

Shorter Time-to-Market: Component-based Architecture and Object-oriented design for Complex Web Applications

(Essay for the course DIF 8901)

Jianyun Zhou

Department of Computer and Information Science (IDI)
Norwegian University of Science and Technology (NTNU)

Phone: +47 735 93488, Fax: +47 735 94466

E-mail: jjanyun@idi.ntnu.no

Abstract

Building large and complex Web applications is a difficult task. It involves graphical interfaces, navigational structures, business models and rules, software architecture and technologies, and many other aspects such as mobile communication. Furthermore, Web applications should be developed with short time. To deal with the extreme complexity, a good solution is to decouple different parts and aspects in Web applications. Decoupling gives also better possibility for reuse and parallel development, which in themselves are a clear way to reduce application development time.

In this paper we examine two aspects in Web application development: software architecture and application-specific design, and explore techniques that can be used in these two fields to improve reuse and achieve shorter time-to-market. For software architecture, a component-based Web architecture is built step by step to remove the tight coupling between a Web application's different parts. For application design, many mature design methods exist. We introduce the object-oriented hypermedia method OOHDM [1] and the aim is to explain how different concerns are addressed in this method independent of software architecture and implementation. Treating conceptual, navigational and interface design as separate concerns provides not only a sound ground for Web application evolution but also a basis for reuse and shorter development time.

In the end, we combine the benefits of the component-based Web architecture and the three different concerns in OOHDM [1]. The resulting architecture maximises opportunities for reuse and parallel development. As a consequent, time-to-market is reduced.

Key words: Web application, time-to-market, decoupling, reuse, component-based architecture, object-oriented design, OOHDM

1 Introduction

Building large and complex Web applications is a difficult task. These applications allow users to browser through large amount of information over Internet and perform operations such as querying or updating on it. In some cases such as ecommerce, Web applications need to be integrated with business logic to provide services to the user. Moreover, different kinds of terminals come into the world. Web applications need to combine new telecommunication technologies to support access from mobile devices. All these factors make developing a complex Web application time-consuming. We need to design graphical interfaces and navigational structures, understand business logic and rules, and come up with flexible and evolving software architecture and other implementation technologies.

However, on the other hand, Web applications should be developed with high reliability and with short time. Time-to-market is a very important attributes for reach financial success with a Web application project. Being the first company to release a certain type of product means that the market anticipates all future products in the same domain resemble this product. This gives a clear market advantage.

Complexity and time limitation bring new challenges for Web application development and need careful thoughts and considerations for systematic engineering.

To deal with the extreme complexity in Web applications, a good solution is to decouple different parts and aspects in Web applications. Decoupling gives also better possibility for reuse and parallel development, which in themselves are a clear way to reduce application development time.

Firstly, we separate application-specific aspects from software architecture and implementation technologies. A good-structured architecture can be reused by a lot of Web applications, and similarly, good-designed application models can be re-implemented using evolving architecture. Development time of Web applications can thus be considerably reduced.

In addition to this macro-decoupling, we consider software architecture and application design respectively to identify possibilities for reuse and high productivity.

As many software engineering experience shows, layered architecture and component-based architecture design [4] can improve reuse and achieve shorter time-to-market. By applying these techniques, we develop a unified view for Web architecture based on components in Section 2 of the paper. Different kinds of components are defined according to their roles in the Web application.

In the Web application design domain, there are a number of mature methods existing, such as HDM [2], OOHDM [1], and WebML [3]. Most of them recognize the clear separation between application behaviour aspects, and navigation and interface concerns. The separation provides possibility for reuse and parallel development of design artefacts, and thus is also an obvious way to reduce development time. In Section 2 we present an object-oriented design method OOHDM [1] for application design. The aim is to explain how different concerns are addressed in this method independent of software architecture and implementation.

In the end, application design models need to be combined with certain software architecture. The independence between them makes the combination flexible. Generally, a Web application can be implemented by any possible architecture, whether component-based or not. Similarly, certain Web architecture can realise any possible Web applications, whether object-oriented designed or not.

In order to maximise reuse possibility and achieve shortest Time-to-Market, we consider the combination of component-based architecture and object-oriented Web application design method. Section 4 shows how different concerns in the application design fit into the component-based architecture. When limitations appear, the architecture can be easily improved due to the component-based nature.

