Summer 2017

Thesis Paper

On

An Object – Oriented Approach to Web – Based Application Design and Implementation

An Object – Oriented Approach to Web – Based Application Design and Implementation

A Thesis

Submitted to the Department of Computer Science and Engineering,

BRAC University, Bangladesh.

By:

MD. ASHIKUL HOQUE (ASHIK)	**13101022**
FARIAH RASHID	**13101090**

Under the supervision of:

Mr. Hossain Arif
Assistant Professor
Department of Computer Science and Engineering
BRAC University, Bangladesh.

In partial fulfillment of the requirements for the degree of Bachelor of Science (B.Sc.) in Computer Science and Engineering

Summer 2017

BRAC UNIVERSITY

DECLARATION

We, Md. Ashikul Hoque (Ashik) and Fariah Rashid, students of the Department of Computer Science and Engineering of BRAC University, aware of our responsibility of the penal law, declare and certify with our signatures that our research-project based thesis entitled "An Object – Oriented Approach to Web-Based Application Design and Implementation" is entirely the result of our own work and has been a bona–fide work under the supervision of Mr. Hossain Arif. Materials of work such as images, figures, tables and citations in this paper are accepted by our Supervisor. We have faithfully and accurately cited all my sources, including books, journals, handouts and unpublished manuscripts, as well as any other media, such as the Internet, letters.

We understand that

- literal citing without using quotation marks and marking the references
- citing the contents of a work without marking the references
- using the thoughts of somebody else whose work was published, as of our own thoughts are counted as plagiarism.

We declare that we understood the concept of plagiarism and I acknowledge that my thesis will be rejected in case of plagiarism.

2017, August, 21

Signature of Supervisor

Mr. Hossain Arif

Signature of the Authors

Md. Ashikul Hoque (Ashik)

Fariah Rashid

ACKNOWLEDGEMENT

We would like to thank our supervisor Mr. Hossain Arif for his constant support towards the successful completion of the project. We would also extend our gratitude towards all the faculty members of the CSE department of BRAC University for their support in different course – works towards our degree.

Last but not the least, we are also very grateful to our parents and friends for their inspiration and support and also BRAC University for giving us the opportunity to complete our B.Sc. Degree.

TABLE OF CONTENTS

CHAPTER 01 – ABSTRACT

Here, in this study, we present a comparative study to analyze the performance differences between Traditional Approach and Object – Oriented Approach using the Object-Oriented Hypermedia Design Method (OOHDM) in Web – Based Application. Traditional approach has a lot of models that deal with different types projects such as waterfall, spiral, interactive and v – shaped, but all of them and other lack flexibility to deal with other kinds of projects like Object – Oriented. The approach of using object – oriented techniques for designing a system is referred to as object–oriented design. Object–oriented development approaches are best suited to projects that will imply systems using emerging object technologies to construct, manage, and assemble those objects into useful computer applications. Object oriented design is the continuation of object-oriented analysis, continuing to center the development focus on object modeling techniques [1]. This study aims to figure out the differences between these two approaches based on features like Requirement Specifications, Maintainability, Correctness, Understanding Requirements, Simplicity, User Involvement, Flexibility and few more. The exploration from this study helped us to find out the best approach out of these two for any average – sized web-based project. We have conducted the study through the use of detail analysis. We have validated the result with experimental results. Finally, we have concluded the paper with some suggestion for future work on this topic.

CHAPTER 02 – INTRODUCTION

Web Services technology is based on the concept of service-oriented computing. Web services are standard that integrate Web-based applications through connecting and sharing of business processes across the network where applications of different vendors, languages, and platforms communicate with each other and with clients.

Web applications refer to applications accessed via Web browser over a network and developed using browser-supported languages (e.g., HTML, JavaScript, Python). For execution, Web applications depend on Web browsers and include many familiar applications such as online retail sales, online auctions, and webmail.

Web applications are needed in the area of business-to-business interaction over networks, e.g., for overseas companies that outsource projects to each other. The adoption of a Web applications infrastructure can provide vital processes such as transfer of funds and updates of pricing information. [8]

During the last few decades, many software development models have been proposed and discussed within the Software Engineering community. All the models can be generally categorized into two sub – categories, Traditional Approach and Object – Oriented Approach. Traditional Approach and Object – Oriented Approach are reviewed separately here due to the number of differences in the two. Both the approaches have merits and demerits. In this study, they will be compared against one

another with real implementations to cross analyze the differences and similarities in how they have impact on selected cost drivers.

Traditional software engineering methodologies, or methodologies for developing information systems using databases, do not contain useful abstractions capable of easing the task of specifying applications that embody the hypertext metaphor. For example, they do not provide any notion of linking, and very little is said about how to incorporate hypertext into the interface. In addition, as the size, complexity and number of applications grows, a systematic approach is needed that helps dealing with complexity, and allows evolution and reuse of previously gathered design knowledge. [9]

Our aim in this article is to be present a practical design implementation for OOHDM. In our approach, the design of a web-based application is based on an object - oriented modeling process that captures Conceptual Design, Navigational Design, Abstract Interface Design, and Implementation. The modeling process is based on the Object-Oriented Hypermedia Design Method (OOHDM).

We showed that our method provides great benefits at solving most of the Web-Based Application Design Application. We discussed how OOHDM designs can be implemented in the Web-Based Application Design Application along with real implementation of this approach and traditional approach. Finally, discuss further research in this area.

CHAPTER 03 – LITERATURE REVIEW

Software development is the process of developing software through successive phases by following a sort of order. This process of developing a software product not necessarily only including the actual writing of code (i.e. implementation) but also the preparation of requirements, design, specification and testing. The order of the successive phases can also be referred to as the methodology. In simple words, we can say that a methodology is a systematic way of moving from one stage to another of software development. There is a growing consensus about the kind of methodology that must be performed with respect to the development of the software.

During the last few decades, many software development models have been proposed and discussed within the Software Engineering community. All the models can be generally categorized into two sub – categories, Traditional Approach and Object – Oriented Approach. Traditional approaches to information systems development tend to be either process-centric (e.g., structured systems) or data-centric (e.g., information engineering). Object-oriented approaches combine process and data into holistic entities (objects). [4] Traditional Approach and Object – Oriented Approach are reviewed separately here due to the number of differences in the two. Both the approaches have merits and demerits. In this study, they will be compared against one another to cross analyze the differences and similarities in how they have impact on selected cost drivers.

Traditional Approach:

The traditional approach, which is also known as the conventional approach or the software development life cycle methodology has distinct stages. The boundaries between each stage are meant to be fairly rigid and sequential. This is basically a sequential development approach in which development is seen as going in one direction. We are expected to move forward from one phase to the other. The basic principles of this approach are: [4]

a) Project is divided into sequential phases, with some overlap and splash back acceptable between phases.
b) Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
c) Tight control is maintained over the life of the project via extensive written documentation, formal reviews, and approval/signoff by the user and information technology management occurring at the end of most phases before beginning the next phase. Written documentation is an explicit deliverable of each phase.

