International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-3, Issue-1)

Research Article | **January 2014**

An Efficient Design of Object-Oriented Hypermedia with J2EE Technology for Web Apps

Kancherla Bala Krishna, Jampani Satish Babu
Assistant Professor, Dept. of Information Technology
NRI Institute of Technology, Perecharla, GUNTUR, India

Abstract:

In any environment, web-based application development is a difficult task, since these applications include various features, like graphical interfaces, navigational structures, business models, and wireless communications, as well as other issues, such as serving a multitude of users, and the need for shorter development time. To overcome these complexities, it is indispensable to use web-based application designs and software architectures. In this paper, we use a multi-tiered, component based software architecture based on the J2EE technology. This technology separates the different parts of a complex application, and the development time is reduced through the use of components and distributed structure. Unfortunately, the J2EE technology lacks rich hypermedia and navigational structures, and the mapping of prepared designs to components is not carried out properly. For this reason, in this article, a combined architecture called OOHDM-J2EE is defined which uses the benefits of both schemes and solves the mentioned problems, provides the possibility of reuse, and further reduces the development time.

Keywords: OOHDM -Object-oriented Hypermedia Design Method, WIS-Web Information Systems, J2EE -Java 2 Enterprise Edition, UML-Unified Modeling Language, MVC-Model-View-Controller, OOHDM-J2EE- Object-oriented Hypermedia Design Method- Java 2 Enterprise Edition, PDA-Personal Digital Assistant.

I. INTRODUCTION:

Nowadays, the web is one of the most important communication channels which can be accessed on various platforms around the clock all over the world. The internet encompasses more than 10 trillion pages and has about 1 billion users. While the nineteenth century was the pinnacle of industrial revolution, the beginning of the current century is characterized by the information revolution, and the web is the principal engine of this revolution. Noting the ubiquity of the web, many information systems are accessible via the web, and they are called Web Information Systems (WIS). Modern WIS applications present the information in meticulously crafted, multimedia-enabled texts. This system is not sufficient for the ever-growing need of publishing high volumes of information over the web. Information-intensive web features, like electronic cities, organizational portals, social engineering sites, e-shopping centers, digital libraries, and the like, are present in many web-based applications.

II. OPERATIONAL TERMINOLOGY:

Navigation: Guiding the users from one place to another. Even though the web is not a physical space, navigational procedures are necessary, so that the users can move in the information space.
Context: A set of navigational nodes with a common property
Container: A place on the server where application components are situated and are executed therein. Containers provide middle-tier services for the components.

PROBLEM DESCRIPTION:

Web information systems allow a user to search in the plethora of information in the internet and perform tasks like research and updating. Some of the specific features in these programs are as follows:

- They must be developed with high confidence in a short period of time, because the time to market is a critical factor in the success of web-based applications.
- They must be able to guide the users in the application space in a satisfactory manner.
- They must be able to respond to large numbers of users.
- They must integrate with modern telecommunication technologies, in order to support access through mobile devices.
- They must support a high degree of reusability.
- Therefore, in this article, we are going to find a suitable solution for designing applications with these features.

THEORETICAL BACKGROUND:

We use the component-based architecture of J2EE (Java 2 Enterprise Edition) to present solutions for reusability, rapid development, multi-user service, and rely on the object-oriented hypermedia design paradigm for navigation,

graphical interface design, and reusability. We believe that the combination of these two methodologies maximizes the capabilities for reusability and rapid development for web-based applications.

OBJECTIVE:

Separation of various parts of web development paves the way for reusability and parallel development, effectively shortening the application development time However, in most cases these benefits are lost in the implementation phase, mostly because the current implementation methodologies have a meager support for mechanisms of abstraction and composition. For this reason, our major objective is a mapping from the design artifacts to implementation components, and adding navigational capability to applications.

