Software Architecture and Desgin for Online Registration System

Issa Hussein Manita
Department of Software Engineering
Misurata University, Libya
eisa@hotmail.com

Abstract—**This paper presents an example of Software Architecture and design for online registration system. Software design is an early phase of the Software Development Life Cycle (SDLC). During this phase, software designers model the system and assess its quality so that improvements may be made before the software goes into the production phase. The goal of software design is to build a model that meets all customer requirements and leads to successful implementation. As software systems continue to grow in scale, complexity, and distribution, their proper design becomes extremely important in software production. The architecture design representation is derived from the system requirement specification and the analysis model. Multi-tier architecture is commonly used for distributed systems. It usually consists of three element types: client, middleware server, and data server. The Unified Modeling Language (UML) used to model the problem domain; describe the user requirements; identify significant architecture elements during software design, such as classes and objects; describe behavior and interactions among these elements; and organize the software structure, specify its constraints, describe the necessary attributes, and more.**

Index Terms: **Software engineering, software designers, Software Architecture, Multi-tier, Unified Modeling Language**

I. INTRODUCTION

The architecture design defines the relationship between major structural elements of the software, the styles and design patterns that can be used to achieve the requirements defined for the system, and the constraints that affect the way in which architecture can be implemented [1]. The architecture design representation is derived from the system requirement specification and the analysis model. Software architects and designers are involved in this process. They translate (map) the software system requirements into architecture design. During the translation process, they apply various design strategies to divide and conquer the complexities of an application domain and resolve the software architecture.

Received 29 May, 2018; revised 20 June, 2018; accepted 11 July, 2018.

Available online Jul 13, 2018.

Software architecture plays a very important role in the Software Development Life Cycle. The architecture design provides a blueprint and guideline for developing a software system based on its requirement analysis specification. The architecture design embodies the earliest decisions that have a decisive impact on the ultimate success of the software product. The design shows how the system elements are structured, and how they work together. An architecture design must cover the software's functional and nonfunctional requirements as well. It serves as an evaluation and implementation plan for software development and software evolution [9].

This paper describes "how the system shall work". It illustrates the three standard components of software design: the architectural design, module interface design and internal module design..

The system shall be accessible, with differing privilege levels, to public users, current students, faculty members, graduate program directors, administrative officers and the system monitors.this system allows potential students to apply for admission and allows current students to register for courses on the web, allows faculty to check related information, such as view class list, view teaching schedule, allows graduate program director to manage course registration information, allows administrative officers to manage students' academic records and faculty members' teaching records, and also allows system monitor to open new accounts, update accounts and maintain the web site [5].

II. ARCHITECTURAL DESIGN

A. Design Rationale

The goal of the architectural design is to derive the structure of the system and determine the major data structures. Due to the nature of the software development life cycle that is the constant changes in both functional requirements and non-functional requirements, the key principle of a good software design is the concept of "design for change". The software should have the potentials to adapt to as many and a severe change as possible during the course of development and even after the system is already in use. The realization of these potentials depends on that the architecture of the system is

designed to bear important features such as information hiding, model-view-control structure and loose coupling of functions between modules. The design should also be abstract and easy to be understood [9 ,8].

The Application Framework for the system is designed to be an integrated and consistent collection of APIs, protocols, services, and convention that provides an open, standard based, scalable and complete foundation for developing and deploying the web application solutions. At the heart of the Framework is a single, unifying Java-based programming model for building system. The JavaServer Page TM (JSP) is a key component serving as the preferred request handler and response mechanism. This technology ensures us to easily develop high quality, maintainable Web applications.

Figure 2.1 illustrates the three-tier conceptual model of this Web application. The client tier is composed of multiple clients, which request services from the middle tier. The middle-tier consists of two sub-tiers, the Web Server and Application Server. The Web server contains JSP pages as the event handler, and the Application server contains JavaBean as the dynamic component of logical unit. They access data through the database tier, apply logical rules to data, and transfer back the results to the client tier. The JSP page and Bean in the server of middle tier play a vital role in the three-tier application.

