Sons, ISBN 0-471-49278-7

[86] Alan Dix, Janet Finlay, Gregory D. Abowd, Russell Beale: *Human Computer Interaction*, Prentice Hall, ISBN 0-13-046109-1

[87] Donald A. Norman: *Emotional Design: Why We Love or Hate Everyday Things*, Basic Books, ISBN 0-465-05135-9

[88] Donald A. Norman: *The Design of Everyday Things*, Basic Books, ISBN 0-465-06710-7

[89] Steve Krug: *Don't Make Me Think*, New Riders Publishing, ISBN 0-7897-2310-7

[90] Jesse James Garrett: *The Elements of User Experience*, New Riders Publishing, ISBN 0-7357-1202-6

[91] Theo Mandel: *The Elements of User Interface Design*, John Wiley & Sons, ISBN 0-471-16267-1

[92] Jeff Johnson: *GUI Bloopers*, Morgan Kaufmann Publishers, ISBN 1-55860-582-7

[93] Jakob Nielsen: *Usability Engineering*, Morgan Kaufmann Publishers, ISBN 0-12-518406-9

[94] Jakob Nielsen: *Coordinating User Interfaces for Consistency*, Morgan Kaufmann Publishers, ISBN 1-55860-821-4

[95] Joel Spolsky: *User Interface Design for Programmers*, Apress, ISBN 1-893115-94-1

[96] Lon Barfield: *The User Interface: Concepts and Design*, Bosko Books, ISBN 0-9547239-0-2

[97] Jeffrey Rubin: *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*, John Wiley & Sons, ISBN 0-471-59403-2

[98] Deborah Mayhew: *The Usability Engineering Lifecycle - A Practitioner's Handbook for User Interface Design*, Elsevier Books, ISBN 1-55860-561-4

[99] Y. y. Tang, P. S. P. Wang, P. C. Yuen: *Multimodal Interface for Human-Machine Communication*, World Scientific Publishing, ISBN 981-02-4594-7

[100] Ian Horrocks: *Constructing the User Interface with Statecharts*, Addison-Wesley, ISBN 0-201-34278-2

[101] Ian Horrocks (not the one at Oxford): *Constructing the User Interface with Statecharts*, Addison-Wesley, ISBN 0-201-34278-2

[102] David Hu: *Object-Oriented Environment in C++: A User-Friendly Interface*, MIS: Press, ISBN 1-55828-014-6

[103] Constantine Stephanidis: *User Interfaces for All*, Lea, ISBN 0-8058-2967-9

[104] Jenny Le Peuple, Robert Scane: *User Interface Design*, Crucial, ISBN 1-903337-19-4

[105] Dave Collins: *Designing Object-Oriented User Interfaces*, Benjamin Cummings, ISBN 0-8053-5350-X

[106] Susanne Bdker: *Through the Interface: A Human Activity Approach to User Interface Design*, Lawrence Erlbaum Associates, ISBN 0-8058-0570-2

[107] Larry E. Wood: *User Interface Design: Bridging the Gap from User Requirements to Design*, CRC Press, ISBN 0-8493-3125-0

[108] Jane Carey: *Human Factors in Information Systems: The Relationship Between User Interface Design and Human Performance*, Intellect L & D E F a E, ISBN 1-56750-286-5

[109] Ernest Edmonds: *The Separable User Interface*, Academic Press, ISBN 0-12-232150-2

[110] Carl Zetie: *Practical User Interface Design: Making Guis Work*, McGraw-Hill Publishing, ISBN 0-07-709167-1

[111] R. J. Torres: *Practitioner's Handbook for User Interface Design*, Prentice Hall, ISBN 0-13-091296-4

