International Journal of Engineering Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 4 Issue 11|| November 2015 || PP.74-82

User experience Design and its Future optimal approaches

Amritanshu Panigrahi1, Bhupesh deka[2], saswati sahoo3
[1,2]*Associate Professor, Department of Computer Science Engineering, Gandhi Institute For Technology (GIFT), Bhubaneswar*
[3] *Assistant Professor, Department of Computer Science Engineering, Gandhi Engineering College, Bhubaneswar*

Abstract :*Modern business web applications are faced with rapidly changing requirements. Users can choose from a wide variety of systems and have a distinct preference when it comes to usability. The forced or required use of one single system is becoming unacceptable. So are systems with poor user experience, even if the business logic behind it is implemented well. Business users demand apps that are effective, intuitive and efficient. They must have fast performance and 24/7 availability.*

Keywords : *User Interface. Business Logic, Requirement*

I. Introduction :

User Experience (UX) has become the major reason for rejecting a system during end user tests or even worse: after go-live. Users have high expectations, based on the frequent use of social media applications, and expect the same standard for their own business systems. Users expect an easy to use interface, fast interface response time, usage on a variety of different devices, easy login and offline availability. To be able to meet these expectations, software developers require short development cycles and full test coverage to support agile development cycles, seamless support for multiple platforms and devices, secure transactions and easy decoupling from backend systems. And during operations, systems managers, need to be prepared for the unpredictable timing and growth of the visitors of business applications. In some cases the system and hosting platforms need to be able to support a burst in demand or the exponential growth of the user community without drastic changes to the application architecture. This also requires a productive development environment with massive scalability for both the number of developers and eventually the number of concurrent end users. Frameworks with an intrinsic agile capability to modify and expand the functionality with a very short time to market. We feel there is no one-size-fits-all solution for UX requirements. We see a shift from technology derived designs towards user centric designs facilitating every end user with a personalized, timely, effective interface. This kind of approach will lead to more effective, easy to use and enjoyable applications. This whitepaper gives an overview of user experience guidelines. These guidelines translate to additional UX requirements when designing and building a new user interface on modern systems. We will also discuss the two major architectural paradigms for user interface development, followed by an overview of the major frameworks and technologies used for implementing this architecture. Finally we will give a number of business examples and the preferred technology for implementing the requirements. User experience is not only about graphic design. User Experience enables end users to effectively do their job. A good user interface design is a small part of a successful user experience. The interface defines the ‘face’ of the application and the user experience defines the impression the application makes on the end user. Successful user experience leaves a pleasant impression with the user when using the application. A successful UX is often taken for granted, while a bad UX is noticed immediately. Therefore it is advisable to include UX verification and UX testing as a formal activity in your application development process.