2 Web architecture design

Architecture design and application-specific design are two aspects of a Web application. When we decouple them from each other we have better opportunities for reuse.

We discuss Web architecture design in this section. With respect to reuse and time-to-market, we view the standard parts in Web architecture as components, which are in turn composed of components. According to the complexity of Web applications, Web architecture is decomposed into different levels.

Component-based architecture

Whether they are static, dynamic or transactional, Web applications almost have multi-tier architecture. Compared to n-tier client/server architectures, client in Web architecture plays a different role. In client/server architecture, the client part is responsible for at least a portion of the functional processes. In Web architecture, however, the browser does not carry out the functional processes except some script codes.

Figure 1 gives the general Web architecture. The standard components in three tiers are:

1. Browser
2. Server
3. Back-end system

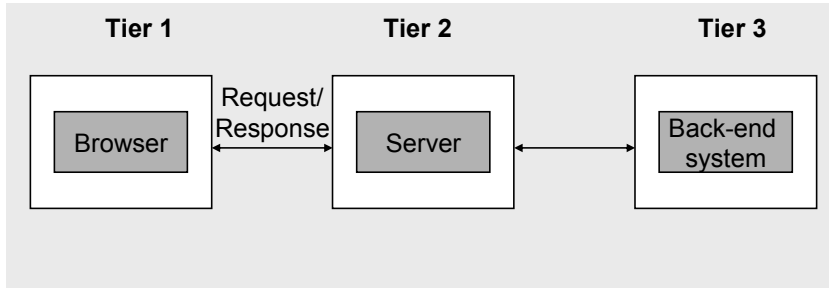


Figure 1 General Web architecture

For the simplest Web applications this architecture is feasible. Server component is a HTTP server that accepts HTTP request from Browser and returns HTML pages to it, and Back-end component is a set of HTML files.

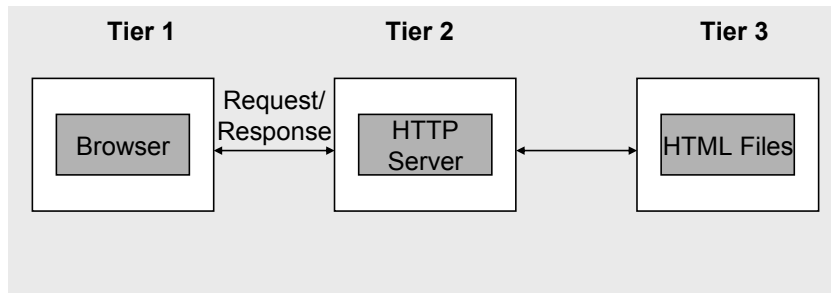


Figure 2 The simplest Web architecture

In order to produce dynamic Web pages, script languages are introduced into Server components. For better reuse, Server component is decomposed in to two kinds of components (Figure 2): HTTP server and application server. Application server is responsible intercepts request from HTTP server, analyse and process the requested source script file including directly database operation in order to return the generated page to HTTP Server.

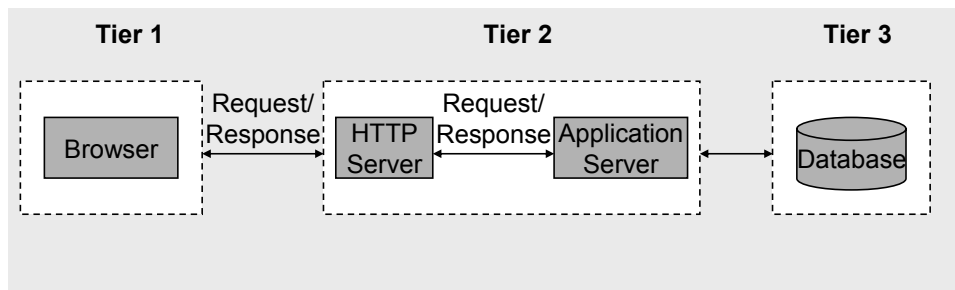


Figure 3 Web architecture with Application Server Component

As more application logic is added to the Web application, the above architecture present limitations, for example, script pages include too much application logic. So we need further component decomposition to improve reusability. As showed in Figure 4 Application sever component can be decomposed into Web component that is responsible for interaction and dynamic page producing, and business logic component that performs application logic and database access functionality.