Many models are developed under this approach. For instance, Waterfall model, Prototyping model, and so on. These methods are applying a disciplined approach where the stages of design and build are well predictable. Detailed stages of analysis and design precede the stage of building the software. These methodologies are well documented and thus are quite complex to apply. [4]

Object – Oriented Approach:

The Object – Oriented approach, which is also known as the modern approach, we are allowed to perform each phase more than once and in any order. In this approach, a system is viewed as a set of objects [1]. This approach follows an iterative and incremental approach to systems development. The systems development life cycle is viewed as consisting of several increments or phases: inception, elaboration, construction and transition. [5]

The object-oriented paradigm emphasizes modularity and re-usability. The goal of an object-oriented approach is to satisfy the "open closed principle". A module is open if it supports extension. If the module provides standardized ways to add new behaviors or describe new states. In the object-oriented paradigm, this is often accomplished by creating a new subclass of an existing class. A module is closed if it has a well – defined stable interface that all other modules must use and that limits the interaction and potential errors that can be introduced into one module by changes in another. In the object-oriented paradigm, this is accomplished by defining methods that invoke services on objects. Methods can be either public or private, i.e., certain behaviors that are unique to the object are not exposed to other objects. This reduces a source of many common errors in computer programming. [6]

The benefits of object – oriented development are reduced time to market, great product flexibility, and schedule predictability and the risks of them are performance and startup costs. [7]

The following table explains some of the ways how the object-oriented approach differs from the traditional structured approach [1, 19]

TRADITIONAL APPROACH	OBJECT – ORIENTED APPROACH
Used to develop the Traditional Projects that uses procedural programming.	Used to develop Object-oriented Projects that depends on Object-Oriented programming.
Program is divided into number of submodules or functions.	Program is organized by having number of classes and objects.
Uses common processes likes: analysis, design, implementation, and testing. DFD & E-R diagram model the data.	Uses UML notations likes: use case, class diagram, communication diagram, development diagram and sequence diagram.
Depends on the size of projects and type of projects.	Depends on the experience of the team and complexity of projects through the numbers of objects.
Needs to large duration sometimes to development the large projects.	Need to more time than Traditional approach and leads that to more cost.
The problem of Traditional approach	The object-oriented software lifecycle

using classical lifecycle [20, 21].	identifies the three traditional activities of analysis, design, and implementation[21].
Software reuse is not possible.	Reusability is possible.

CHAPTER 04 – BACKGROUND

4.1) The Traditional Approach –

The traditional approach, which is also known as the conventional approach or the software development life cycle methodology has distinct stages. The boundaries between each stage are meant to be fairly rigid and sequential. This is basically a sequential development approach in which development is seen as going in one direction.

There are a number of phases common to every development, regardless of methodology, starting with requirements capture and ending with maintenance. With the traditional approach, will be expected to move forward gracefully from one phase to the other. The list below describes the common phases in software development [11, 15].

A. Requirements:

Requirements capture is about discovering what is going to achieve with new piece of software and has two aspects. Business modeling involves understanding the context in which software will operate. A system requirement modeling (or functional specification) means deciding what capabilities the new software will have and writing down those capabilities [11].

B. Analysis:

Analysis means understanding what are dealing with. Before designing a solution, it needs to be clear about the relevant entities, their properties and their inter-relationships. Also needs to be able to verify understanding. This can involve customers and end users, since they are likely to be subject-matter experts [11].

C. Design:

In the design phase, will work out, how to solve the problem. In other words, make decisions based on experience, estimation and intuition, about what software which will write and how will deploy it. System design breaks the system down into logical subsystems (processes) and physical subsystems (computers and networks), decides how machines will communicate, and chooses the right technologies for the job, and so on [11].

D. Specification:

Specification is an often-ignored, or at least often-neglected, phase. The term specification is used in different ways by different developers. For example, the output of the requirements phase is a specification of what the system must be able to do; the output of analysis is a specification of what are dealing with; and so on [13].

E. Implementation:

In this phase is writing pieces of code that work together to form subsystems, which in turn collaborate to form the whole system. The sort of the task which is carried out

during the implementation phase is ‘Write the method bodies for the Inventory class, in such a way that they conform to their specification’ [7].

F. Testing:

When the software is complete, it must be tested against the system requirements to see if it fits the original goals. It is a good idea for programmers to perform small tests as they go along, to improve the quality of the code that they deliver [7].

G. Deployment:

In the deployment phase, are concerned with getting the hardware and software to the end users, along with manuals and training materials. This may be a complex process, involving a gradual, planned transition from the old way of working to the new one [11].

H. Maintenance:

When the system is deployed, it has only just been born. A long life stretches before it, during which it has to stand up to everyday use – this is where the real testing happens. The sort of the problem which is discovered discover during the maintenance phase is 'When the log-on window opens, it still contains the last password entered.' As the software developers, we normally interested in maintenance because of the faults (bugs) that are found in software. Must find the faults and remove them as quickly as possible, rolling out fixed versions of the software to keep the end users happy. As well as faults, users may discover deficiencies (things that the system

should do but doesn't) and extra requirements (things that would improve the system) [13, 15].

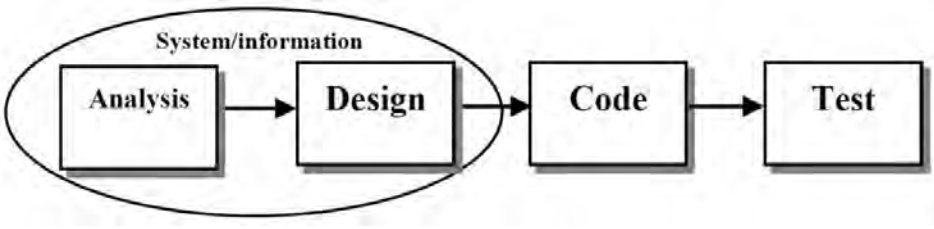


FIGURE 01 – The Linear Sequential Model

4.2) The Object – Oriented Approach –

This is a new way of thinking about problems using model based on real world concepts. One of the main principles in the object - oriented approach is that abstraction, not of data structures and processes separately but both together [16].

Object-oriented approaches to developing information systems, technically speaking, can use any of the traditional methodologies such as (waterfall development, parallel development, phased development, prototyping, and throwaway prototyping). However, the object-oriented approaches are most associated with a phased development RAD methodology. The primary difference between a traditional approach like structured design and an object-oriented approach is how a problem is decomposed. In traditional approaches, the problem decomposition process is either process-centric or data-centric. However, when modeling real-world systems, processes and data are so closely related that it is difficult to pick one or the other as the primary focus. Based on this lack of congruence with the real-world, new object-oriented methodologies have emerged that use the RAD based sequence of SDLC phases but attempt to balance the emphasis between process and data by focusing the decomposition of problems on objects that contain both data and processes. Both approaches are valid approaches to developing information systems [17].

In object-oriented approach, a system is viewed as a set of objects. All object-orientation experts agree that a good methodology is essential for software

development, especially when working in teams. Thus, quite a few methodologies have been invented over the last decade. Broadly speaking, all object-oriented methodologies are alike – they have similar phases and similar artifacts – but there are many small differences. Object-oriented methodologies tend not to be too prescriptive: the developers are given some choice about whether they use a particular type of diagram, for example. Therefore, the development team must select a methodology and agree which artifacts are to be produced, before they do any detailed planning or scheduling. In general, each methodology addresses [1]:

a) The philosophy behind each of the phases [1].

b) The workflows and the individual activities within each phase [1].

c) The artifacts that should be produced (diagrams, textual descriptions and code) [1].

d) Dependencies between the artifacts [1].

e) Notations for the different kinds of artifacts [1].

f) The need to model static structure and dynamic behavior [1].

Some methodologies, especially the more comprehensive ones, have alternative development paths, geared to different types and sizes of development [11,14].

The benefits of Object-Oriented Development are reduced time to market, greater product flexibility, and schedule predictability and the risks of them are performance and start-up costs [7].

The basic construct is object which combines both data structure and behavior in a single entity. Objects are used to model real world entities, and they have mainly two faces: the state (the structural semantics) of an object and the behavior of an object (the behavioral semantics).

Mainly, two structuring mechanisms characterize the object - oriented approach. The first is a categorization of objects called classification. Objects which have common structural and behavioral semantics are gathered in classes to model sets of homogeneous entities. The second is a relationship between classes that is called generalization. Generalizations represents situations where classes are combined to form new classes that define the set of shared structural and semantics. This concept is also called inheritance [18].

4.3) What is Object-Oriented Hypermedia Design Method (OOHDM) –

The Object-Oriented Hypermedia Design Method (OOHDM) is a model-based approach for building large hypermedia applications. In other words, it is method for the development of Web applications. It is originally developed by Gustavo Rossi (Universidad Nacional de la Plata) and Daniel Schwabe (PUC Rio, Brazil). Its development started in 1995 and still continues.

