# WWM: A Practical Methodology for Web Application Modeling

Chanwit Kaewkasi[1] Wanchai Rivepiboon[2]
Software Engineering Laboratory
Department of Computer Engineering
Chulalongkorn University, Bangkok, Thailand
Tel. +66-2218-6988, Fax. +66-2215-6955.
chanwit@customix.net, wanchai.r@chula.ac.th

## Abstract

*Web applications are becoming more complex and the way to manage the complexity is to model them. This paper presents a methodology to model Web applications directly from the object-oriented fashion on the top of the event-driven programming concept. Our approach, WebForm-based Web application modeling Methodology (WWM), provides guidelines to model Web application architectures from higher point-of-view than the Web element perspective. We employ the Unified Modeling Language (UML) and a subset of Use Case Maps (UCM) for our methodology. The case study is developed to present how a Web application can be modeled, built and integrated with the business objects. A set of components is developed in order to use in the case study. Connection of the presentation to business layer shows that it could be done transparently via the components. The various issues are discussed to further extend our work.*

## 1. Introduction

Business sectors highly demand to anchor their transactions into the Internet environment. This trend makes Web applications more complex. Traditional Web applications were developed using Common Gateway Interface (CGI). Because of performance limitation and persistent state maintainability issues [15], it should be avoided for enterprise-class Web applications. With more flexible features comparing to CGI, Active Server Pages (ASP) [16] and Java Server Pages (JSP) [14] are current popular technologies. Microsoft's ASP is server-side engine that processes scripting languages embedded as special tags into HTML page. JSP acts in the same manner to ASP, but its major objective is to improve Java servlet. Java servlet is a Java server-side executable program that processes the requests from client. JSP gives more maintainability to Java servlet [14, 21]. Both of them have the following limitations: 1) the content and code are not separated; 2) the component-based architecture cannot be applied to the user-interface design.

To build complex Web applications, we need an appropriate tool. Fortunately, the evolution of Web application development drives itself into the new era. Coming of the tools that support event-driven programming to Web application development may change the trend. There are some existing tools, such as Microsoft's ASP.NET with Visual Studio .NET [17], and AtoZed Software's IntraWeb [11] in the marketplace that support event-driven programming. Those tools make Web applications development easier because they hide many complexities away from developers. However, it is not enough for making a high-complex application; we still need to model it first.

Object oriented techniques become popular. They are widely used for implementing business logic. In order to build a Web application that connects to those business objects, its presentation layer should be modeled in the same fashion, not only for maintainability issue, but also extensibilities. The work reported in [8, 9] shows that it is possible to model Web application architectures with the notations from the Unified Modeling Language (UML) [20]. The author defined an extension to UML, called Web Application Extension (WAE), including the concept and some stereotypes, such as <<JavaScript>> and <<Form>>. WAE is mainly focused to the Web element semantics, not object-oriented perspective. We propose a methodology to model high-complex Web applications on the top of the event-driven programming scheme. Our methodology shows a modeling approach from the object-oriented point-of-view, and it is also based on UML. To make our methodology more efficiency and practical, we additionally apply Use Case Maps (UCM) to our work for describing the scenarios of the Web application systems. We show that the methodology can be used with the use-case driven development process to build not only a practical Web application model, but also the real-world software.

Section 2 reviews the event-driven tools that support Web application development, Use Case Maps, and the work on Web modeling techniques. Section 3 describes our methodology. Section 4 presents how to model and

[1] Master student.

[2] Associate Professor, Head laboratory.

Proceedings of the 26 th Annual International Computer Software and Applications Conference (COMPSAC'02)
0730-3157/02 $17.00 © 2002 IEEE

build Web application with our techniques as a case study. Section 5 gives conclusion and discussion of our work and its further extension.

## 2. Related Works

### 2.1 Web application development tools

Event-driven programming has been widely used in many desktop application development environments. There are few existing frameworks in the market that apply this programming scheme to Web application development. ASP.NET, the member of the Microsoft's forth-coming .NET Application framework [17], and AtoZed Software's IntraWeb [11] are such new kind of the development framework that support event-driven programming for Web application. ASP.NET provides the necessary services for developers to build the enterprise-class Web applications. IntraWeb acts the same functions as ASP.NET and it could be used for Borland Delphi. The technique behind frameworks is a back-end engine, which acts as the input/output manager of Web applications. The engine provides transparently interactive simulation for developers feel as they are writing a desktop application with normal event-driven programming scheme.

