Vol 21, No 2 (2014), JOURNAL OF FACULTY OF ENGINEERING & TECHNOLOGY (JFET) Pages 60-81

Reverse Object Oriented Design Methodology (R-OODM)

Muhammad Usman Ghani Khan, Mobina Zafar, Khadim Hussain Asif
Junaid Arshad, and Abad Ali Shah
Department of Computer Science and Engineering,
University of Engineering and Technology, Lahore Pakistan
Corresponding Author: usman.ghani@kics.edu.pk

Abstract

Software design plays a vital role in better understanding of the software system and its architecture. It also provides help throughout the software development life cycle and afterwards in its maintenance as well. Web applications have been emerging enormously and subjected to continuous changes due to high competition in the market. But unfortunately, most of the web applications are implemented without producing any formal design documentation for its subsequent maintenance and evolution. Therefore, the maintenance of these applications becomes a challenging problem as the complexity of the web application grows. The reverse engineering techniques has been used to support effective web application maintenance. We have proposed a Reverse Object Oriented Design Methodology (R-OODM) that extracts the design of web application using design phase models of OODM [1][15], based upon water fall model. A tool has also been developed for implementation of this reverse object oriented design methodology.

1. Introduction

The Web and its applications have become the most powerful medium of communication for commercial domain of all kinds. Most of the Web applications are not well structured and properly documented because during their development the basic software engineering principles are not practiced, hence maintainability and reusability of these applications become difficult and not manageable. Therefore, the maintenance of such applications becomes a challenging problem, and this problem becomes more complex as the complexity of web applications grows. Also, in most of cases, the design documents of web applications are not available because those applications are not developed by following the principles of software engineering and a proper software development technique is not used in their development. To address these situations the reverse engineering can be one possible solution to support effective web application maintenance [2].

In this research, we propose a technique to extract design of web applications. The extraction of design of web applications makes maintenance easy and effective. It covers the deficiencies and deals with challenges that the existing reverses engineering techniques and tools that are facing today [3].

Main objective of this technique is to extract design of web application by following the rules of web design methodology OODM [15].

Problems:

i) During the development of most of the web applications no software development technique has been followed, hence reverse engineering becomes difficult.

ii) In the development of many legacy web applications the structured approach has been followed, any object oriented software development technique has not been followed. Therefore, it becomes difficult to extend functionality of the applications, and their maintenance became difficult [4].

iii) The existing reverse engineering techniques for the web applications and tools based on these techniques are applicable only to those web applications that have been developed using the object oriented approach.

iv) The input of our proposed technique is XML+HTML and output is complete design of web application in the form of web development technique termed as Reverse Object Oriented Design Methodology (R-OODM).

Considering all above mentioned problems and issues we have propose a technique that extracts design from source code of web applications. Using our proposed technique, we have developed a tool that extracts design of web application.

1.1 Benefits

Our proposed technique will facilitate us with followings benefits.

i) The proposed technique is applicable for both types of web applications, structured and object oriented too. A structured based web application is transformed into object oriented format, than is parsed by proposed technique to extract its design. It is done using XML- Schema translation by mapping XML-Schema into object oriented database. For this mapping object graph generally termed as components graph is extracted from given XML schema [7].

ii) A dynamic analysis has been performed by this technique to capture the dynamic features and functionalities of web design.

iii) Design extraction process is in the form of development methodology. We have used Object Oriented Design Methodology OODM [15] that has been transformed into Reverse-Object Oriented Design Methodology (R-OODM).

iv) The remainder of this research work is organized as follows. In Section 2, a detail and in-depth literature survey is given and a comparison of the existing reverse engineering techniques is also given. Section 3 describes the proposed work. In Section 4, we presented case studies. The result analysis and discussion is given in Section 5. Section 6 describes the conclusion and future work of the research work. A complete list of references is provided at the end.

2. Related Work

Reverse Engineering is defined as a process of analyzing a subject system at high level of abstraction. Reverse engineering can be used in following situations.

- If the Source code is already available for the software, but high level aspects of programs are not valid or well documented.
- If no source code is available; an effort to discover one possible source code is termed as reverse engineering.

Reverse engineering is a knowledge-intensive and an iterative process. Reengineering is the examination and alteration of a legacy system to reconstitute it in a new form and the subsequent implementation of the new form.

Before going into depth of the reverse engineering process we must learn some basic concepts that will be helpful in giving us a clear picture of the process

2.1 The reverse engineering process

The figure below gives a brief description of where the reverse engineering lies and how it works and helps in re-engineering and re-designing the software system [5].