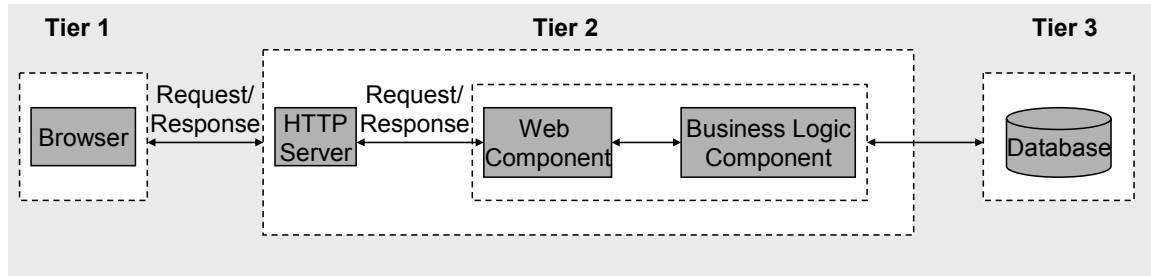


Figure 4 Web architecture with Business Logic Component

Furthermore, Web component can be divided into two kinds according to their roles:

1. One kind (Controller) deals with application flow. It acts as a controller, intercepting request from HTTP server and assigning it to other components.
2. The other kind (presentation) is responsible for producing dynamic HTML pages. It does not include any logic processing; only simply gets data from other components and dynamically inserts it into pages.

So we have got the following architecture (Figure 5) for Web applications. It is a component-based architecture and conforms to Model-View-Controller (MVC) architecture [9]. Different kind components play different roles and realise different functionality. It can greatly improve reusability of the components. At the same it makes reusing of components possible during development process and thus reduces time-to-market.

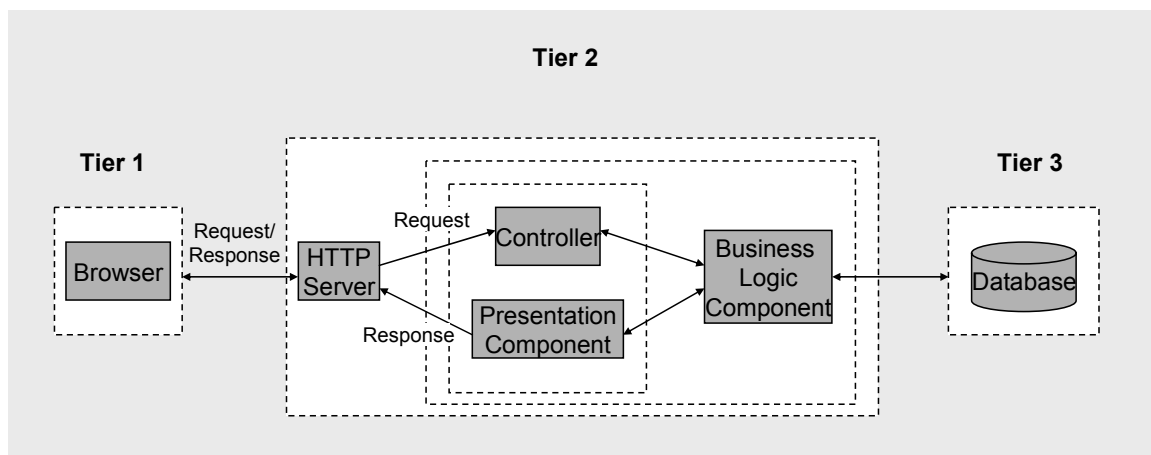


Figure 5 MVC Web architecture

Furthermore, as business logic becomes more complicated, 3-tier architecture showed in Figure 4 and Figure 5 can be easily extended to multi-tier architecture by introducing more tiers with business logic components between Tier 2 and Tier 3.

Implementation -- J2EE architecture

Today a major solution to complex Web application architecture is J2EE [11]. J2EE architecture in its nature is a component-based architecture and J2EE applications are composed of components. In this section, we will show how to implement the component-based architecture showed by Figure 5 in J2EE technology.

The J2EE specification defines the following J2EE components [11]:

1. Application clients and applets are components that run on the client.
2. Java servlet and java server page (JSP) components that run on the server
3. Enterprise JavaBeans (EJBs) components are business components that run on the server.

