WEB APPLICATION'S RELIABILITY IMPROVEMENT THROUGH ARCHITECTURAL PATTERNS

Md Umar Khan[1] and Dr T.V. Rao[2]

[1] Assoc. Professor, Prakasam Engineering College, Kandukur, , AP, India

[2] Professor& HOD in CSED, PVP Siddartha Institute of Technology, Kanuru, Vijayawada, AP, India

ABSTRACT

Scalability and availability are two highly desirable attributes pertaining to reliability of a web application that renders state-of-the-art services to online users. In simple words scalability is the ability to grow, the ability to serve increased number of requests or clients. Building a scalable application with round the clock availability is a challenging problem in the light of ever increasing population of potential users. Dramatic increase in users to web application causes bursts of requests that put the application to acid test. On the other hand web application availability represents the degree of operational continuity. High availability and unlimited scalability are the two indispensable quality attributes a web application in the real word. These features bestow rich user experience as far as operational continuity and ability to handle growing workload are concerned. By taking server side measures it is possible to achieve these two desirable features. However, there is possibility to have architectural pattern along with underlying design patterns to promote these quality features of web application. In this paper we enhance our architectural pattern eXtensible Web Application Development Framework (XWADF) that can leverage the quality of web application design as it result in highly scalable with high availability. As the application is designed in XWADF framework it promotes scalability assuming server side supports in terms of resources. The application also ensures availability as the design supports maintenance without letting the application down. The empirical results revealed that our architectural approach increases reliability of web applications significantly in terms of availability and scalability

KEYWORDS

Web application development, scalability, availability, reliability

1. INTRODUCTION

In simple words, scalability is the ability to grow, the ability to serve increased number of requests or clients. In the context of web application design, it is essential to keep reliability [21] in mind. In this paper we focus on two reliability attributes such as scalability and availability. These two are highly desirable features that improve the quality and performance of web applications. According to Sanderson [8] a web application is scalable when each user gets same quality irrespective of the number of users concurrently accessing the application. Poor usability[22] and poor scalability results in web applications when designers are not aware of web design patterns [10]. Web frameworks were also written without considering scalability usability and simplicity [11]. Enterprise web applications that generate content dynamically and data grids throw scalability challenges [12]. There are many good reasons to use design patterns as they can get rid of reinventing the wheel by promoting reuse. Moreover they are proven, and expressive.

Model View Controller (MVC)[23] is one of the well known architecture which renders many advantages to web applications including maintainability[24], availability and scalability [14]. The modern web applications use three tier or n-tier architecture.

The entire flow together actually forms a functional web applications. However, in this paper, our focus is to promote scalability and availability through best practices or design patterns that are the underlying proven blueprints within our architectural pattern XWADF[25]

In this paper our contributions are twofold. They are identifying design patterns that can improve scalability and availability of web application and implementing them in our architectural pattern[26]. They are as follows

- Enhancing the architectural pattern XWADF proposed by us earlier. In this paper we improved the architecture by incorporating the design patterns identified for promoting scalability and availability of web applications into the architectural pattern.

- Building two case study applications that demonstrate the usefulness of our enhanced architecture as far as scalability and availability are concerned.

- In addition to these, we used reliability metrics to test the case study applications and provide reliability improvements achieved through the use of our architectural pattern XWADF

This paper remaining part is as follows. Section II gives the literature pertaining to improving web application scalability and availability in terms of architectural and design patterns usage. Section III presents XWADF and its limitations. Section IV presents the enhanced XWADF architectural with identified design patterns incorporated. Section V illustrates case study applications that make use of our enhanced XWADF. Section VI evaluates the reliability attributes such as scalability and availability using appropriate metrics. Section VII concludes about this paper and provides future work directions.

2. RELATED WORKS

Because of the rapid growth of different users of web applications, scalability and availability has become cornerstone of web application design. Zhao and Schulzrinne [1] built an architecture that promotes dynamic scalability of web applications. Their architecture was named DotSlash [2], [3] that has been improved further [1] to increase scalability. The design goals of the system are scalability, transparency[27] and self configuration. They employed query result caching [4] that could reduce extensive computations. The solution was made in distributed environment. Their architecture has a provision for rescue server which is made available to client when origin server is unable to scale. A distributed query result cache is designed and implemented in both the servers which will help to process client requests faster. The system is capable of making rescue servers on demand so as to make it highly scalable to bursts of client requests. The DotSlash has configuration of cache where the cache can be activated or deactivated as situation demands.

