Mockup-Supported Web Requirements Engineering

Jia Zhang
infiNET Solutions
1425 E. Busch Parkway
Buffalo Grove, IL 60089, USA
jiazhangchicago@yahoo.com

Konstantin Läufer
Department of Computer Science
Loyola University Chicago
Chicago, IL 60626, USA
laufer@acm.org

Zhiguo Gong
Faculty of Science and Technology
University of Macau
P.O. Box 3001 Macao, PRC
fstzgg@umac.mo

Abstract

Requirements engineering for web applications entails new demands, compared to traditional software applications. This paper proposes a mockup-driven fast-prototyping methodology to help elicit and finalize system requirements, as well as facilitate adjustments to quickly changing user requirements typical of web applications. This strategy minimizes the risk of wasting valuable development efforts because of ambiguous or incomplete requirement specifications.

1. Introduction

The emerging term *web requirements engineering* (WRE) refers to requirements engineering activities specific for web applications. In the broadest sense, a web application is an application accessible through the Internet. Because of the distinctive features of web applications as compared to traditional software applications [4][6], web application development usually starts from ill-structured or vague requirements [3]. In addition, web applications constantly change their requirements so as to meet the volatile demands from the market. Consequently, WRE entails new demands. The last decade has witnessed numerous comprehensive notations, models, and methodologies that have been introduced in the requirements engineering (RE) field; however, little attention has been paid to methodologies coping with requirements elicitation and finalization specific to web applications [1]. Our goal is synergistically to apply proven concepts in RE to web applications, in order to provide an efficient methodology for WRE. In this paper, we propose a mockup-driven fast-prototyping methodology (MODFM) to help elicit and finalize system requirements, as well as facilitate adjustment to quickly changing user requirements typical to web applications.

The rest of this paper is organized as follows. In Section 2, we discuss related work. In Section 3, we present enhancements to our web system generator. In Section 4, we introduce our mockup-driven fast-prototyping methodology. In Section 5, we discuss and evaluate our methodology. In Section 6, we summarize the contribution and innovation of MODFM, discuss assessments, and describe future work.

2. Related Work

There has been a considerable amount of research in RE. Among the variety of research achievements, four of the most promising RE approaches form a solid foundation for WRE: fast prototyping, structured analysis, use case methodology, and architecture-driven requirements engineering. Szekely defines the concept of fast prototyping as constructing a small-scale version of a complicated system in order to acquire critical knowledge required to build a full system [11]. This prototype not only provides clients with a complete picture of what the final product will be, but it also facilitates requirements validation, elicitation, and revision. Since its inception in 1970s and 1980s, structured analysis has been considered as a powerful and natural approach to analyze complicated software application requirements [13]. The use case approach [8], originally proposed by Jacobson and colleagues, utilizes a scenario-driven mechanism and has been extensively adopted and widely considered as the most popular requirements elicitation technique in industry [10]. Software Architecture Based Requirements Engineering (SABRE) [2] aims at bridging between software architecture (SA) and RE; which core concept is that SA supports RE whilst RE helps firm up SA as a high-level solution. SABRE concentrates on utilizing a function-class decomposition in order to identify and validate user requirements incrementally.

Our previous research resulted in a structured software architecture for J2EE-oriented [9] web applications [14]. A web application system can be organized as a collection of functional modules, and each module can be in turn divided into a list of interacting mini-modules. Each mini-module can be implemented as a tiny web application consisting of a set of components, which can be structured into a front-end tier and a back-end tier. These two tiers represent web presentation and business logic, respectively, each exhibiting a Model-View-Controller structure [5]. According to this architecture, any module in a web application can be realized by the composition of JSP pages, form beans, pre-actions, post-actions, service methods, EJB components, and database schemas. As a

result, automatic code generation becomes high practical.