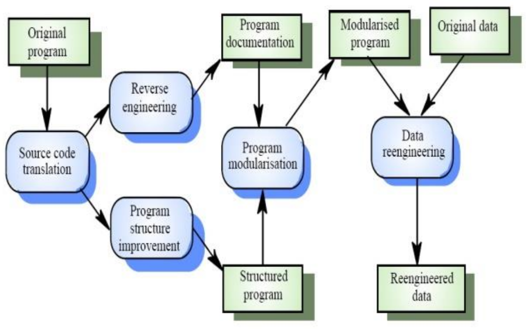

Figure 2. Reverse Engineering in action [15]

It takes original program as input and its source is translated for reverse engineering and structure improvement process. Reverse engineering process access the program documentation and transform into program modularization, at the same level program structuring process return an improved structure which ultimately translates to the modularization level. After this modularization, data engineering is performed with the help original data which return re-engineered data [6].

Reverse engineering is a process capable of enabling us to re-engineer and re-design a software system. It is a very broad term on high end it includes design recovery and on the other end recompilation and disassembly but the real meaning of reverse engineering is to get the understanding of something that is missing in the current working system that can be either source code, documentation or the software design [6]. It is an iterative process that can be started at or from any step of the software development life cycle to achieve and get its pre-requisite product or design [7].

2.2 Design extraction process

The process of design extraction from the source code is an iterative process. The following figure is a try to describe the reverse engineering process completely in a three step iteration process. The process can be well defined with the help of the following diagram:

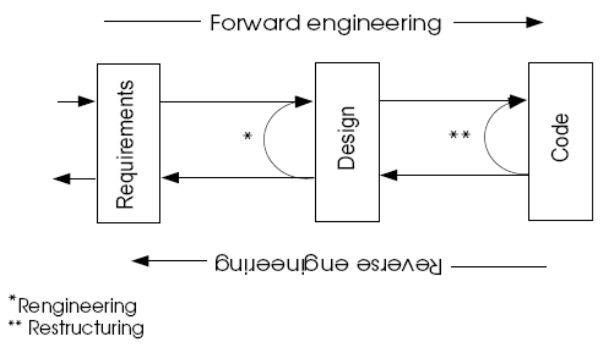

Figure 2.The Engineering Process

The figure 2 describes that how the process leads to achieve software design out of the software system.

The process of extracting design from the source code may differ from various techniques as there the era and techniques of software engineering have been varying from time to time from the classical approaches to web based approaches, from classical UI and console based applications to Object Oriented web designs [8]. So with the difference of the design methodologies the outputs and inputs for these methodologies also change, following figure gives a brief description of the mentioned scenario:

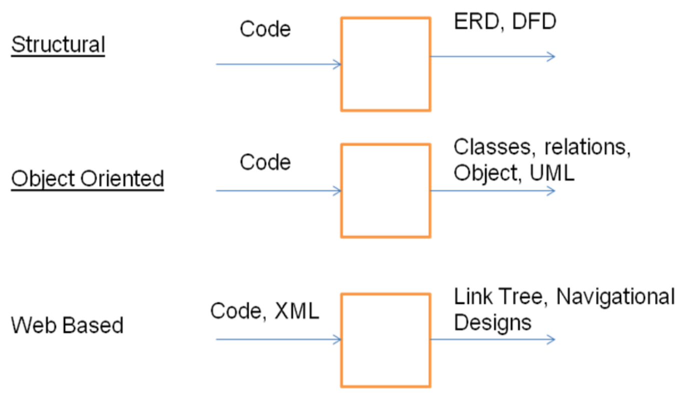
Structural
Code
ERD, DFD
Object Oriented
Code
Classes, relations, Object, UML
Web Based
Code, XML
Link Tree, Navigational Designs

Figure 1. Inputs and Outputs of Different S/W Design Methodologiees

2.3 Reverse Engineering Tools and Techniques

While going through the studies of different types of techniques have come into our studies some of the mentioned techniques are as following:

a) **PINOT**

Pattern Interface Recovery Tool (PINOT) detects all the Gang of Four (GoF) patterns that have concrete definitions driven by code structure or system behavior. The tool is faster, more accurate, and targets more patterns than existing pattern detection tools [14].

b) **Two Phase design**

The design consists of two different phases that consist of kind of phases i) going through the code of the system and ii) identifying the main methods and operations within the system. Then this technique uses recursive functions to identify and make a system diagram in the end [15].

c) **Manual Reverse Engineering (BOS/X)**

This is the system that is very unique and goes through reverse engineering process manually. This technique requires the user to go through the system manually line by line and by following the systems calls and then determine the main components of the whole system[13].