MOTIVATION:

The traditional design of WIS applications has created dissatisfaction among users due to long development and response time, lack of navigational capabilities, and dozens more problems. To overcome these shortages, we have no other way but to combine software engineering approaches (component-based architecture) with web engineering techniques.

SNAPSHOT OF THE ARTICLE:

This paper consists of 4 sections. In Section 2, we will review previous studies, including component- base architecture (J2EE) and object-oriented hypermedia design. In Section 3, we will focus on the mapping of the design constructs and we will add navigational capability to the components. Finally, in the last section, conclusion and future research directions will be discussed.

RESEARCH BACKGROUND:

Multi-tiered Design:

When an application program is developed for a standalone computer, it is expected naturally to include all input from functions, data processing logic, or data access routines. Figure 1 shows such a program at run time.

In Figure 1, if anything is changed in one part of the program, it will affect all other parts of the part as an independent layer, we can provide a possibility for reusability, decoupling of program parts from each other, more flexibility, and freedom in implementation of each part independent of other parts of the application. In addition, more flexibility in presentation design and the possibility of running the application on a server makes this design methodology an ideal prototype for web environments.

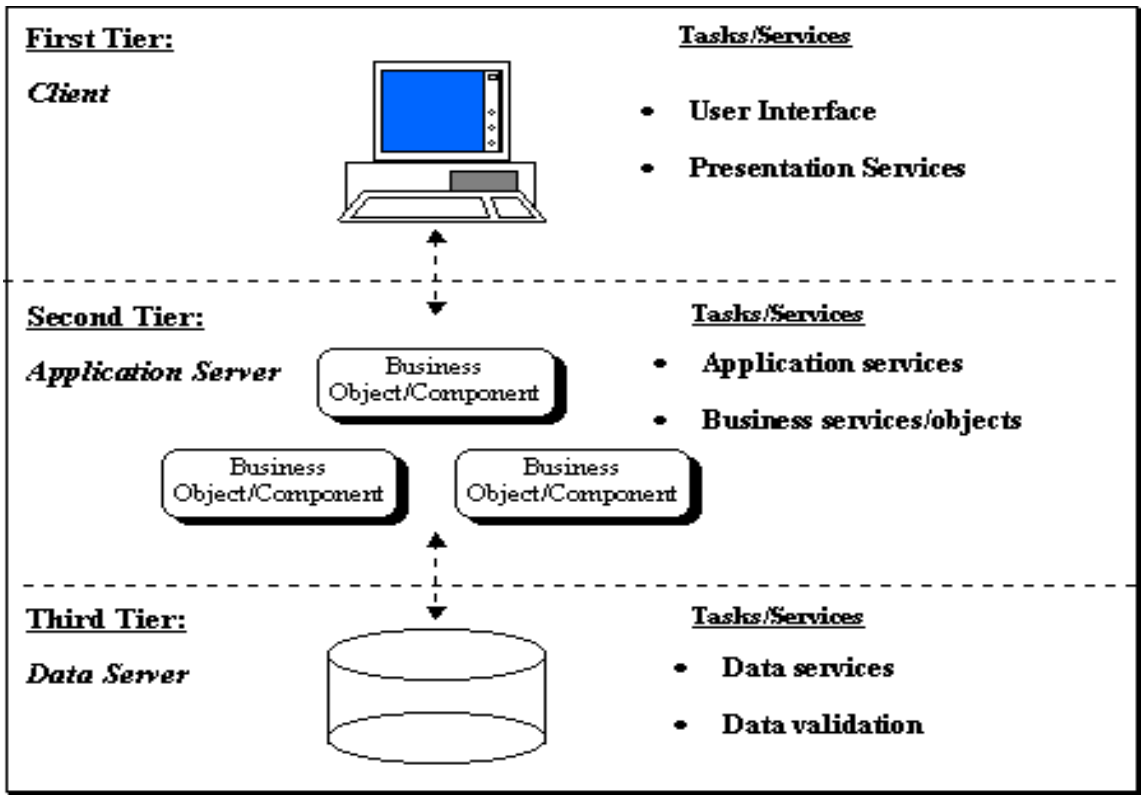

Fig. 1: Architecture of an Integrated System.