User interface design (UID) involves interaction from users with any devices such as mouse and keyboard. It is the foremost and replaceable components of any software [1]. The aim is to produce the interface that is easy to use and easy to understand, which also meet the desires of the future users and provision users in the chores they wish to undertake [2]. UID for Web applications involve users who use the Web to complete required tasks on the Website. Web applications offer the simple interface for accessing Web facilities over the Internet [3]. UID specifies what users need in term of look and feel of a software system and what software engineers understand based on user requirements. However, sometimes software engineers develop user interfaces with little or few supports or guidance from professional user interface designers [4]. Besides, UID is an overwhelming process basic to the achievement of a software system such that planning interactive system, which is attractive, accessible, and easy to use is a challenging task [5]. Occasionally, software engineers do not have enough time to study the UID that also leads to the misunderstanding of what users want. For example, Nasir et al. [6] state that MyEG as one of the G2C Web portal has been overwhelmed with UID problem that leads to complications in using the Web portal and causes the negative perceptions on MyEG UID. UID is the design of interfaces that users can see from any devices such as computers and tablets. Galitz [7] defines UID as "a subset of the field of study called human-computerinteraction (HCI)". UID acts as an intermediary for the user to interact with the system, and it only focuses on interfaces. Thus, UID needs to be understandable and easy to use by people. Good UID will make a mix of well-design input and output mechanisms that free from doubts and meet the user's needs, capabilities and conditions in the best way possible [7]. This paper adopts systematic literature review (SLR) to study the issues and the solutions related to the UID in Web applications. The main goal of this paper is to gather more information that is related to the UID for Web applications. The following Section II and Section III elaborate on the related work and the review process respectively. Section IV reports the result and discussion. Finally, Section V concludes the study and states its future work. II. RELATED WORKS UID should be useful to users. This requires an application that does not just focus on the important parts of users' tasks; it should likewise allow users to interact with the application in ways that are instinctive and normal. Hence, the UID should be simpler yet easy to understand by the users. Friendlier software with constrained abilities is seen to be more usable as the user interface greatly affects the quality of the software product [8]. Furthermore, Web application is a common application to people nowadays. Ginige and Murugesan [9] describe that Web-based applications can be categorized into seven classes as shown in Table 1 in spite of the fact that a given application may belong to more than one class. The UID is important to make a Web application understandable and easy to use regardless of categories. There are more and more applications nowadays are being relocated into Web applications. Thus, the design of the Web application interface should focus on its functionality to offer the simple, intuitive and responsive user interfaces that give the user a chance to complete things with less exertion and Journal of Telecommunication, Electronic and Computer Engineering 58 e-ISSN: 2289-8131 Vol. 9 No. 3-4 time. For example, the work by Zhu et al. [10] portray the elements of a Web application and the UID that develops a particular element by depicting the presentation of input and output parts of the user interface. In addition, Islam and Bouwman [11] state that interface sign is the important elements of Web user interfaces such as navigational links, small images, thumbnails, short text, and button. Some studies explain on behavior change and persuasive factors in HCI related to UID. Most of the works follow the Fogg Behavior Model (FBM) to examine the behavior change and its persuasive factors [12]. FBM is used to define and identify three influences which are motivation, ability, and triggers that control whether a behavior is performed. Thus, the FBM provides the guidance to designers and researchers on how to systematically think about the unexpressed behavior change such as in the existing works [13-15] that adopt FBM. However, such studies are not considered in this SLR as they are more towards HCI issue related to user experience or UX rather than UID. There are limited existing works that use the SLR as their method to review UID issues systematically with different focuses. For instance, the work by Ngadiman et al. [16] discuss the attractiveness and learnability factors in Web applications. The SLR focuses on the issues, proposed work, and strengths and weaknesses of the related work that are limited to the attractiveness and learnability factors in UID. Besides, Islam [17] also uses the SLR as the method to identify the strengths, gaps, and challenges of the related work for the semiotics perception in user interfaces. Moreover, an SLR has also been used to a physical exploration of some related works to identify the criteria and gaps of usability and security in UID respectively [18]. Thus, the SLR in this paper aims to gather information about existing works in the UID for Web applications and specifically in the elicitation process of UID. Eisenstein and Puerta [20] state that the design issues involve the stylistic preference and the flexible standard of achievement as human designers frequently follow the organization and ignoring any strict rule-based processes. This means that the proposed components make design tasks to be difficult to computerize. Hussey [21] highlights that recording and documenting of UID has a deficiency of accurate methodsto be used by implementers and maintainers of the system. The experimentation, exploration and continual assessment are the creative processes that are commonly non-linear and iterative which the modern user interfaces are inexplicitly supported [22]. The user-centered method is still underused and hard to understand by software development teams and organizations as this technique is developed independently Journal of Telecommunication, Electronic and Computer Engineering 60 e-ISSN: 2289-8131 Vol. 9 No. 3-4 from software engineering community [23]. Wenting et al. [24] state that with different background of users having a different cognitive processing that affects the way they use the computer may lead the UID to be a significant issue in information processing system. Besides, Obeidat and Salim [25] report that usability complication is unbearable as it affects the software systems negatively. Janeiro et al. [26]

II. Requirements for modern business UX

The difference between a modern user interface for business environments and for the consumer market is starting to fade. People expect business apps to work in the same way as the systems they use for their personal activities. Thus we can merge the requirements of these two kinds of interfaces.

