Mockup-based Navigational Diagram for the Development of Interactive Web Applications

I. Bouchrika, L. Ait-Oubelli & A. Rabir
Dept of Mathematics and Computer Science
University of Souk-Ahras
Souk-Ahras, 41000, Algeria
imed@imed.ws

N. Harrathi
Department of Computer Science
University of Mentouri - Constantine
Constantine, 25000, Algeria
harrati.nouzha@yahoo.fr

ABSTRACT

The web industry has evolved very rapidly since its beginning from read-only static content to fully interactive applications. As opposed to traditional software applications, the development of a web application is a challenging and complex task involving many phases with different type of stakeholders. In fact, the majority of web applications do get shredded away or fail to gain popularity due the simple fact that the application does not meet and achieve usability goals with the accuracy and level outlined by the experts at the early. In this research studies, we propose a communication channel to automate the process between user interface prototyping and formalized software architecture in order to bridge the gap between the different stakeholders. As user interfaces provide a rich source of user requirements, an approach is being investigated to formalize requirements analysis from user interface using a proposed Mockup-based navigational diagram that can be translated to other architectural models and therefore the possibility of automated code generation.

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: Evolutionary prototyping

General Terms

HCI, Mockups, Prototyping, Software Engineering

1. INTRODUCTION

The web development industry has evolved very rapidly since its beginning from read-only static content to fully interactive and rich distributed web applications. As a result, the web has become an essential part of our daily activities from watching television, travel booking, reading news and shopping. A number of technologies, tools and platforms have emerged to ease and formalize the development of web applications as well as to alleviate the common problems encountered during the development by providing solutions to common activities such scalability, authentication, database access and session management. As opposed to traditional software applications, the development of a web application is a challenging and complex task [12] involving many phases with different type of end-users and contributors throughout the development phases from software developers, end-users, HCI experts, User Experience (UX) designers and so on. In fact, the majority of web applications do get shredded away or fail to gain popularity due the simple fact that the application does not meet and achieve usability goals [13, 9] or wishes of end customers with the accuracy and level outlined by the experts at the early stage of the requirements analysis.

The failure of web applications in meeting usability goals stems from a number of factors mainly due to the lack of communication between the different development teams which are usually from different disciplines including end-users, HCI specialists, UX designers, Quality Assurance (QA) staff and marketers. In fact, such complexity is as result of the development approach and methodologies where there is no common bridge or platform between software developers and other members of the projects as end-users or HCI experts who outline the strategies or requirements for the system to interact with the end-user. Requirements analysis should be the results of an intensive, continuous and iterative communication with the customers or experts to provide a representation of the business and usability decisions related to the expected applications. In practice, many projects start ambitiously with the technical design or even the implementation with the gap from other teams growing wider and larger to the extent that the development process becomes a functionality or code-driven project ignoring severely other aspects of the applications that can be a factor for its success. Furthermore, to architect or redesign the software to meet existing or new requirements, can be an expensive and unbearable process. This is mainly because software are fully driven by needs to satisfy usability goals and please end customer rather than providing extra and fancy functionalities that can drive away end-users.

Arguably during the requirements analysis, the end user or teams from other disciplines cannot precisely define the formal specification of the design even using a simple UML use case diagram. On the other hand, specifications drawn using a simple prototype or mockup interface [16] can influence deeply the formal specification or technical design of the system. The involvement of end-users and HCI specialists is

ISDOC'13 July 11-12, 2013, Lisbon, Portugal.

a key factor for the success of a given web application. Prototyping of user interfaces has been recently used as a way to communicate requirements and project specification between the different teams. However, prototypes are usually ignored when starting the development where the technical and implementation aspects are being laid down from the ground. This is because user interface prototypes are static drawings introduced to provide insights to software architects and developers of the system's functionalities.

In this research studies, we propose a communication channel to automate the process between user interface prototypes and formalized software architecture in order to bridge and narrow the gap between end-users and software developers. As user interfaces provide a rich source of user requirements, an approach is being investigated to formalize requirements analysis from user interface using a Mockup-based navigational diagram. The proposed approach is aimed to devising a way for automating the translation of user prototypes which are not formalized towards to formalized description as UML diagrams or Architecture Definition Language (ADL) using XML. This can be a major milestone for developing web application with usability goals as a primary priority during the iterative and continuous development phases.