Figure 2 shows a three-tiered structure for this integrated design.

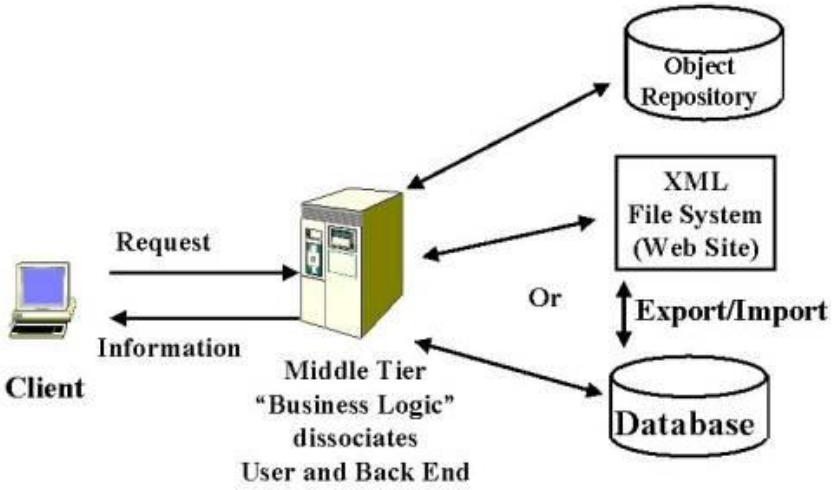

Fig. 2: Architecture of a Three-tiered System

If we have an integrated system and convert it to a distributed (multi-tiered) system, so that many users can connect to it concurrently and it can be connected to multiple servers and databases at the same time, there are a few things that we may worry about. These include remote client access, session and transaction management, security, and resource management. Now if we decide to implement and maintain all of these functionalities, which are collectively called system service or middleware layer, we will have a difficult time doing the job.

Therefore, today, there are technologies that provide us with the necessary middleware services. One of these is the J2EE technology, which is one of the main solutions for construction of complex web-based applications (Keogh, 2002). J2EE is inherently a component-based architecture. Figure 3 shows two J2EE-based applications which are divided into several layers, including:

- Client component layer, which runs on the client machine.
- Web component layer, which runs on the J2EE server.
- Business component layer, which runs on the J2EE server.
- Enterprise Information System (EIS) layer, which runs on the EIS server