2.1 User Centric

Most user interfaces are designed around the product or service itself, the supplying organization or the developers of the applications. However, new user interfaces require a user centric approach. The system must be aware of the context, interests, goals and responsibilities of the user and implements this in the user interface. Users demand a my.organization.com environment. An interface aimed at the user and that displays the tasks, products, messages, customers and orders of that particular user. And not just all of the available content, tasks, products, messages and orders. This is vital to new user interfaces: Only show to the information that is important in the context of current specific user.

2.2 Device independent

Users demand applications that operate on all the devices they use and they also want consistency; personalization should apply to each device they use. It should not be limited to the office infrastructure. Working from home and Bring Your Own Device are trends that require systems to operate on a wide variety of devices, browsers and operating systems.

2.3 Fast response

Users expect a timely response of the application. For both the interaction with the user interface and the time it takes to present information from the systems resources (database). Make sure the user gets timely feedback from the application by showing the effect of their actions. By executing the action immediately or by showing a timer or progress bar.. If the application is lacking immediate action feedback, the user will get irritated or confused and submit the same action again and again.

2.4 Scalable

The times when there was a steady and predictable flow of users of your application have gone. Almost all systems must be prepared for peak loads. Due to seasonal influences, additional popularity of certain items or just rising reporting demands. Offering applications via a cloud based infrastructure might resolve some of these peak loads, but the solution is using the appropriate architecture for scalable solutions

2.5 Short development lifecycle

Short time to market and fast response on changing business requirements is essential for a modern user interface. This requires a toolkit with a short development lifecycle, supporting agile software development. The development toolkit needs to support quick development, prototyping, be equipped with a fast test framework and it must be manageable for a large group of developers. The time between development and deployment should be as short as possible.

2.6 Decoupling of back end systems

A user interface has a shorter life cycle than back end systems. The interface has to respond and adapt to fast changing demands of stakeholders and users. The expertise and technology for designing and creating a user interface is also different to that required for developing back end systems. Effective development and fast response to changes is possible when there is a clear decoupling of the user interface from the back end system. This way your web designers can create a user experience without having to bother with the possibilities or limitations of the interface services connecting the back end system.

2.7 Rich user experience

Users expect a rich user experience from (web) applications. This means instant feedback on data manipulation, drag and drop functionality, visualization with extensive charts and (info)graphics and animations to grasp attention and give feedback to the user. The user interface offers information in a more condensed manner and responds to the user by expanding or reducing information based on the users interaction with the system.

2.8 Ability to work offline

Working offline is an often requested feature of user applications. Even with good mobile coverage and availability of wireless connection points there is still a need for working offline. Often to bridge temporal loss of network connection when switching networks or when the user is temporarily in a networking blackspot. Users expect the system to keep working during these moments and catch up automatically when the network connection restores. This means a local copy of the edited data needs to be stored in the client that synchronizes the moment the connection restores. Data security is also important, since users expect their sensitive data to be secure on the local software. The ability to work offline is an architectural requirement. Since user sessions with the system can stop and start at any time, the server needs to be able to handle this.

2.9 Mashability

Usually applications offer the combination, enrichment and filtering of data in the backend system. A Service Oriented Architecture will enable the application to combine several sources to one functional data stream on the server side. In some architectures there is less need for combining this information since the data source is publicly available. You can then use a mashup to enrich data from your private company system with public data. The data is combined in the client, resulting in faster response and reduced load on the server. Examples: combining the visitors address of your customer with public map data (including traffic information); the invoice amount in a default currency combined with public currency rates to display the amount in different currencies or combining private research information with public news sources on the same subjects. A mashup offers the possibility to include real time public data sources in your application and offer extra information to your user. Mashup will not require additional server resources since the mashup is done in the client interface (web browser). An user experience can be composed of several client side mashup applications. These applications can share resources, making it possible to define a uniform skin, language or other preferences.

III. Architecture

When you starting to create an attractive, appealing user interface, you need to consider the best architecture and the best frameworks for your interface. There is no one size fits all solution, no silver bullet framework to support all needs and requirements of a modern demanding user for every type of business, market or application.
Use a customized approach and select the best architecture and frameworks for each situation. However, this takes time and experience.