It has been used to design different types of applications such as web sites, web applications, database and information system, multimedia presentations, etc. The Object-Oriented Hypermedia Design Method (OOHDM) mainly breaks down the hypermedia design in four main activities. They are:

1. Conceptual Design;
2. Navigational Design;
3. Abstract Interface Design; and
4. Implementation.

They are performed in a mix of incremental, iterative and prototype-based development styles. During each activity a set of object-oriented models describing particular design concerns are built or enriched from previous iterations. [10]

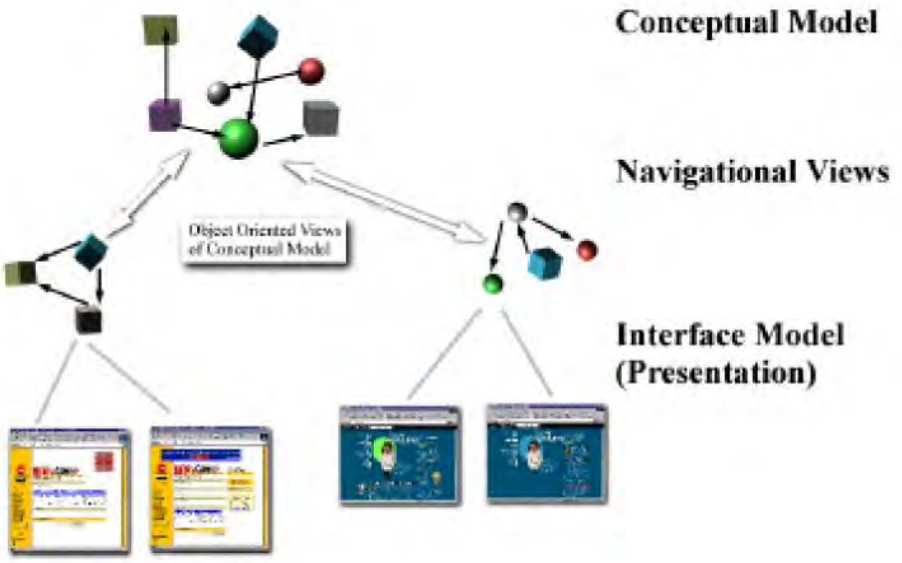


FIGURE 02 – OOHDM Design Models [9]

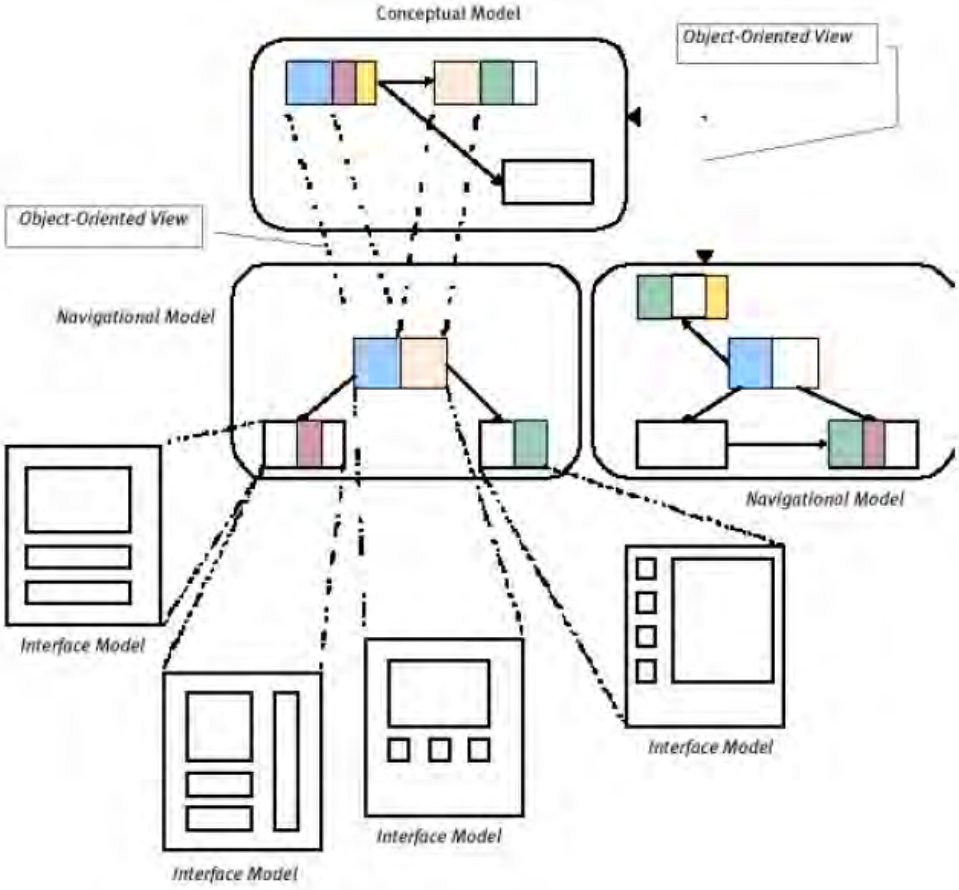


FIGURE 03 – Relationships Between the Different Activities [9]

The cornerstones of this approach are:

1. Navigation objects are views (in the database sense) of conceptual objects;

2. The use of appropriate abstractions to organize the navigational space (navigational contexts); and

3. The separation of interface issues from navigation issues [9].

4. An explicit identification that there are design decisions that need only be made at implementation time [9].

Now, we will explain each of the main activities, i.e. Conceptual Design, Navigational Design, Abstract Interface Design, and Implementation, with a brief summary of the OOHDM methodology.

Conceptual Design:

In the Conceptual Design phase, the model of the application domain is elaborated that determines the universe of the model. This is done by using Object – Oriented principles. In simple words, a model of the application domain is build using well known Object – Orientation (OO).

In OOHDM, the conceptual schema is built upon classes, relationships and sub-systems. Classes are described as usual in object-oriented models, though attributes may be multi-typed representing different perspectives of the same real-world entity. In OOHDM the class schema consists of a set of classes connected by relationships.

Objects are instances of classes, and thus, when a relationship holds between classes, it abstracts corresponding object-to-object relationships. Classes will be used later during Navigational Design to derive Nodes, and Relationships will be used to derive Links [9].

The main concern of this phase is to capture the domain semantics as "neutrally" as possible:

- Represents objects, their relationships and collaborations;
- Little concern for the types of users and tasks;
- If the application requires dynamic updates, this model will evolve into the object model for the application.

Since OOHDM is a design method and not an implementation framework, we shall assume that methods for getting the value of attributes of an object are automatically generated. When other computations must be done, corresponding methods should be specified. Using the terminology of object-oriented methods, we may say that the Conceptual Model will contain features of both Analysis and Design Models. When the application will run on top of an environment supporting (distributed) objects, classes in the conceptual model will be implemented directly as they are defined, specifying multiple perspectives as separate attributes. If not, they will serve as design specifications for the navigational and interface design activities [9].

Navigational Design:

In OOHDM, the navigation design is considered to be an important step in the design of the application. It is as a view over the conceptual model. This allows the building of different models according to different users' profiles. Each navigational model provides a "subjective" view of the conceptual model [22]. Each navigational model provides a "subjective" view of the conceptual model [9].

The navigation design takes into account [9]:

- Which objects will be navigated;
- Composition structures (how we are creating composites);
- What is the underlying structure of the navigation: in which contexts will the user navigate?
- Whether navigational objects might look different depending on the context in which they are navigated;
- Which connections and access structures exist among the navigable objects;
- How does navigation proceed when the user "jumps" from one object to another?

