ISSN:0975-9646

Designing a Framework for Web Testing using Object-oriented Testing Approach

Nimish Kumar[#1], Reena Dadhich[*2], Aditya Shastri[#3]

[#]*Apaji Institute of Mathematics & Applied* Computer *Technology*
Banasthali University, India
[*]*Department of Computer Science & Informatics*
University of Kota, India

***Abstract*— The problem of how to effectively and efficiently test a Web-based application is an open issue. Also, Web-based applications are turning into the centre business for practically all dominating significant organizations of the business sector in all perspective and all ranges. As Web applications have become more common, the quality assurance of Web applications has become more imperative. Underlying complexity of Web applications makes testing of Web applications more difficult than conventional software. It is critical to develop efficacious methodologies and framework to test a Web application.**

As Web applications are part of global economy, it is critical for Web applications to be tested exhaustively. The growing demand for use of internet is a powerful economic reason to produce and maintain high quality Web-based applications. Most of the researchers have focused on narrow range of certain test area based on architecture, model, attributes, scenario, domain functionality and other non-functional qualities but Object-oriented approach of Web testing is still warrant more research.

***Keywords*— Web-based Applications, Page Navigation Diagram, Object-oriented Framework, Hierarchical Framework, Object State Diagram.**

I. INTRODUCTION

Software testing plays an important role in traditional as well as Web application software development. Testing is performed in intent to find an error in a program or software application and is conducted thoroughly before deploying or delivering it to clients. In other words, testing ensures that the implemented software is what the client wants and should be faultless before delivery. The formal testing methods that are applied to conventional software development are not suitable for the ever changing, time-to-market constrained Web application software. For this informal techniques are used, which are tedious and confusing software testing jargon but are widely accepted by software industry.

Current Web implementation is Object-based and standard and nonstandard resources of Web can be viewed as object. Considering Web as Object-oriented system can yield an extensible solutions that are capable of supporting existing functionality and allows the faultless integration of more complex resources and services. The Web depicts both static and dynamic behaviour of a Web application that captures both structural and behavioural test aspects of Web applications.

Due to heterogeneous technologies and complex structure of Web pages controlling and testing of large scale Web application is a time consuming and expensive process. This paper presents a new framework based on Object-oriented testing methodologies. The hierarchical design proposed in this paper is a framework which will decrease complexity and nested relations between Web pages. This architecture supports both Object-oriented and structured programming methods. Hierarchical architecture increases control and security of a Web application in its different levels. Flexibility of adding new modules and ease of maintenance are the key features of this architecture.

II. RELATED WORK

The world is moving towards transparent computational environment offered by internet. Many frameworks have been proposed by researchers to develop Web applications depending upon the choices of the users. The frameworks of Web-based applications provide flexibility to the designers in choosing proper development tools for the implementations of Web-based applications. Several categories of solutions for Web application development have been investigated and identified by Fraternali [1].

Switching from traditional approaches to Object-oriented approach has many advantages. Munassar and Govardhan [2] compared these models and stated that traditional approach has a lot of models that deal with different types of projects but other lacked flexibility to deal with projects like Object-oriented. Sivakumar and Vivekanandan [3] also analysed and compared various approaches of testing process carried out in conventional software development, Object-oriented software development and agent-oriented software development. Lucca and Fasolino [4] presented the main differences between Web-based applications and traditional ones. The focus is mainly on testing the functional and non-functional requirements of a Web-based application with some indications about future trends in Web application testing.

Gellersen et al. [5] introduced the Web Composition Markup language (WCML), which is an XML application and allows describing design and code in an Object-oriented way. Kung et al. [6] presented a methodology that uses an Object-oriented Web Test Model (WTM) to support Web application testing. Both structural and behavioural test artifacts of Web applications are captured by the model and represent the artifacts from the object, behaviour, and structure perspectives. Based on the test model, both

structural and behavioural test cases can be derived automatically to ensure the quality of Web applications.

In their work, Liu et al. [7] extends traditional data flow testing techniques to Web applications. In the test model, each component of a Web application is modelled as an object. The data flow information of the Web application is captured using flow graphs. Rossi and Schwabe [8] discussed the use of an Object-oriented approach for Web-based applications design, based on Object-oriented Hypermedia Design Method (OOHDM).

A testing techniques based on analysis model has been given by Ricca and Tonella [9]. Girgis et al. [10] proposed Web testing approach that guaranties the satisfaction of two Web application testing criteria, namely page coverage criterion and hyperlink coverage criterion. Sampath et al. [11] proposed the application of concept analysis for clustering user sessions and a set of heuristics for test case selection. Andrews at el. [12] evaluated a solution that uses constraints on the inputs to reduce the number of transitions, thus compressing the FSM.

Mansour and M. Houri [13] proposed new techniques for white box testing of Web applications developed in the .NET environment with emphasis on their event-driven feature. They extended recent work on modelling of Web applications by enhancing previous dependence graphs and proposing an event-based dependence graph model.