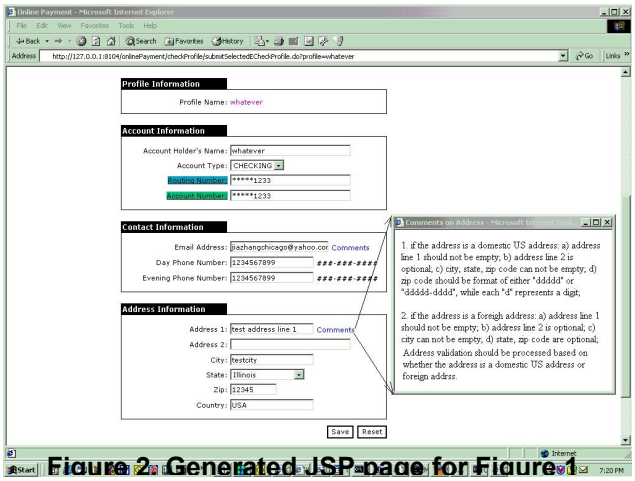

Figure 2. Generated JSP page for Figure 1

3. Web System Generator (WSG)

Our structured architecture makes it reasonable and feasible to apply automatic code generation to web application development. Our previous research thus resulted in a J2EE-oriented web system generator (WSG) [15][16]. Three types of files must be provided by users:

```
<WGenerator>
 <OBJ NAME="ECheckPaymentProfile" TNAME="echeck_profile">
   <ATTR ATNAME="Key" ATYPE="String" UNIQ="YES"/>
   <ATTR ATNAME="HolderName" ATYPE="String"
VALID="YES"/>
   <ATTR ATNAME="AccountType" ATYPE="String"
VALID="YES"/>
   <ATTR ATNAME="RoutingNum" ATYPE="String"
VALID="YES"/>
    <ATTR ATNAME="AccountNum" ATYPE="String"
VALID="YES"/>
   <ATTR ATNAME="EmailAdd" ATYPE="String" VALID="YES"
          COMMENTS=”Has to be valid email address”/>
   <ATTR ATNAME="DayPhone" ATYPE="String" VALID="YES"/>
   <ATTR ATNAME="NightPhone" ATYPE="String" VALID="YES"/>
   <ATTR ATNAME="AddressLine1" ATYPE="String" UNIQ="NO"
  COMMENTS=”1. if the address is a domestic US address:
                    a) address line 1 should not be empty;
                    b) address line 2 is optional;
                    c) city, state, zip code can not be empty;
                    d) zip code should be format of either "ddddd" or
"ddddd-dddd", while each "d" represents a digit;
                  2. if the address is a foreigh address:
                    a) address line 1 should not be empty;
                    b) address line 2 is optional;
                    c) city can not be empty;
                    d) state, zip code are optional;”/>
   <ATTR ATNAME="AddressLine2" ATYPE="String" UNIQ="NO"/>
    <ATTR ATNAME="City" ATYPE="String" UNIQ="NO"/>
   <ATTR ATNAME="State" ATYPE="String" UNIQ="NO"/>
   <ATTR ATNAME="Country" ATYPE="String" UNIQ="NO"/>
   <ATTR ATNAME="ZipCode" ATYPE="String" UNIQ="NO"/>
  </OBJ>
</WGenerator>
```

data files define the data models that can be applied to template file(s) to generate target file(s); template files themselves are code files that contain tags that need to be replaced at generation time by the corresponding values from data files; and configuration files specify the many-to-many relationships between the former two as well as the criteria of generation. Both data files and

configuration files are XML files. Seamlessly integrated with the architecture described above, WSG offers a J2EE-oriented template system. As a result, for every unit-level functional requirement, one merely needs to provide data models and some criteria; and the WSG will generate a running unit with all related code from the front-end to the back-end.

Here, we enhance WSG to include support for requirements elicitation. Figure 1 illustrates an enhanced data file of a user's payment profile of electronic check. Once a user defines his payment profile, he can utilize the profile to pay bills without re-typing all related information. For each item of the corresponding data model, one can define the item's name and data type. *ECheckPaymentProfile* contains fourteen data items: holder name, account type, routing number, account number, email address, day time phone number, evening phone number, address line 1, address line 2, city, state, country, and zip code. The data item key is used to store the primary key internally. Detailed information about the specification rules for each item can be found in [14]. It should be noted that we provide the ability for users to specify comments on data items. As shown in Figure 1, two data items are associated with comments: email address and address line 1. Comments of the former item say that email address had to be valid email address; comments of the latter item declare the validation rules for address.

We also enhance the JSP page generation accordingly. If one data item is associated with comments, a clickable link with word "Comments" will be shown beside the data item on the generated page. If a user clicks on the link, a window will pop up displaying the comments defined in the corresponding data file. Taking Figure 1 as an example, Figure 2 shows its corresponding generated JSP page. It can be seen that there are two links next to the email address and address line 1, respectively; each link is in blue color. Figure 2 also illustrates the window popped up when the link next to address line 1 is clicked. This window contains exact information defined in Figure 1 as comments of data item address line 1, which is the validation rule for address.