2.4 Reverse Web Development Techniques and Tools

All distinguishing features of existing reverse web development techniques and tools, brief description of each technique and tool is given bellow:

- **ReWeb**: In [10], A tool take traditional source code analysis of Web applications as input and represent the Web application as a graph structure and undertaking various types of analysis such as reachability, flow and traversal analysis. ReWeb can download and analyze Web applications. [5].
- **TERESA** [4] is a source code statistical analysis tool that produces a task-oriented model of a Web application.

2.5 Comparison of existing Reverse Engineering Techniques and Tools

The Table 1 gives a very clear idea on how the different techniques support various parameters and what is lacking in all of them.

Table 1. A Comparison b/w the All techniques

Tool	Object Oriented	Classic Approach	Web Applications	Design Patterns	Language Specific

PINOT	✓	✗	✗	✓	✓
BOS/X	✓	✓	✗	✗	✗
Cliché mining techniques	✓	✗	✗	✗	✓
Rationale Graphs	✗	✓	✗	✗	✓
2 Phase design	✗	✓	✗	✗	✓
ReWeb	✗	✗	✓	✗	✓
TERESA	✗	✗	✓	✗	✓

Parameters that have been identified for comparisons are object oriented approach, classical approach, web application, design patterns and language specific.
PINOT technique used object oriented development methodology rather classic approach. This technique does not work for web applications. It Support design patterns but it is language specific and support JAVA, AWT, JHotDraw, Swing and Apache Ant [9].
BOS/X technique also termed as manual reverse engineering comprises combination of object oriented and classical approach. It's not design for web applications. It does not support design patterns but it is language independent.
Cliché mining techniques used object oriented approach. This technique does not support web applications and design patterns. It's is language specific [11].

Rationale Graphs technique used classical approach but does not support web applications. It also does not support design patterns. It's language specific too.
2-Phase design technique used classical approach but does not support web applications. It also does not support design patterns. It's language specific too.

ReWeb technique does not support object oriented or classical approach rather it uses web development approach. It does not support design patterns. But it is language specific.

TERESA characterize same features as capture by ReWeb.

3. Proposed Design Extraction Technique

OODM [15], is the first object oriented design methodology for web applications. It covers all principles of software engineering and software development life cycle.

We have proposed a tool for reverse web engineering by following reverse object oriented design methodology named as R-OODM.

Our propose work maintain the web application at design phase of OODM [15].

R-OODM tool involve following steps for deign extraction of a web application.

Step-1: It takes input of source code in form of xml and any server scripting language.

Step-2: DTD (Document Type Definition) is extracted from source code.

Step-3: Components of design phase of OODM [15].

Step-4: Algorithm for each component to extract design node, that fall in relevant component model.

Step-5: Integration of all components to sum up the design.

3.1 Design Phase of OODM [15]

Design phase of OODM [15] involves key features like presentation of information that how it's presented to user, user navigation paths, implementation of each operation and user interface.

Design phase mainly composed of four models, Component model, Navigational model, Operation portioning model and User interface model [12].

Component model defined a set of related multimedia attributes of a page class. Each web page consists of components and each component is further comprising the set of multimedia attributes. Navigation model design the information structure of a Web Application. This information structure provides an orientation and guidance to the users while navigational through Web pages. The processing unit Building Operation-Partitioning Model further is divided into four processing units, namely Building Object-Interaction Graph, Building an Algorithmic Form for Operations, Partitioning Operations into Client and Server Operations, and Completing and Refining Operation. User interface model deals with the designing of user's perception and interaction with web application. The design process first determines user interface elements (for example, pages, forms, frames, colors, command buttons, bars, and check boxes) for the objects, e.g., page-classes, components, navigation types, operations, and navigation primitives [11].

3.2 Outline of Reverse Object Oriented Design Methodology(R-OODM)

Table 2. Outline Of R-OODM

Development Phase	Input	Output	Output Components Nodes
Design Phase			
Building Component Model	Document Type definition (DTD) in form of Information Model	Component Model	Component Nodes and component access sequence nodes
Building Navigation Model	Component Model + User navigation model in form of DTD	Navigation Model	Local Navigation Nodes, Global Navigation Nodes, Menu Navigation Nodes
Building Operation Model	DTD in form of operation Model	Operation Partitioning Model	Client and server operation nodes Event handler
Building User Interface	User Navigation Model + Component Model + Navigation Model + Operation Model	User Interface Model	Component User Interface, Navigation primitive user interface, Form User Interface Element, Presentation Nodes

R-OODM comprises same design phase model as OODM [9][15]. It extract these model at design phase. R-OODM adds value to existing OODM [15] by introducing reverse engineering technique. For extraction of design, this technique take DTD (Document Type Definition) of whole schema as input and out all design phase models.