Navigation design is classified in two schemas. They are:

1. Navigational Class Schema:

The Navigational Class Schema defines the navigable objects of the application (created from the conceptual schema). It defines three types of navigational classes:

- Nodes

- Links
- Access structures: they represent possible ways to access the nodes (i.e. indexes, guided tours)

2. Navigational Context Schema:

Once the navigation classes have been decided, it is necessary to structure the navigation space that will be made available to the user. In OOHDM this structure is defined by grouping navigation objects into sets called contexts. Each context definition includes, besides which elements are included in it, the specification of its internal navigation structure, an entry point, access restrictions in terms of user classes and operations, and an associated access structure [10].

The Navigational Context Schema defines how navigable objects are clustered together and navigated. It is basically a set of nodes, links, context classes and other (nested) navigational contexts. It is defined by enumeration or by stating a property. There are seven different types of Navigational Context Schema. They are:

- Simple class based: each element of the context should satisfy a property
- Class based group: it is a set of contexts, of which each is a simple class context
- Link based: the selection is based on a relationship, usually 1-to-n
- Link based group: collection of link based contexts
- Enumerated: elements are explicitly enumerated

- Dynamic: if they are defined by the navigation or interaction of the user (shopping basket, history, or user-modifiable database)
- Static: if not dynamic

Abstract Interface Design:

The Abstract Interface Design specifies interface objects that are responsible for mediating user interaction with navigation objects. The interface model specifies which interface objects the user will perceive; which interface objects will activate navigation; how multimedia interface objects will be synchronized; and the interface transformations that will take place [9].It should be recognized that there is a distinction between navigation operations and interface operations; not everything that happens in the interface is navigation related [10].In OOHDM we use the Abstract Data View (ADV) design approach for describing the user interface of a hypermedia application [23].

Implementation:

Finally, the Implementation phase is responsible for mapping conceptual objects, navigation objects and interface objects onto the particular runtime environment being targeted. In this phase, the designer will actually implement the design. Up to now, all models were deliberately constructed in such a way as to be independent of the implementation platform; in this phase, the particular runtime environment is taken into account [9].

CHAPTER 05 – IMPLEMENTATION

5.1) Introduction –

Our aim is to figure out the differences between these two approaches Traditional Approach and Object – Oriented Approach based on features like Requirement Specifications, Understanding Requirements, Cost, Guarantee of Success, Resource Control, Cost Control, Simplicity, Risk Involvement, Expertise Required, Changes Incorporated, Risk Analysis, User Involvement, Flexibility and few more. For this we have created two different systems using Traditional Approach and Object – Oriented Approach which helped us to do find the difference between them. We have validated the result with experimental results.

For both the approaches, we have developed an e-commerce based website, keeping all the functionalities same.

5.2) The Traditional Approach Implementation –

The traditional approach, which is also known as the conventional approach or the software development life cycle methodology has distinct stages. With the traditional approach, will be expected to move forward gracefully from one phase to the other. The list below describes the common phases in software development [11, 15].

While implementing the traditional approach, we have ensured that the coding is done in the traditional way, i.e. we did not use any object – oriented approach concept. We used PHP and MySQL only to write the code. We have not used any function or any objects. We coded as simple as possible.