### 2.2 Use Case Maps

Use Case Maps (UCM) [6, 7] is a set of semi-formal notations for describing the scenario of the system. It gives a scenario-based model to the system. UCM model fulfills the gap between the high-level architectural and the detailed design phase. UCM has been successfully applied to wide range of the systems, including telecommunication systems [1, 3]. Previous works reported the use of UCM with UML [2, 6]. In the literature [18], the authors purposed methodology and techniques for deriving Message Sequence Chart from UCM model.

In [5], the authors applied this scenario-based modeling together with goal-oriented requirements technique to model the architecture of system. The objective is to make more strength connection between requirements and design during the early stages.

UCM visually illustrates causal relationships between *responsibilities* on the organizational structures of *components*. Responsibilities represent general behaviors (actions, tasks, etc.). Components are also general and could represent software entities (objects, processes, Web pages, etc.). They represent non-software entities, such as actors as well. The relationships are said to be causal because they link causes to effects by arranging responsibilities in sequence, as alternatives, or in parallel. Normally, UCMs show related and interacting use cases in a map-like diagram, and causal paths display the sequence of behaviors that explain a scenario along a use case.

### 2.3 Web application modeling techniques

The methodology reported in [12], Relationship Management Methodology (RMM), is for structured design of hypermedia systems. RMM uses for supporting the design of Web sites, and provides integration with databases. It employs the Relationship Management Data Model, which focuses on the design phase.

In the recent years many object-oriented modeling languages have been developed for software engineering. UML is the synthesis of many of such notations and it is considered as a standard for the industrial development software. It is a modeling language for object-oriented design and analysis.

Recent works [8, 9] introduced a set of extension notations to UML, named Web Application Extension (WAE), and it could be used with use-case driven software development process [13]. WAE employs UML, but it uses UML only as the notations. The extension gives the model not much object-oriented semantic since it sees the model from Web element perspective. WAE covers both server- and client-side of Web application architecture. The server-side elements (e.g. Server Page), and the client-side notations (e.g. ActiveX, Java applet, JavaScript code, and etc.) have been described by using the stereotype concept of UML. However, the object-oriented concepts (e.g. inheritance) are not concerned enough by the extension. A class notation in class diagram is used for representation an HTML page. WAE mainly focuses on the scripting page technology, such as ASP [16] and JSP [14].

In [10], the authors compared a number of methodologies for the design of a hypermedia application. The guidelines, stated in that work, gave some useful comparison aspects as the following:

- Methodology: it considered that the approach provides models and guidelines;
- Completeness: it considered that the most part of the lifecycle is deeply treated;
- Model: it considered that the approach gives some models;
- Tool CASE: there exists a tool CASE supporting the considered approach;

We follow the above features to compare our work with [8, 9].

## 3. WebForm-based Web Application Modeling Methodology

There are many definitions for a Web application [9]. In this paper, we scope a Web application to a program

that presents its output as an HTML document, transfers data via HTTP, and changes a state of business logic only on the server-side. The following subsections discuss the definitions, notations, and steps that appear in the methodology.

### 3.1 Definitions

Firstly, we recall two terms stated in [8, 9] *a client page* and *a server page.* A client page is an HTML page that displays on the client side, rendered by the user's Web browser program. A server page is a program on server that can produce one or more client pages. We introduce two of new terms *a WebForm class* and *a WebForm object*. The term "WebForm" is borrowed from Microsoft's' ASP.NET framework [17]. We define this term as WebForm because we look Web application architecture from higher point-of-view. Instead of modeling a Web application using the semantic of HTML pages directly, we put the related pages together and group it into a class using object-oriented concept, including its relationship. A WebForm class is an object-oriented class, which is defined as a template of a WebForm object. A WebForm object is an object of its corresponding class. The object produces a client page and acts like a server page stated in [9]. We do not define the semantic of any client page since its notation will be eliminated from our model. There is the difference between the semantic of the server page and the WebForm class that we need to point them out. The server page cannot be instantiated to object by its sense, but the WebForm class can. It is possible to create two distinguish WebForm objects from the same class; both of them have their own state for each connecting session.