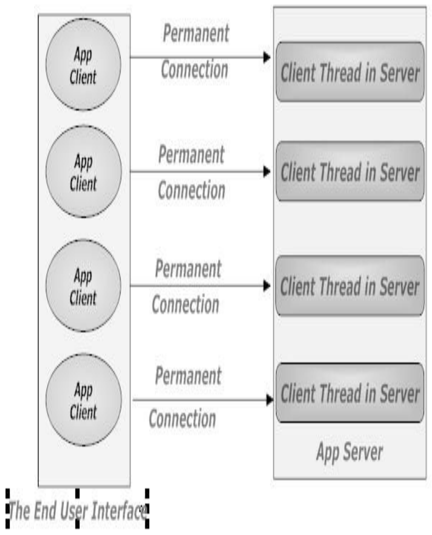
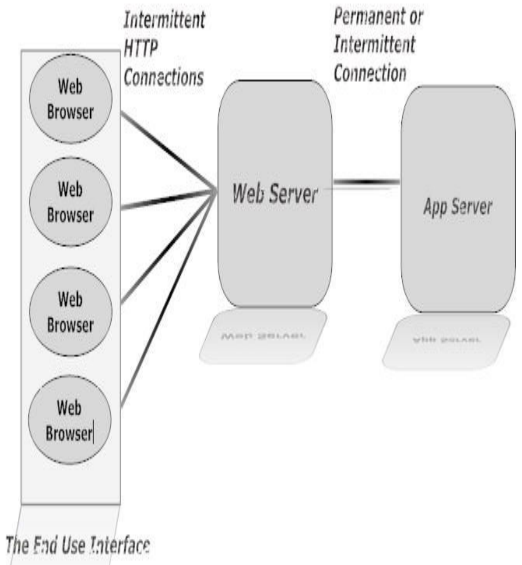
[112] Stuart K. Card, Jock Mackinlay, Ben Shneiderman: *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann (Series in Interactive Technologies), ISBN 1-55860-533-9

[113] QSI Corporation: *Operator Interface Buyer's Guide*, QSI Corporation White Paper

[114] QSI Corporation: *Rugged Operator Interface Terminals: Build Versus Buy*, QSI Corporation White Paper

[115] Torsten Stapelkamp: *Screen- und Interfacedesign*. Springer Science Business+Media, Berlin 2007, ISBN 3-540-32949-8

Table 1: Comparison between Web-Application & Client-Server Application

Client-ServerApplication	Synchronous Web Application
 App Client Permanent Connection Client Thread in Server App Server The End User Interface Figure 1: a) Classic Client-Server Architecture	 Intermittent HTTP Connections Permanent or Intermittent Connection Web Browser Web Server App Server The End Use Interface Figure 1:b) Web Application Architecture
Figure 1-A illustrates how client server applications are usually organized - each user has his own client which maintains a permanent connection to the server which maintains the state of each session by allocating a thread for each client. The thread embodies the state of the session [8,9].	As shown in figure 1-B ,The architecture of Web Applications is somewhat different from that of classic client-server applications. Fundamentally, they all work by receiving requests made by users accessing the application using a Web Browser, and producing Web files (most often HTML Pages) which represent the answer to the requests received. The basic architecture is shown below. Issues related to web application architectures - A Web Browser is used as the Application Client software. - HTML is used by the application to interact with the user. - A Web Server (or, more precisely, a piece of Web Server Software) acts as an intermediary between the Application Client and the Application Server. - There is a many-to-one relationship between the number of Application Clients and the Web Server software; the Web Server acts as a multiplexer/demultiplexer, routing requests from multiple Application Clients to a single Application Server and the corresponding responses back the other way, making sure that requests and responses

	match up correctly [8,9]

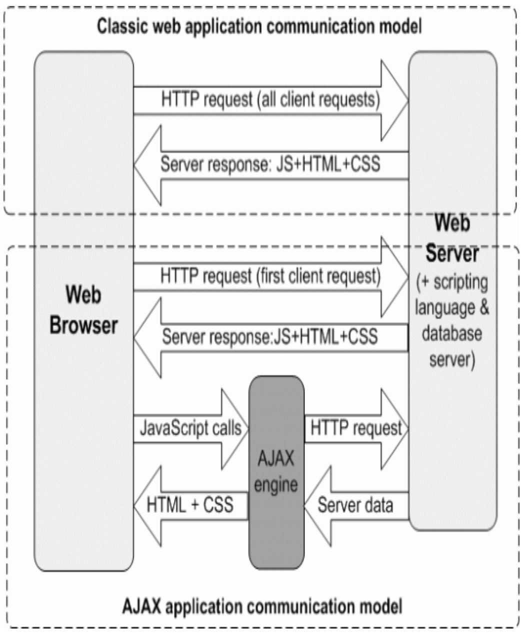
Asynchronous WebApplication
 Classic web application communication model HTTP request (all client requests) Server response: JS+HTML+CSS Web Browser Web Server (+ scripting language & database server) HTTP request (first client request) Server response:JS+HTML+CSS JavaScript calls AJAX engine HTTP request HTML + CSS Server data AJAX application communication model Figure 1: C) Synchronous Web Application Architecture vs. AJAX Architecture
As shown in figure 1-C ,The core idea behind AJAX is to make the communication with the server asynchronous, so that data is transferred and processed in the background. As a result the user can continue working on the other parts of the page without interruption. In an AJAX-enabled application only the relevant page elements are updated, only when this is necessary [10,11,12,13].