When J2EE architecture is chosen to implement Web applications, Java servlet and JSP components are used to interact with a browser, which implement the Web component and EJBs implement the business logic component.

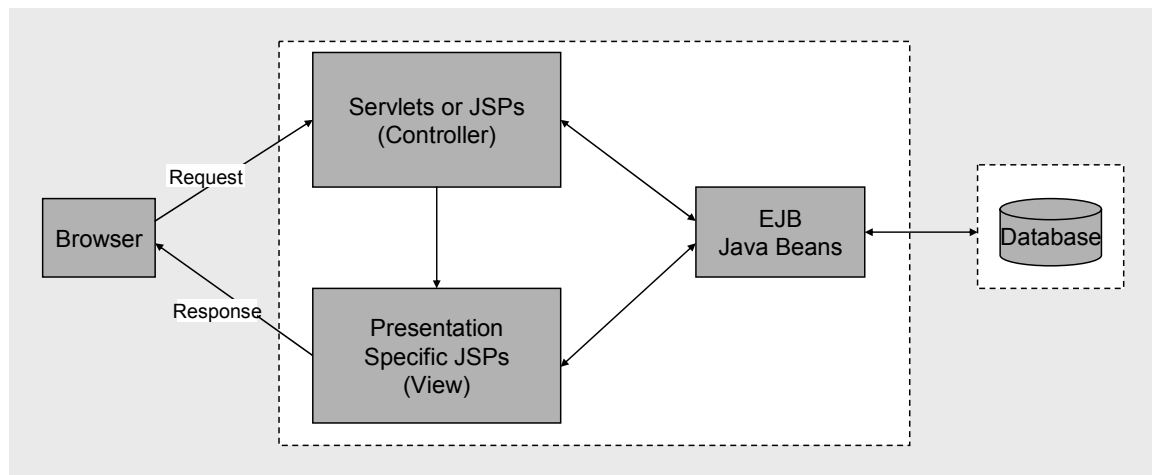


Figure 6 J2EE architecture

In this architecture (Figure 6) Controller implemented by Servlets or JSPs deals with client requests, initiate EJBs, and assign tasks to Presentation-specific JSPs and EJBs. Presentation-specific JSPs contain no or little application logic, and only get data from Controller and EJBs to produce dynamic pages.

3 Application-specific design

In addition to architecture design discussed in Section 2, Web applications have another important issue: application-specific design. As mentioned before, this separation is beneficial for reuse and independent evolution of architecture and application.

It has been argued that good Web applications should be first good hypermedia applications [8], since the Web is based on the hypertext paradigm, in as much as it is composed of pages (nodes) which can be linked to each other through links (URLs).

However, the Traditional software design methods don't contain useful rules to deal with hypertext issues [12]. For example, they do not provide any notion of linking, and talk about how to integrate hypertext into interface. When many companies are moving their existing interactive applications to WWW, their task is to add hypertext functionality on the top of existing application behaviour.

Building good navigation structures and integrating it with conceptual models and interfaces are thus extra tasks of Web application design, which is beyond traditional software design.

In addition, as the size, complexity and number of applications grows, design methods are needed to help dealing with complexity, which can only be dealt with by separating of modelling concerns in a clear and modular way. Separation provides not only a sound ground for Web application evolution but also, as we explained before, a basis for reuse and shorter development time.

Nowadays there are a number of solutions existing for hypertext design, such as HDM [2], OOHDM [1], WebML [3], etc. They differ from each other in that some are object-oriented, some are not; some are based on conceptual model, some are not. In this section we briefly introduce the Object-Oriented Hypermedia Design Method (OOHDM), which is both object-oriented and based on conceptual model. The introduction here provides an overview of OOHDM method, and a general understanding about how object-oriented techniques are used in the method. The aim is to emphasize how the rational in the method is beneficial to reuse and time-to-market. So for a complete understanding of method and its application, please refer to other documents such as [1, 6, 7].

OOHDM

The key concept in OOHDM is that Web application models involve a conceptual model, a navigation model and an interface model. Treating conceptual, navigational and interface design as separate concerns allow the developer to concentrate on different concerns one at a time; in particular, application (business) data is separated from the Web page contents and these contents, in turn, are separated from the pages (look-and-feel) [5]. It is thus easy to move traditional applications to WWW by building navigation model on the top of existing conceptual model. Time to build Web application is reduced.