. Cache[28] on and cache off are used to activate and deactivate caching. The system has three states namely normal state, SOS state and rescue state. There are three load regions known as light load, desired load and heavy load based on which the system will operate in different states to cope with burst of requests. The DotSlash architecture is somewhat similar to our architectural

pattern as far as caching is concerned. However, ours is the approach at application level which promotes scalability. Our caching mechanism is described in Section III.

Wei [5] focused on scalability of web application with respect to database management. Their research was on complex query support, storing huge amount of data, strong data consistency, scalability and elasticity[29] and fault tolerance. Unlike our framework explored in section IV, this work has thrown light into backend side improvements for promoting scalability. Manjhi etal. [6] studied the tradeoff between security and scalability of web applications. They employed a third party product known as Database Scalability Service Provider (DSSP)[30] for designing a scalable architecture for data-intensive web applications. Their architecture focuses on providing security and scalability simultaneously. As there is relation between scalability and security these researchers had an empirical study to ascertain facts pertaining to scalability-security tradeoffs
This is achieved by DSPP by maintaining cache. It takes care of data invalidations based on the information exposed to it. The scalability tradeoff is achieved by knowing statically which data can be encrypted[31] without compromising scalability. This will help administrators to get relieved from the burden of managing tradeoff. The work in [6] is pertaining to scalability. This work is similar to our work in the enhanced XWADF as far as caching is concerned with the intention of promoting scalability. However, our work does not focus on security issues explicitly.

Tripp, Pistoia and Cousot [7] employed ANDROMEDA tool for security analysis of web applications in scalable fashion. The tool's main focus is on monitoring integrity violations such as log forging, SQL injection, and cross side scripting (XSS)[32]. It focuses on scalable security analysis rather than promoting scalability of web applications. Sanderson [8] presents Google App Engine and its infrastructure for building scalable web applications. However, in this paper our focus is on building web applications using original J2EE[33] technologies such as Servlets and JSP along with design patterns. Garrod and Manjhi [9] presented a proxy query result cache named "Ferdinand".

As far as caching is concerned the Ferdinand also makes use of it for improving scalability. However, it uses it both at local level and distributed level making it more scalable. Plattner and Alonso [12] introduced a middleware named "Ganymed" which takes care of database replication for improving scalability of web applications. Oracle [13] presents architecture of MySQL database server which has out of the box replication features that ensures high availability and scalability of web applications.

It is evident that MYSQL[34] supports a master and multiple slaves. Master holds original copy and replicas are held in slaves. All read operations are performed in slaves while writes are made in master and then replicated. This will improve scalability while achieving high availability of database server which will enhance the overall reliability of web applications. We use this feature in case study applications. However, it does not affect our XWADF architectural pattern. However it provides said benefits purely from service improvements from backend.

J2EE technologies can be used to leverage scalability and availability of web applications. Kwon and Bang [14] proposed web application design method which demonstrate the effective usage of Servlets and JSP pages with proper design so as to avoid availability and scalability problems in web applications. Especially the design of SQLServlet for all SQL operations andForwardServlet for forwarding requests to corresponding JSPs can ensure the web application scalability and availability to some extent. In our XWADF framework we

adapted ForwardServlet as an underlying design pattern while indirectly used the SQL Servlet feature.

Srikanth and Savithri [15] studied GOF[35] design patterns for improving quality of web applications. Especially they focused on three design patterns namely Flyweight, Composite and AbstractFactory for improving quality in terms of expandability[36], understandability[37] and reusability[38] respectively.

3. XWADF AS AN ARCHITECTURAL PATTERN

In our previous work we proposed an architectural pattern with many underlying design patterns for improving quality of web applications. XWADF is the name of the framework that leverages the advantages of design patterns at design level. Detailed information about our architecture can be found in [16].