The remainder of this research paper is organized as follows. The next section summarises the previous related work for the different approaches being used for prototyping-based development. The theoretical description of the proposed approach for formalizing user requirements from user interfaces using the Mockup-based navigational diagram presented in Section 4. A case study is discussed in the Section 5 followed with conclusions and future work.

2. RELATED WORK

The capturing and formalizing of specifications for web applications is a challenging and complex task due to the nature and characteristics of such systems such as the rapid changes and evolution of requirements, participation of the different stakeholders in the development process and technical constraints. During the last few years, a variety of tools and methods are being proposed for the capturing of web requirements using UML Diagrams including use cases and sequence diagrams [1], task models [14] and navigational models [7, 18]. Furthermore, a number of advanced tools are being freely or commercially available to capture user requirements for the graphical user interface such as Balsamiq. However, despite the outstanding advances in web science and software engineering, existing methods still have a number of limitations as it lacks the capability of providing an efficient two-way communication channel between the different multidisciplinary participants. For instance, a number of tools provide an informal platform for specifying requirements which cannot therefore be validated whilst other tools lacks the ability to evolve with newly added specification.

The process of capturing and constructing requirements models have been investigated by Koch *et al.* [7] who have worked on the derivation of requirement models with the aim of automatically generating UWE models. The authors have presented a modelling language named WebRE in which they have outlined a set of transformation rules specified at a meta-model level defined in the QVT language [8]. The transformation process covers the generation of content, navigational and presentation models. In the same way, Montero *et al.* [11] Introduced a case tool which produces design models from requirement models following the ADM model-driven approach used for generating light prototypes of the applications. In [19] Valderas *et al.* proposed the idea of including other team members within the development life cycle such graphics designers by extending their automatic code generation strategy of the Object-Oriented Web Solution (OOWS).

Rivero *et al.* [17, 10] proposed a hybrid model-based agile methodology named Mockup-Driven Development (MockupDD) where the user interface mockups are designed and constructed using a commercial software Balsami during the requirements analysis phase with the active participation of end users. Drawn mockups from Balsamiq are exported to an XML format and later translated to an abstract user interface model. Based on a tagging approach for the presentation models of mockups, the authors in [17, 10, 5] introduced the navigational specification which is utilized along a heuristic approach to infer the content model. Brogneaux *et al* [2] discussed in their research study a methodology of drawing user interfaces using 3rd party software which exports drawing to USIXML documents. Based on the form-components within the interface, the database conceptual schema is deduced using a semi-automated approach.

3. CONCEPTS & CHALLENGES

In software engineering, a number of modelling methods and CASE tools have been defined to specify user and software requirements [6]. The classical way to capturing and define the system functionalities is to use UML diagrams. Unified Modeling Language (UML) is used for the conceptual and logical modelling of mostly any system whilst it supports both static and dynamic modelling with the ability to model complex systems [15]. UML notations include a number of diagrams such as the use case diagram, activity diagram and interaction diagram. The use case diagrams are textually specified enabling the definition of a sequence of steps that describe the interaction between one or more actors and the system. Use case diagrams are important as they lay the foundation for subsequent development phases by providing a functional view where each use case describes one use of the system. However, the discussion of whether UML diagrams can be used to communicate ideas and requirements between the different stakeholders can be a concern. This is mainly because current modelling tools do not visualize the end-user mental model for a desired system to be delivered.

For instance, a simple case scenario for an online system setup for an online visitor to sign-up or login in order to view a private video is considered during the course of this research studies. The use case diagram is plotted as shown in Figure (1) showing the cases or the main functionalities of the system which are: signup, login and show video. However, in practice, such a simple system can contain a large number of use cases such as reset forgotten passwords, ban user, update member details, update video, show help or advertisement, and so on. Without doubt, a use case diagram or other UML diagrams are central for a model-driven engineering approach facilitating software abstraction and development productivity. However, as a vital process for requirement engineering, a number of critical and essential use cases for the system can be missed or ignored that highly related to the usability of the system.