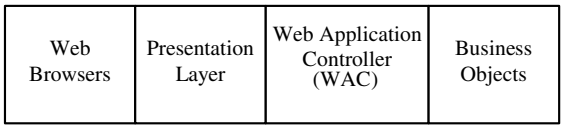


**Figure 1: Four-tier model for an event-driven Web application.**

Recall an event-driven programming framework for Web application development given in section 2. It has a back-end engine that manages the connecting sessions and binds itself with the presentation layer. We call that engine the *Web Application Controller* (WAC). Figure 1 shows the system architecture in the four-tier fashion.

We define the *BlankForm* class as the base class of all WebForm classes. In fact, a client page that is produced by the BlankForm class does not contain any Web element. Figure 2 shows the BlankForm class and its derivative. A WebForm class contains zero or more *Web components*. A Web component is not only an atomic HTML element, but also a compound one. A Web component receives an event, from WAC, to perform its appropriate action. A WebForm object works in the following manner. An object is created, and rendered as a client page by WAC. When user makes the requests for changing state of the Web application, WAC receives those requests, and then tells the object to change its state and re-produce a new client page.

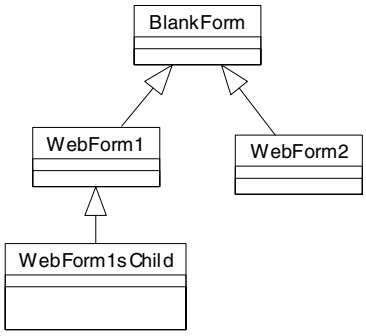


**Figure 2: BlankForm class, WebForm classes and their relationships.**

### 3.2 Notations and Methodology

This section describes our methodology, *WebForm-based Web application modeling Methodology* (WWM). It models Web applications from requirements specification, and use-case diagrams into UML and UCM diagrams using the definition of WebForm above. The final results of WWM are a set of UML diagrams, which consist of class diagrams, sequence diagrams, and a set of UCM scenario diagrams. The resulting class diagrams usually contain WebForm classes. A WebForm class in WWM must contain stereotype <<WebForm>>. The resulting sequence diagrams present how a WebForm object sends message to itself and the others. These represent hyper links among client pages. WWM utilizes the sequence diagram to model those links and their interaction. For more clarification of Web application modeling, we also apply the subset of UCM notations as a portion of our methodology. UCM is useful for describing the scenarios of the system. WWM describes architectural flows of Web application systems using its notations. We do not describe the list of UML notations here because of its well known. The UCM notations that used in WWM and a UML class in UCM's component concept are shown in figure 3.

The following steps describe how WWM performs to model Web applications starting from use-case model:

a) Define all actors of the system.
b) Define all candidate classes of the system.
c) Filter *WebForm* classes from the set of candidate classes, then construct a class diagram for them and put all WebForm classes into the class diagram.
d) For each use case, create its corresponding use-case realization.

e) For each use-case realization, construct UCM scenario diagram for it using UCM notations, and then put related WebForm classes (from the class diagram) as UCM components into the diagram.

i) Define a causal path.

ii) For each WebForm class on UCM scenario diagram, define its responsibilities and name them.

f) Repeat e) on the same UCM scenario diagram, if there exists causal paths that should be defined.

g) Remove WebForm class that has no responsibility from the UCM scenario diagrams.

h) Construct a sequence diagram for each UCM scenario diagram, and each causal path. Model all message sending that corresponds to each responsibility on the causal path in the sequence diagram.

i) Refine all diagrams with the above steps if necessary.

The above guidelines include steps from use-case driven development process [13]. WWM results both UML and UCM diagrams. It is iterative methodology, thus all diagrams can be further revised.

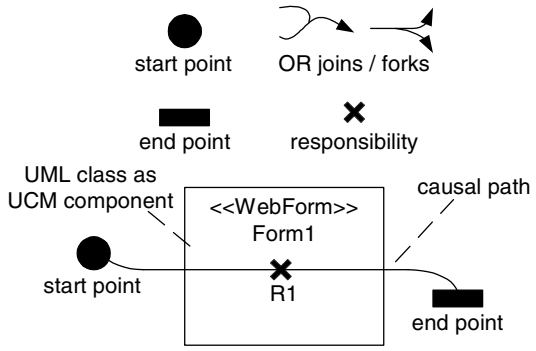


**Figure 3: UCM notations and UML class notation that are used in WWM.**

The comparison of our work with [9] is done following the partial guideline stated in [10]. The summarized result is shown in table 1.