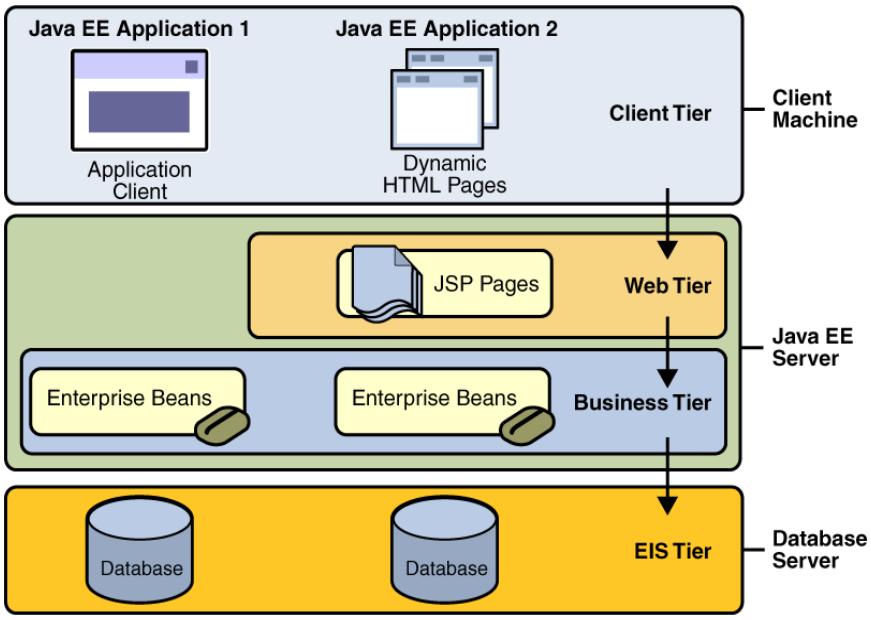

Fig. 3: J2EE Application Layers

In Figure 3, each of these layers is composed of components that are executed inside a container. The container provides these components with middleware services. Additionally, the J2EE server runs in a multithreading mode, which is an important factor and enables us to respond to large numbers of users simultaneously, minimizing the response time (Sun Microsystems, 2006).

III. OOHDM:

Good application programs must be good hypermedia applications. Since the web is based on the hypertext design principle, we can use web-based design guidelines to support the hypermedia and navigation functionalities (Ceri & Fraternali, 2000). By separating the various parts of web based development, we can reduce problem complexity, enhance reusability, and shorten the development period. According to the Object-oriented Hypermedia Design Method (OOHDM), the development of hypermedia applications is broken up into four steps, including conceptual design, navigational design, abstract interface design, and implementation. These activities are performed as a mixture of iterative and incremental methods.

CONCEPTUAL MODELING:

During this activity, we build a conceptual schema representing objects, their relationships and collaborations existing in the target domain. We model the target domain using the Unified Modeling Language (UML), an inherently object-oriented modeling tool, in the form of classes and associations between them. Figure 4 contains a simple schema for an online magazine. In this model, there are stories, which can be essays, translations or interviews; an interview is an aggregation of questions and answers. Every story has an author, and an interview is also related to the person who grants the interview

Object-oriented modeling uses aggregation, generalization/specialization, and a packaging concept, called sub-systems, to enhance reusability and shorten development time (Schwabe & Rossi,1998).