Algorithm has been written for each design phase model which helps to extract node of each model.

3.3 XML Extraction Algorithm

XML extraction algorithm has been written to extract data nodes from the source code (XML schema). As we are proposing reverse engineering methodology, input of this algorithm is source code and out will be the data nodes and associated values like node type attributes, depth and content.

3.3.1 Output File Layout

This table describes the output file format. Output nodes will be categories into different models based upon its values.

Document node and element node are two basics types of nodes. Document node does not hold any attribute, its depth level is 1 and it contains inline nodes. Rest of the schema mainly consists of element nodes. Element node hold optional attributes, there can be no attribute or one attribute or more than one attributes. Depth of element node is equal or greater than two. Content of element can be text, image, form etc.

Table 3. Output file Layout

Node	Type	Attribute	Depth D	Content
Document Node	DOCTYPE	Nil	D=1	Inline Nodes
Element Node	ELEMENT	Optional	D>=2	Text, Image, Form etc.

With the help of below algorithm we come up with DTD (Document Type Definition). Two steps of information processing, mentioned in section 3, have been done up till now. At third step, components of design phase of OODM [15] are considered. Algorithm for each component is defined to extract the relevant set of nodes.

```
InputFile= XML_file ;
OutputFile=DTD_file;        /* All nodes, their attributes, depth and text will be written in this file */
Processing[ ];              /* For xml manipulation */
Output[ ];                  /* For output */

Operation XMLExt:  GetExtractionXMLDOC( inputFile ):outputFile
Check to see if file open (1)
If yes then
Parse the file (2)
If successfully read nodes then
 Write contents into output file (3)
            If file opens then
            Return successful output
                          Else
Return an error message “could not open output file”
Else
Return error message
Else
Return a error message “could not open XML input”

Method  XMLExt :Read(inputFile) :Array
If user read valid xml file then
Return data
Else
Error Message
Method  XMLExt :Parsing(source) :Array
If source is not empty then get nodes (2.1)
```

```
        If yes then
Get the depth, attribute and contents of the node
Else
Error Message
Else
Error Message
Method  XMLExt :getNode(Source) :Boolean
If source not empty
Return true
Else
False
Method XMLExt: Write (outputFile, Output) :Boolean
If Output [ ] is not empty and contain valid data and file open successfully
Return true
Else Return false
```

3.3.2 Proposed DTD structure

Before we start individual component model extraction Document type definition structure is describe below. XML schema starts with root element and descended by child elements.

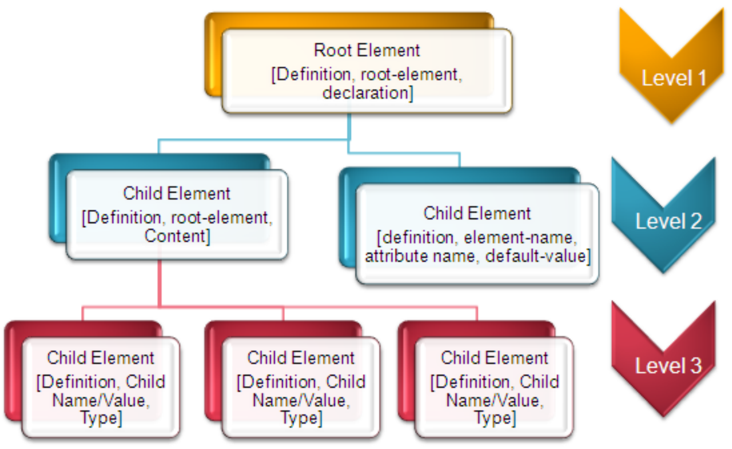
Root Element
[Definition, root-element, declaration]
Level 1
Child Element
[Definition, root-element, Content]
Child Element
[definition, element-name, attribute name, default-value]
Level 2
Child Element
[Definition, Child Name/Value, Type]
Child Element
[Definition, Child Name/Value, Type]
Child Element
[Definition, Child Name/Value, Type]
Level 3