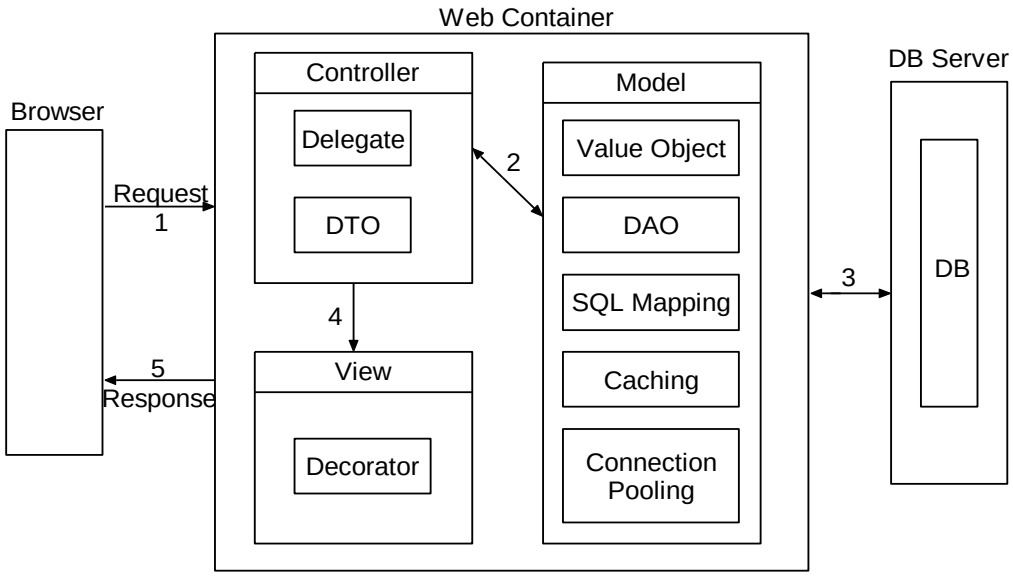

Figure 1 – Overview of XWADF

When we see in Figure 1, it is shown that the architectural pattern is based on MVC pattern which is at base level and renders layered services to web application design. The design patterns shown in the architectural pattern spread across all the three layers of the design to render quality services. Our paper in [16] also has road map (refactoring algorithm) for converting normal web applications into XWADF compatible ones. In this paper we enhanced the XWADF architectural pattern to focus on scalability and availability.

4.DESIGN PATTERNS IDENTIFIED FOR IMPROVING SCALABILITY AND AVAILABILITY

There are different design patterns that are identified for improving scalability and availability. They are Flyweight, Mediator, Factory Method, and Singleton. They are

1. Flyweight

When there is almost simlar nature of creating high number of objects, Flyweight is used. When there is high number of objects then it consumes high memory. So it requires flyweight design pattern which reduces the load on memory by sharing objects.
It is achieved by using two types of properties intrinsic and extrinsic.

2. Mediator

Mediator design pattern is a behavioral design pattern which is widely used. Mediator introduces a layer in between two objects so that the interaction between that two objects taken place.The intermediate mediator object communicate between the two objects, and also helps in interaction for a set of different objects.

3. Factory Method

One of the creational pattern is a factory method. It is used to instantiate an object from a set of classes. The factory method will have the super class. So that, you can program for the interface and not for the implementation.

4. Singleton

There are only two points in the definition of a singleton design pattern,

1. Only one instance of a class is allowed.
2. That single instance accessed globally.

5.ENHANCED XWADF WITH PROPOSED DESIGN PATTERNS FOR SCALABILITY AND AVAILABILITY

Design patterns which are already part of the XWADF contribute to performance of web application. When web application performance improves, it adds to the scalability of the application at design level. It is to be notices that our research is not into server side measures to make web applications scalable. However, we throw light on the design of web applications for improving performance using an architectural pattern with many underlying design patterns. Our previous paper was particular about improving latency and throughput of web applications for improving performance of web applications. However, the latency and throughput attributes can be used to measure scalability of web applications. This is presented in section V.

The design patterns identified to enhance web application's scalability and availability include Flyweight, Mediator, Factory, and Singleton. Influenced by the functionality of these patterns we proposed the two design patterns that contain the required functionality to improve reliability of web applications. The patterns we proposed include Object State Pattern and Interaction Pattern which will provide seamless interaction between objects involved in the request processing of a web application.

Incorporating Design Patterns into XWADF (Enhanced XWADF)

The proposed patterns Object State Pattern and Interaction Pattern, now we are embedding them into our architectural pattern by enhancing XWADF. The improved architectural pattern is as shown in Figure 2.