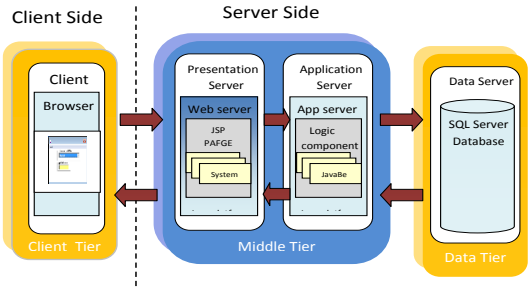

Figure 2.1: System Conceptual Model

The separation of user interface and program logic in a JSP page allows for a very convenient delegation of tasks between web content authors and developers. As a result of this, the implementation will be more reusable and manageable.

A web-based application such as this system differs from a traditional client-server system in that the connection between the client and server only exists during a page request. Once the request is fulfilled, the connection is broken. All activities on the server, as affected by the user, occur during the page request. Business rules on the server are only activated by the execution of JSP pages inside the requested pages. In other words, business objects are not always accessible when handling individual user interface requests.

III. SYSTEM ARCHITECTURE

A. *Architectural Diagram*

The system has to be able to respond to asynchronous events from users. Users are allowed to interact with the system by viewing or querying and even updating information through the system. The interactive architecture best addresses these requirements. On the other hand, the system contains no predefined sequence of action and only responds according to user inputs (either data input or control input), thus qualifies it as an event-driven interactive system. The system should also have the characteristics of the user friendliness and the database orientation.

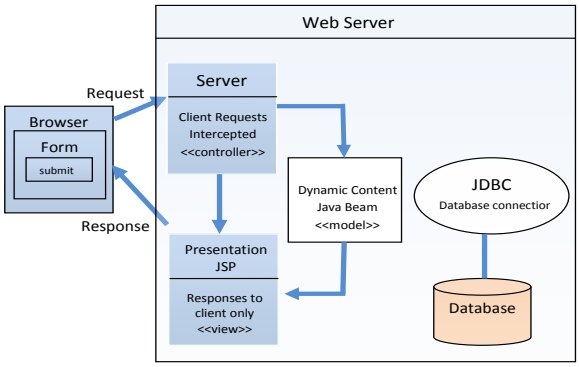

Figure 2.2: Architecture Diagram of the System

The architecture, shown in Figure 2.2, is an abstract approach for serving dynamic request processing, by using JSP. The system uses JSP to generate the presentation layer and to perform process-intensive tasks. I choose Model-View-Controller (MVC) as the predominant pattern for system design. MVC separates the Model from the View and the Controller, thus offering a way of increasing the system modularity. Here, the JSP page acts as the ***controller*** and is in charge of the request depending on the user‘s actions. The JSP page forwards the request to the JavaBean as ***model***. Note particularly that there is no processing logic within the JSP page itself; it is simply responsible for retrieving any objects or beans that may have been previously created, and extracting the dynamic content from that server for insertion within static templates.

As shown in the system architecture diagram in Figure 2.2, the system consists of the following subsystems:

User Interface: The UI allows general users to interact with the system through the web using an internet browser, as well as authorized users to use the system directly through their personal computer with different accessing privileges through inputs (mouse clicks, keyboard strokes etc.).

Web Server: This is the major part of the design, which is to handle the user requests from user inputs. The JSP page in Web server uses logical rules in JavaBean dynamic component, to get the results. **JSP Package (the Controller)** consists of sub-components for university module and six users, as shown as in Figure 2.3. **Presentation JSP Pages (the View)** generates the presentation pages of the response from JavaBean, and send them to clients for displaying in the user interface browser.
JavaBean: It is a dynamic component served as **the Model**. It receives the information from Web server and connects the database with the JDBC. It contains formatting beans, command beans and application logic.
JDBC: It is the connection between JavaBean and database.