Application behaviour

The Conceptual Model is conforming to traditional software design and presents two kinds of objects: those that will be eventually perceived as nodes in the navigational model, and those that provide computational support for the application, encapsulating behaviours such as algorithms, access to databases, etc. It reflects also relationship between objects. In OOHDM the conceptual model is specified using UML notation. The resulting model may possible serve as a basis for many applications, and does not include any navigation specific information.

Consider the domain of conference paper review systems (i.e., Web-based applications that help manage program committees in the process of receiving and evaluating papers for a conference) [10]. A possible conceptual model for this domain is given in Figure 7.

Navigation modelling

Navigation is one essential distinguishing feature of hypermedia applications, in which the user of an application in this domain navigates in a space made out of objects [13].

The navigation model consists of navigation schema and navigation context. The navigational schema contains Node classes and Link relationships that indicate the basic navigation architecture. These nodes are not the same as the conceptual objects, but rather objects customized to the user's profile and tasks. This customization is achieved using the view mechanism between objects, analogously to views in databases.

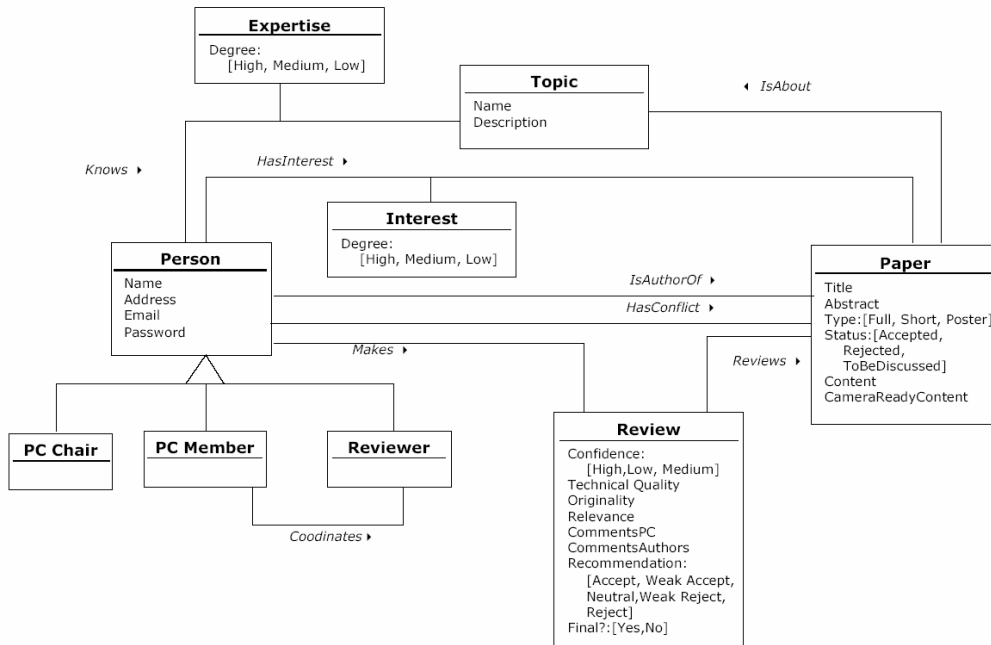


Figure 7 Conceptual model example [10]