3.1 Architectural models for modern user experience

There are two architectural models for modern user interface in enterprise applications; thin server and thin client. “Thin” refers to the size of the application and/or the hardware on either the server or the client. Thin server means the user interface is completely created in the client and thin client means most of the user interface is generated on the server. We decided not to discuss a third group of models comprised of scripted or database generated interfaces in this whitepaper. They are not very suitable for an enterprise architecture and they offer less usability features. These interfaces are build in languages like Oracle APEX and PHP. They are perfect for delivering user interfaces in a somewhat straightforward landscape but they do not offer the flexibility, scalability and user experience required for enterprise applications. Therefore, we do not consider them a valid alternative for application development for large scaled enterprise systems..

3.2 Thin Client, Server side processing

Thin client architecture is designed for systems requiring limited intelligence and processing power on the client side. The server generates the complete interface and sends it to the client. The client only has to render the interface. The majority of the processing is done on the server, so that there only limited requirement to the client hardware. This means you can use simple terminals as clients. The server is always aware of the actual state of the user sessions on the client and they can easily be transferred to other terminals. The server has total control over the actions of the user and these can be completely traced since all actions result in a callback to the server. This makes control over complex multi-step transactions very easy. Thin clients work really well when little or no data can be stored on the client and the session of the user needs to be completely controlled by the server. Examples of these frameworks are Spring, JSF PrimeFaces / RichFaces and Oracle ADF Faces.

3.3 Thin Server, Client side processing

Thin server architecture is designed to bring the application as close to the user as possible. Most of the processing is done on the client side (e.g. the web browser of the user or the mobile app). The interface is rendered in the client by combining data with the user interface design and logic. Business logic is brought to the server side and interface logic to the client side. This is less complex since client and server development is done separately; Communication between client and server goes via an independent protocol and does not depend on the architecture of the server (eg SOAP or JSON). The thin server architecture moves session management and page rendering to the client side. This results in a radical reduction of server load and the possibility of massive scalability. In fact, you are utilizing the processing resources of the client and every new user brings hisown processing power (own client machine/browser). Examples of frameworks used to implement this architecture are Bootstrap, AngularJS, Polymer, GWT and Ember.

3.4 Global difference in these models

The difference between these frameworks is the location of the session state and the interface logic. Thin Client architecture keeps the session and logic on the server; Thin Server architecture keeps the logic and session on the client side (the "machine of the user").

The advantages of Thin Server are fast response to user actions, ability to work offline, fine grained control over UI rendering, native CSS control and scalability. Thin server works well for systems with a lot of data read actions, systems with a requirement for fast response time and a fairly unpredictable number of users. The disadvantage of thin server is the abstraction of the server side code towards the frontend developer. The developer is not aware of the structure and capabilities of the backend system. This can also be explained as an advantage since the complexity is shielded from the developer.

The advantages of Thin Client are traceability of the users actions, suitability for complex data manipulations and complete server side control over the users session.

The disadvantage of thin client is server side session management. With a large number of concurrent users all sessions need to be stored on the server. In the case of failure, all session information needs to be transferred to another server within the same cluster. It takes a lot of memory and processing power on the server side.

IV. Use case typologies

Below we describe several use case typologies. For each typology we describe the characteristics of the applications and the preferred general architecture for these kind of applications. Remember these are not actual customer cases, they are virtual, 'ideal' situations.

4.1 Electronic banking application

Banking: The ABC Bank is a large European bank. 80% of its consumer- and 90% business transactions are handled online. Clients look for ease of payment, integration and a multi-channel experience (pc, tablet, mobile). Delivering a secure environment is obligatory. That means that external users cannot modify messages during the transaction process. Users want an interface that is trustworthy and works seamlessly in different browsers, including the less obvious ones. Besides fast and secure transactions the scalability of the services is very important since the application usage van fluctuate drastically during the day. ABC Bank needs to offer a stable and reliable service.