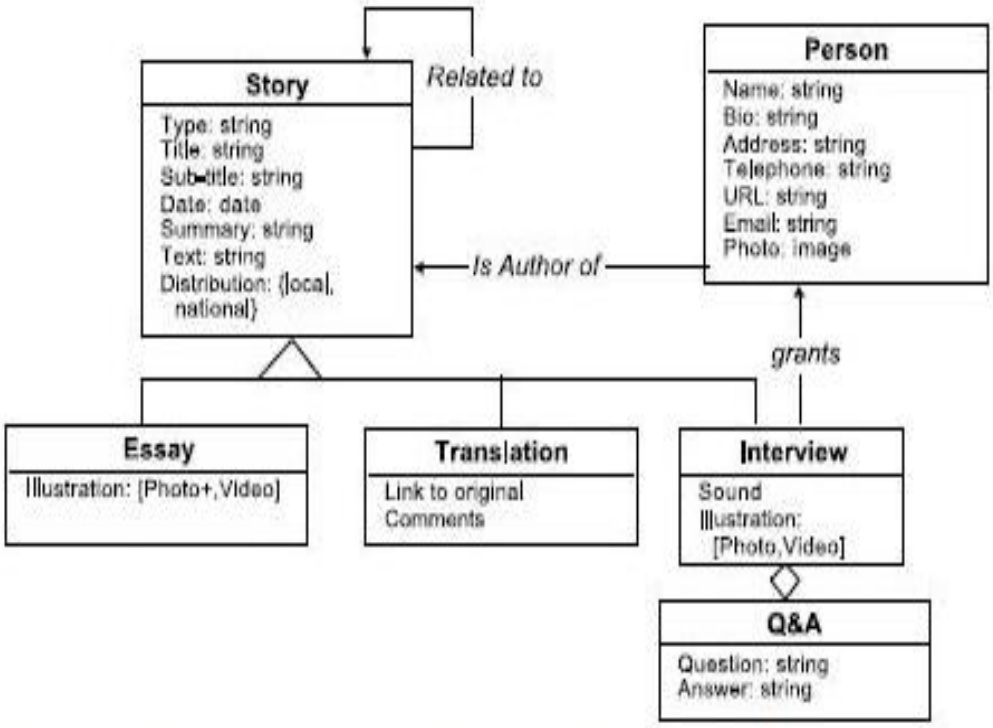

Fig. 4: Conceptual Model for an Online Magazine (Schwabe & Rossi, 1998).

Navigation Design:

Navigation is an important aspect in web-based application development, so that the user should be able to reach the desired part of the program with just three clicks. Unfortunately, most organizations do not pay enough attention to navigation and hypermedia functions when deploying their web applications and this confuses the users, so that they cannot reach their desired goals. And in some applications, even though the developers have considered navigation, a consistent navigational structure is not applied to the whole application, so that the site becomes a huge set of disheveled links and nodes

. In OOHDM, navigation is built as view over a conceptual model, thus allowing the construction of different models according to different users' profiles. Navigation design is expressed in two schemas, the Navigational Class schema and the Navigational Context schema. In OOHDM, there is a set of pre-defined types of navigational classes: nodes, links, and access structures. Navigation nodes, which are built as views of conceptual objects, are called navigation objects. If some of the classes in the conceptual model are not related to navigation, they may be deleted in navigational objects or be converted to attributes in other classes. This case is shown in Figure 5.

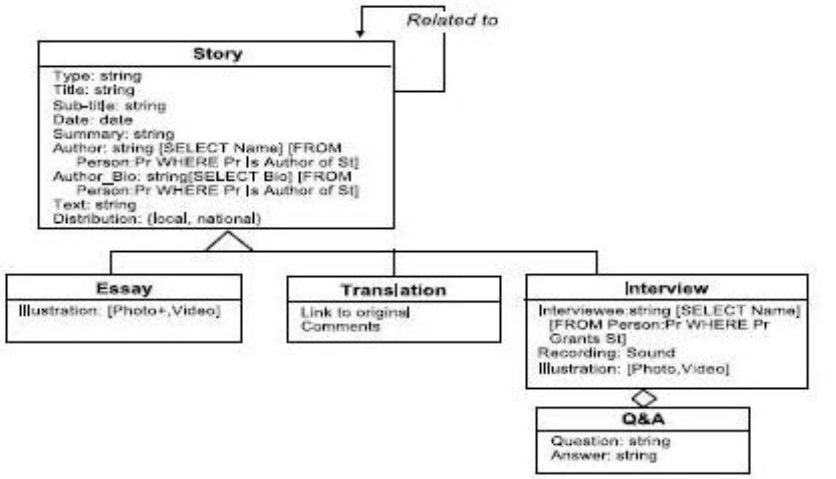

Fig. 5: Navigational Schema of the Online Magazine.

Access structures, such as indices and guided tours; represent possible ways of accessing nodes. Navigational design must take into account the way in which the user explores the hypermedia space.
Unfortunately, nodes and links are not enough to fulfill this goal—this is the point where navigational contexts appear. The purpose of defining contexts is to make it possible to classify navigational objects in many different ways. For example, we can create "Story by Section" or "Story by Author" sections. Notice that class Person does not appear; the author's attributes are now part of the Story. The same happens to the Person who grants an Interview.
In Figure 6 we show the Navigational Contexts Schema for the online magazine.