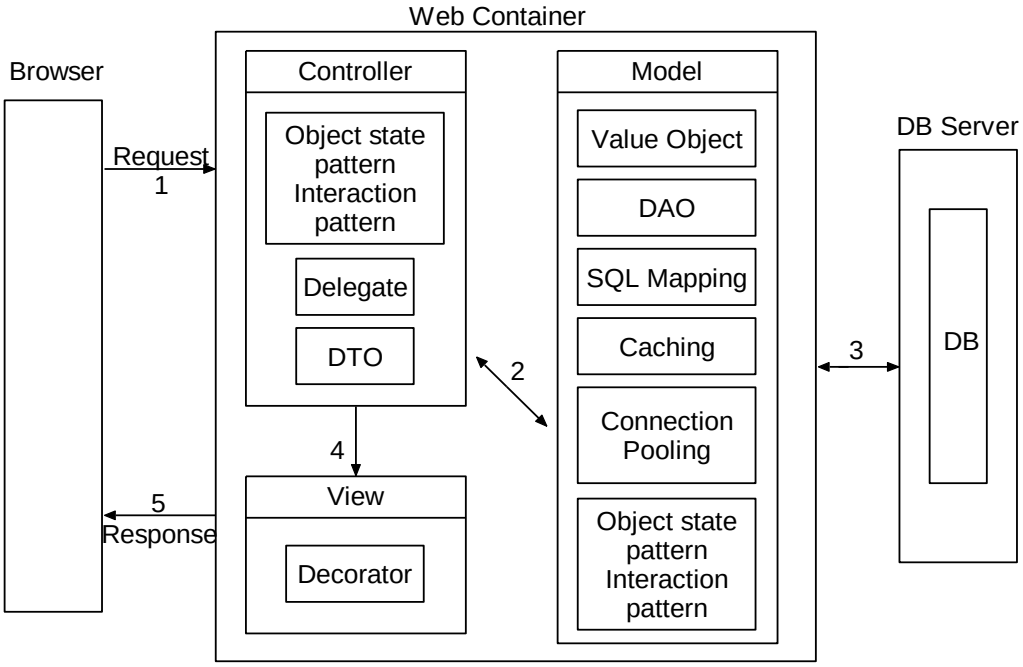

Figure 2– Enhanced overview of XWADF

As can be seen in Figure 2, it is evident that more design patterns are included into the architecture so as to make the application highly scalable with high availability. Interestingly these patterns can be used in either model layer or controller layer based on the complexity that arises in those layers in case of complex enterprise web applications.

Working steps of enhanced XWADF

1. start
2. Send request to controller
3. The controller delegates to model
4. The controller transfer data in the form of object to model
5. The controller use Object State Pattern & Interaction Pattern for fast communication
6. The model interact with database
7. The model send data in the form of object to database
8. The model access result from database in the form of object(DAO)
9. More number of manipulations carried at model by taking data from database(SQL Mapping)
10. Caching (local memory) is used to hold data objects which are frequently accessed from database
11. Efficient access of data using connection pooling

12. More efficient and fast communication using Object State Pattern and Interaction Pattern
13. The model send results to controller
14. The controller send results to view
15. The view use Decorator Pattern to display and send as response to browser.

6. CASE STUDY AND EXPPERIMENTAL EVALUATION

Experimental Setup

The environment used for experiments include a PC with 4 GB RAM, Core 2 dual processor running Windows 7 operating system. Java/JEE platform is used for building case study applications. The scalability of applications is tested using two metrics namely latency and throughput. The testing is done using a testing tool known as LoadUIWeb 2.

Case Study Applications

Two existing web applications [16] are used for experiments. Hospital Management System (HMS) from health care domain and Library Management System (LMS) from education domain were considered to adapt enhanced XWADF. The applications prior to adaption of the proposed architectural pattern and after are subjected to automated testing in terms of CRUD [18] operations. The database used is MYSQL with replication support. The HMS database has one lack records while the LMS has 50000 entities.

Scalability Evaluation

As explored in [17] latency and through metrics can be used to measure the scalability of web applications. Latency refers to the time taken in seconds to obtain response while the throughput denotes the number of responses per second.