Artzi and Pistoia [14] leveraged combined concrete and symbolic execution and several fault-localization techniques to create a uniquely powerful tool for localizing faults in PHP applications. Yang et al. [15] presented a software testing architecture that integrates several common but enhanced testing tools as sub-architecture to fit a common Web application framework. The architecture reuses several software patterns and architectures from traditional testing environments. Marchetto et al. [16] proposed state-based testing technique, developed to test AJAX based applications and compared to existing Web testing techniques, such as white-box and black-box ones. The results show that state-based testing is complementary to the existing Web testing techniques and can reveal faults.

Garousi et al. [17] retrieved 147 papers in the area of Web application testing, which have appeared between 2000 and 2011. The results of their systematic mapping can help researchers to obtain an overview of existing Web application testing approaches and indentify areas in the field that require more attention from the research community. Li et al. [18] presented a broad survey of recent Web testing advances and discuss their goals, targets, techniques employed, inputs/outputs and stopping criteria. A large class of Web application testing is identified by Doğan et al. [19]. Authors reviewed functional testing of Web application through a systematic literature review (SLR) study. They outcome with the following results: (1) the list of test tools in the Web application testing area and their capabilities, (2) the types of test models and fault models proposed in this domain, (3) the way the empirical studies in this area have been designed and reported, and (4) the state of empirical evidence and industrial relevance. Their work also concluded the emerging trends in Web application testing, and discussed the implications for researchers and practitioners in this area.

III. ANALYSING ARCHITECTURE

A typical Web-based application generally consists of two kinds of pages:

- Standalone single pages that are distinct enough to form a section of their own e.g. About-us, Contact-us or Privacy-policy etc.
- Pages that can be grouped together under a common section e.g. all the pages belonging to the blog section or portfolio section.

A Web-based application testing method represents objects from three different perspectives:

a) Web object: This view point describes Web application entities as objects and their inter relationships

b) Web behaviour: This view point represents navigational and state-dependent behaviours of a Web application

c) Web structure: This view point shows control and data flow information of a Web application

A Web-based application generally consists of more than one page that includes scripts from diverse programming languages. These pages are called separately or nested pages. The primary purpose of nested pages is to create a hierarchy of pages that can be represented as the Web application menu.

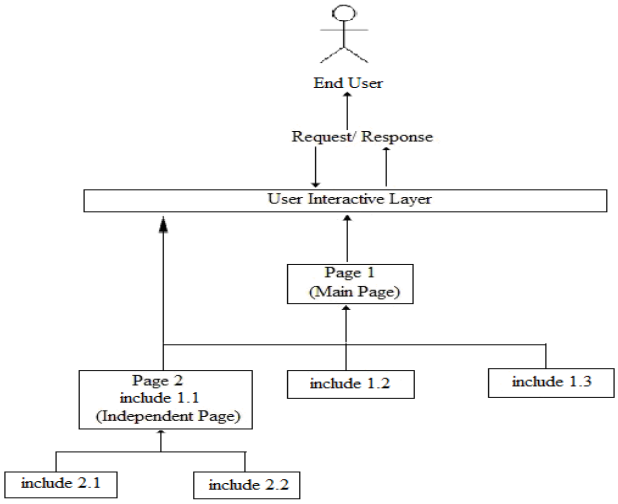

Fig. 1 Implementation of Nested Web-page Call in Response to User Request

As shown in fig. 1, a user may call *main page* (*page 1*) which may include set of scripts from diverse programming languages and independent pages (*page 2*) to accomplish requested process. As an example in an online shopping system, suppose *page 2* is a "Customer Authentication" page and *page 1* is a "Login or Check out" page. If a customer authentication is required in check out process, it needs to include "Customer Authentication" page in other pages. These nested includes are significantly increasing and creating serious problems in software development.

IV. ADVANTAGES OF USING HIERARCHICAL FRAMEWORK

- In a team work each group is assigned some task and after completion of task by each group further the task is integrated. It is very hard to maintain and implement change requests coming from customers. In totality it is difficult to see the advantages of team working. Although Object-oriented programming and class view help programmers to work together and use their sub-class in further uses but using hierarchical architecture will help them to drastically increase their collaboration.
- Web-based applications are often poorly structured and documented due to rapidly evolving environment and short time to develop. Adding new modules and features to the software is an issue. But due to the lack of relationships between different pages and integrated architecture, software maintenance is easy.
- Security is another important issue in software development especially in Web-based applications. In hierarchical architecture security is considered on each level thus preventing unauthorized access to the whole or some parts of software.
- Hierarchical framework diminishes redundancy in coding.
- File such as CSS, JavaScript global classes and libraries need not be included in each page of Web application, instead they should be included in the top levels.
- Creating a log file can be easily performed in each level and for every user.
- It is possible to add new modules, classes, CSS files in each level.
- All the Object-oriented methodologies can be easily accessed using this framework.
- Reusability of code is possible.
- All Object-oriented testing methodologies can be easily applied to the framework.