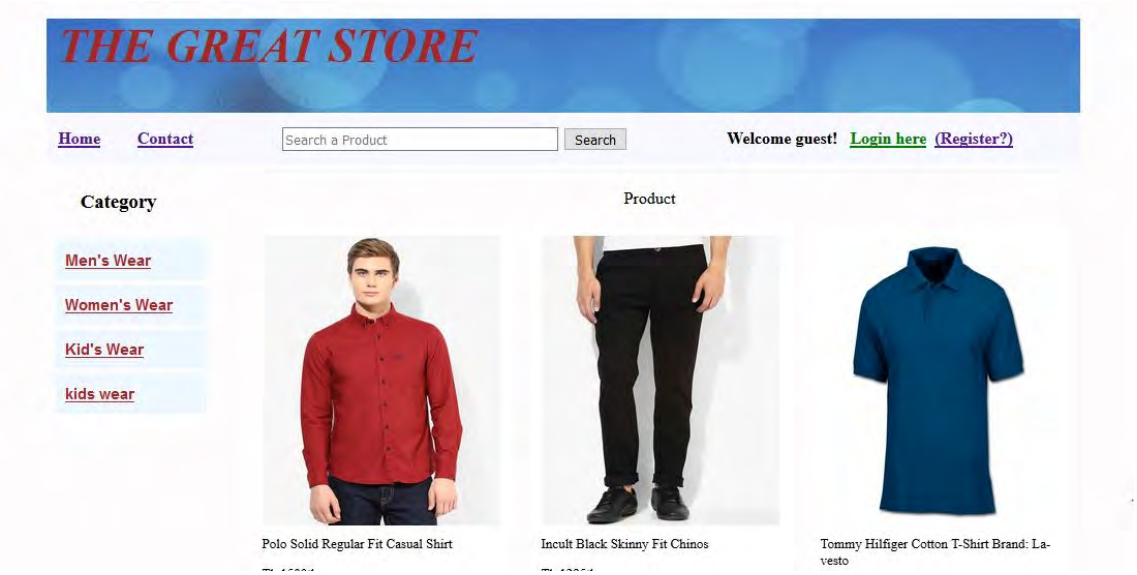


Figure 04 – Home Page of the E-commerce site built upon the Traditional Approach

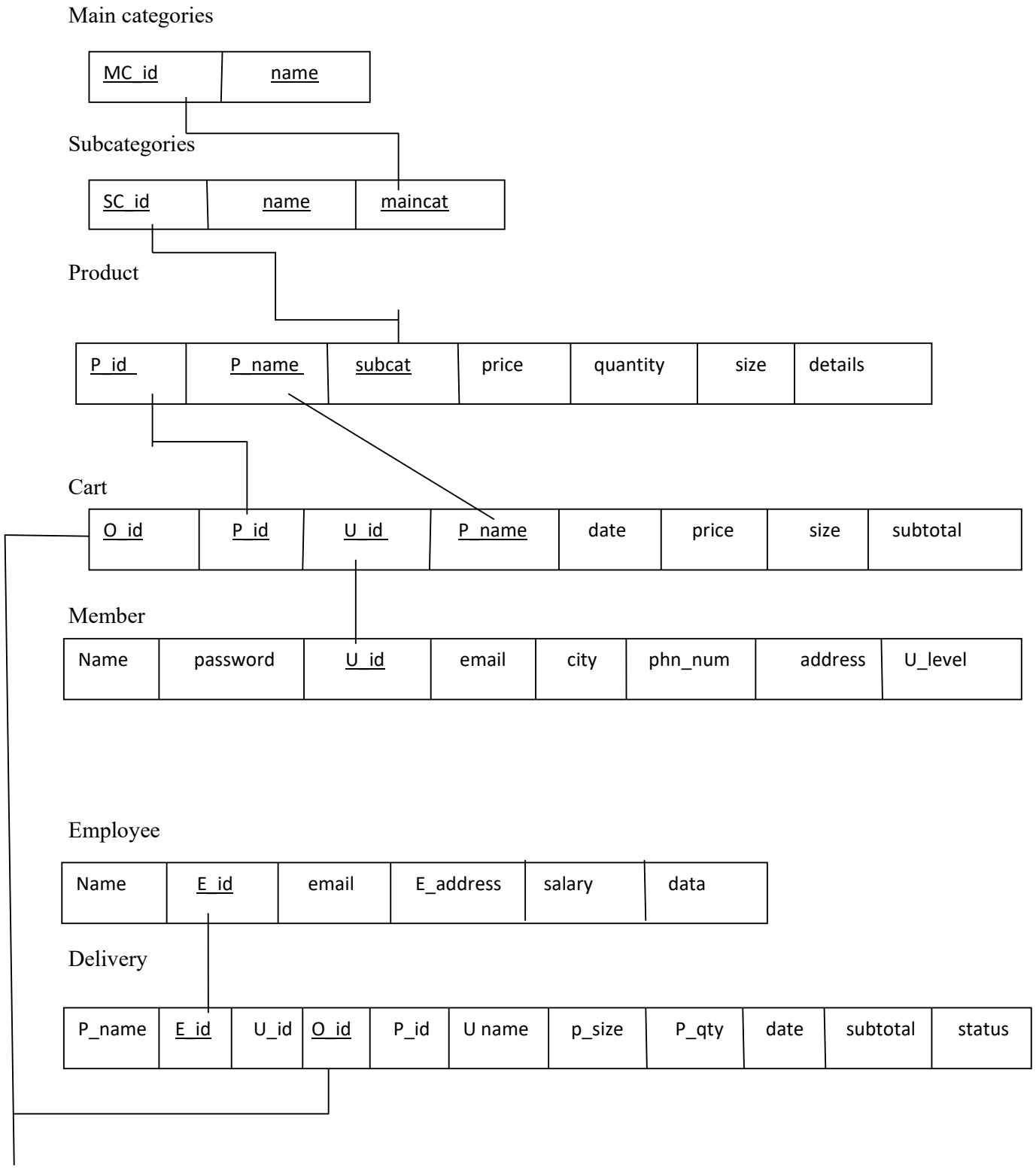


Figure 05 – Relational Model of the E-commerce site built upon Traditional Approach

5.3) The Object – Oriented Approach Implementation Using OOHDM –

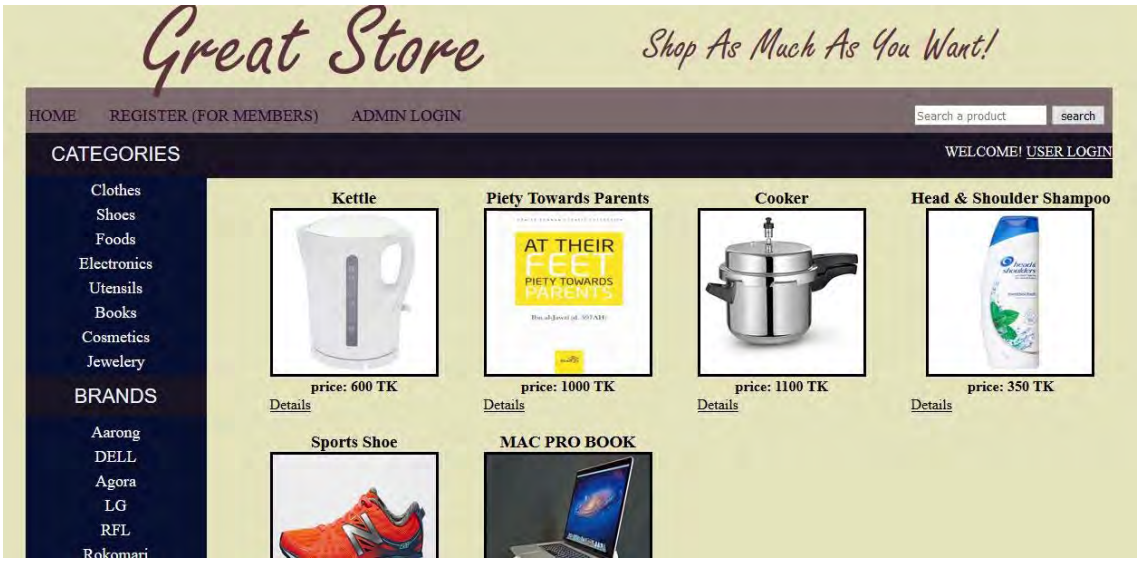


Figure 06 – Home Page of the E-commerce site built upon the OOHDM

Since we already know that the OOHDM model has four main activities, i.e. Conceptual Design, Navigational Design, Abstract Interface Design and Implementation.

While implementing the object – oriented approach using OOHDM, we have ensured that the coding is done in the object – oriented way, i.e. we used classes and objects. We used Java Script, PHP and MySQL only to write the code. We have used functions to store the values in the variable.

Conceptual Design:

Firstly, we did the Conceptual Design. In OOHDM the class schema consists of a set of classes connected by relationships. Objects are instances of classes, and thus, when a relationship holds between classes, it abstracts corresponding object-to-object relationships. Classes will be used later during Navigational Design to derive Nodes, and Relationships will be used to derive Links [9]. Figure 08 contains the class diagram for our implemented e-commerce website using the OOHDM model. In this diagram, there are products, which has its own category, brand and which can be added to cart. Every user has an id, name and password using which the user can login and order any product.