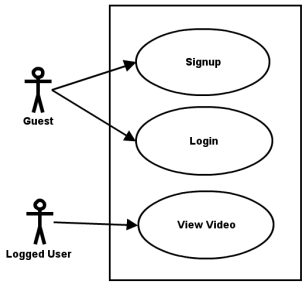

Figure 1: The Use Case Diagram

The UML activity diagram showing the operational workflow of the online system is being illustrated in Figure (2). The large black dot located at the top is the starting point of the flow. Rounded rectangles represent actions whilst diamonds represent decisions with true and false outputs. In fact, the activity diagram provides rich information about the system core behaviour for software developers. However, other stakeholders may find it difficult to interpret or collaborate using such a simple medium. This is mainly because the activity diagram is a graphical representation of the algorithm or pseudo-code. Interaction styles, event triggers or other essential user interface components are not considered which are essential for the usability of the system as well the development process. For instance, the transitional arrow between actions or decision node in the diagram shown can be implemented as simple HTML links. However, a HCI expert may argue that having an AJAX-driven interface [4] for the system is far more superior for a better usability experience. Therefore, the user would be able to login or register on the welcome page using a *lightbox* or a *modal box* without leaving the homepage. Indeed, such a simple argument can affect the implementation and development process at an expensive cost.

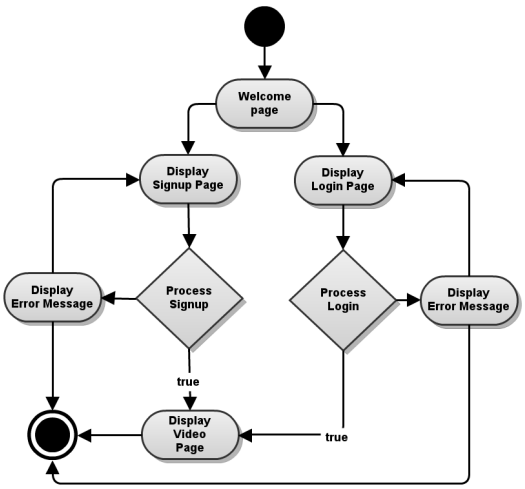

Figure 2: The UML Activity Diagram

Whilst Model-Driven methodologies facilitate software specification, abstraction as well as productivity they fail in providing an agile interaction across the different stakeholders. This is mainly because concrete results are obtained too late at the final phase of the development process [17]. In fact, MDA methodologies would not be ideal for developing web applications that require different multidisciplinary stakeholders to be involved and communicating constantly from HCI experts, web designers, quality assurance staff, end-users as well as software developers. On the other hand, agile methods promote early and constant interaction with the customers and other stakeholders to assure that the software complies with their requirements [3]. This is usually observed by delivering prototypes in short periods of time meanwhile software specification would emerge naturally and therefore enhancing former prototypes along the development process until the final product is obtained.

4. THE PROPOSED APPROACH

To ease the development process for web application, there needs to be a solid and self-describing communication protocol for all different team members involved at the early stage of development. This is in order to discuss, derive and capture the set of requirements that might even change from time to time quickly. Mock-ups and wire framing for software application using drawing tools such as Balsamiq have been reported to be useful for communicating ideas. However, such tools are commercial and proprietary and therefore cannot be extended to translate their mock-ups to other architectural models in addition to the fact that mock-ups are not considered as models and they are trashed after requirement engineering [17]. Further, produced mock-ups are usually cluttered with design elements where it is impossible to integrate the navigational and interactional data or business decisions.

In this research studies, we propose a mockup-based navigational approach that can be translated to other architectural models and therefore the possibility of automated code generation. An overview of the proposed approach is being illustrated in Figure (3) where all different stakeholders from HCI experts, QA staff, end-users or software developers should be able to communicate together using a common protocol in order to visualize the requirements of the desired system. The mockup-based diagram contains only essential components relating to the user interaction and navigation through the use of the application. The Mockup diagram is exported to an XML file complying with a pre-defined XML schema where the transformation can be devised heuristically in order to generate different models such as UML diagrams and testing cases.