V. HIERARCHICAL ARCHITECTURE OVERVIEW

In the proposed hierarchical architecture all the Web pages are integrated in root page through which all requests are made to the system. When a user requests an HTML forms (templates), the request is sent to the requested section (Section, Subsection variables) and then to the requested function (Action variables).
Every form has generally two states:

- First, it is not submitted and the user will see a raw form to fill it out.
- Second, it is submitted and the information is sent to the server.

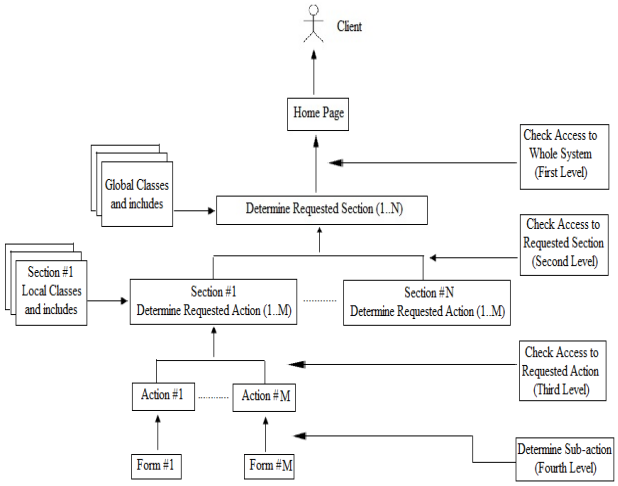

Fig. 2 Hierarchical Framework Integrating all Web-pages in Root Page

These two states can be checked by using "do-action" variable as shown in Table I. Figure 2 shows logical view of our proposed hierarchical architecture. All of the programming parameters such as such as GET, POST, SESSION, and SERVER variables are called and exchanged using main page. The pseudo code of the proposed architecture is implemented PHP as shown in Table I but it can also be implemented in any programming language.

TABLE I

IMPLEMENTATION OF HIERARCHICAL ARCHITECTURE IN PHP PROGRAMMING LANGUAGE FROM [20], [21]

index.php

```
<?php
//Get section, subsection, action variables
$section=$_GET['section']; $subsection=$_GET['subsection']; $action=$_GET['action'];
/* Determine movement path in hierarchical architecture by using section and subsection variables */
include ("include/determine_movement_path.php");
/* … Your Custom Script Code … */
?>
<html>
<title>Web Application Index Page</title>
<body>
<!-- Your Custom HTML Code -->
<?php include($form_to_show); ?>
<!-- Your  Custom HTML Code -->
</body>
</html>
```

Implementing root sections

```
<?php
switch ($section) {
        //Section 1
        case "section1" : {
                checkUserAccess ($_SESSION['user_id'], "section1");
                include ("class/section1_class.php");
                $section1_obj = new Section1();
                //Continue determining path by subsection variables
                include ("class/section1_determine_path.php");//Determine requested action in next level
                break;
        }
        //Section 2
        case "section2" : {
              checkUserAccess ($_SESSION['user_id'], "section2");
              include ("class/section2_class.php");
              $section2_obj = new Section2();
              include ("class/section2_determine_path.php");//Determine requested action in next level
              break;
        }
}
?>
```

Implementing section1 actions

```
<?php
switch($action) {
        case "action1" : {
        checkUserAccess($_SESSION['user_id'], "section1", "action1");
        if ($_POST['do_action']) {
                $result = $section1_obj->action1();
                if ($result == false) {
                        //Print error message and show the related form again
                        $form_to_show = "templates/section1_action1.tpl";
                } else {
                        /*Action done successfully so show proper message/form to user and do your custom tasks*/
                }
        } else {
                $form_to_show = "templates/section1_action1.tpl";
        } }// End of case
}// End of switch
?>
```

Based on the requirements levels in the architecture can be added or deleted. This gives facility to extend or enhance a model. Hierarchical structure allows rapid navigation and access to the different contents of the Web application. For example server files and directories are organized as shown in fig. 3. Here *Classes* consists of PHP class files, *Includes* consists of CSS, JavaScript, library files and all that needed to include globally or locally and *Templates* contains all forms in template format.

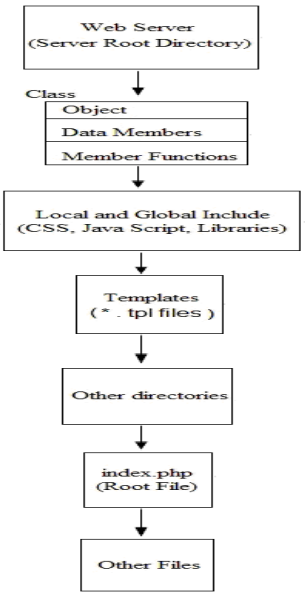

Fig. 3 Hierarchical Organization of Server Files and Directories

VI. Applicability of Framework for Object-Oriented Testing Methodologies

A Web-based application employs an entirely new and different way to deploy software solution to the end users. Web applications are complex interactive programs that employ huge GUIs, several back-end software components, number of new languages, technologies, and programming models that are integrated with each other in novel ways. Models represent the components of the application and their interconnections under test. A test case for a Web-based application is a sequence of pages to be visited and the inputs values provided to the pages containing them. To test proposed hierarchical framework both structural and behavioural test cases can be derived automatically to ensure the quality of Web applications. According to Kung et al. [6], Web-based application testing method represents objects from three different viewpoints:

a) *Web Object Perspective:* This view point describes Web application entities as objects and their inter relationships represented by an object relation diagram.

b) *Web Behaviour Perspective:* This view point represents navigational behaviour using a page navigation diagram and state-dependent behaviour of a Web application using object state diagram.

c) *Web Structure Perspective:* This view point shows control and data flow information of a Web application and is represented using flow graphs.