Table 2 Basic web application technologies [1,2,3,4,5,6,16,20,21]

UI language	Short Description	Publication date	Development (Creator)	License	Development environment	Runtime environment
BXML	(Binary XML) is an straightforward, open, patent-unencumbered binary-encoding format for XML data that is a stand-alone work-alike drop-in replacement for an XML file that mirrors the XML markup structures in a way that is similar to the in-memory representations of many parser libraries.[**3**]	2003	*Backbase*	LGPL	Text editor / Eclipse / Visual Studio	BPC AJAX
Curl	Is an object-oriented programming language designed to replace HTML, JavaScript, and related tools as a means for creating interactive Web pages.	1990s	David A. Kranz	Not Commercial	text editor	Curl RTE
GladeXML	Graphical user interface builder	April 1998	GNOME	LGPL	Glade	GTK+

Gul2/Xul	A free XUL interpreter	November 2005	redsofa	GPL	Not required (e.g. text editor)	PHP-GTK 2
LZX	Open source platform for rich Internet applications	July 2003	Laszlo Systems	CPL	Not required (e.g. text editor, Eclipse IDE available)	Flash Player 5 or above, DHTML, JavaME announced
MXML	XML-basedintroduced by Macromedia	March 2004	Adobe	MPL	Adobe Flex or free Flex 3 SDK	Flash Player 9 or above
QuiX	Is a javascript engine that renders rich UIs defined in XML and "runs" on the latest builds of Mozilla and Internet Explorer 6.0.	June 2005	inno:script	Commercial	Quill UI Designer	Internet Explorer, Mozilla based browsers
SVG	Scalable Vector Graphics An XML markup language for describing two-dimensional vector graphics, both static and animated	1999	World Wide Web Consortium	W3C License	Not required (e.g. text editor)	Internet Explorer, Mozilla based browsers
UIML	Basically UIML tries to reduce the work needed to develop user interfaces. It allows you to describe the user interface in declarative terms (i.e. as text) and abstract it.	December 1997	OASIS	GNU	Vary	Vary
WasabiXML	XML markup language used to define the graphical interface in Wasabi applications.	2002-2003	Nullsoft	Isn't restricted by GNU	Not required (e.g. text editor)	Internet Explorer, Mozilla based browsers
XAML	is a declarative XML-based language used to initialize structured values and objects	March 2005	Microsoft	Commercial	Not required (e.g. text editor), Microsoft Expression Blend, Microsoft Expression Design, Visual Studio 2008, Vectropy	.NET Framework 3.0 (formerly WinFX), XBAPs for WPF and plugins for Silverlight in internet browsers
XRC	The XML-based resource system, known as XRC, allows user interface elements such as dialogs, menu bars and toolbars, to be stored in text files and loaded into the application at run-time.	1998	wxWidgets	wxWindows Library Licence	Not required (e.g. text editor), wxGlade, XRCed, wxDesigner, DialogBlocks	Vary
XUL	Is an XML user interface markup language developed by the Mozilla project for use in its cross-platform applications, such as Firefox.	December 1998	Mozilla Foundation	GPL / LGPL / MPL	Not required (e.g. text editor)	Gecko-based applications ECMAScript, C++
XAL	eXtensible Application Language	January 2000	Nexaweb	Commercial	Not required (e.g. text editor, Eclipse IDE available)	Java JRE 1.1 and up / MSJVM, DHTML
XForms	XML format for the specification of a data processing model and user interface(s) for the web forms.	March 14th 2006	World Wide Web Consortium	W3C License	Not required (e.g. text editor)	Many- many implementations in browsers, plug-ins, extensions and servers
XFDL	Extensible Forms Description Language	1998	UWI.Com and Tim Bray\	W3C License		UWI.Com's InternetForms Viewer
ZUML	ZK markup language for rich user interface definition	November 2005	Potix	GPL	text editor or Eclipse	ZK Ajax Framework