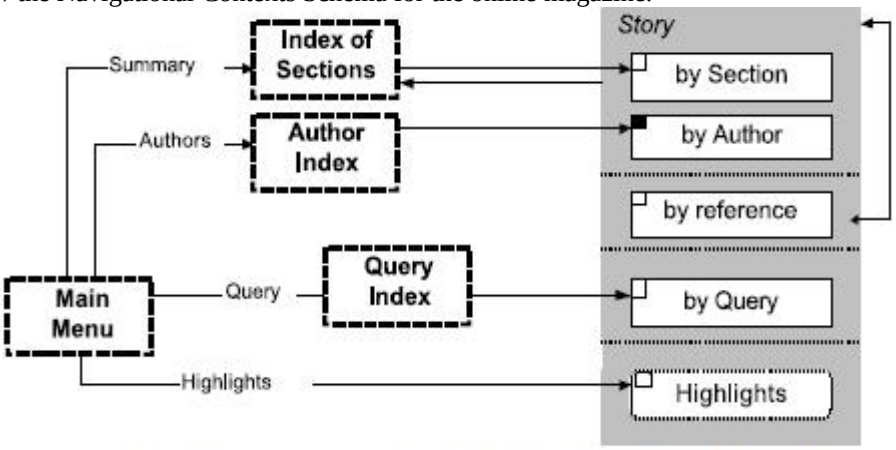

Fig. 6: Navigational Context Schema for the Online Magazine Application

Abstract Interface Design:

Eventually, what the user sees is the interface. Even the navigational objects are not felt directly by the user, but they are presented to the user by the abstract interface. For a single navigation model, we can create different interfaces, so that the users can access the application from different devices, like mobile phones, or various types of browsers.

IMPLEMENTATION:

Any application-specific model, like OOHDM design, must be eventually implemented using an Implementation technology. As it was mentioned previously, we have chosen the J2EE technology for implementation, because it supports component-based, multi-tiered, multithreaded designs and has many other advantages, as well. In the remaining sections of this paper, we will describe the combination of OOHDM and J2EE, OOHDM-J2EE, which is an architecture paradigm that makes it possible to decouple design decisions concerning the application domain from decisions about navigation and interface design. Furthermore, this methodology exploits the benefits of both technologies and maximizes the reusability opportunities.

Why We Need Software Architecture?

Mapping design documents into implementation artifacts is a difficult task. For example, as it is shown in Figure 7, the designer may erroneously include business rules inside JSP pages, instead of placing them inside EJB components. For this reason, there is a visible trend in web application architecture toward modular approaches. For example, we can

consider that the MVC architecture (which is consisted of three modules: model, view, and controller) is used for improving the development of J2EE-based applications, and has enhanced earlier proposals in which most architectural design decisions were left to the programmer's savvy or ability (Jacyntho 2005).

In the MVC architecture, the "View" which is built by JSP components is responsible for implementing the user interface (page layout), and the "Model" which is made of EJB components is responsible for implementing the business logic of the application.

While the MVC provides a set of structuring principia for building modular interactive applications, it does not completely fulfill the requirements of web applications for providing rich hypermedia structures. Concretely, in a naïve use of the MVC, nodes and their interfaces are handled by the same JSP components. In addition, navigational contexts are totally absent in the MVC architecture. If we want to support navigational contexts in JSP, we have to use the context as a parameter for the JSP page, and write conditional statements to insert context sensitive information as appropriate. The JSP becomes overloaded, difficult to manage and evolution becomes practically unmanageable. Now, if we use different JSP pages for different contexts, we will have even more problems. Summarizing, the main problem is that the navigational logic (node and context management) is dealt with in the JSP page, which mixes it with the interface layout, which is also a responsibility of the JSP page.