A. Web Object Perspective

Each entity in a Web-based application can be viewed as an object, which describes structural aspects of a Web-based application. Web application generally describes three types of objects:

- **Client pages:** A client page consists of an HTML document which is accessed through the Web browser on the client side.
- **Server pages:** A server page is a Common Gateway Interface (CGI) script executed by the Web server on the server side. Other server side scripts are Active Server Page (ASP), a Java Server Page (JSP), or a servlet.
- **Components:** A component can be any program module that interacts with the client pages, server pages, or other components. Examples are an HTML element, a Java applet, an ActiveX control, a Java Bean or a server-side included HTML file.

In a Web-based application, a server page can generate different client pages dynamically in response to various HTTP requests. These generated pages have dynamic behaviours when they are interpreted by a Web browser and are embedded within the server page as static output statements. Thus, on the client-side, these generated client pages have their own structures and behaviours similar to HTML files. That's why generated client pages are treated as independent objects separate from their server pages. This view point describes Web application entities as objects and their inter relationships represented by an Object Relation Diagram (ORD).

In testing proposed hierarchical framework, an *ORD = (V, L, E)* is a directed tree, where objects are represented by a set of nodes *V*, relationship types by *L***,** and set of edges representing the relations between the objects by *E*. Therefore, an object can be represented hierarchically to provide levels of abstraction and detail.

To accommodate the new features of Web applications, new relationship defined are: *navigation, request, response,* and *redirect* which are used to model the navigation, HTTP request/ response, and redirect relations introduced by Web applications, respectively. Thus in ORD, the set of label is defined by seven relationships as:

$$L = \{Iht, Agg, Asso, Nav, Req, Res, Rdt\}$$

TABLE II

SHOWS WEB APPLICATIONS RELATIONSHIPS AND THEIR MEANING

Relationships	Meaning
Iht	Inheritance
Agg	Aggregation
Asso	Association
Nav	Navigation
Req	Request
Res	Response
Rdt	Redirect

Various edges and their definitions represented by ORD are as follows:

TABLE III

SHOWS EDGES AND THEIR DEFINITION REPRESENTED BY ORD

Edges	Definition
Request (E_{Req})	Represents an HTTP request relation between a client page and a server page.
Response (E_{Res})	Represents an HTTP response relation between a client page and a server page. For any two pages V_1 and V_2, E_{Res} indicates that the client page V_1 is generated by the server page V_2.
Navigation(E_{Nav})	*E*dges representing a navigation relation between two client pages.
Redirect(E_{Rdt})	Edges representing a redirect relation between two server pages. For any two server pages V_1 and V_2, E_{Rdt} indicates that the server page V_1 redirects an HTTP request to the server page V_2.
Inheritance(E_{Iht})	Represents inheritance relationship between server and client pages.
Aggregation(E_{Agg})	Represents aggregation relationship between server and client pages.
Association(E_{Asso})	Represents association relationship between server and client pages.