Database: To store the whole information of the system, to receive the SQL queries and to give the query results out.

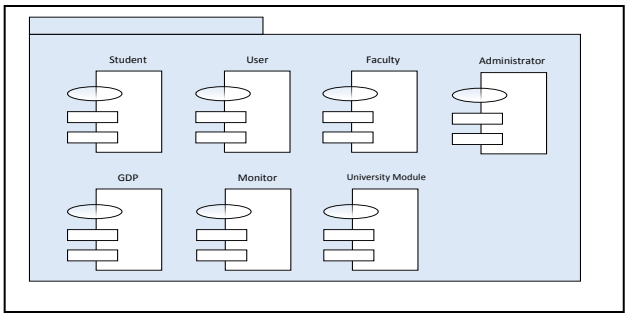

Figure 2.3: The Sub-Components in JSP Package

B. *Deployment Diagram*

The deployment structure for the system is relatively simple (Figure 2.4). The client Web Browser is on the client PC. The connection between Web Browser and Web Server is standard internet protocols, TCP/IP. It is important to note that because Tomcat has a standalone HTTP server built in; it must be run on a different port than Apache. The system architecture of the system ensures that it is platform independent.

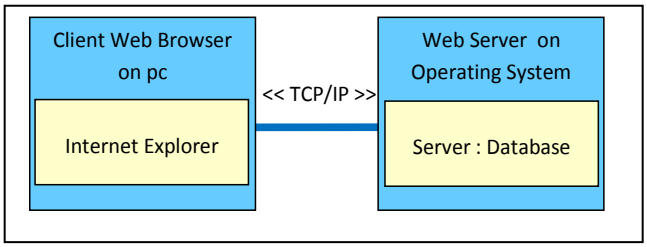

Figure 2.4 The Deployment Diagram of the System

C. *Object Model and Class Diagram*

The object model demonstrates the static structure of the real-world events and organizes them into abstract components. It describes real-world object classes and their relationships to each other. The analysis of our object model comes from the problem analysis in the SRD, our knowledge of the application domain of student services, which I am very familiar with, and finally the general knowledge of the real world.

Class diagram illustrates classes and the relationships between them. The class diagrams of each component of the system are divided into two sections. For the purpose of simplifying the architectural design, i wrapped the user interface (UI), WebServer, BeanTemplate, JDBC and Database into the section of the major modules.

IV. MODULE INTERFACE SPECIFICATION

A. *User Interface Subsystem*

The UI subsystem is to present the graphic user interface to the system users. UI receives the input from users and sends the user requests to the Web server. This subsystem contains one class UI.

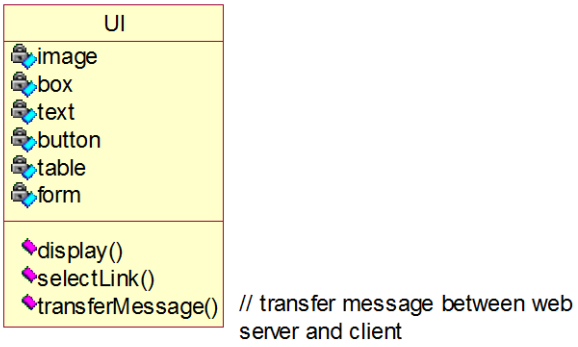

Figure 3.1 The Class Diagram of UI

B. *Web server subsystem*

The Web server subsystem receives the client's request, compiles the request processing program, executes the program, and returns the results to the client by calling user interface's transferMessage(). The Web server subsystem contains one class WebServer. (Figure 3.2)

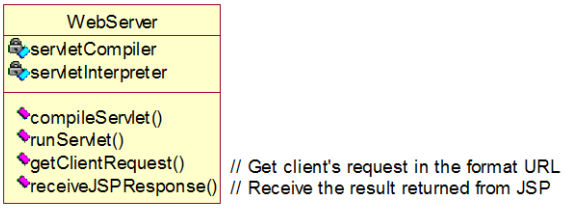

Figure 3.2 The Class Diagram of WebServer

C. JDBC subsystem

The JDBC subsystem frees the logic programmer from caring about handling the details of concrete databases. This is a layer between the logic component and the database. The logic programmer just declares a driver for the target database, creates a SQL statement, passes it to JDBC to execute it, and gets results. JDBC also provides services to map the type in host program language with the type in the concrete database, such as String type in Java is mapping Char type in Oracle, and traverse the return results.