## 4. Case Study

A case study illustrated in this section shows how WWM can be used to model a Web application. Firstly, we model a Web application starting from a use-case diagram, and then use the result for developing software. Our demonstrating environment is Borland Delphi. We use IntraWeb [11] as WAC, and BoldSoft's Bold for Delphi [4] as an implementation tool for the business objects. We have chosen IntraWeb as WAC because it is only product in the market that contains the inherent state management. A set of components, named IWBold, is developed in order to use with the case study. IWBold is built on the top of both IntraWeb and Bold architectures to connect them together. It acts as the Presentation-to-Business layer bridge.

| | WWM | WAE | Description |
|---|---|---|---|
| Methodology | + | | Our work introduces the methodology. WAE is a set of notations. |
| Completeness | + | | Our work covers use-case, scenario descriptive, static, and behavioral analysis. WAE can be used with the use-case driven development process, but it does not cover the steps for early scenario description. |
| Model | + | + | WAE is a rich set of UML extension, covering both server- and client-side, that gives model as Web elements. Our work emphasizes on using UML and UCM notations to model Web application with the object-oriented concept. |
| CASE Tool | + | + | Both of them employs UML, thus a tool such as Rational Rose can be used. However, our UCM scenario diagram's editing tool is under development. |

**Table 1: The Comparison of WWM and WAE.**

The interactions on the Web application are as the following:

1. The client browser receives the input from the user, and submits it to the WAC via HTTP.
2. The WAC receives the inputs, parses and sends them to the active WebForm object and IWBold components. IWBold components invoke the business objects implemented with Bold for Delphi to perform the business rules via Object Constraint Language (OCL) expression.
3. The business objects update their states and the corresponding database.
4. Each IWBold component on the active WebForm object produces HTML fragment by retrieving data from business objects using OCL expression.
5. The WAC sends the rendered HTML page back to the browser via HTTP.

Our case study is based on the Buildings and Owners demo came with Bold for Delphi. The given class diagram that represents business entities is in figure 4. We create an additional use-case diagram for starting analysis by WWM. The use-case diagram is in figure 5.

The building and owners class diagram in figure 4 contains three classes, the *Person* class, the *Building* class, and the *Residential Building* class. The Residential Building is derived from the Building class. A person object from the Person class has zero or one home, whereas a home is from the Residential Building class. The *Rent home* use case, illustrated in figure 5, assigns a home to a person. From the *Rent home* use case, we create the UCM model contains the causal path *rent_home*.