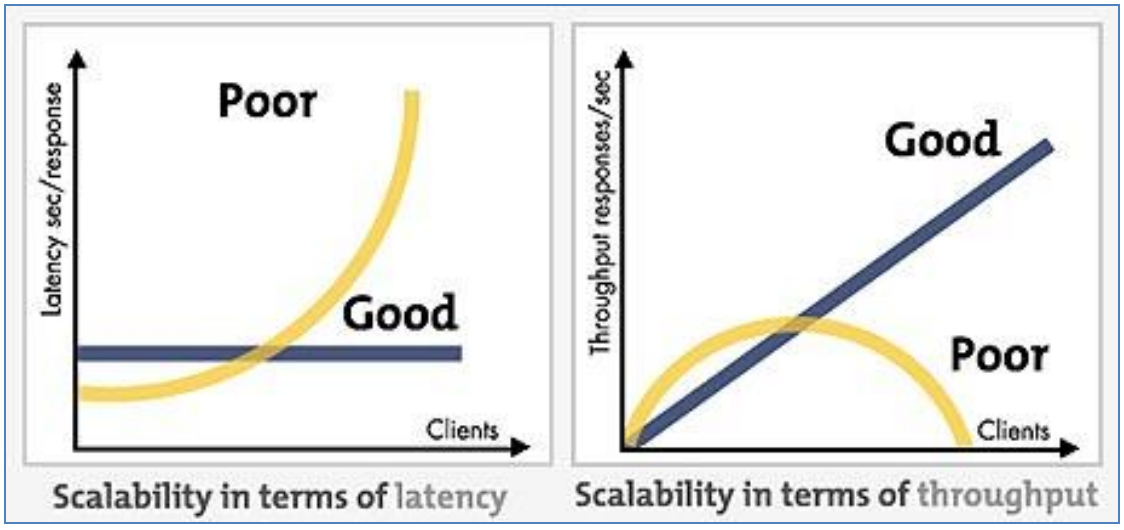

Figure 3 – Latency and scalability to measure scalability

Table 1 : Experimental results for Latency of HMS to measure scalability

No. of users	Response Time in seconds per user	
	Without XWADF	With Enhanced XWADF
1	74.8	50.33
2	76.9	56.66
5	84.62	58.53
10	95.32	62.32
15	107.23	66.46
20	118.32	70.22
25	130.42	80.43
30	142.36	82.56
35	155.42	90.43
40	172.67	92.56
45	184.32	96.76
50	196.42	98.56

Latency Results for HMS (Hospital Management System) are shown in the Table 1. Latency (response time) is a measure that tells how long user waits to get response to a query. The response time in seconds for without XWADF and with Enhanced XWADF is tabulated. The results reveal that there is considerable improvement in response time when the application with Enhanced XWADF is used for experiments.

Table 2 : Experimental results for Latency of LMS to measure scalability

No. of users	Response Time in seconds per user	
	Without XWADF	With Enhanced XWADF
1	20.33	15.45
5	24.2	17.06
10	32.92	20.33
15	44.92	22.32
20	54.11	28.65
25	65.21	32.69
30	78.22	35.62
35	90.15	42.36
40	112.35	55.65
45	125.11	62.32
50	134.51	68.36

Latency Results for LMS (Library Management System) is shown in Table 2. Latency (response time) is a measure that tells how long user waits to get response to a query.

The response time in seconds for without XWADF and with Enhanced XWADF is tabulated. The results reveal that there is considerable improvement in response time when the application with Enhanced XWADF is used for experiments.

Table 3 : Experimental results for Throughput of HMS to measure scalability.

Time in minutes	**Throughput in KB/sec per user**	
	Without XWADF	**With Enhanced XWADF**
1	5.01336	7.45082
2	4.87646	6.61843
5	4.43157	6.40697
10	3.93411	6.01733
15	3.49715	5.64249
20	3.16937	5.34036
25	2.87532	4.66244
30	2.63416	4.54215
35	2.41281	4.14685
40	2.17177	4.05143
45	2.03450	3.87557
50	1.90917	3.80479

Throughput results for HMS (Hospital Management System) are shown in Table 3. Throughput is the amount of work accomplished for a given unit time. Throughput without XWADF and with Enhanced XWADF is tabulated. The results reveal that there is considerable improvement in throughput when the application with Enhanced XWADF is used for experiments.