JDBC subsystem contains one class JDBC. (Figure 3.3)

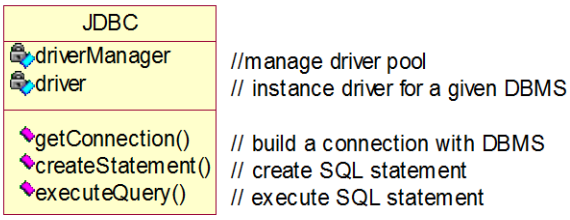

Fig. 3.3 The Class Diagram of JDBC

D. Database subsystem

This subsystem stores all tables specified in the database design. The SQL mechanism resides in the database subsystem. This subsystem processes SQL statements transferred from JDBC subsystem, interprets SQL statements, executes SQL statements, and returns query results to JDBC subsystem.

Database subsystem contains one class DB. (Figure 3.4)

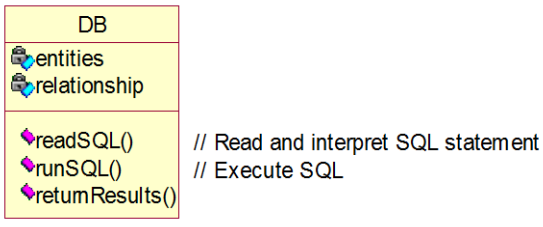

Fig. 3.4 The Class Diagram of DB

V. CONCLUSION

It has been shown that better visualization for software life-cycle can be achieved using OOD approaches while Clearance and common understanding between users, customers, developer and designers can be achieved using UML. The OOD techniques such as, prototyping have been used in the development life-cycle of software for efficient and faster system implementation. The use of OOD and OOP can make the system more flexible and the change of requirement can be handled easily. The use of iterative and evolution models can save a considerable amount of time.

VI. FUTURE WORK

Measuring the software quality based of the OOD and OOP is an open area of research. Comparing the quality of OOD and structural programs in term of speed and performance might be trade-off. Testing in software engineering can follow many paths and critical software engineering need different testing techniques.

REFERENCES

[1] Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrion, "*Fundamentals of software engineering*" ,2003.

[2] James A. O'Brien, McGraw-hill Irwin, "*Introduction to information systems for the e-business Enterprise*,"2003.

[3] H. Poor, "*An Introduction to Signal Detection and Estimation. New York: Springer-Verlag*", 1985.

[4] John W.Satzinger, Southwest-Missouri state University, Robert B. Jackson, Brigham Young University,Steohen D. Burd, University of New Mexico, "*System Analysis and Design, in a changing world , course Technology*", 2002.

[5] Stephen R. Schach, Vanderbilt University, *"Introduction to Object-Oriented Analysis and Design with UML and unified process*", 2004.

[6] Stephen R. Schach, Vanderbilt University, "*Object-Oriented and classical software engineering"*, McGraw-hill Higher Education, 2005.

[7] Christopher fox, "*Introduction to software engineering design processes principles and patterns wit UML2*" , Pearson Addison Wesley, 2006.

[8] David Avison, Guy Fitzgerald, *"Information system development methodologies techniques and tools*," 4th edition, McGraw-hill companies, 2006.

[9] Kai Qian, Chong-wei Xu, Xiang Fu, Jorge L. Díaz-Herrera, Lixin Tao "*Software Architecture and Design Illuminated*", 2010