This enhancement facilitates WSG to serve for requirements elicitation. In the process of RE, it is normally sufficient to define business rules in informal language, or even natural language. This enhancement enables business rules to be associated with corresponding data items on the web page, so that not only clients can review the rules, but also developers can implement these rules and replace the comments accordingly.

4. Mockup-driven Fast-prototyping Methodology (MODFM)

On the basis of our structured web system architecture and web system generator (WSG), we propose a mockup-driven fast-prototyping methodology (MODFM) serving for WRE. According to *The American Heritage Dictionary*, a mockup is "a usually full-sized scale model of a structure, used for demonstration, study or testing" [7]. In this paper, a mockup refers to a running, navigable, partial or full-sized model of a web application, used for requirements elicitation, validation, and finalization. Meanwhile, MODFM is applicable to a web application if the application satisfies the following four assumptions. First, the application is a typical web application, which is navigable through a set of web pages. Second, the application can be decomposed into modules exhibited by a hierarchical menu system. In other words, all the web pages can be organized into a menu system. Third, the project at least starts with a very high-level functional description of the system. Fourth, the hierarchy of the menu system is no greater than three. In addition, we assume that a test server has already been set up so that all mockups can be deployed to the test server, while clients can test the mockups and provide feedback remotely.

The essential tenet of MODFM is always to use a running mockup system to elicit and validate user requirements. This iterative process applies a top-down approach to decompose system functionalities to web pages while simultaneously generating navigable running mockup system, with functionalities organized by menu system. Figure 3 summarizes this iterative process. A module acts as the container for a list of web pages that exhibit high cohesion and low external coupling. In a typical web application, a module is normally realized as a menu item. Functional decomposition tasks consist of identifying menu items and sub-items, and organizing them in a menu system. Scenario analysis tasks consist of identifying web pages, finding out the navigational relationship among pages, and allocating pages to appropriate menu item. Mockup construction then occurs bottom up.

Figure 3. Mockup-Driven Fast-prototyping Methodology Algorithm

1. Functionally decompose the system into menus. Every menu item is identified as one web page. It can be as simple as a "Hello world" page.
2. Generate a mockup containing only menu system and dummy pages into a released version.
3. Deploy the mockup to test server. Go back to step 1 if client would like some changes; otherwise go to step 4.
4. Iterate on every menu item step 4 through 7. If finished, go to Step 8.
5. Develop scenarios of each menu item page. Identify a list of pages, gather page information, and record the business logic between pages.
6. Generate a mockup, with business logic displayed on every page as written document.
7. Deliver the mockup to client. Go back to step 5 if client would like some changes; otherwise jump to step 4.
8. Deliver the mockup to developers.

5 Evaluation of MODFM

Here we seek to provide efficient support for the end-to-end process of generating and eliciting web requirements. In contrast with previous approaches, we accomplish this objective by providing a generic framework to support generating a running mockup application and by using automatic code generation techniques that exploit state-of-the-art tools and technologies for web development.

Building mockups as system prototypes for requirements engineering has several advantages. First, the mockup helps elicit and finalize the system requirements. Second, clients can view the visual layout of the final system at extremely early stages. Third, since a mockup will become the skeleton of the final system, even requirements elicitation work will not be wasted. Fourth, the mockup can be reused by other similar applications since they do not involve coding application logic. Fifth, constructing the mockup blends functional decomposition [12] with object-oriented design principles.

MODFM offers three compelling features: short release cycles, incremental requirements elicitation, and suitability for non-technical analysis. The mockup starts out as a menu system only, and incrementally adds new pages and scenarios. Clients can get involved in this process at any time. The web system generator makes extremely short release cycles practical. In addition, with the assistance of the web system generator, the mockup can be generated by a non-technical person who knows nothing about Java programming. In general, MODFM is *extreme* in the sense that it takes four well-known requirements engineering concepts and "best practices" to their logical extremes -- turning them all up to "10". This early, incremental, and client-focused approach is expected to reduce project risks considerably.