In practice, a web application is composed of web pages that are traversed through transition events triggered usually by the user such as clicking or pressing some keys. During the transition across different pages or within the same page, business logic is being conducted either at the server, client level or both. Our approach is based on three main components which are: pages, transition events and business entities for visualizing the system behaviour for all team members. Mockup diagram can be considered as a graph diagram where the nodes are either the pages or business entities whilst the edges linking the nodes are the transition events. Special notations are being utilized for distinguishing the different naming for the elements contained within the diagram.

The main element is a web page conventionally noted as **#nameofthepage** as shown in Figure (4). The page can be either a standalone web page or loaded as embedded within other active pages using a *lightbox*. The name of the page

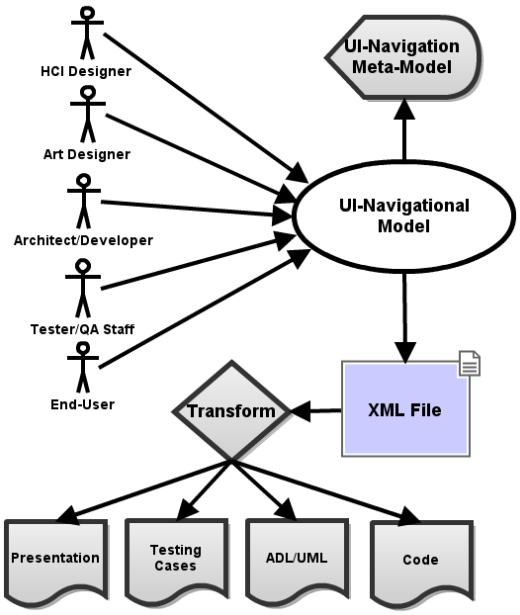

Figure 3: Overview of the Mockup-based Navigational Development Process

is written on top left corner meanwhile an icon is placed on the top right corner to signify the type of the page. Widgets are essential elements contained within a web page or a virtual container to aid the interaction process between the system and the user such as a button, label, textfield, checkbox, radiobox, image or video. For the naming of a widget element as referenceable, the @ symbol is utilized to access an element.

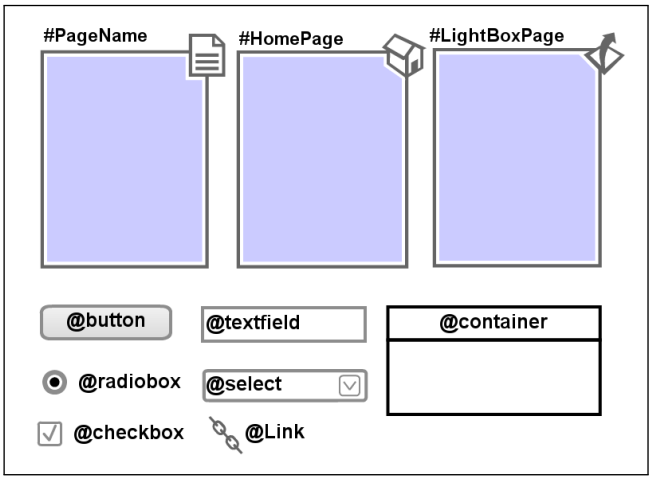

Figure 4: Pages & Elements being used for the Mockup Diagram

The transition between different nodes is triggered through interaction events between the user and the system such as clicking, pressing a key, scrolling or even closing a page. Transitions are plotted using an arrow with the annotation of the event and the corresponding arguments. The form of the annotation is written as follows:

e(@element).Event(EventArgs);

The function *e*() is for accessing a widget element using its reference. The values of *Event* and *Action* which are discussed in this section are summarized in Table (1). The *AjaxAct* is introduced to use *Ajax* functionalities like dynamically updating, showing as well as adding content.

Element Events	Page Events	Actions
Click	Loaded	Lightbox
MouseOn	Scroll	Redirect
MouseOut	Closed	ProcessForm
KeyPressed		AjaxAct
KeyUp		
KeyDown		

Table 1: Events & Actions for the Widget Elements

To further illustrate the use of transition, an example is considered for the event of clicking on a link named @*help* in order to show a *lightbox* which loads the content from a webpage named #HelpPg; it is annotated as:

e('@help').Click();

Meaning that on clicking the link named @*help*, a lightbox should be opened within the active page as furtherly shown in Figure (5). Transition arrows do not have to be linked to the element itself, we have sought that linking to its page container would be sufficient and therefore avoid diagram clutter with arrows.