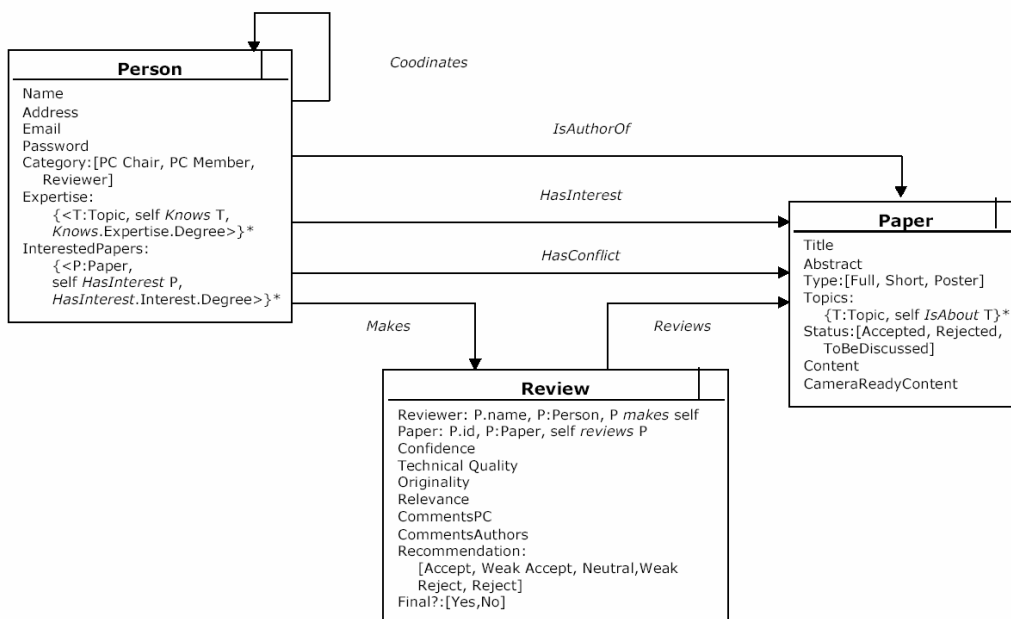


Figure 8 Navigation class model example [10]

“View” for conceptual model in Figure 7 is showed in Figure 8. Each Node class can be defined by combining attributes from different conceptual classes, while links indicate navigation paths and reflect conceptual relationships. For example, “Person” has incorporated “Expertise” and “Topic” as an attribute; similarly, “Paper” has also incorporated “Topic” as an attribute. Links in navigation schema are derived from “semantic” relationships (e.g. the relationship IsAuthorOf between Person and Paper); we can not express “finer grained” links such as those among papers of the same author or papers reviewed by the same reviewer.

In OOHDM Navigational Contexts are defined as Sets of nodes that may be accessed using indexes and traversed in some order or freely. An index is a navigation object that acts as an access structure containing references to other navigation objects. Navigational contexts are specified using a slight addition to the UML notation by taking into account sets of nodes.

In Figure 9 we show a possible navigational context schema of the example. Rectangular boxes with full borders denote contexts (sets of nodes); the label of each grey area indicates the class of the elements in the contexts within it; boxes with dashed borders indicate access structures (indices).

In the example, it is specified that, for instance, papers may be navigated in several ways, such as “By Reviewer” (i.e., all papers assigned to some given reviewer), or “By Topic” (i.e., all papers classified under a given topic), etc.