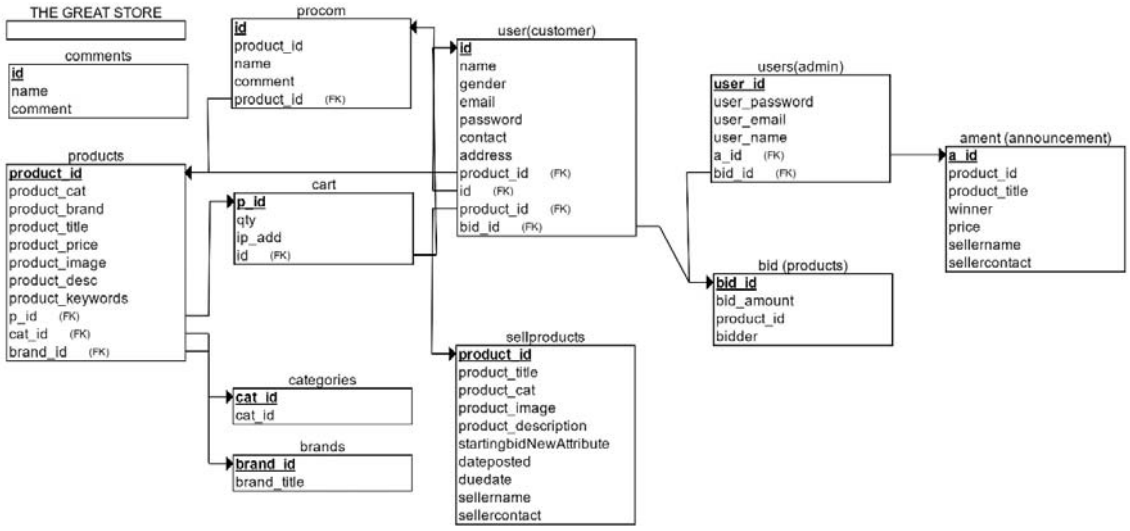


Figure 07 – Class Diagram of the E-commerce Website built upon the OOHDM

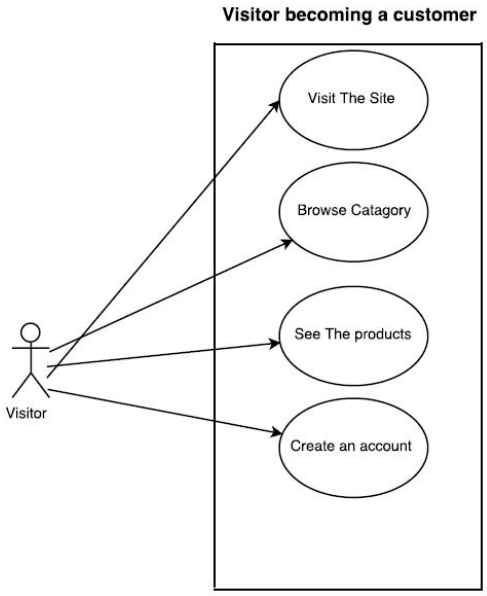


Figure 08 – Use Case Diagram for Visitor in the online store becoming a Customer

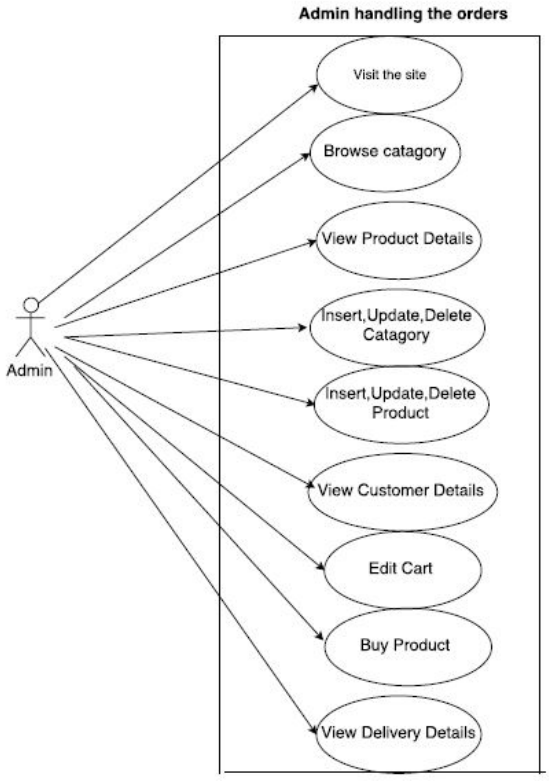


Figure 09 – Use Case Diagram for Admin handling the orders

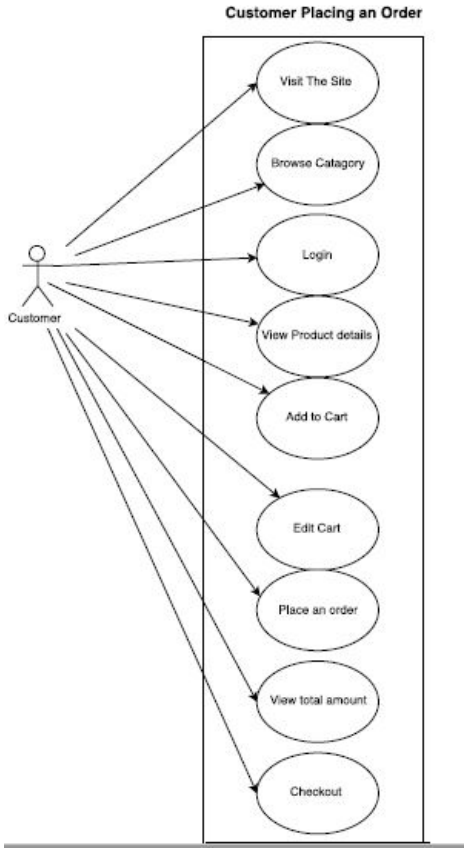


Figure 10 – Use Case Diagram for Customer placing an order

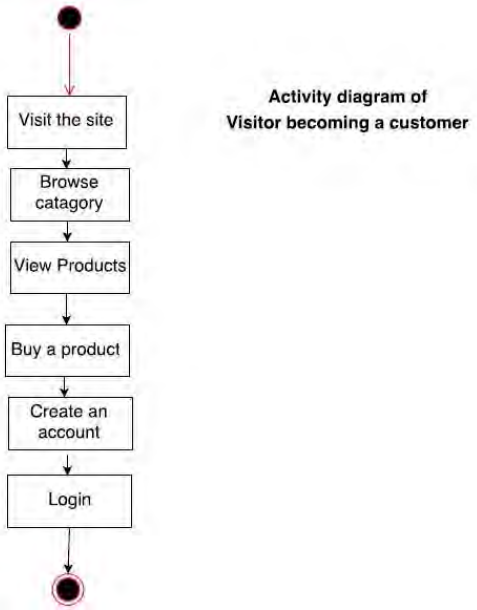


Figure 11 – Activity Diagram of Visitor becoming a Customer

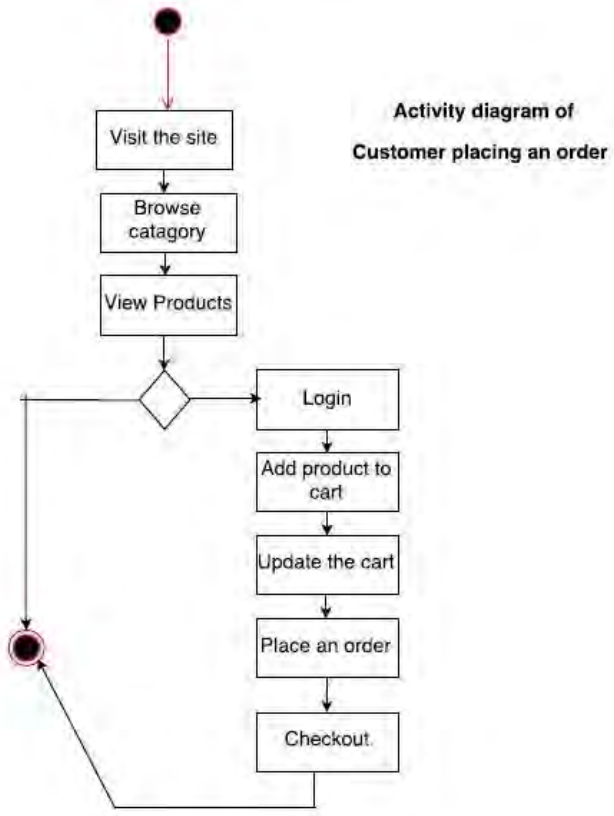


Figure 12 – Activity Diagram of Customer placing an Order

Navigational Design:

A Navigational Model is built as a view over a conceptual model, thus allowing the construction of different models according to different users' profiles. Each navigational model provides a "subjective" view of the conceptual model [24]. We now briefly describe the notation used in the diagram.

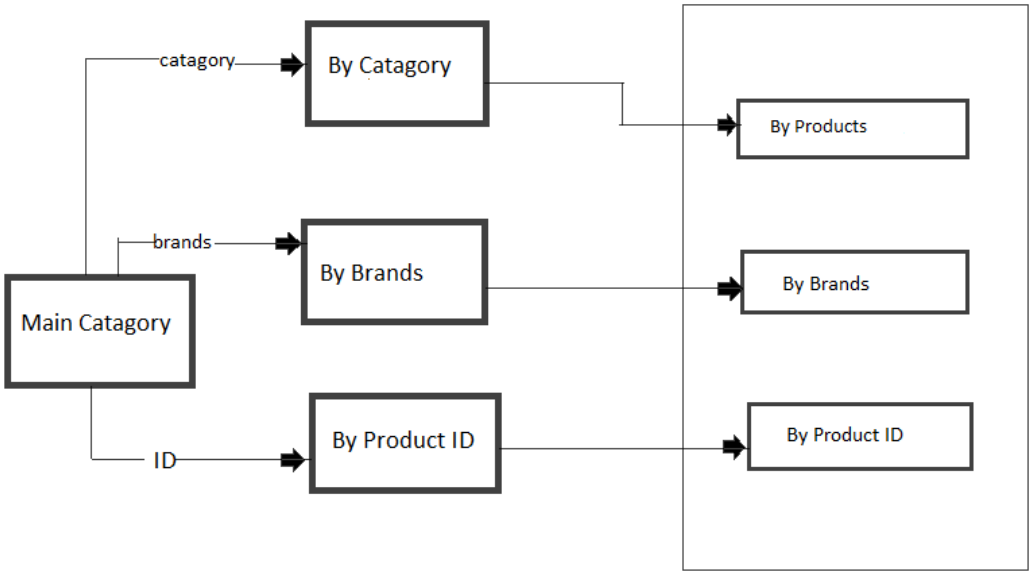


Figure 13 - Navigational Context Schema for the e-commerce web-based application

In many situations, there will be groups of contexts that encompass nodes of the same class, according to different criteria. For example, in our case, Products, may be organized according to the different categories or brands. We denote this by:

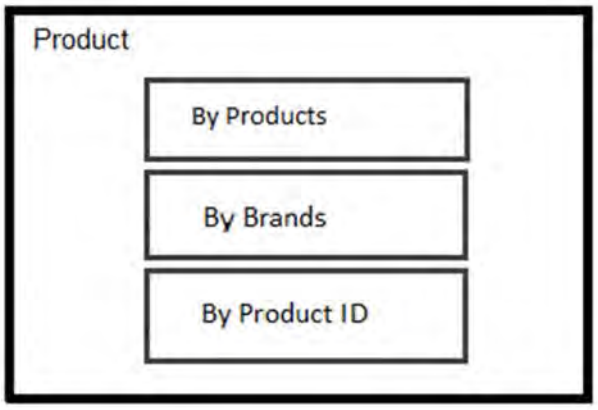


Figure 14 - Notation for Product

Abstract Interface Design:

Once the application's navigational structure has been defined, we must specify its interface aspects. This means defining the way in which different navigational objects will appear, which interface objects will activate navigation and other application functionality, and which interface transformations will take place and when. [9]

In OOHDM, we use the Abstract Data View (ADV) design approach for describing the user interface of a hypermedia application [23]. ADVs are objects in that they have a state and an interface, where the interface can be exercised through messages (in particular, external events generated by the user). ADVs are abstract in that they only represent the interface and the state, and not the implementation. ADVs have been used to represent interfaces between two different media such as a user, a network or a device (a timer, for example) or as an interface between two or more Abstract Data Objects (ADOs). ADOs are objects that do not support external, user-generated events [23]. From an architectural point of view, ADVs are observers for ADOs, so the communication protocol among interface and application objects follows the rules described in the Observer Design Pattern [25].

An ADV used in the design of web applications can be viewed as an interface object. In the context of OOHDM, navigational objects such as nodes, and indexes will act as ADOs, and their associated ADVs will be used for specifying their appearance to the user. [9]

To end this section, we show in Figure 09 the real interface of our e-commerce website and how abstract interface objects are related with their implemented counterparts.

Figure 15 – ADVs and their relationship with "real" interface objects

Implementation:

In this phase, we have actually implemented the design. Up to now, all models were actually constructed in such a way as to be independent of the implementation platform. We have concentrated on how OOHDM designs can be implemented in the web-based application.

When the implementation phase is reached, we have already defined previously the information items that are part of the problem domain. We also have identified how these items should be organized according to the user's profile and tasks; he has

decided what the interface will look like, and how it will behave. In order to implement all of this in the web-based application environment, we have stored all the information items (both conceptual and navigation objects) in MySQL database.

CHAPTER 06 – ANALYSIS & COMPARISONS

After we have implemented both the approaches using an online e-commerce site, now we have figure out the differences between these two approaches based on features like Requirement Specifications, Understanding Requirements, Guarantee of Success, Resource Control, Cost Control, Simplicity, Risk Involvement, Expertise Required, Changes Incorporated, Risk Analysis, User Involvement, Flexibility and few more. We have also test the quality of the implementation of the two approaches in terms of technical quality parameters, such as reliability, capability, maintainability, performance and also user quality parameters. We have also used McCall's Factor Model to test the quality of the two approaches.

6.1) Technical Quality Parameters:

1. Correctness:

To check the correctness for both the approaches, we have measured the lack of bugs and defects. We have measured in terms of defect density (number of bugs per lines of code). In our traditional approach implementation, we have in total 3331 lines of code and in our object – oriented approach implementation, we have in total 2708 lines of code. In our traditional approach implementation, we found 78 bugs for the total of 3331 lines. As for the object – oriented approach implementation, the number of bugs were 29 bugs, which is considerably very low.

Therefore, defect rate of the traditional approach implementation is:

$$\text{defect density} = \frac{\text{number of bugs}}{\text{lines of code}} = \frac{78}{3331}$$

$$= 0.023416$$

$$= 2.3416\ \%$$

And the defect rate of the object – oriented approach implementation is:

$$\text{defect density} = \frac{\text{number of bugs}}{\text{lines of code}} = \frac{29}{2708}$$

$$= 0.010709$$

$$= 1.0709\ \%$$

2. Capability:

To check the capability for both the approaches, we have measured in terms of requirements coverage. We have checked whether they do all that is required or not.

$$\text{capabilities} = \%\ \text{of required operations implemented}$$

For our web-based application, the functional requirements, as collected from the users, have been categorized as follows to support the types of user interactions that the system shall have. In the following table, we have shown the coverage of the approaches in accordance with the requirements.

REQUIREMENTS	**Traditional approach implementation**	**Object – oriented approach implementation**
The users shall be able to view the categories on the application's home page.	✓	✓
The users shall be able to view items in different categories.	✓	✓
The users shall be able add items to the cart.	✓	✓
The users shall be able to view more information about an item before adding it to the cart.	✓	✓
The users shall be able to able view the shopping cart.	✓	✓
The users shall be able to browse through the available items.	✓	✓
The users shall be able to view the items added to the cart.	✓	✓
The users shall be able to delete items from the cart.	✓	✓
The users shall be able to check out items only when there are items in the shopping cart.	✓	✓
The users shall login or register using the user authentication form.	✓	✓
The users shall not login or register if the information is incomplete or invalid.	✓	✓
The users shall place an order by completing the	✓	✓

information in the order form.		
The users shall not be able to place an order if the information in the order form is invalid or incomplete.	✓	
The administrator shall be able to view all the users' information that completes the order form and the checkout process.	✓	✓
The administrator shall be able to add new items to the list of shopping items.	✓	✓
The administrator shall be able to modify/update an item's price and description.	✓	✓
The administrator shall be able to delete items from the main page of the shopping-cart application.	✓	✓
The administrator shall be able to view the entire history of the checked-out items.	✓	✓
The administrator shall be able to view the entire history for the users who successfully complete the checkout process.	✓	✓

Therefore, capability or requirement coverage of the traditional approach implementation is:

$$\text{capability} = \frac{\text{number of required operations implemented}}{\text{number of total required operations}} = \frac{19}{19} \times 100$$

$$= 100\ \%$$

And the capability or requirement coverage of the object – oriented approach implementation is:

$$\text{capability} = \frac{\text{number of required operations implemented}}{\text{number of total required operations}} = \frac{18}{19} \times 100$$

$$= 94.74\ \%$$

3. Maintainability:

To check the maintainability for both the approaches, we have tried to add a new feature for both the implementations. We have added a new feature "Bidding" in the Traditional and Object – Oriented approach implementation and checked whether is it easy to change and adapt to new requirements or not. We measured in terms of change logs (time and effort required to add a new feature) and impact analysis (number of lines affected by a new feature).

Therefore, maintainability of the traditional approach implementation is:

$$\text{time required or given to add the new feature} = 3 \text{ hours}$$

$$\text{impact analysis} = \text{number of lines affected by a new feature} = 500$$