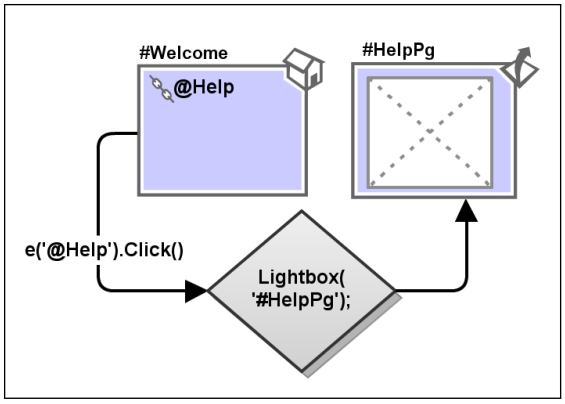

Figure 5: Transition between Pages

For the business entities introduced in the Mockup-based diagram, they are modelled as diamond object in the same way as UML activity diagram that does refer to business decisions within the application. Usually business entities are annotated with actions to be carried out either at the client level or at the server level. A number of actions are listed in Table (1). The *Lightbox* action is utilized for showing modal box loaded with content from an external page within the actual active page as mentioned in the previous example and illustrated in Figure (5). The *Redirect* action is aimed to navigate or send the user to a different page. *ProcessForm* is used to redirect the user to a different page along with form data for processing at the server level. The *AjaxAct* is introduced to add dynamicity and improved interactivity to the application page taking the following syntax:

AjaxAct('subAction','Arg1','Arg2',...)

The *subAction* can take various values including *'AjaxFormProc','Hide','Show', 'UpdateContent', 'Toggle', ...* For example, in order to hide a button referenced as *@ProcessPay* after *Click* event trigger to avoid double payment processing, we use the following annotation:

AjaxAct('Hide','@ProcessPay')

The *AjaxFormProcess* subAction processes the form within the same active page in contrast to the *ProcessForm* action. The subAction takes two arguments which are the references for the form and a message container to display any returned results. This is illustrated in the following example for a login form:

AjaxAct('AjaxFormProcess', '@LoginForm','@LoginMsg')

For advanced business actions that need to be carried out at the server level, an OOP-based annotation can be used by software developers. For example, to show a business entity for processing user authentication, the following annotation is being adopted:

$User.Login():Boolean

where this is a function named *login* belonging to the object *User* introduced to process its authentication and should return a *Boolean* value.

Listing 1: DTD for the Mockup-Based Diagram

```
<?xml version="1.0"?>
<!DOCTYPE System [
<!ELEMENT System ( Pages ) >
<!ELEMENT Pages ( page+ ) >
<!ELEMENT page ( Widgets ) >
<!ATTLIST page name CDATA #REQUIRED >
<!ATTLIST page type CDATA #IMPLIED >
<!ELEMENT Widgets ( widget* ) >
<!ELEMENT widget ( Triggers ) >
<!ATTLIST widget name CDATA #REQUIRED >
<!ATTLIST widget type CDATA #REQUIRED >
<!ELEMENT Triggers ( trigger* ) >
<!ELEMENT trigger (event, process, Args, Returns)>
<!ELEMENT event ( #PCDATA ) >
<!ELEMENT process ( #PCDATA ) >
<!ELEMENT Args ( arg* ) >
<!ELEMENT arg ( #PCDATA ) >
<!ELEMENT Returns ( return* ) >
<!ATTLIST Returns type CDATA #REQUIRED >
<!ELEMENT return (value, action*)>
<!ELEMENT value ( #PCDATA ) >
<!ELEMENT action ( #PCDATA ) >
]
```

The Mockup-based navigational diagram is usually exported to an XML document as an initial phase for a transformation process to other models. Therefore, the XML file can be parsed and utilized for automated code generation, derivation of UML diagrams or testing scenarios. The Document Type Definition (DTD) is outlined in Listing (1) showing the structure or format for the generated XML document of the Mockup diagram.