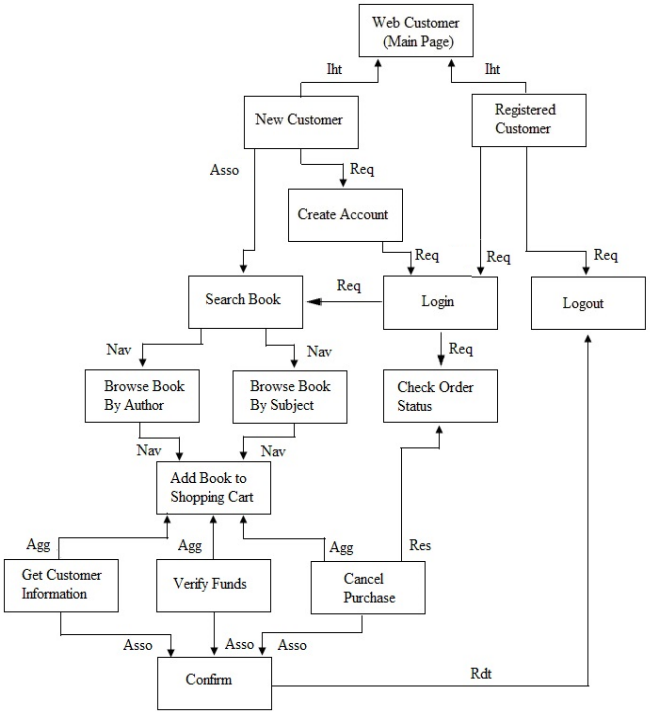

Fig. 4 ORD of Online Purchasing of a Book from Shipping Cart

To illustrate ORD, consider a Web application on Online Purchasing of a Book from Shipping Cart as shown in fig 4. Simplified diagram of objects and their relationships are depicted in the fig. 4. In figure, *Web customer* (main page) client-page consists of two inherited pages, *New customer* client-page and *Registered customer* client-page respectively. *New customer* client-page can associate *Search book* client-page and can make a request for *Create account* Client-page. In the similar way *Registered customer* client-page can make a request for *Login* and *Logout* client-page. After logging in *Registered customer* can also make request for *Search book* client-page. It can also make request for *Check order status* page and if book is not required, receive response of cancellation. Through *Search book* client-page one can navigate to *Browse book by Author*, *Browse book by subject* and *Add book to shopping cart* pages. *Add book to shopping cart* page is an aggregation of *Get customer information, Verify funds* and *Cancel purchase pages*. These three pages can again make an association for *Confirm* client-page. After confirmation request is redirected to *Logout page* and back to *Web customer* page.

ORD depicts structural and dependent relationships of objects in a Web-based application. Entities of a Web-based application are treated as objects in the ORD. Each object in ORD can have its own supporting test methods to assist Web application testing. It also helps to obtain a cost effective testing strategy for regression testing**.**

B. The Behaviour Perspective

The dynamic behaviour of a Web-based application can be modelled from two aspects:

- Navigation behaviour between Web pages
- State-dependent behaviour of interacting objects.

1) Page Navigation Diagram

Through hyperlinks, Web application introduces new navigational behaviour among the Web pages. However, an unreachable page causes a bad impression among users. Most of the tools allow checking the accessibility of the hyperlinks, but they do not provide a way to analyse the dynamic navigation behaviour. To check navigation behaviour, a Page Navigation Diagram (PND) is employed. The PND is a Finite State Machine (FSM) and its each state represents a client page. The hyperlink is represented by transition between the states and is labelled by the URL of the hyperlink. The PND of a Web-based application can be constructed from an ORD. The algorithm is as follows:

1. Initialize S_0 as initial state, where S_0 represents home page
2. Create a set S of reachable states and initially set S = $\{S_0\}$
3. Select any state S_i from set of reachable states *S* and identify the transition for S_i.
   - For each state S_i there exists a transition if there is a hyperlink relationship from S_i to a client page S_j.
   - For each state S_i there exists a transition if there is a request relationship from S_i to a server page S_p and, from S_p, there is a response or redirect relationship that led to a client page S_j.
   - Connect S_j from S, for the known transition.
4. If S_j, has not occurred before, add S_j to *S*.
5. Remove S_i**,** from *S* and repeat the process from step 3 until S is empty.

It may possible that the same navigation hyperlink may lead to different client pages. In order to model this dynamic situation, a *flag condition*, enclosed in brackets, is required on the transition in the PND. In order to fire the transition, the flag condition that specifies the conditions of

the submitted data or internal system states must be true. Figure 5 shows Page Navigation Diagram.

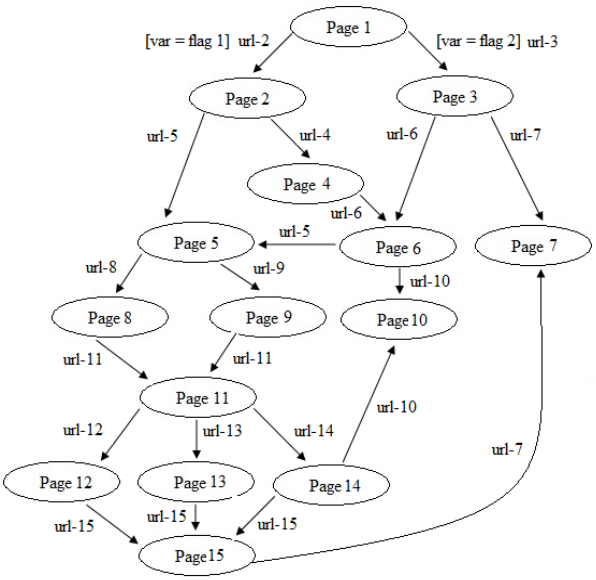

Fig. 5 PND of Online Purchasing of a Book from Shipping Cart

To test navigational error, a navigation test is employed. This navigational test tree is a spanning tree constructed from PND. Figure 6 shows the navigational test tree for Online Purchasing of a Book from Shipping Cart. In figure nodes represents states and edges represents transitions in the PND.