Table 4 : Experimental results for Throughput of LMS (Library Management System)

Time in minutes	**Throughput in KB / sec per user**	
	Without XWADF	**With Enhanced XWADF**
1	5.90260	7.76699
5	4.95867	7.03400
10	3.64520	5.90261
15	2.67141	5.37634
20	2.21770	4.18848
25	1.84020	3.67085

30	1.53413	3.36889
35	1.33111	2.83286
40	1.06809	2.15633
45	0.95915	1.92555
50	0.89212	1.75541

Throughput results for LMS (Library Management System) are shown in Table 4. Throughput is the amount of work accomplished for a given unit time. The throughput without XWADF and with Enhanced XWADF is tabulated. The results reveal that there is considerable improvement in throughput when the application with the proposed architectural pattern is used for experiments.

The experimental results are visualized through a series of graphs as presented below.

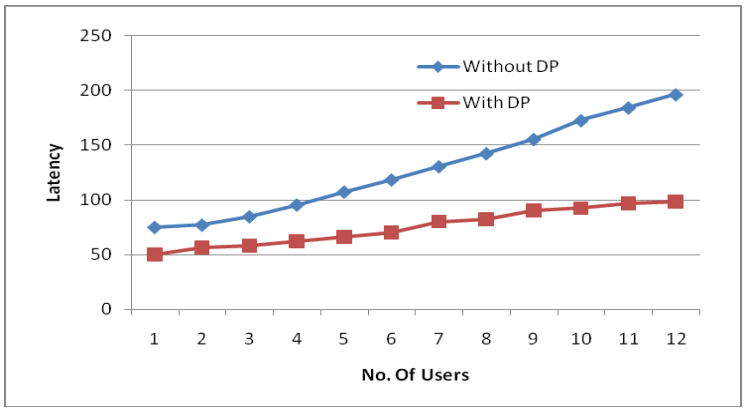

Figure 4– Scalability in terms of latency for HMS

From the above fgure, without using design patterns,when the number of users increases the latency is also increases for HMS. By using design patterns when the number of users increases the latency decreases for HMS.

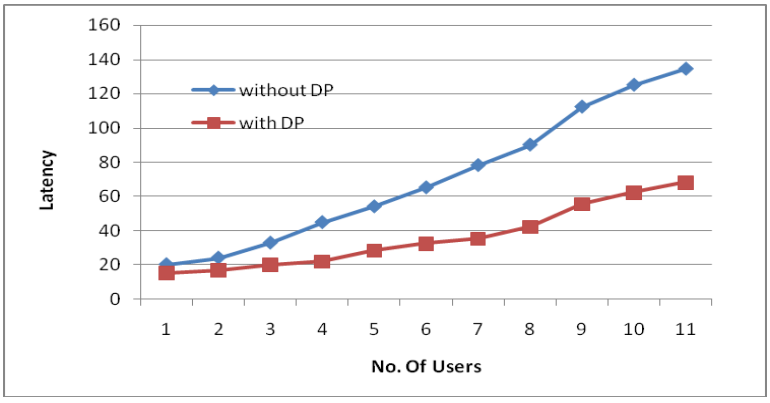

Figure 5– Scalability in terms of latency for LMS

From the above fgure, without using design patterns,when the number of users increases the latency is also increases for LMS. By using design patterns when the number of users increases the latency decreases for LMS.

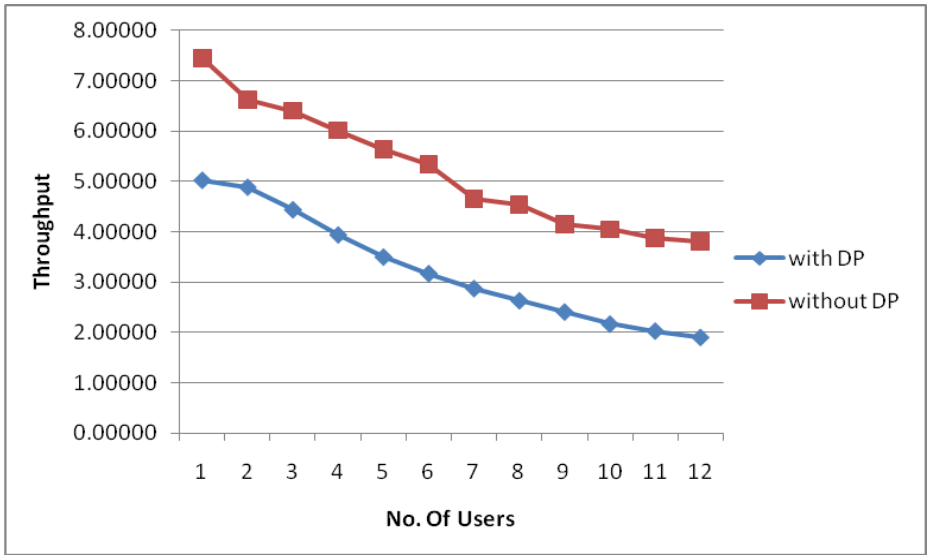

Figure 4– Scalability in terms of throughput for HMS

From the above fgure, without using design patterns,when the number of users increases the throughput is decreases for HMS. By using design patterns when the number of users increases the throughput increases.