5. CASE STUDY

For the case being shown in Figure (2) for an online application where a user can login or register for an account in order to view a private multimedia content, the UML activity diagram without doubt bears little knowledge for teams to reinforce or communicate their ideas as it is rather an abstract description for the system. Using the proposed Mockup-Base navigational diagram, the same system is being modelled and illustrated in Figure (6).

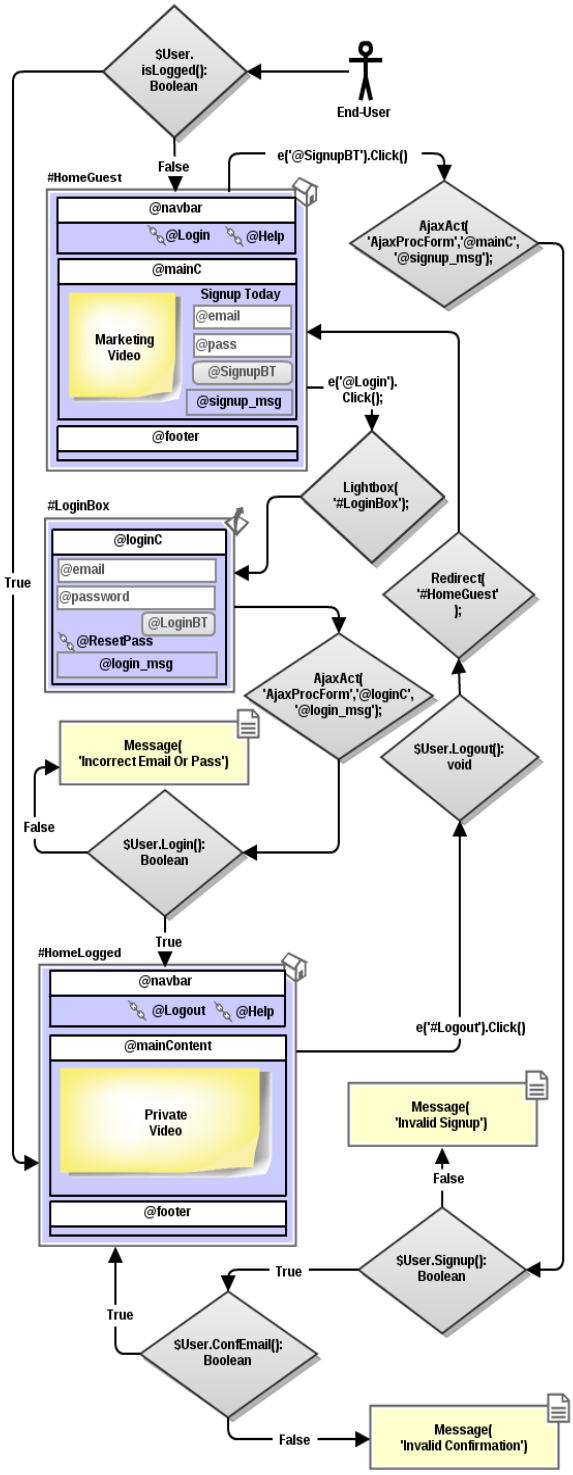

Figure 6: Mockup-Based Navigational Diagram for an Online Application

Upon the user initial visit, a business entity is placed to check the authentication status of the user to redirect to the corresponding page. For the case where the user is logged in, they will be redirected to the page named **#HomeLogged** which have the private content. In case, the user is not logged in, a normal guest page is being loaded where the user can login or signup if they do not already have an account. Upon clicking the button @*Signup*, a *Lightbox* with the content of the page **#LoginBox** will be shown to proceed to the registration process. Indeed, it can be observed that a lot of interaction as well as functionality cases covered inside the Mockup-based diagram including login, signup, logout, forget password. Furthermore, all team members from different disciplines can understand and collaborate on constructing the navigational mockup diagram. For simplicity sake, other cases for the system are ignored such as resetting forgotten password and updating user account details.