And the maintainability of the object – oriented approach implementation is:

$$\text{time required or given to add the new feature} = 3 \text{ hours}$$

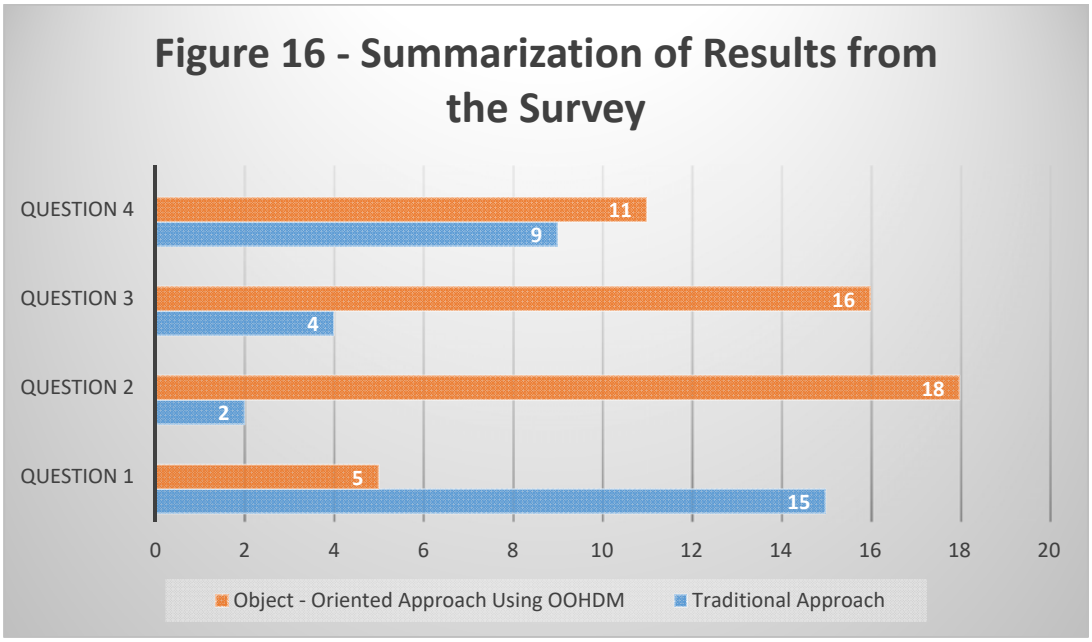
$$\text{impact analysis} = \text{number of lines affected by a new feature} = 238$$

6.2) User Quality Parameters:

To check the user satisfaction and usability for both the approaches, we have done a survey among 20 people. Our survey questions were very simple. For instance,

- Which approach do you think is more user friendly?
- Which approach implementation do you think was faster?
- In terms of performance, which implementation is better?
- Which approach implementation do you think is more reliable?

In the following bar chart, we have summarized our results from the survey.



CHAPTER 07 – CONCLUSION & FUTURE WORK

The contribution of this paper lies in the identification of advantages OOHDM modeling could offer for the Web-Based Application. Also, guidelines have been provided to illustrate the nature of OOHDM modeling and how that could be applied in the specification of Web-Based Application. This paper also attempts to compare between traditional and object - oriented approach using OOHDM for Web-Based Application.

In this paper, we have implemented an e-commerce website using Traditional Approach once and Object - Oriented Approach (used OOHDM) another time. While implementing the OOHDM based e-commerce website, we have also implemented the four main activities, i.e. Conceptual Design, Navigational Design, Abstract Interface Design and Implementation.

Due to the simplicity of the approach and comparisons presented in this paper, it can be used in a great variety of Web - based Application where top-down decomposition in the design is a largely used technique. To properly implement the OOHDM approach, the designer must perfectly do the four main activities, i.e. Conceptual Design, Navigational Design, Abstract Interface Design and Implementation. All the other important steps that are mentioned previously must be handled carefully due to

its importance in the model. To properly implement the OOHDM, we have used the UML diagrams along the real – life implementation of an e-commerce website using the both the approaches.

From our study, we suggest that the usage of OOHDM is far better the Traditional Approach in average – sized Web-based Applications. We compared them based on features like Requirement Specifications, Maintainability, Correctness, Understanding Requirements, Simplicity, User Involvement and few more. Future work can be done to find out how efficient and productive the OOHDM becomes for huge or large – sized Web-based Applications.

As there are numerous advantages, it is hoped that future work on this area of using OOHDM will enhance the understanding of the structure of large Web-Based Application. Interviews of project managers working in different companies can be carried out to understand what type of approach are in use in real world and what the most relevant aspects found in developing the web – based application. Future work can also be done on the issues of security, to see which approach gives the best security. Also, it can be investigated that if a new component or feature is added to the web-based application, which approach gives us the most benefit. This entire rich and dense OOHDM can also be implemented in the field of desktop-based software development.

CHAPTER 08 – REFERENCES

[1] N. M. A. Munassar and Dr. A. Govardhan, "*Comparison between Tradition Approach and Object – Oriented Approach in Software Engineering Development"*, in IJACSA, Vol. 2, No. 6, 2011.

[2] Alan Dennis, Barbara Haley Wixom and David Tegarden, "*Systems Analysis and Design with UML Version 2.0*", John Wiley & Sons, Inc. , Second Edition, 2005, Page # 25.

[3] Centers for Medicare & Medicaid Services (CMS) Office of Information Service (2008), "*Selecting a development approach"*.

[4] Stoimen, "*Traditional Software Development Methodologies"*: http://www.stoimen.com/blog/2011/04/15/traditional-software-development-methodologies/.

[5] Abrahamsson P. et.al, "*Agile Software Development Methods: Review and Analysis*", ESPOO, VTT Publications 478, VTT Technical Research Centre of Finland. http:/www. fi/pdf/publications/2002/P478.pdf, 2002.

[6] Meyer, Bertrand (1988) ,*Object-Oriented Software Construction*. Cambridge: Prentise Hall International Series in Computer Science. p. 23. ISBN 0-13-629049-3.

[7] Grady Booch, "*Object-Oriented Analysis and Design with applications*", Addison Wesley Longman, Inc, second Edition, 1998.

[8] Sabah Al-Fedaghi, "*Developing Web Applications*", International Journal of Software Engineering and Its Applications, Volume 5 No. 2, April, 2011.

[9] Daniel Schwabe and Gustavo Rossi, "*An Object Oriented Approach to Web-Based Application Designs*".

[10] Daniel Schwabe and Gustavo Rossi, "*Developing Hypermedia Applications Using OOHDM*".

[11] Mike O' Docherty, "*Object-Oriented Analysis and Design Understanding System Development with UML 2.0*", John Wiley & Sons Ltd, England, 2005.

[12] Magnus Christerson and Larry L. Constantine, "*Object-Oriented Software Engineering- A Use Case Driven Approach*", Objective Systems, Sweden, 2009.

[13] Iansommerville, "*Software Engineering*", Addison Wesley, 7th edition, 2004.

[14] Pankaj Jalote, "*An Integrated Approach to Software Engineering*", Springer Science Business Media, Inc, Third Edition, 2005.

[15] Roger S. Pressman, "*Software Engineering a practitioner's approach*", McGraw-Hill, 5th edition, 2001.

[16] Paul Lewis, "*The Object Oriented Approach to Software Engineering"*: http://users.ecs.soton.ac.uk/phl/ctit/oo/oo.html.

[17] Alan Dennis, Barbara Haley Wixom and David Tegarden, "*Systems Analysis and Design with UML Version 2.0*", John Wiley & Sons, Inc. , Second Edition, 2005, Page # 35-36.

[18] Danny B. Lange, "*An Object - Oriented Design Method for Hypermedia Information Systems*", IBM Research, Tokyo Research Laboratory, Computer Science Institute.

[19] "*An Object Oriented Approach*":https://www.tutorialspoint.com/system_analysis_and_design/system_an

alysis

_and_design_object_oriented_approach.htm.

[20] MM Lehman, "*Process Models, Process Programs*", Programming Support", ACM, 1987.

[21] Tim Korson and John D. McGregor, "*Understanding Object-Oriented: A Unifying Paradigm*", ACM, Volume 33, No. 9, 1990.

[22] W. Harrison and H. Ossher, "*Subject-Oriented Programming (a critique of pure objects)*", In Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'93), pp. 411-428, Washington, September 1993.

[23] D. D. Cowan and C. J. P.Lucena, "*Abstract Data Views, An Interface Specification Concept to Enhance Design for Reuse*", IEEE Transactions on Software Engineering, Vol.21, No.3, March 1995.

[24] W. Harrison and H. Ossher. "*Subject-Oriented Programming (a critique of pure objects).*" In Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'93), pp. 411-428, Washington, September 1993.

[25] Gamma, R. Helm, R. Johnson and J. Vlissides: "Design Patterns: Elements of reusable object-oriented software", Addison Wesley, 1995.