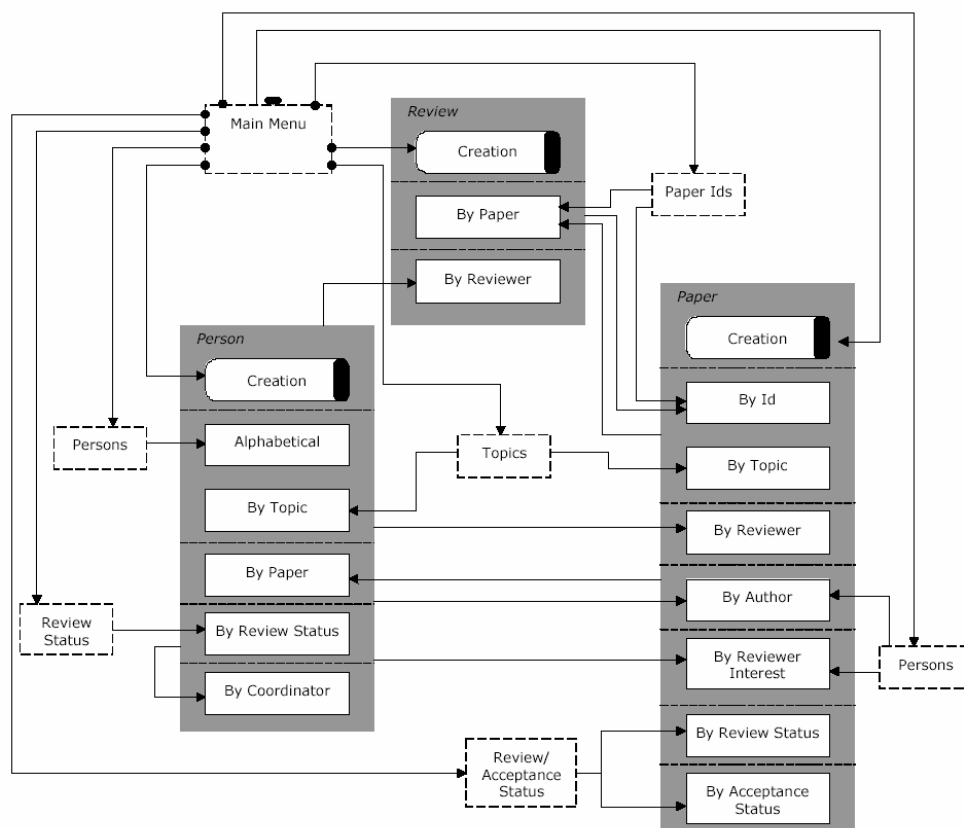


Figure 9 Navigation context schema example [10]

Interface design

Finally, the third concern involves the specification of the user interface. Navigation objects are not directly perceived by the user; rather, they are accessed via interface objects. For a given navigational model (i.e. a particular user profile) we may have to define different interface models according to the particular interface device he/she will use, e.g. an internet browser, a mobile phone, etc.

From the above introduction, we can see that the most outstanding contribution of the OOHDM approach is the clear identification of these three design concerns in an implementation-independent way [6]. Whether we choose what software architecture and implementation technology, the design models can be reused.

4 Mapping design models into Component-based architecture

OOHDM and other mature Web application design methodologies provide modular design and reuse of design documents. However they concentrate on the application domain specific aspects. How to implement these models is also not an easy thing. We will in this section discuss the combination of these models with the component-based architecture discussed in section 2. In this way we can combine the advantage of the object-oriented design method and the component-based architecture, to maximize the reuse possibility and shorten time-to-market.

Limitations of the component-based architecture

The component-based architecture discussed in Section 2 provides rational for building modular Web application. However, it does not completely fulfil the requirements of Web applications since it does not consider characteristics of Web applications as the hypermedia application. Especially, it does not take into the navigation concern into account.

As showed in Figure 5, the Presentation component gets the data from Business logic component, and produces dynamic HTML pages. When relating this process to the three different concerns in OOHDM, application data are kept by Business logic component, while both navigation model and interface are handled by the same component Presentation component (a JSP object in J2EE architecture).

In addition, the basic idea that navigation always occurs within a context is absent in the architecture. For example, if we want that the same node has a slightly different structure depending on the access context, we have to use the context as a parameter for the Presentation component, which contains conditional statements to differ from various contexts. The presentation component becomes overloaded, difficult to manage and evolve. It is clearly an obstacle for component reuse.

In a word, the limitation of the component-based architecture (Figure 5) is that navigation aspect and interface aspect are mixed and realised by the same component. Thus the object-oriented design method will lose its most charming advantage.

Revised architecture

To solve the above problem, the further decoupling of the Presentation component is demanded. We propose the creation of Navigational Node component to encapsulate all navigational logic and Page Layout component for interfaces.

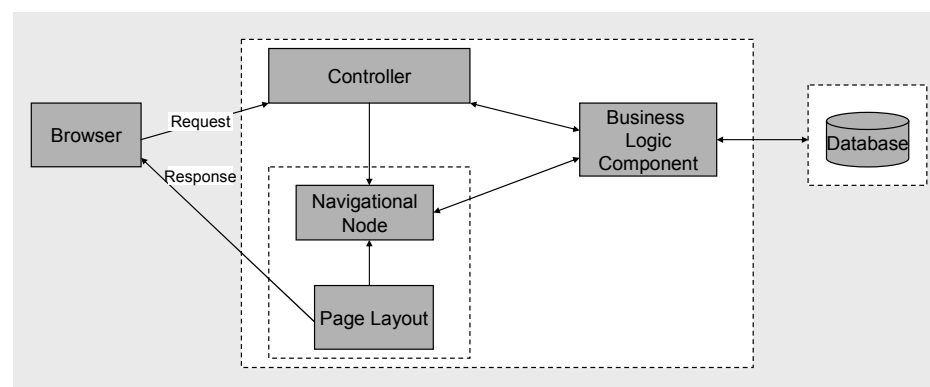


Figure 10 Revised Web architecture

Figure 10 illustrate the separation. In the new architecture, while the Page Layout component is responsible for the page layout structures and producing response pages, the Navigational Node component manages the node contents, updates them according to data from Controller and Business Logic component, and deals with context-specific information.