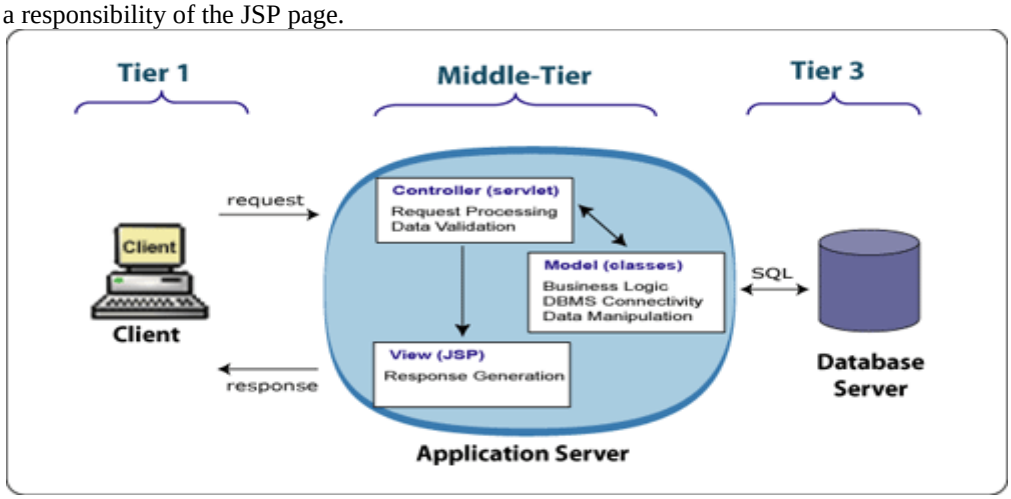

Fig. 7: The MVC Architecture and mapping of design documents

OOHDM-J2EE:

To solve the above-mentioned problem, we propose to build a navigation layer that includes all of the navigational logic. While JSP is responsible for the page layout, the navigational layer manages the node content and handles context-specific information. The OOHDM-J2EE architecture corrects the view part of the MVC architecture by adding a navigational layer. Now, the modified view sections comprises of the user interface (page layout) which is created by JSP and navigational nodes.

Figure 8 shows the top-level components of the OOHDM-J2EE architecture and the interactions between them at the time of handling a request.

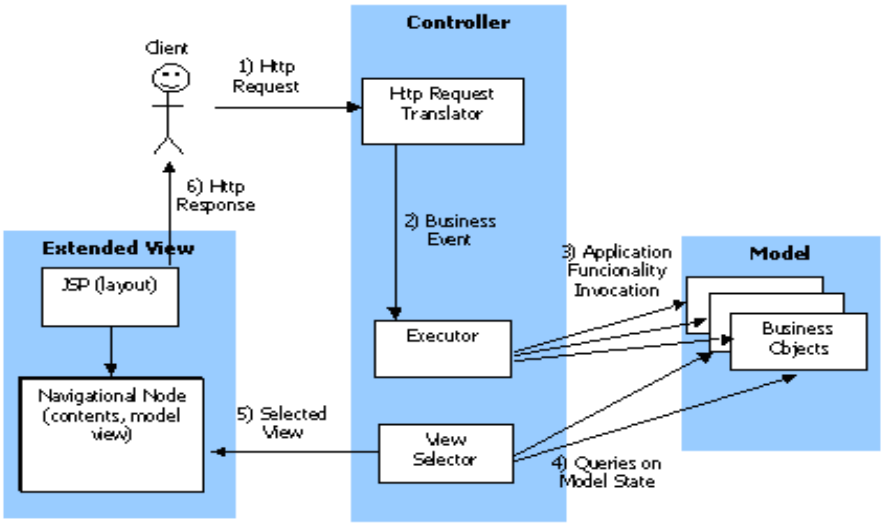

Fig. 8: Architecture of OOHDM-J2EE

IV. RESULTS AND CONCLUSION:

In this paper, we tried to decompose integrated systems into multi-tiered systems, in order to create light-weight clients and enable rapid development of web-based applications. For the implementation of this three-layered structure, we used the J2EE framework, which is a component based technology developed with the Java programming language. The design model used in this paper was object-oriented hypermedia design methodology.