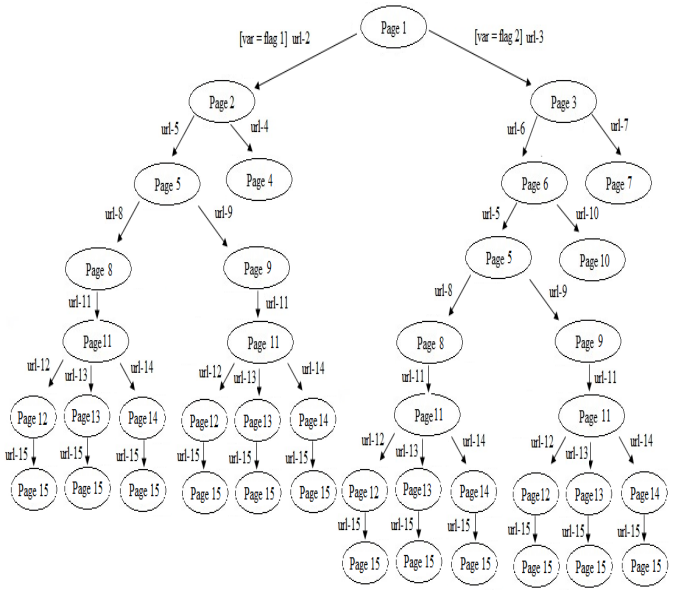

Fig. 6 Navigational Test Sequences from Page 1 to Page 15

The spanning tree is then used to generate the test cases, each of which is a path of edges starting from the root (initial state) to any particular node. Table IV shows some possible navigational test sequences starting from page 1 to page 15. Properties such as reachability and deadlock of the navigation behaviour can be checked using tree structure.

TABLE IV

NAVIGATIONAL TEST SEQUENCES STARTING FROM PAGE 1 TO PAGE 15

S.No.	Probable Test Case Sequences
1.	[var = flag 1] url-2, url-5, url-8, url-11, url-12, url-15
2.	[var = flag 1] url-2, url-5, url-8, url-11, url-13, url-15
3.	[var = flag 1] url-2, url-5, url-8, url-11, url-14, url-15
4.	[var = flag 1] url-2, url-5, url-9, url-11, url-12, url-15
5.	[var = flag 1] url-2, url-5, url-8, url-11, url-13, url-15
6.	[var = flag 1] url-2, url-5, url-8, url-11, url-14, url-15
7.	[var = flag 2] url-3, url-6, url-5, url-8, url-11, url-12, url-15
8.	[var = flag 2] url-3, url-6, url-5, url-8, url-11, url-13, url-15
9.	[var = flag 2] url-3, url-6, url-5, url-8, url-11, url-14, url-15
10.	[var = flag 2] url-3, url-6, url-5, url-9, url-11, url-12, url-15
11.	[var = flag 2] url-3, url-6, url-5, url-9, url-11, url-13, url-15
12.	[var = flag 2] url-3, url-6, url-5, url-9, url-11, url-14, url-15

2) Object State Diagram

Object State Diagram (OSD) describes behaviour of interacting objects and is similar to State chart that is a set of hierarchical, concurrent, and communicating state machines. OSD represents state-dependent behaviour of an object in a Web-based application. Figure 7 shows login and logout session of a customer in online purchasing of a book from shopping cart. Whereas fig. 8 shows addition and deletion of a book from cart after checking its cardinality.

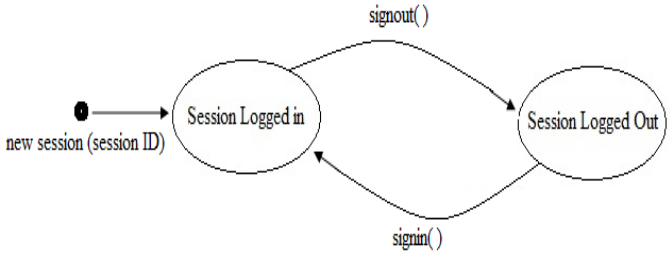

Fig. 7 State Chart for Logging in and Logging out

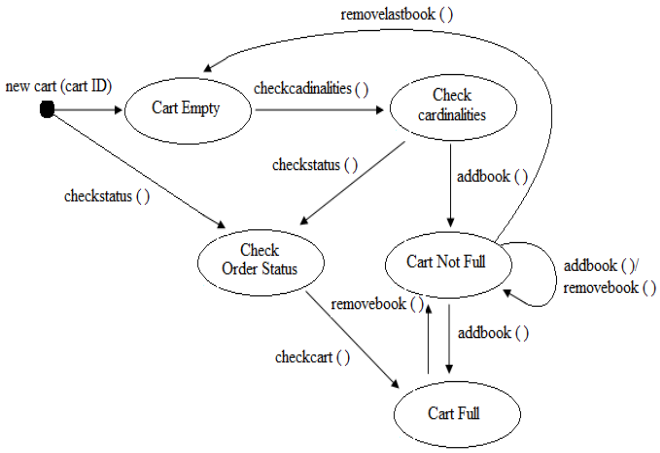

Fig. 8 State Chart for Adding/ Removing Book from Cart

Figure 9 shows the OSDs of the major interacting objects for online purchasing of a book from shipping cart. The state-dependent behaviour can be accessed by integrating various objects of the online shopping cart system.