General architecture: the actual transactions are not very complex. The number of different types is limited. Reliability, volume and security are the most important aspects of this system. Most of the business logic resides at the server side or even in the backend system. The logic on the device is fairly simple. To support fast response, a reliable enterprise service bus is used for stable communication between frontend and backend. Due to the vast number of concurrent users, it is advisable to have the session reside on the client. The backend services are hidden via this mechanism. To support a clean frontend structure with fast response, a thin server architectures is most advisable. All security, access and tracing rules are implemented in the service bus and backend.

4.2 News website

Media: Medi@ is a news website where national and international news is combined. The news site has a large amount of daily visitors using PC, tablet and mobile. All news items include photo and video material, and users can contribute to the items by adding comments and media. Content syndication towards other news sites is also offered

Keywords: **fast, multi-channel, multimedia,**

General architecture: After being published, the news articles hardly change. Users read most articles only once. To ensure fast performance, caching of data and styling is important. In this scenario a thin client architecture is the best choice. Content can be cached with styling for different devices or even be perfected for the most frequently used devices. Almost all recent content can be placed as complete rendered pages in the servers memory. Due to SEO considerations it is advisable to keep the pages and structure of the content as stable as possible. Server side control will provide this stability.

4.3 Customer Retail Shop

Consumer Retail: Sphere is an online retail shop that focusses on a complete customer experience. They are not only selling products, but they also focus on aftersales and they give their consumers the opportunity to review products. Social sign is provided and it gives Sphere the opportunity to provide the user with product suggestions. The shop also has social media sharing options. Users use the application at home on their laptop for product browsing and also on their mobile phone to check prices while shopping in a physical shop.

General architecture: Customers of Sphere expect personalized content with push functionality when a newly desired product becomes available. Part of the content is general for all users and part of the content is personalized . The users will use multiple devices to access their accounts. Product prices will also vary between customers depending on their purchase history and loyalty programs. The best architecture Sphere is a combined or hybrid model. The generic information is stored as cached information on the server. The personal information including the shopping cart and wish list will be handled as a thin server application, so that the list is always saved on the server side to provide the same cart when the user logs in with another device. .

4.4 Logistics information provider

ShipIT is a large logistics company located in a seaport. Their monitoring application must be able to show a user friendly overview of all the goods, with different kinds of markings for monitoring purposes. Different departments consult the application for specific data and customers with exclusive access to their own data.It therefore has to be easy to filter data. The number of data entries is huge; ShipIT handles 50-100.000 shipments a day. The information about these packages needs to be accurate and timely. The size of the individual information packages is relatively small and most information requests come from small mobile devices.

4.5 Extension on CRM system such as SAP or JDEdwards

The Holding Inc. is a large conglomerate of construction stores selling to both consumers and building companies. They are based in 8 countries and have a total of 240 stores. The Holding has 8 distinct brands that are independent. All stores are managed by one central JDEdwards implementation that supplies them with logistical information and that reports monthly to the holding. The high-end stores offer additional value to their key customers by offering online services to integrate with their planning system. They also offer specific software to detect planning faults and to prevent ordering of incompatible material within the same project. These services are offered as a web-service and as web page interface. They are custom extension to the existing software.

The most important characteristic of this solution are extendibility, customization and resilience. Software updates of the CRM system should have no effect on the custom service. The architecture of these extensions consists of a service layer responsible for the decoupling of the CRM system and the custom software. The custom software consists of several logical modules providing the additional services. These modules will most likely reside on the server. The provided service layer for valuable customers is created with a webservice implementation such as Oracle SOA Suite or JaxWS. The web pages interface is a thin server implementation based on the same web services. This will facilitate a lean architecture on the server side and fast response for the specific customers.

4.6 Document management

DocIT is a project management organization involved in large projects. For every project they work with a fixed document structure where several employees, with different user rights, have access to during the project. PM is looking for a solution that allows different users to work simultaneously on the documents online, and also offline on their laptop or tablet. They also need to be able to contribute to the documents and make annotations. Some of the projects are under non-disclosure and data cannot leave the (virtual) office unsecured.