This architecture combines the benefits of component-based architecture and object-oriented design method. Possibility for Reuse and parallel development is maximised and can be achieved at different levels.

J2EE architecture can be convenient extended to implement this revised architecture.

5 Summary

In this paper we try to examine different aspects in Web application development with respect to time-to-market and reuse issues.

Two primary concerns are software architecture design and application-specific design. At first we respectively discuss techniques that can be used to realize reuse and shorter time-to-market in these two fields.

A component-based architecture has the potential to remove the tight coupling of an application's parts, eases the reuse and parallel development of different parts and therefore reduces development time. In Section 2 a component-based architecture for complex Web applications is developed step by step.

Object-oriented design methods for Web applications are the topic of Section 3. We present an object-oriented design method OOHDM [1] and the aim is to explain how different concerns are addressed in this method independent of software architecture and implementation. Treating conceptual, navigational and interface design as separate concerns allow the developer to concentrate on different concerns one at a time [6]; in particular, application (business) data is separated from the Web page contents and these contents, in turn, are separated from the pages. Separation provides not only a sound ground for Web application evolution but also a basis for reuse and shorter development time.

In the end, we combine the component-based Web architecture and the object-oriented Web application design method to maximise reuse possibility. As a result, a revised architecture is proposed.

6 References

- [1] D. Schwabe, G. Rossi: "An Object Oriented Approach to Web-Based Application Design", Theory and Practice of Object Systems (TAPOS), Special Issue on the Internet, v. 4#4, pp.207-225, October, 1998
- [2] F. Garzotto, P. Paolini, D. Bolchini, S. Valenti: "Modeling-by-Patterns of Web Applications", Proc. International Workshop on the World Wide Web and Conceptual Modeling, WWW CM'99, Paris, November 1999
- [3] S. Ceri, P. Fraternali, A. Bongio: "Web Modeling Language (WebML): a modeling language for designing Web sites", Proceedings of the 9th International Conference on the WWW (WWW9), Amsterdam, May 2000
- [4] L. Bass, P. Clements, R. Kazman: "Software Architecture in Practice", Addison Wesley, 1998
- [5] D. Schwabe, G. Rossi, L. Emerald, F. Lyardet: "Web Design Frameworks: An approach to improve reuse in Web applications", Proceedings of the WWW9 Web Engineering Workshop, Springer Verlag LNCS, 2000

- [6] G. Rossi, D. Schwabe, F. Lyardet: "Web application models are more than conceptual models", Proceedings of the First International Workshop on Conceptual Modeling and the WWW, Paris, France, November 1999
- [7] S. Daniel, V. Patricia: "The OOHDM notation", available at <http://sol.info.unlp.edu.ar/notacaoOOHDM/>
- [8] L. Baresi, F. Garzotto, P. Paolini: "From Web Sites to Web Applications: New Issues for Conceptual Modeling", Proc. ER'2000 Workshop on Conceptual Modeling and the Web, Lecture Notes in Computer Science 1921, Springer Verlag, Heidelberg, 2000
- [9] G. Krasner, S. Pope: "A cookbook for using the model-view controller user interface paradigm in Smalltalk-80", Journal of Object-Oriented Programming, 1(3), August/September 1988
- [10] D. Schwabe, G. Rossi, L. Esmeraldo, F. Lyardet: "Engineering Web Applications for reuse", IEEE Multimedia, 2001
- [11] SUN Microsystems, Java 2 Enterprise Edition (J2EE) Official site, <http://java.sun.com/j2ee/>
- [12] G. Rossi, D. Schwabe and A. Garrido: "Design Reuse in Hypermedia Applications Development", Proceedings of ACM International Conference on Hypertext (Hypertext'97), Southampton, April 7-11, 1997
- [13] G. Rossi, A. Garrido and D. Schwabe: "Navigating between Objects: Lessons from an Object-Oriented Framework Perspective", ACM Computing Surveys, 1999