Figure 3.4. Propose DTD Structure

Component Model Extraction

Procedure1: Component Model

Input: DTD

Output: Component nodes (Component Model)

```
NodesCollection [] =input;
Check for each node
If (DepthOf (Node) ==1)
    Attribute=AttributeOf(Node);
   If(AttributeOf(Node)==SRC OR
Node is parent Node
Else
Continue
If (ContentOf(Node)! ="")
Content Node found
else
Continue
```

Navigational Model Extraction

Procedure 2: Navigational Model

Input: DTD

Output: Navigational Nodes (Navigational Model)

```
Start Procedure:
NodesCollection [ ] =DTD
NavigationNode([0]=>UL ,[1]=>A,[2]=>LI)
Attributes[]=All Attribute Found
i=0;
While(NodeCollection[i])
{
If(AttributOf(Node)==HREF AND ValueOfAttribute(Node)!="")
{
OutPut[i]=Node[i];
}
}
end while
End procedure:
```

Building User Interface Model (Presentation Model) Extraction

Procedure 3: Building User Interface Model (Presentation Model):

Input: DTD

Output: Presentation Nodes (Presentation Model)

```
Start Procedure:
NodesCollection [ ] =DTD
Attributes[]=All Attribute Found
i=0;
While(NodeCollection[i])
{
If(AttributOf(Node)==STYLE OR AttributOf(Node)==WIDTH OR AttributOf(Node)==HEIGHT OR AttributOf(Node)==CLASS)
{
OutPut[i]=Node[i];
}
}
end while
End procedure:
```

Operation Partitioning Model Extraction

Procedure 4: Operation Partitioning Model:

Input: DTD

Output: Operation Partitioning Nodes (Operation Partitioning Model)

```
Start Procedure:
NodesCollection [ ] =DTD
Attributes[]=All Attribute Found
i=0;
While(NodeCollection[i])
{
If(AttributOf(Node)==OnClick() OR AttributOf(Node)==SCRIPT OR AttributOf(Node)==OnSelect() OR AttributOf(Node)==onChange())
{
OutPut[i]=Node[i];
}
}
```

```
end while
End procedure:
```

After defining procedures few rules have also been defined to grouping the nodes.

3.4 Grouping of Nodes

3.4.1 Define Rules for Grouping

According to our purpose algorithm, few rules have been extracted to define nodes.

1. **Depth**

Depth defines the node type.

Depth 0 means document node, depth 1 means parent node and depth 2 or greater than 2 means child nodes.

2. **Node Type value**

Values of the node predict node type according to DOM manipulation rules.

Main Types of nodes

XML document consists of many syntax structures that do not contribute towards XML content. But they provide interface to access element data.

- **Structural node**

DocumentType, Processing Instruction, Notation, entity, attribute, CharacterDataType, CDATASectioon

- **Content Node**

Element Node, Text node

Extracted nodes are needed to classify.

I. **Navigational Nodes**

Hyperlink node, Menu node, Index Node and Guided Node

II. **Representation Nodes**

CDATA Node, Font Node, Formatting Nodes

III. **Behavioral Node**

a. Attribute Nodes

3.5 Components of R-OODM:

3.5.1 Component Model

A set of related multimedia attributes of a page-class is referred to as a component.

Content nodes come under the heading of component model.

3.5.2 Building Navigation Model

The main function of this unit is to design the information structure of a web application. This information structure provides an orientation and guidance to the users while navigational through Web pages. This model include the navigational nodes

3.5.3 Building Operation Partitioning Model

The processing unit Building Operation-Partitioning Model further is divided into four processing units, namely, Building Object-Interaction

Graph, Building an Algorithmic Form for Operations, Partitioning Operations into Client and Server Operations, and Completing and Refining Operation Partitioning Model. Structural nodes involve in this model.

3.5.4 Building User Interface

This processing unit deals with the designing of user's perception and interaction with a web application.

The design process first determines user interface elements (for example, pages, forms, frames, colors, command buttons, bars, and check boxes) for the objects, e.g., page-classes, components, navigation types, operations, and navigation primitives.

Presentation nodes come under this heading.

4 Case Studies

1. Case Study : Language and learning Online(LALO)
2. Case Study: Book Store

4.1 Case Study 1: Language and learning Online(LALO)

Language and learning online is a XML-based web application of Monash University, which provides online learning facility for students. Students can improve their writing, reading and speaking skill.

Its data flow diagram and trace of XML source is given below.