The most important characteristics are security and data storage. The number of users is known, so they are able to size the server for this requirement. Need for collaboration and secure storage of data is implemented by server side session management. A thin client architecture is the best option for this requirement .

4.7 Combining information from different sources

ZAPP energy is an energy and gas company providing end user services and supporting transportation. Their website is a mashup application consisting of Twitter and Google Maps. Based on geo tagged twitter massages like #gasleak they show the information on the map. Clients and employees can see this information on their internet device. This information is combined with information on actual planned work on the gas network so the operators of ZAPP are able to analyze if the public notifications correspond with actual planned maintenance work on the gas network.

General Architecture: This solution needs a mashup frontend architecture. Private data about planned maintenance data is combined with public data. The combination of these two sources is designed as a mashup application in the client application. This architecture is thin server.

We made a short study and assessment of several frameworks to be considered when implementing a thin server or thin client architecture. This study can be a starting point for a new application since these technologies are optimized to work together. The frameworks are selected based upon four characteristics.

V. Several frameworks to consider in these architectural models

There is a vast list of available frameworks and development languages. We limited ourselves to those frameworks used for professional enterprise applications. We excluded scripted frameworks, frameworks that are infrequently usage and frameworks that are still in a very immature development stage and. Our list presents frameworks that are suited for our area of expertise: Oracle and Java Technology.

CSS Frameworks
CSS is just styling, no logic or server communication. These CSS frameworks combine HTML5 and CSS technologies in some very useful components. We prefer Bootstrap. Other choices are Topcoat and YUI.

● Bootstrap (OpenSource)
○ Combines HTML + CSS (+ some JavaScript)
○ Supports responsive design for multiple device formats.
○ Easy integration with other frameworks for frontend development.
○ Tailoring with JavaScript for integration with other frameworks.

CSS support: editing and modifying css structures

● LESS or SASS
○ Editors for modifying CSS structures. Configuration of Bootstrap is possible via parameters..

JavaScript libraries

These libraries are especially valuable for filling in the missing links between the JavaScript and other frameworks. They assist developers in fast development and hide the browser specific anomalies found in the use of JavaScript.

● jQuery
○ This is a fairly simple layer on the low-level browser DOM API. JQuery is commonly used as a layer on top of the low level browser DOM-API. The concepts of jQuery are also included in the DART language.
○ jQuery is often used to extend functionality that is not by default included in another framework. ● Alternatives: YUI (JavaScript) Bootstrap (JavaScript)

Client-side development languages

● Dart
○ Also known as the new JavaScript. Is actively developed and promoted by Google as the replacement for JavaScript. Dart needs it's own VM to run the code. A growing number of browsers are supporting native Dart execution. This number is probably going to grow for the upcoming months. Dart has a faster execution (about 5 times faster compared to JavaScript) and is therefore more energy efficient. This will result in a better battery performance for mobile devices. For browsers not able to support native Dart the fallback scenario is to use the JavaScript compiled output of Dart.

Advantages compared to JavaScript:

■ Browser independent;
■ Clear and readable syntax (contrary to JavaScript);
■ Contains the best practice we learned from the using of jQuery;
■ Is, just like Java, very suitable to develop large scaled applications (This is very difficult in JavaScript).

You can recognize most frameworks using DART since they carry the word in the framework name (AgularDart).

● Java
○ Very well know and commonly used (not only for front end technology). Does not execute directly in the browser, only with the use of specific development plugins like Google Web Toolkit (GWT). In GWT the Java code needs to be compiled to JavaScript. This compiled JavaScript is most often better performing compared to home-written JavaScript. But not as good as Dart.
■ Frontend usage in GWT, Errai, Vaadin, etc.
● JavaScript
○ Most of the other front end frameworks use JavaScript.

- ○ Large support from all kinds of browsers for this kind of JavaScript. However there are a lot of differences between browser(versions).
- ○ No good development support. Developing and debugging in JavaScript is complex, error prone and very time consuming.

JavaScript Frameworks
These frameworks are more complete and offer good integration capabilities.