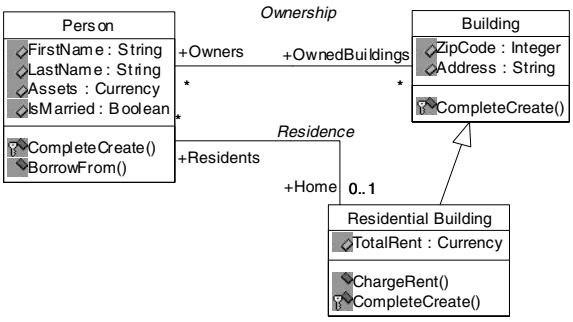


**Figure 4. The UML class model of Bold for Delphi's Buildings and Owners demo.**

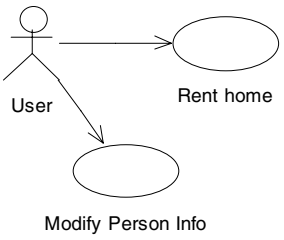


**Figure 5: Additional use cases created for the case study.**

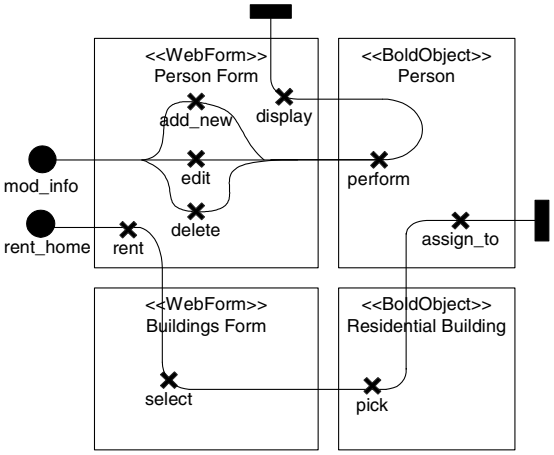


**Figure 6: The UCM model for the case study.**

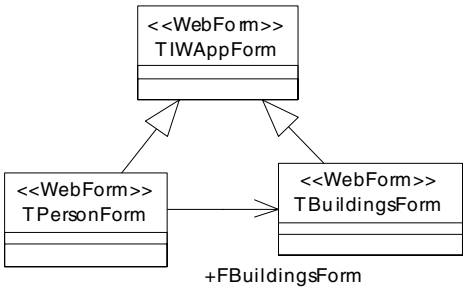


**Figure 7: WebForm classes that are derived from UCM model.**

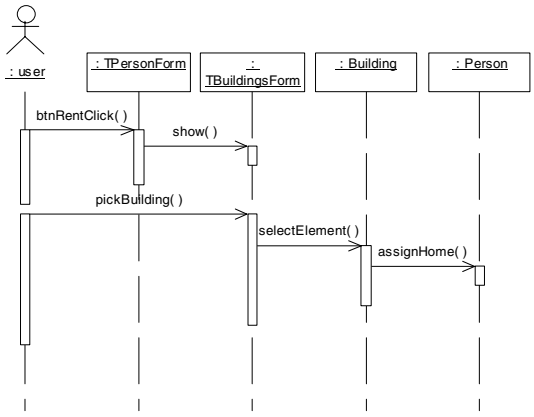


**Figure 8: Sequence diagram of the use case *Rent home* that is derived from UCM model and WebForm class diagram.**

Figure 6 shows two components as WebForm, with two components as BoldObject. The BoldObject stereotype does not have semantic in WWM, it only notes that an object of this type is implemented with Bold for Delphi. There are four responsibilities created on the rent_home causal path. Figure 7 shows that two UCM components, stereotyped with WebForm, become WebForm classes. They are derived from *TIWAppForm* class, which has the same semantic to the *BlankForm* class in WWM concept. TIWAppForm is also the real-world class defined in IntraWeb framework [11]. The sequence diagram in figure 8 is created using UCM model and WebForm class diagram from figures 6, and 7 respectively. We complete the sequence diagram by defining the four responsibilities as classes' method. The partial screen shots of design-time and compiled application are in figure 9.

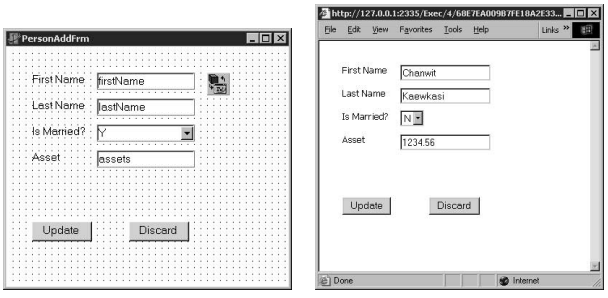


**Figure 9: The partial screen shots.**

The observation during development of the case study states:

- The analysis and design of a Web application system are straightforwardly done.
- UCM model clarify the scenario of the problem domain. This makes WebForm class, and sequence diagram easier to define.

- Client page modeling is omitted, thus the size of model is reduced.
- A WebForm class does not represent only a scripting page or a Web element. It is a real object-oriented class and has inheritance semantic, so it can be reused in both of model and code.

## 5. Conclusion and Discussion

We have developed WWM, the methodology for Web application modeling based on UML and UCM with fully object-oriented point-of-view. WWM is compatible and can be integrated to use in the use-case driven development process. Our methodology employed use-case diagram, class diagram, and sequence diagram from the concept of UML. Additionally, we complete the analysis steps for WWM by describing the scenarios of the Web application system using the subset of UCM notations. This paper shows that WWM covers all phases needed for modeling Web applications with various notations. We remains all object oriented semantic to the model, including class concepts, such as the inheritance. UCM notations clarify the problem domains with scenario diagrams. Moreover, we have done a few combination of UCM to the world of UML by assigning the stereotype concept to the components of UCM. Eliminating the client page semantic from our methodology makes the use of fewer notations comparing to previous works [8, 9]. The more complex Web applications are modeled the more significant benefit.