4.1.1 Trace of Algorithm for website of Language and learning DTD:

Node	Type	Node Title	Attribute	Attribute value	Depth	Content
Document Node	DCOTYPE	Home	Nil	Nil	0	Nil
Root Node	HTML	HTML	XMLNS	http://www.w3.org/1999/xhtml		
			XML:LANG	en		
			LANG	en		
Parent Node		HEAD			1	

Child Node		META	HTTP-EQUIV	Content-Type	2	
			CONTENT	text/html; charset=utf-8		
		TITLE			2	Language and Learning Online
		BASE	HREF	http://www.monash.edu.au/lls/llonline/	2	
		LINK	REL	stylesheet	2	
			HREF	assets/styles/lls-styles.css		
			TYPE	text/css		
			MEDIA	all		
Parent Node		BODY			1	
Child Node		DIV	ID	accessibility	2	
		A	HREF	javascript:accessQueryString();	3	Accessible version
		DIV	CLASS	spacer	2	
		TABLE	WIDTH	100%	2	
			BORDER	0		
			CELLSPACING	1		
			BGCOLOR	#000000		
			SUMMARY	Layout for site-wide navigation		
		DIV	ID	search		
			STYLE	float: right	8	
		FORM	NAME	search-form	9	
			ACTION	search.php		
			METHOD	get		
		LABEL	FOR	search-field	13	search field
		INPUT	TYPE	text		
			NAME	q		
			ID	search-field		
			CLASS	searchfield	13	
		DIV	ID	breadcrumbs	3	
		Home				
		DIV	CLASS	header	2	
		UL			3	
		LI	ID	current	4	
		A	HREF	index.xml	5	HOME
		LI			4	
		A	HREF	reading/index.xml	5	Reading
		LI			4	
		A	HREF	writing/index.xml	5	Writing
		LI			4	
		A	HREF	speaking/index	5	Speaking

				.xml		
		LI			4	
		SPAN	STYLE	float:right	6	
		IMG	SRC	assets/images/logo150.gif	8	

4.1.2 Design Phase Model for Case Study 1

4.1.2.1 Component Model

- Home
- HTML
- HEAD
- META
- TITLE
- BASE
- LINK
- SCRIPT
- BODY

4.1.2.2 Navigation Model

Attributes for navigation model:

- HREF, UL, LI, A

4.1.2.3 User Interface Model

- Table, TD, TR, Span, Strong., H2
- Size, Width, Height, Class, Style, font, color, background, background color, border
- IMG

4.1.2.4 Operational Partitioning Model

javascript:accessQueryString();

4.2 Case Study 2 : Book Store

This case study is a XML document of a book store, which keeps a catalogue of books. Each books has its detail, book title, price, author name, publishing date and description. Trace of its Documents Type Definition (DTD) is given below.

Table 4.2. Trace of Algorithm for Book Store

Node	Type	Node title	Attribute	Attribute value	Depth	Content
Document Node	DCOTYPE	Books	Nil	Nil	0	Nil
Root Node	ELEMENT	CATALOG	Nil	Nil	0	Nil

ParentNode	ELEMENT	BOOK	ID	bk101	1	Nil
Child Node	ELEMENT	AUTHOR	Nil	Nil	2	Gambardella, Matthew
	ELEMENT	TITLE	Nil	Nil	2	XML Developer's Guide
	ELEMENT	GENRE	Nil	Nil	2	Computer
	ELEMENT	PRICE	Nil	Nil	2	44.95
	ELEMENT	PUBLISH_DATE	Nil	Nil	2	2000-10-01
	ELEMENT	DESCRIPTION	Nil	Nil	2	An in-depth look at creating applications with XML.
ParentNode	ELEMENT	BOOK	ID	bk102	1	Nil
Child Node	ELEMENT	AUTHOR	Nil	Nil	2	Ralls, Kim
	ELEMENT	TITLE	Nil	Nil	2	XML Developer's Guide
	ELEMENT	GENRE	Nil	Nil	2	Computer
	ELEMENT	PRICE	Nil	Nil	2	44.95
	ELEMENT	PUBLISH_DATE	Nil	Nil	2	2000-12-16
	ELEMENT	DESCRIPTION	Nil	Nil	2	A former architect battles corporate ….

4.2.2 Design Phase Model for Case Study 2:

4.2.2.1 Component Model

Nodes that are classified as component model:

Books, Catalogue, Book, Author, Price, Publishing Date and Description

Content and attributes of all nodes are included in this model.

4.2.2.2 Navigation Model

- HREF

4.2.2.3 User Interface Model

List of attributes that comes under user interface model is provided below: WIDTH/HEIGHT, FONT, COLOR and ID.