- ● AngularJS
- ○ AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. Angular's data binding and dependency injection eliminate much of the code you currently have to write. AngularJS is executed within the browser, making it an good frontend for any server technology.
- ○ AngularJS has great integration features with CSS frameworks like Bootstrap. Combined with the Bootstrap UI module this is a great combination for seamless integration between these two frameworks. .
- ○ AngularJS supports:
- ■ Dependency Injection
- ■ Modules
- ■ Routing
- ■ Deep-linking
- ■ DOM-based templating (contrary to string-based templating)
- ■ Model-View-Whatever architecture
- ■ "Web Components" extension based upon HTML with a custom syntax.
- ○ Angular is build from a test automation perspective. The support for dependency injection is important. It is rather easy to build test cases for Angular.
- ○ AngularJS is more complex compared to Knockout or Backbone (partly due to the larger set of features and possibilities). However Angular has a large community of users and a lot of examples, questions and answers on stackoverflow.
- ○ Angular developers are actively involved in the development of new HTML5 standards. New versions of AngularJS will support the future standards in HTML development.

AngularDart

- ○ AngularDart is the Dart implementation of AngularJS and is being created by the same team as AngularJS.
- ○ AngularDart is very forward. A lot of the concepts in AngularDart will be ported in the HTML5 Web Component Standards..
- ● Ember.js
- ○ Ember is functionally comparable to AngularJS.
- ○ Ember uses more conventions. Is more leaning towards pure HTML and is as flexible as HTML.
- ● KnockoutJS
- ○ Knockout is mainly a binding framework.
- ○ Unobtrusive.
- ● Polymer
- ○ Polymer is the implementation (polyfill) of new HTML5 Web Components in current standard versions of the browsers not supporting these HTML5 standards.
- ○ Polymer is very strict on follwing the standards. A change in these standards can result in breaking code. Polymer is still in a pre-alpha version.
- ● Vaadin
- ○ Vaadin is a development language to be used as Java code and is linked to the knowledge of the modern Java developer.
- ○ Combined with CSS knowledge this is a rather complete framework.
- ○ Vaadin uses serverside state management.
- ○ Vaadin uses a lot of components using boilerplating. This means the developer needs to create these components first. Java purists are very enthousiastic about this, but this might take too much time during development.
- ○ Vaadin works great for large data sets and supports lazy loading of data.
- ○ Vaadin has a large community and most components are being build in GWT. !
- ○ Other Play 2 and GWT

Other UI frameworks

- ● Oracle ADF

Oracle ADF is a quite extensive framework consisting of both server side and client side processing. In the thin server / thin client architecture ADF uses both of these models. ADF is very suitable for building systems on top of an Oracle database or a Fusion Middleware services layer.

● Node.js

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Conclusion : The paper discussed about the features of developing an USER interface and also the framework available in the market. The paper describes these framework and how to choose a framework depends

References

[1]. W. Jiancheng, L. Xudong, and L. Loi, "A model of user interface design and its code generation," in IEEE International Conference on Information Reuse and Integration, 2007, pp. 28-133.

[2]. D. Stone, C. Jarett, M. Woodroffe, and S. Minocha, User Interface Design and Evaluation. Morgan Kaufmann, 2005.

[3]. K. Sharma, and N. Kumar, "SWART: Secure web application response tool," in International Conference on Control Computing & Materials (ICCCCM) 2013, 2013, pp. 1-7.

[4]. J. Johnson, GUI Bloopers 2.0: Common User Interface Design Don'ts and Dos". Amsterdam: Elsevier Inc., 2008.

[5]. J. C. Quiroz, S. J. Louis, A. Shankar, and S. M. Dascalu, "Interactive genetic algorithms for user interface design," in 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 1366-1373.

[6]. K. Nasir, N. Ariffin, and F. Shuib, "User interface design using cognitive approach: A case study of Malaysian government web portal," in International Conference on User Science and Engineering 2010, 2010, pp. 174-178.

[7]. W. O. Galitz, The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques. Canada: Wiley Publishing, 2007.