PROOF OF THE THEORY:

In this article, we presented a solution for the problem discussed in Section 1.2, namely:

1. In Section 2, we used OOHDM methodology to guide the users in the site and applied the hypermedia principles for the application.
2. Using a multi-tiered, component-based architecture, we realized decoupling of layers form each other and enhanced reusability capacity and shortened development time.
3. We exploited the advantages of the J2EE technology, like multithreading and rich libraries, to respond to large numbers of users.
4. In Section 3, with separation of the presentation layer form other parts and with definition of multiple presentation logics, we could serve users on different devices (PC, PDA, etc.).

THE GOAL:

Using a combination of J2EE and OOHDM, we mapped the design artifacts created in OOHDM easily into components and maximized reusability and time to market. Subsequently, we added navigation as an important concept to the web-based applications.

V. RELATED WORKS, DISCUSSION, AND COMPARISON:

J2EE technology, in comparison with other technologies like .NET, has advantages like scalability which is evident in some of use cases like electronic cities, university exam results portal, gas smart card, centralized account of Bank Melli Iran, Ministry of Interior's national polling site, and many others. Using a J2EE application server, we can exploit the many libraries and classes of this technology for various purposes. An application server is a software infrastructure that is responsible for application lifecycle management. In other words, it responds to the needs of a large-scale application, including database, web server usage, mail sending (via SMTP protocol support in J2EE), authentication (using Kerberos technology support in J2EE), smart card transactions (using the smart card API in J2EE), and many others, throughout the lifecycle of the application. Currently, J2EE is the only system that can be used for supporting smart cards in the application. Scalability is an important advantage of J2EE over other technologies like .NET framework, and for this reason, this technology is used for large projects where a mistake in choosing the suitable architecture and occurrence of incompatibilities could be unforgivable. The difference between these two technologies is that learning .NET is easier and its complexity is lower than J2EE framework. The learning curve for Java is less steep (a steep curve would mean easier learning), while learning .NET is much more rapid. The run time in .NET is similar to Java. The JSP server has the capability of clustering and the servlets run in a multithreading environment. All of these advantages make J2EE preferable over .NET for serving large population of users.

FINDINGS OF THE ARTICLE:

The OOHDM-J2EE architecture provides these capabilities:

- Centralized control of HTTP requests.
- Centralized control of business events execution.
- Centralized control on the selection of response interfaces.
- Further separation between navigation logic and interface aspects.
- Since the components are run in a multithreading environment, it has short response time and it can respond to large groups of users simultaneously.
- It adds navigation support to J2EE technology.
- Clear separation between application and presentation logic.
- It supports navigation contexts and sets of navigational nodes.
- Decoupling between JSP pages and business events.
- Single entry point for serializing requests of the same user.
- Centralized control of navigation logic.
- Changes in the navigational structure do not impact on the business logic, and vice versa.

FUTURE RESEARCH TOPICS:

We plan to make a tool for automatic creation of OOHDM-J2EE code. Additionally, we intend design methodologies for the semantic web. These will be based on model-driven web methodologies. In other words, we will use all the advantages of model-driven methods to create a semantic web methodology, and we will add information integration concepts to this methodology.

REFERENCES:

1. Ceri, S., & Fraternali, L. (2000). Web modeling language (WebML): A modeling language for designing Web Sites. http://sol.info.unlp.edu.ar/notacaoOOHDM
2. Frasincar, F. (2005). *Hypermedia presentation generation for semantic web information systems.* Doctoral thesis, Technische Universiteit Eindhoven, pp.1-29.
3. Jacyntho, M. (2005). *A software architecture for structuring complex web applications*. Rio de Janeiro, Brazil: Department Information.
4. Keogh, J. (2002). *J2EE The complete reference*. Berkeley: McGraw-Hill, Osborne.
5. Schwabe, D. (2004). *A software architecture for structuring complex web applications*. Departamento de Informática. PUC-Rio, Rio de Janeiro, Brazil. Retrieved April 2007 from www.inf.puc-rio.br