We have tested the MODFM on three industrial web applications. The first one is an e-hospital service suite that allows users to customize a new hospital web site at run time, as well as provides to hospitals with web-based capabilities such as managing accounts, administrations, notifications, etc. The second one is an e-university administrative system that encompasses modules such as student records, admissions, financial aid, financial services, registration, payment management, faculty, etc. The third one is an e-payment system that supports on-line electronic payments, including functionalities such as both e-check and credit card payment, bill presentment as both paper bill and PDF format, bill loading, and payment history presentment, etc. We believe that MODFM possesses high potential in the software industry, specifically in the area of web application development. Based on our experiences, MODFM promises more complete and accurate requirements gathering, more client satisfaction, and more efficient human resource usage.

6. Contributions, Assessments, and Future Work

MODFM improves the efficiency of requirements elicitation for web applications. This is accomplished by utilizing a structured J2EE-oriented architecture and web system generator. MODFM guarantees to deliver running web applications by incrementally eliciting, validating, and finalizing user requirements early and consistently; consequently, MODFM reduces the cost of nearly inevitable changes to business rules, programming environment, or software design. Many of these practices have been part of conventional wisdom for years, such as fast-prototyping, use case analysis, and SABRE; but rethinking their interaction in the context of web application development is the value of MODFM.

The efficiency and effectiveness of MODFM rely on our architectural model and web system generator; both techniques are J2EE-oriented. Hence, if the web technology changes significantly, these two techniques need to be upgraded accordingly. However, we believe that the concept and approach of MODFM is widely applicable; therefore it can be fully reused by future web applications, based on J2EE or other enterprise application platforms, as long as a corresponding platform-specific web system generator is plugged in.

Our future work will include an Interactive Development Environment (IDE) in order to support MODFM for non-technical analysts. Meanwhile, we plan to conduct more case studies, so as to test the effectiveness and efficiency of MODFM over different types of web applications.

7. References

[1] D. Bolchini and P. Paolini, “Capturing Web Application Requirements through Goal-Oriented Analysis”, The 5th Workshop on Requirements Engineering, Valencia, Spain, Nov. 11-12, 2002.

[2] C.K. Chang, J.C. Huang, S. Hua, and A.K. Combelles, “Function-class Decomposition: A Hybrid Software Engineering Method”, *IEEE Computer*, Dec. 2001, pp. 87-93.

[3] L.L. Constantine and L.A.D. Lockwood, “Usage-centered Engineering for Web Applications”, *IEEE Software*, Mar./Apr. 2002, pp. 42-50.

[4] Deshpande, Y. and Hansen, S, “Web Engineering: Creating a Discipline among Disciplines”, *IEEE Multimedia*, Apr.-Jun. 2001, pp. 82-87.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns*, Addison Wesley 1994.

[6] A. Ginige and S. Murugesan, “The Essence of Web Engineering – Managing the Diversity and Complexity of Web Application Development”, *IEEE Multimedia*, Apr. 2001, pp. 22-25.

[7] *The American Heritage Dictionary of the English Language*, 4th edition, Copyright 2000 by Houghton Mifflin Company, Published by Houghton Mifflin Company.
[8] I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard, *Object Oriented Software Engineering – A Use Case Driven Approach,* Addison-Wesley, 1992.
[9] http://java.sun.com/j2ee.
[10] W.J. Lee, S. D. Cha, and Y. R. Kwon, "Integration and Analysis of Use Cases using Modular Petri Nets in Requirements Engineering", *IEEE Transactions on Software Engineering*, vol. 24, no. 12, Dec. 1998, pp. 1115-1130.
[11] P. Szekely, *User Interface Prototyping: Tools and Techniques,* USC/Information Sciences Institute, 1994.
[12] R. Wieringa, "A Survey of Structured and Object-Oriented Specification Methods and Techniques", *ACM Computing Surveys*, Dec. 1998, pp. 459-527.
[13] E. Yourdon, *Modern Structured Analysis*, Yourdon Press, Upper Saddle River, N.J., 1989.
[14] Jia Zhang and Ugo Buy, "An Approach to Develop Web Applications", the 8th IEEE Symposium on Computers and Communications (ISCC 2003), To appear.
[15] J. Zhang and J.Y. Chung, "Web Code Generator", Proceedings of the IASTED International Conference on Applied Modeling and Simulation (AMS 2002), Nov.4-6, 2002, Cambridge, MA, USA, pp. 505-510.
[16] J. Zhang and J.Y. Chung, "A New Model for Web Application Development", Proceedings of the IASTED International Conference on Applied Modeling and Simulation (AMS 2002), Nov.4-6, 2002, Cambridge, MA, USA, pp. 429-434.