4.2.2.4 Operation Partitioning Model

No operation partitioning model

5 Results and Analysis

In this section, we have summarized our results of case studies. Trace of algorithm and design phase model has been provided in previous section.

Now the models of design phase of object oriented design methodology- OODM [15] are compared and analyzed for each case study.

Table 5.1. Analysis of case studies

Case Study No.	Component Model		Navigation Model		User Interface Model		Operational Partitioning Model	
	Status	% Weight	Status	% Weight	Status	% Weight	Status	% Weight
1	Yes	Full	Yes	Full	Yes	Full	Yes	Full
2	Yes	Full	Yes	Full	Yes	Full	No	Empty

In the design phase, the presentations of information to users, user navigation paths, implementation of each operation, and user-interface elements are designed.

5.1 Design Phase of R-OODM

Design phase of R-OODM consists of four stages, same like OODM. Main processing units are Building Component Model, Building Navigation Model, Building Operation Model and Building user interface model. In next sections, we describe these four units.

In this section we will extract the design of **Language and Learning Online (LALO)** and **Book Store** describe in section 4.1 using DTD extracted from XML source code.

5.1.1 Building Component Model

We have extracted components nodes in section 4.1.2.1. For building component models, these component nodes are basically the meaningful and logical units that made up pages-class. These logical units are termed as components. Each component contains related multimedia attributes that is a part of page class, so a page class is mainly consists of components. In this section we will extract the design of **Language and learning Online (LALO)** using DTD extracted from XML source code.

We have extracted components nodes in section 4.4.2.1 for building component models. Its page class detail is given in below table.

The access order of these is as follows: Learner, Reading, Writing, speaking, Learning, Listening, Grammar, and Quick Reference.

Table 5.2. Building Component Model for Language and Learning Online Case study

Components	Multimedia Attributes	Order Of Access
Learner	Learner Detail	1
Reading	Reading text	1
Writing	Writing Text	2
Speaking	Speaking Text	2
Grammar	Grammar Text	3
Quick Reference	Quick References	3
Learning	Learning Text	2
Listening	Listening Text	2

5.1.2 Building Navigation Model:

As per OODM [15], this unit deigns navigational paths that enable hypermedia navigation in web application. These navigation paths have been divided into four categories, local, instance, global and menu navigation. Local navigation for each component of page class is built by using building component model. For each component, an index guided tour or local navigation is built to access information from component. Figure shows the graphical representation of page-class *Language and Learning Online.*

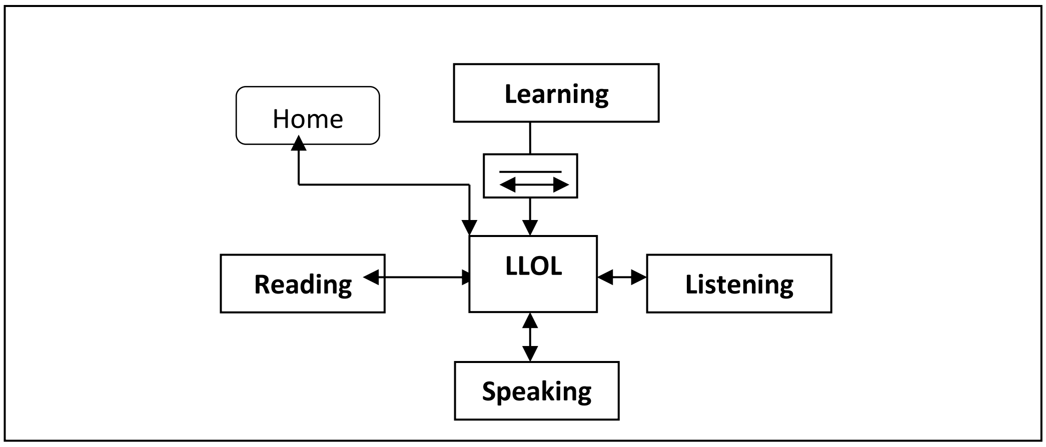

Figure 5.1. Graphical Representation for navigation Language and Learning Online Page-class

Local Navigation:

A user navigates through his/her profile, providing a list of options.

Global Navigation

A user may navigate from a learner to Listening area, from a learner to speaking area, from a learner to writing area.

Instance Navigation

A user may navigate from a listening area to learner area, from a speaking area to learner area.

5.1.3 Building Operation Partitioning Model

The processing unit Building Operation-Partitioning Model further is divided into some processing units, namely, Building Object-Interaction Graph, Building an Algorithmic Form for Operations, and Partitioning Operations into Client and Server Operations.