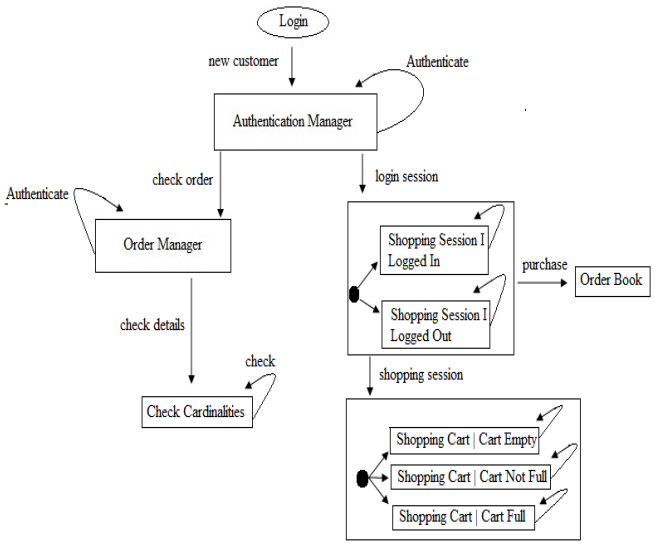

Fig. 9 OSD for a New Customer

The errors related to the object's state behaviours can be achieved by constructing a test tree as shown in fig. 9. The nodes of the test tree represent the states of the OSD. The edges of the tree represent the transitions between the states. The test cases are then the sequences of the transitions in each path of the tree that starts from the root and ends at any node. Table V enlists various test cases:

TABLE V

TEST CASES FOR ONLINE SHOPPING CART OF A NEW CUSTOMER

S.No.	Test Cases
1 1.1 1.1.1	newCustomer: AuthenticationManager newSession: Shoping Session newBook: OrderBook
2 2.1 2.1.1 2.1.2	newCustomer: AuthenticationManager newSession: ShoppingSession newBook: OrderBook newCart: ShoppingCart
3 3.1 3.1.1 3.1.2 3.2 3.2.1	newCustomer: AuthenticationManager newSession: ShoppingSession newBook: OrderBook newCart: ShoppingCart billOrder: OrderManager deductAmt: CheckCardinalities

C. The Structure Perspective

The structure perspective access both control flow and data flow information of a Web-based application. Existing traditional testing techniques for control and data flow are still useful to evaluate reliability of Web applications. To capture control flow and data flow information, the Block Branch Diagram (BBD) and Function Cluster Diagrams (FCD) are used. The BBD is similar to a control flow graph and is constructed for each individual function of a Web application to describe the control and data flow information.

Where as the FCD is a set of function clusters within an object. Each function cluster is a tree T = (V, *E),* where V represents set of nodes for the individual functions and E represents set of edges or calling relations between the nodes.

Due to inter-procedural calls test cases can be derived to cover the test paths involving more than one function. Furthermore, in Web-based applications script functions of the client page are invoked in an arbitrary fashion depending upon the user interaction. To capture function invocation sequences single-entry point and single-exit point is used.

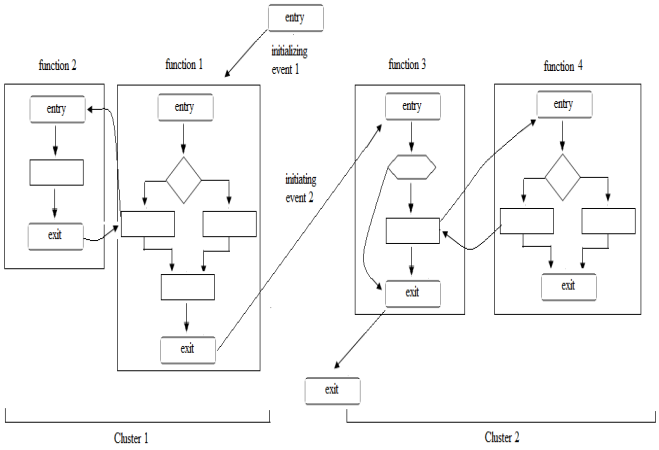

Fig. 10 Sample FCD for an Object

In fig. 10 there are two function clusters, Cluster 1 and Cluster2. Each function in the cluster represents calling relationships between the functions. This is further expanded by BBD which shows internal control structure. The events initiated shows data flow information for the function invoking sequence.

CONCLUSION

The paper presents a hierarchical architecture as new approach for developing Web applications. The proposed architecture is suitable for almost all Web applications (Small, Medium and Large scale). Designing the Web pages in a hierarchical manner facilitates developers to easily implement security compliances, adding new modules, CSS and include files with low integration cost. Reusability of code is possible. Hierarchical architecture allows all Object-oriented testing methodologies be applied to the framework thus improving quality of Web application. Kung's Object-oriented testing methodology is applied on the hierarchical framework to derive various test cases.