Our case study shows the working Web application that is modeled using WWM and built with our set of components. Usage of WWM makes the sample Web application model clearly connect to business objects since both of them employ object-oriented technology. The observation during development phase also found that the impact of changes do not effect the whole system directly because of the use of the components provided with WWM.

The integration of WWM with use-case driven development process is demonstrated in this paper. OMG's Model-Driven Architecture (MDA) [19] can be applied to generate source code of selected object-oriented language from the model. Since WWM models Web applications in the object-oriented fashion, it is possible to qualify the quality of a Web application with object-oriented techniques. The quality measurement metrics, such as coupling and cohesion can also be applied to analyze the model of Web application at the analysis phase.

We are now developing a tool to support a subset of UCM as an add-in of Rational Rose. Our methodology shows that stereotype concept in UML can also assign to components found in UCM. The tool is being implemented for making the concept realized with Rational Rose.

## References


1. Amyot, D., *Specification and Validation of Telecommunications System wtih Use Case Maps and Lotos*, in *School of Technology and Engineering*. 2001, University of Ottawa: Ottawa.
2. Amyot, D. and G. Mussbacher. *On the Extension of UML with Use Case Maps Comcepts*. in *<<UML>> 2000, 3rd International Conference on the Unified Modeling Language*. 2000. York, UK.
3. Andrade, R.M.C. *Applying Use Case Maps and Formal Methods to the Development of Wireless Mobile ATM Networks*.
4. BoldSoft, *Bold for Delphi Documentation*. 2001, BoldSoft AB.
5. Bruin, H.d. and H.v. Vliet. *Scenario-Based Generation and Evaluation of Software Architectures*. 2001.
6. Buhr, R.J.A., *Use Case Maps as Architectural Entities for Complex Systems.* IEEE Transactions on Software Engineering, 1998. **24**(12): p. 1131-1155.
7. Buhr, R.J.A. and R.S. Casselman, *Use Case Maps for Object-oriented Systems.* 1995: Prentice Hall.
8. Conallen, J., *Building Web Applications with UML.* The Addison-Wesley Object Technology Series. 1999: Addison Wesley.
9. Conallen, J., *Modeling Web Application architectures with UML*, in *Communication of ACM*. 1999. p. 63-70.
10. Costagliola, G., F. Ferrucci, and R. Francese, *Web Engineering: Models and Methodologies for the Design of Hypermedia Applications*, in *Handbook of Software Engineering and Knowledge Engineering*, Y. Zhai, Editor. 2001, World Scientific Publishing.
11. Hower, C., *IntraWeb Documentation*. 2001, AtoZed Software.
12. Isakowitz, T., A. Stohr, and E. Balasubramanian, *RMM: A methodology for structured hypermedia design*, in *Communication of ACM*. 1995. p. 34-44.
13. Jacobson, I., G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. The Addison-Wesley Object Technology Series. 1999: Addison-Wesley.
14. JavaSoft, *Java 2 Platform, Enterprise Edition http://www.javasoft.com/j2ee*, Sun Microsystems.
15. Kochikar, V.P., *The Object-powered web*, in *IEEE Software*. 1998. p. 57-92.
16. Microsoft, *Developer Resources for Active Server Pages http://msdn.microsoft.com/asp*, Microsoft Corp.
17. Microsoft, *.NET Framework http://www.microsoft.com/net*. 2002, Microsoft Corp.
18. Miga, A., et al. *Deriving Message Sequence Charts from Use Case Maps Scenario Specifications*. in *SDL' 01*. 2001.
19. OMG, *Model Driven Architecture http://www.omg.org/mda*, Object Management Group.
20. OMG, *Unified Modeling Language Specification version 1.3*. 1999, Object Management Group.
21. Saimi, A., et al. *Presentation Layer Framework of Web Application Systems with Server-Side Java Technology*. in *Proceedings of the The Twenty-Fourth Annual International Computer Software and Applications Conference (COMPSAC)*. 2000.