In our extracted design, few operations have been extracted that perform validation check and data loading.

5.1.4 Building User Interface Model

Nodes entitle as form, style, width, color, frame, paragraph, table, table data, table row and all navigation primitive have been collected under this user interface model.

5 Conclusion and Future:

We propose a Reverse object oriented design methodology (R-OODM), and we have developed a tool based on this reverse object oriented design methodology that extracted design of a web application. This reverse engineering methodology has followed the phases of Water Fall software development life cycle model. Our proposed methodology has extracted the design of web application, by extracting the components of design phase of OODM [15].This methodology has overcome the deficiencies of existing reverse engineering methodology. It provides support for maintaining existing web applications. Now with the help of this methodology, maintenance of legacy system has become easy.

According to our result and conclusion, we have extracted three main components completely but operation portioning partially. We are working over it to extract this component completely.

References

[1] Aurore, F., Jean H., Jean-Luc H.; Web Reverse Engineering: Fabrice Estiévenart1, , IEEE Explore, 2008.

[2] Irina A.; Toward the Semantic Web – An Approach to Reverse Engineering of Relational Databases to Ontologies: Advances in Databases and Information Systems: proceedings of the 9th East-European Conference, ADBIS 2005, Tallin, September 12-15, 2005. - ISBN 9985-59-545-9. – Tallin.

[3] Irina A., Bela S.; Reverse Engineering of Relational Databases to Ontologies: An Approach Based on an Analysis of HTML Forms, Australia, 2004.

[4] Building ontologies from relational databases using reverse engineering methods, Computer technologies, 2007, ISBN:978-954-9641-50-9.

[5] Girardi, C., Pianta, E., Ricca, F. and Tonella, P.; Restructuring Multilingual WebSites: (2002).Proc. 18th Int. Conf. on Software Maintenance (ICSM'02), IEEE, PP. 290-299.

[6] Antoniol, G., Canfora, G., Casazza, G. and De Lucia, A.; Web Site Reengineering Using RMM: (2000). Proc. 2nd Int. Workshop on Web Site Evolution, PP. 9-16.

[7] Conallen, J.; Building Web Applications with UML: (1999). Addison Wesley. ISBN: 0-201-

61577-0.

[8] Di Lucca, G.A., Di Penta, M., Antonniol, G. and Casazza, G.; An Approach for Reverse Engineering of Web-Based Applications: (2001). Proc. 8th Working Conference on Reverse Engineering, WCRE'01, IEEE, PP. 231-240.

[9] Reverse Engineering: Accessed on 2014 http://www.answers.com/topic/reverse-engineering.

[10] Harris; David, R. ; Reubenstein, Howard B. ; Yeh, Alexander S. Recognizers for Extracting Architectural Features from Source Code: Reverse Engineering, 1995., Proceedings of 2nd Working Conference; 14-16 July, 1997. PP. 252-261. by David R. Harris', Howard B. Reubensteint, Alexander S. Yeh 'The MITRE Corporation".

[11] Lerner, M. ; Tomin Corp., Wellesley, MA ; A Process of Re-engineering Large and Complicated Systems: Systems, Man, and Cybernetics, 1991. 'Decision Aiding for Complex Systems, Conference Proceedings., 1991 IEEE International Conference, 13-16 October, PP. 479-485 Vol. 1 (1991).

[12] Hiroyuki Sugawara, Tsuneo Hagiwara, and Tetsuya Numajiri; Design Extraction System for Rapid Development of Object-Oriented Switching Software: Communications, 1997. ICC'97 Montreal, Towards the Knowledge Millennium: 1997 IEEE International Conference. 8-12 June, 1997; PP. 231-235 (vol. 1). NTT Network Service Systems Laboratories, Tokyo, Japan.

[13] Smith, J.M.; Stotts, D. ; SPQR: Flexible Automated Design Pattern Extraction from Source Code; Automated Software Engineering: 2003. Proceedings 18th IEEE International Conference on October 6-10; PP. 215-224 North Carolina, Chapel Hill, NC, USA.

[14] Yang, S.; Zhou, X.W.; and Zhang, M.Q.; Approach on aspect-oriented software reverse engineering at requirements level: Computer Science and Software Engineering, 2008 International Conference on. Vol. 2. IEEE, 2008., PP. 321-324,Dec 12-14 (2008).

[15] Shah A. OODM: An Object-Oriented Design Methodology for Development of Web Applications: IDEA GROUP PUBLISHING, USA., ITB8759, PP. 189-229 (2003).