Appendix A:

Table 3: Frameworks Features comparison [14,15]

Project	Language /Technology	Ajax	MVC framework	MVC Push/Pull	i18n & l10n?	ORM
Ajile	JavaScript	Yes	Yes	Push & Pull	Yes	
Akelos PHP Framework	PHP	Yes, Prototype, script.aculo.us	Yes, Active record pattern	Push	Yes	Yes, Active record pattern
Apache Struts	Java	Yes		Push	Yes	Yes
Apache Struts 2 (ex. WebWork)	Java	Yes		Push & Pull	Yes	Yes
Aranea MVC	Java	Yes		Pull	Yes	Yes
BarracudaDrive LSP	Lua	Yes and JSON-RPC	Yes	Push & Pull	Yes,Limited	Yes,Limited
BFC	.NET	Yes	Yes, but not mandatory	Push & Pull	Yes	Yes, through active data dictionary
CakePHP	PHP	Yes, Prototype, script.aculo.us	Yes, Active record pattern	Push	Yes, Development branch	Yes, Active record pattern
Camping	Ruby	No	Yes	Push	No	Yes, Active record pattern
Catalyst	Perl	Yes, multiple (Prototype, Dojo...)	Yes	Push in its most common usage	Yes	Yes, multiple (DBIx::Class, Rose::DB...)
CherryPy	Python				Yes	
Click Framework	Java	Yes	Yes	Pull	Yes	Yes, integrates with Hibernate and Cayenne
CodeIgniter	PHP	Yes, Framework extension	Yes, Modified Active record pattern	Push	Yes	Yes, framework extension
ColdBox Framework	ColdFusion	Yes, various libraries	Yes	Push & Pull (via Viewlets)	Yes	Yes, Transfer & Reactor
Django	Python	Yes	Yes	Push	Yes	Yes, Django ORM, SQLAlchemy
DotNetNuke	.NET	Yes	No	Pull	Yes	Yes, SubSonic, NHibernate
Drupal	PHP	Yes, jQuery			Yes	No
eZ Components	PHP	No			Yes	Yes
Flex	Actionscript, MXML	No				
FUSE	PHP	Yes	Yes	Push	Yes, custom	Yes
Fusebox	ColdFusion, PHP	Yes	Yes, but not mandatory	Push	No, custom	Yes, via lexicons for Transfer and Reactor
Grails	Groovy	Yes	Active record pattern	Push	Yes	Yes, GORM, Hibernate
Grok (web framework)	Python	Yes,	Yes	Pull	Yes	Yes, OODBMS called ZODB, SQLAlchemy, Storm
JBoss Seam	Java	Yes		Pull	Yes	Yes, JPA, Hibernate
jZeno	Java	Yes	Yes	Pull	No, custom	Yes Hibernate
Kohana	PHP		Yes	Push	Yes	Yes, framework extension
Lift	Scala	Yes	Yes		Yes	Yes
Mach-II	ColdFusion	Yes, via CF or any JavaScript Library	Yes	Push	Yes, via custom plugin	Yes Transfer, Reactor, Hibernate