6. CONCLUSIONS

The web development industry has evolved very rapidly since its beginning from static content to fully interactive and rich distributed web applications. A number of technologies, methodologies and platforms have emerged to ease and formalize the development of web applications. As opposed to traditional software applications, the development of a web application is a challenging and complex task involving many phases with different stakeholders throughout the development phases from software developers, end-users, HCI experts, User Experience designers and so on. In fact, the majority of web applications do get shredded away or fail to gain popularity due the simple fact that the application does not meet and achieve usability goals outlined by the experts. We propose a communication channel to automate the process between user interface prototypes and formalized software architecture in order to the gap between end-users and software developers. As user interfaces provide a rich source of user requirements, an approach is being investigated to formalize requirements analysis from user interface using a Mockup-based navigational diagram that can be translated to other architectural models and therefore the possibility of automated code generation. This can be a major milestone for developing web application with usability goals as a primary priority during the iterative and continuous development phases.

7. REFERENCES

[1] A. I. Anton, R. A. Carter, A. Dagnino, J. H. Dempster, and D. F. Siege. Deriving goals from a use-case based requirements specification. *Requirements Engineering*, 6(1):63–73, 2001.

[2] A.-F. Brogneaux, R. Ramdoyal, J. Vilz, and J.-L. Hainaut. Deriving user-requirements from human-computer interfaces. In *Proc. of 23rd IASTED Int. Conf.* Citeseer, 2005.

[3] J. Ferreira, J. Noble, and R. Biddle. Agile development iterations and ui design. In *Agile Conference (AGILE), 2007*, pages 50–58. IEEE, 2007.

[4] J. J. Garrett et al. Ajax: A new approach to web applications, 2005.

[5] J. Grigera, J. M. Rivero, E. R. Luna, F. Giacosa, and G. Rossi. From requirements to web applications in an agile model-driven approach. In *Web Engineering*, pages 200–214. Springer, 2012.

[6] M. Jackson. *Software requirements & specifications*, volume 8. ACM Press New York, 1995.

[7] N. Koch, A. Knapp, G. Zhang, and H. Baumeister. Uml-based web engineering. In *Web Engineering: Modelling and Implementing Web Applications*, pages 157–191. Springer, 2008.

[8] I. Kurtev. State of the art of qvt: A model transformation language standard. In *Applications of Graph Transformations with Industrial Relevance*, pages 377–393. Springer, 2008.

[9] P. Lew, L. Olsina, and L. Zhang. Quality, quality in use, actual usability and user experience as key drivers for web application evaluation. In *Web Engineering*, pages 218–232. Springer, 2010.

[10] E. R. Luna, G. Rossi, and I. Garrigós. Webspec: a visual language for specifying interaction and navigation requirements in web applications. *Requirements Engineering*, 16(4):297–321, 2011.

[11] S. Montero, P. Díaz, and I. Aedo. From requirements to implementations: a model-driven approach for web development. *European Journal of Information Systems*, 16(4):407–419, 2007.

[12] S. Murugesan, Y. Deshpande, S. Hansen, and A. Ginige. Web engineering: A new discipline for development of web-based systems. In *Web Engineering*, pages 3–13. Springer, 2001.

[13] J. Offutt. Quality attributes of web software applications. *Software, IEEE*, 19(2):25–32, 2002.

[14] F. Paternò, C. Mancini, and S. Meniconi. Concurtasktrees: A diagrammatic notation for specifying task models. In *Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, volume 96, pages 362–369, 1997.

[15] C. Phillips, E. Kemp, and S. M. Kek. Extending uml use case modelling to support graphical user interface design. In *Software Engineering Conference, 2001. Proceedings. 2001 Australian*, pages 48–57. IEEE, 2001.

[16] F. Ricca, G. Scanniello, M. Torchiano, G. Reggio, and E. Astesiano. On the effectiveness of screen mockups in requirements engineering: results from an internal replication. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, page 17. ACM, 2010.

[17] J. M. Rivero, J. Grigera, G. Rossi, E. R. Luna, and N. Koch. Improving agility in model-driven web engineering. In *CAiSE Forum*, volume 734, pages 163–170, 2011.

[18] J. M. Rivero, J. Grigera, G. Rossi, E. R. Luna, and N. Koch. Towards agile model-driven web engineering. In *IS Olympics: Information Systems in a Diverse World*, pages 142–155. Springer, 2012.

[19] P. Valderas, V. Pelechano, and O. Pastor. Introducing graphic designers in a web development process. In *Advanced Information Systems Engineering*, pages 395–408. Springer, 2007.