ACKNOWLEDGMENT

The authors are thankful to Prof. Aditya Shastri, Vice Chancellor, Banasthali University, Banasthali, Niwai (Rajasthan), India, for providing the necessary facilities for the preparation of the paper.

REFERENCES

[1] P. Fraternali, "Tools and Approaches for Developing Data-intensive Web Applications: A Survey," *ACM Computing Surveys*, vol. 31, pp. 227-263, 1999.

[2] N. M. A. Munassar, A. Govardhan, "Comparison between Traditional Approach and Object-oriented Approach in Software Engineering Development", *International Journal of Advanced Computer Science and Applications (IJACSA),* vol. 2, no. 6, pp. 70-76, 2011.

[3] N. Sivakumar, K. Vivekanandan, "Comparing the Testing Approaches of Traditional, Object-oriented and Agent- Oriented Software System." *International Journal of Computer Science & Engineering Technology (IJCSET)*, vol. 3, no. 10, pp. 498-504, October 2012.

[4] G. A. Di. Lucca, A. R. Fasolino, F. Faralli, U. DeCarlini, "Testing Web Applications," *In Proceedings of the IEEE International Conference on Software Maintenance*, pp. 310-319, Montréal, QC, Canada, October 2002.

[5] H. W. Gellersen, R. Wicke, M. Gaedke, "WebComposition: An Object-oriented Support System for the Web Engineering Lifecycle," *Computer Networks and ISDN Systems*, vol. 29, pp. 1429-1437, 1997.

[6] D. Kung, C. H. Liu, P. Hsia, "An Object-oriented Web Test Model for Testing Web Applications," *In Proceeding of IEEE 24th Annual International Computer Software and Application Conference (COMPSAC2000)*, Taipei, Taiwan, October, 2000.

[7] C. Liu, D. C. Kung, P. Hsia, C. Hsu, "Object-Based Data Flow Testing of Web Applications," *In Proceedings of the First Asia-Pacific Conference on Quality Software*, IEEE Computer Society Press, Los Alamitos (CA), pp. 7–16, October, 2000.

[8] G. Rossi, D. Schwabe, "Object-oriented Design Structures in Web Application Models," *Annals of Software Engineering*, vol. 13, no. 1-4, pp. 97-110, June 2002.

[9] F. Ricca, P. Tonella, "Building Quality into Web Applications: Analysis and Testing of Web Applications," *IEEE 0-7695-1050-7/01*, 2001.

[10] M. R. Girgis, T. M. Mahmoud, B. A. Abdullatif, A. M. Zaki, "An Automated Web Application Testing System," *International Journal of Computer Applications,* vol. 99, no. 7, pp. 37–44, August 2014.

[11] S. Sampath, S. Sprenkle, E. Gibson, L. Pollock, S. Greenwald, "Applying Concept Analysis to User-Session-Based Testing of Web Applications," *IEEE Transactions on Software Engineering*, vol. 33, no. 10, pp. 643–658, 2007.

[12] A. A. Andrews, J. Offutt, R. T. Alexander, "Testing Web Applications by Modeling with FSMs," *Software Systems Modeling*, vol. 4, no. 3, pp. 326–345, 2005.

[13] N. Mansour, M. Houri, "Testing Web Applications," *Information and Software Technology*, vol. 48, no. 1, pp. 31–42, January, 2006.

[14] S. Artzi, M. Pistoia, "Practical Fault Localization for Dynamic Web Applications," *Proceedings of the 2nd ACM/IEEE International Conference on Software Engineering*, vol. 1, pp. 265-274, May, 2010.

[15] J-T. Yang, J-L. Huang, F-J. Wang, C. C. William, "Constructing an Object-oriented Architecture for Web Application Testing", *Journal of Information Science and Engineering*, vol. 18, no. 1, pp. 59–84, January, 2002.

[16] A. Marchetto, F. Ricca, P. Tonella, "A Case Study-Based Comparison of Web Testing Techniques Applied to AJAX Web Applications," *International Journal on Software Tools for Technology Transfer*, vol. 10, no. 6, pp. 477– 492, October, 2008.

[17] V. Garousi, A. Mesbah, A. Betin-Can, S. Mirshokraie, "A Systematic Mapping of Web Application Testing," *Elsevier Journal of Information and Software Technology*, vol. 55, pp. 1374–1396, 2013.

[18] Li, Yuan-Fang, Das, K. Paramjit, Dowe, L. David, "Two Decades of Web Application Testing- A Survey," *Recent Advances. Information Systems*, vol. 43, pp. 20–54. 2014.

[19] S. Doğan, A. Betin-Can, V. Garousi, "Web Application Testing: A Systematic Literature Review," *Published in Journal of systems and Software*, vol. 91, pp. 174–201, May, 2014 Elsevier Science Inc. New York, NY, USA.

[20] Welling, Luke, Thomson, Laura, PHP and MySQL Web Development, SAMS Publishing, 2001.

[21] Achour, Mehdi, Friedhelm, Betz. PHP Manual, PHP Documentation 00Group, 2008-11-21, <http://www.php.net/docs.php>.