MonoRail	.NET	Yes, Prototype	Active record pattern	Push	Yes	Yes, Active record pattern
Nitro	Ruby	Yes, jQuery	Yes	Push	Yes	Yes, Og
onTap	ColdFusion	Yes, native features + Prototype + script.aculo.us	Yes, but not mandatory	Push	Yes	Yes
OpenXava	Java	No	Yes, Model Driven		Yes	Yes, JPA, Hibernate and EJB2 CMP
Pal	PHP5	Yes, Ajax Components	Yes	Push and Pull	No, roll your own	Yes, optional Active record pattern, arbitrary SQL
PRADO	PHP5	Yes, Active Controls	Yes	Push	Yes	Yes, Active record pattern, SQLMap
Pylons	Python	Yes, helpers for Prototype and script.aculo.us	Yes	Push	Yes	Yes, SQLObject, SQLAlchemy
Qcodo	PHP5	Yes, built-in	Yes, QControl	Push	Yes	Yes, Code Generation-based
RIFE	Java	Yes, DWR (Java)	Yes	Push & Pull	Yes	Yes
Ruby on Rails	Ruby	Yes, Prototype, script.aculo.us	Yes, ActiveRecord, Action Pack	Push	Yes, Localization Plug-in	Yes, ActiveRecord
Seaside	Smalltalk	Yes, Prototype, script.aculo.us, ...			Yes	Yes, GLORP, Gemstone/S, ...
SilverStripe/Sapphire	PHP 5.2+	Yes, Prototype, script.aculo.us	Yes, Active record pattern	Push & Pull	Yes	Yes, Active record pattern
Spring Framework	Java		Yes		Yes	Yes hibernate, iBatis, etc
Stripes	Java	Yes	Yes	Push	Yes	Yes, Hibernate
Symfony	PHP5	Yes, Prototype, script.aculo.us, Unobtrusive Ajax with UJS and PJS plugins	Yes	Push	Yes	Yes, Propel, Doctrine
Tapestry	Java	Yes	Yes	Pull	Yes	Yes, integrated with Hibernate(tapestry-hibernate module)
Tigermouse	PHP5	Yes, it is mostly Ajax-only framework	Yes, Active record pattern	Push	Yes	Yes, Active record pattern
TurboGears	Python	Yes, MochiKit	Yes	Push	Yes	Yes, SQLObject, SQLAlchemy
web2py	Python	Yes	Yes	Push	Yes	Yes
WebObjects	Java	Yes	Yes	Push & Pull	Yes	Yes, EOF
Wicket	Java	Yes	Modular event driven	Push	Yes	Yes
Widgetplus	JavaScript	Yes it is mostly Ajax-only framework	Yes	Push & Pull	Yes	No,
Zend Framework	PHP5 (>=5.1.4)	Yes, various libraries	Yes	Push	Yes	Yes, Table and Row data gateway
ZK Framework	Java	Yes, 170+ Ajax components	Yes	Push & Pull	Yes	Yes, any ORM frameworks, such as Hibernate, TopLink
Zope2	Python		Yes	Pull	Yes	Yes, native OODBMS called ZODB, SQLObject, SQLAlchemy
Zope3	Python	Yes, via add-on products, e.g. Plone w/KSS	Yes	Pull	Yes	Yes, native OODBMS called ZODB, SQLObject, SQLAlchemy
ztemplates	Java jdk 1.5 or newer	Yes, integrates YUI, Google etc. with annotations	Yes	Push, multiple actions per url	Yes, standard Java	Yes, use any J2EE ORM framework
Lion Framework	PHP	Yes	Yes	Yes Push &	Yes	No

				Pull		

Table 4: Frameworks Features comparison (cont) [14,15].

Ajile	Yes, jsUnit			Yes	Yes	
Akelos PHP Framework	Yes, Unit Tests	Yes		Yes	Yes	Yes
Apache Struts	Yes, Unit Tests			Yes, Jakarta Tiles framework		Yes, Jakarta Validator framework
Apache Struts 2 (ex. WebWork)	Yes, Unit Tests			Yes		Yes
Aranea MVC						
Barracuda Drive LSP	Yes		Yes, ACL-based	Yes	Yes, See CMS for demo	Yes, Limited
BFC	Yes, Unit Tests	Yes, SQL Server, Oracle, DB2, Sybase, MySQL	Yes, security groups and rules	Yes	Yes, metadata and result sets	Yes, data dictionary-driven
CakePHP	Yes, Unit Tests	Yes	Yes, ACL-based	Yes	Yes, Development branch	Yes
Camping	Yes, via Mosquito	Yes	No	Yes	No	No
Catalyst	Yes		Yes, multiple (ACL-based, external engines...)	Yes, multiple (Template::Toolkit, HTML::Template, HTML::Mason...)	Yes, multiple (Memcached, TurckMM, shared memory,...)	Yes, multiple (HTML::FormValidator,...)
CherryPy	No, because unittest and doctest are standard Python modules			Yes, CherryTemplate	Yes	
Click Framework				Yes, Velocity and JSP		Yes, built-in validation
CodeIgniter	Yes, Unit Tests	No	Yes	Yes	Yes	Yes
ColdBox Framework	Yes, Unit Tests	No	Yes, via plugins or interceptors		Yes ColdBox Cache Manager and externally pluggable.	Yes
Django	Yes	No	Yes,	Yes	Yes	Yes