[8]. A. Sajedi, M. Mahdavi, A. Mohammadi, and M. M. Nejad, "Fundamental usability guidelines for user interface design," in International Conference on Computational Sciences and Its Application (ICCSA), 2008, pp. 106-113.

[9]. A. Ginige and S. Murugesan, "Web engineering: An introduction," IEEE Multimedia, vol. 8, no. 1, pp. 14-18. 2001.

[10]. B. Zhu, H. Miao, and B. Song, "User interface design of web application using object-z," in Eighth IEEE/ACIS International Conference on Computer and Information Science, 2009, pp. 1200- 1204.

[11]. M. N. Islam and H. Bouwman, "Towards user-intuitive web interface sign design and evaluation: a semiotic framework," International Journal of Human-Computer Studies, vol. 86, 2016, pp. 121-137. A Systematic Literature Review on User Interface Design for Web Applications e-ISSN: 2289-8131 Vol. 9 No. 3-4 61

[12]. B. Fogg, "A behavior model for persuasive design," in Persuasive '09 Proceedings of the 4th International Conference on Persuasive Technology, 2009, p. 1-7.

[13]. D. Freeney, Usability Versus Persuasion in an Application Interface Design. Master's Thesis in Innovation and Design. Malardalen University, Sweden, 2004.

[14]. A. Nemery, E. Brangier, and S. Kopp, "First validation of persuasive criteria for designing and evaluating the social influence of user interfaces: justification of a guideline," in First International Conference on Design, User Experience, and Usability, 2011, pp. 616- 624.

[15]. M. Lykke, "Persuasive design principles: means to improve the use of information organisation and search features in web site information architecture?," in American Society for Information Science and Technology SIG-Classification Research 20th Research Workshop, Vancouver, Canada, 2009.

[16]. N. Ngadiman, S. Sulaiman, and W. M. N. W. Kadir, "A systematic literature review on attractiveness and learnability factors in web applications," in International Conference on Open System (ICOS), 2015, pp. 22-27.

[17]. M. N. Islam, "A systematic literature review of semiotics perception in user interfaces," Journal of Systems and Information Technology, vol. 15, no. 1, pp. 45-77, 2013.

[18]. U. O. Nwokedi, B. A. Onyimbo, and B. B. Rad, "Usability and security in user interface design: a systematic literature review," International Journal of Information Technology and Computer Science (IJITCS), vol. 8, no. 5, pp. 72-80, 2016.

[19]. B. Kitchenham, S. Charters, D. Budgen, P. Brereton, M. Turner, S. Linkman, M. Jorgensen, E. Mendes, and G. Visaggio, "Guidelines for performing systematic literature reviews in software engineering", EBSE Technical Report version 2.3, Durham, UK, 2007.

[20]. J. Eisenstein and A. Puerta, "Adaptation in automated user-interface design," in Proceedings of the 5th International Conference on Intelligent User Interfaces, 2000, pp. 78-81.

[21]. A. Hussey, "Formal object-oriented user-interface design," in Proceedings of Software Engineering Conference, 2000, pp. 129-137.

[22]. M. Terry and E. Mynatt, "Recognizing creative needs in user interface design," in Proceedings of the 4th Conference on Creativity & Cognition, 2002, pp. 38-44.

[23]. A. Seffah and E. Metzker, "The obstacles and myths of usability and software engineering," Communications of the ACM-The Blogosphere, vol. 47, no. 12, pp. 71-76, 2004.

[24]. L. Wenting, S. Tian, Z. Hong, and Y. Ying, "Interface design in the ship navigation information system," in 11th International Conference on Computer-Aided Industrial Design & Conceptual Design (CAIDCD), 2010, pp. 395-400.

[25]. M. Obeidat, and S. Salim, "Integrating user interface design guidelines with adaptation techniques to solve usability problems," in 3rd International Conference on Advanced Computer and Engineering, (ICACTE), 2010, pp. V1-280-V1-284.

[26]. J. Janeiro, S. Barbosa, T. Springer, and A. Schill, "A flexible model for improving the reuse of user interface design patterns," in Proceedings of the 28th ACM International Conference on Design of Communication, 2010, pp. 215-221.