			ACL-based			
DotNetNuke	Yes, Unit Tests	Yes	Yes, ACL-based, (OpenID, LiveID, Active Directory, LDAP, CardSpace, ASP.NET Forms Auth)	Yes	Yes, Pluggable	Yes, ASP.NET Validators, built-in API
Drupal	Yes, simpletest, devel	Yes, Schema API	Yes, multiple (OG, Node Privacy By Role, ACL, Taxonomy Access List)	Yes, multiple (PHPTemplate, Smarty, XTemplate, others)	Yes, multiple (builtin, memcache, APC)	Yes, Form API
eZ Components	Yes	Yes	Yes	Yes	Yes	Yes
Flex	Yes,FlexUnit					
FUSE	Yes, SimpleTest		Yes	Yes	Yes	Yes
Fusebox	Yes, CFUnit, CFCUnit		Yes, multiple plugins available		Yes, via lexicon for ColdSpring	Yes, via qforms or built in cf validation
Grails	Yes, Unit Test	No	Yes	Yes	Yes	Yes
Grok (web framework)	Yes, Unit Tests, Functional Tests	Yes, ZODB Generations	Yes	Yes	Yes	Yes
JBoss Seam	Yes, JUnit, TestNG		Yes, JAAS integration	Yes, Facelets		Yes, Hibernate Validator
jZeno	Yes, JUnit, TestNG		Yes, role-based	Yes, Composite Pattern	Yes	Yes
Kohana	Yes, unit_test module				Yes, File and SQLite Driver based caches	Yes
Lift						
Mach-II	Yes, CFUnit, CFCUnit		Yes, via plugin		Yes, ColdSpring	
MonoRail	Yes, Unit Tests		Yes, via ASP.NET Forms Authenti	Yes	Yes	Yes

			cation			
Nitro	Yes, RSpec	Yes (automatic)	Yes	Yes	Yes	Yes
onTap	Yes, CFUnit, CFCUnit	has potential - supports multiple db platforms	Yes, Members onTap Plugin	Yes	Yes	Yes, client + server
OpenXava	Yes, JUnit	Yes, Hibernate tools	Yes, uses JSR-168 portal security	UI is automatically generated	Yes, uses portal and JPA caching	Yes
Pal	No, still not completed		Yes, via plugins and filters	Yes, via plugins	Yes, allows both memcache and caching pages	Yes, built-in extensible validation
PRADO	Yes, PHPUnit, SimpleTest, Selenium (software)		Yes, modular and role-based ACL	Yes	Yes	Yes
Pylons	Yes, via nose			Yes, pluggable (mako, genshi, myghty, kid, ...)	Yes, Beaker cache (memory, memcached, file, databases)	Yes, preferred formencode
Qcodo		Inherent		Yes, QForm and QControl	Yes	Yes
RIFE	Yes, Out of container testing		Yes	Yes	Yes, Integration with Terracotta	Yes
Ruby on Rails	Yes, Unit Tests, Functional Tests and Integration Tests	Yes	Yes, Plug-in	Yes	Yes	Yes
Seaside	Yes, Unit Tests, SUnit			No, intentionally		Yes, Magritte
SilverStripe/Sapphire	Yes, Unit Tests	Yes (Automatic)	Yes incl. OpenID	Yes (object oriented)	Yes	Yes
Spring Framework			Yes, Acegi	Commons Tiles, velocity etc		Commons Validator
Stripes	Yes		Yes, framework extension	Yes		Yes
Symfony	Yes	Plugin exists (alpha	Yes, plugin	Yes	Yes	Yes

		code, though)				
Tapestry			Yes, tapestry5-acegi library	Yes		Yes, great built-in validation system
Tigermouse	No	No, Multiple RBMSes and access libraries supported	Yes, through intercepting filters (ACL-based, customizable)	Yes	No	Yes
TurboGears	Yes, nose	No	Yes, pluggable authentication providers, user->group<-permissions schema	Yes, pluggable: Kid, Genshi, any Buffet-compatible engine	No	Yes, TurboGears widgets, ToscaWidgets, both utilizing FormEncode
web2py	Yes	Yes	Yes	Yes	Yes	Yes
WebObjects	Yes, WOUnit (JUnit), TestNG, Selenium (software)	Yes, in Project WONDER		Yes	Yes	Yes
Wicket			Yes	Yes	Yes	Yes
Widgetplus	Yes,	No,	Yes	Yes	Yes	Yes
Zend Framework	Yes, Unit Tests	Yes	Yes, ACL-based	Yes	Yes	Yes
ZK Framework	Yes, Unit Tests, Functional Tests		Yes, plugin, like Acegi	Yes, DSP, Velocity, JSP, others pluggable	Yes	Yes, constraint, event validation, others pluggable
Zope2	Yes, Unit Tests		Yes, ACL-based	Yes	Yes	Yes, CMFFormController
Zope3	Yes, Unit Tests, Functional Tests	Yes, ZODB generations	Yes, ACL-based	Yes	Yes	Yes
ztemplates	Yes, Unit Tests		Yes, annotation based	Yes, Velocity, JSP, others pluggable		Yes, ZProperty
Lion Framework	Yes	No	Yes	Yes	Yes	Yes

Appendix B:
Abbreviations & Acronyms

Abbreviations	*Acronyms*
AGPL	The GNU **A**ffero **G**eneral **P**ublic **L**icense
ACL	**A**ccess **C**ontrol **L**ist
AJAX	**A**synchronous **J**avaScript **A**nd **X**ML
ASP	**A**ctive **S**erver **P**ages
Avg	**A**int's **V**ector **G**raphics
BSD	**B**erkeley **S**oftware **D**istribution
BeOS	**Be** Inc
BXML	**B**ackbase e**X**tensible **M**ark-up **L**anguage
CGI	**C**ommon **G**ateway **I**nterface
CSS	**C**ascading **S**tyle **S**heets
CLI	**C**ommand **L**ine **I**nterface
Dlls	**D**ynamically **L**oadable **L**ibraries
DHTML	**D**ynamic **HTML**
GEOS	**G**raphic **E**nvironment **O**perating **S**ystem
GPL	**G**eneral **P**ublic **L**icense
GUI	**G**raphical **U**ser **I**nterface
Html	**H**yper **T**ext **M**arkup **L**anguage
Http	**H**yper **T**ext **T**ransfere **P**rotocol
IIS	**I**nternet **I**nformation **S**ervices
i18n	**18** of letters between the **i** and the **n** in
LGPL	**L**esser **G**eneral **P**ublic **L**icense
L10N	**10** letters between the "**l**" and the "**n**" in localization
MXML	**X**M**L**-basedintroduced by **M**acromedia
MathML	**Math**matical **M**arkup **L**anguage
MIT	**M**assachusetts **I**nstitute of **T**echnology
MVC	**M**odel**-V**iew-**C**ontroller
NeWS	**N**etwork **E**xtensible **W**indow **S**ystem
NLS	**On**-**L**ine **S**ystem
OPL	**O**pen **P**ublication **L**icense
OS	**O**perating **S**ystems
ORM	**O**bject **R**ole **M**odeling
PD	**P**ermitted **D**evelopment
PHP	PHP **H**ypertext **P**reprocessor
SDL	**S**ervice **D**escription **L**anguage
SVG	**S**calable **V**ector **G**raphics
UI	**U**ser **I**nterface
UIML	**U**ser **I**nterface **M**arkup **L**anguage
URL	**U**niversal **R**esource **L**ocator
VBScript	**V**isual **B**asic **S**cript
WMLScript	**W**ireless **M**arkup **L**anguage
WebML	**Web M**arkup **L**anguage for GIS applications
WasabiXML	XML markup language used to define the graphical interface in **Wasabi applications**
XAL	e**X**tensible **A**pplication **L**anguage
XAML	**E**xtensible **A**pplication **M**arkup **L**anguage
XFDL	**Ex**tensible **F**orms **D**escription **L**anguage
XForms	**XML form**at for the specification of a data processing
XHTML	E**X**tensible **H**yper**T**ext **M**arkup **L**anguage
Xml	**E**xtensible **M**arkup **L**anguage
XRC	XML **R**esour**c**e
XUL	XML **U**ser **I**nterface **L**anguage
ZUML	**Z**K **M**arkup **L**anguage for rich **U**ser Interface
ZPL	**Z**ope **P**ublic **L**icense