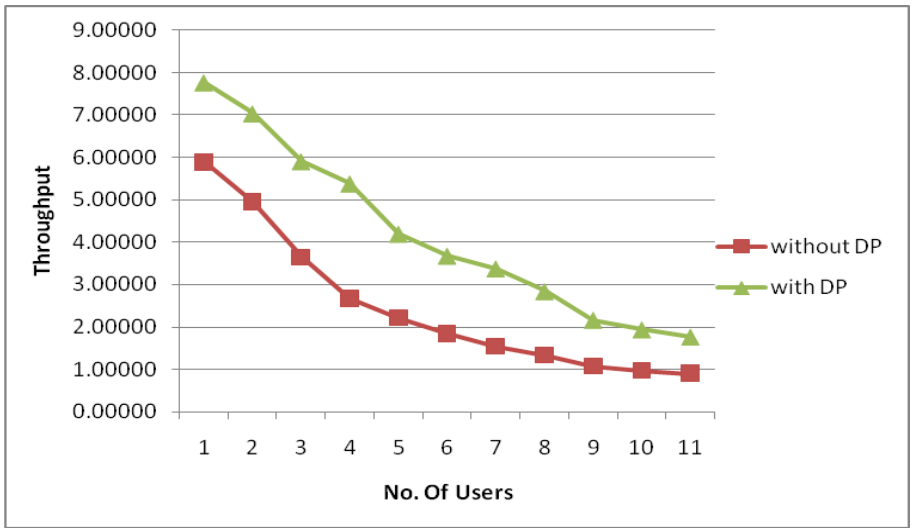

Figure 6– Scalability in terms of throughput for LMS

From the above fgure, without using design patterns,when the number of users increases the throughput is decreases for LMS. By using design patterns when the number of users increases the throughput increases.

Availability Evaluation

Availability is a feature of web application which represents operations continuity. The measurement for availability is the actual quantification of failure recovery time [19]. In other words, it is technically known as mean time to recovery (MTTR) [20]. The MTTR is also known as MTBI (mean time between failures and interruptions) [20]. A web application that fails once in every twenty minutes and exhibits one minute recovery time is considered to have 95% availability [19].

AVAILABILITY MEASURE	DOWNTIME PER YEAR	DOWNTIME PER WEEK
98%	7.3 days	202.15 minutes
99%	87.6 hours	101.08 minutes
99.5%	43.8 hours	50.54 minutes
99.8%	1,052 minutes	20.22 minutes
99.9%	526 minutes	10.11 minutes
99.95%	4.38 hours	5.05 minutes
99.99%	53 minutes	1.01 minutes
99.999%	5 minutes	6.00 seconds

Figure 7– Outages causing web application down time

As shown in Figure 6, it is evident that availability measures are established based on the downtime per year and downtime per week. Even when a web application causes 6 seconds downtime per week, it amounts to 5 minutes in a year. However, this measure is the highest availability that any web site can achieve for almost 100% availability.

This measure is applied to our case study applications. The downtime is recorded when they used XWADF and when they do not use it. The experiments revealed that XWADF makes the web applications not only scalable but also reduce downtime of applications thus promoting availability attribute. Both scalability and availability attributes reflect the reliability of web applications or performance to put in other words. We studied HMS and LMS applications for 24 hours and the results are scaled to 1 week and then the availability measure is applied.

TABLE 5- Availability of HMS

Without XWADF			**With XWADF**		
Downtime Per Year	**Downtime Per Week**	**Availability Measure**	**Downtime Per Day**	**Downtime Per Week**	**Availability Measure**
8.00 Days	300 minutes	90%	50.56 minutes	43.2 minutes	99.6%

As can be seen in Table 5, it is evident that the HMS with enhanced XWADF availability is more in the orders of magnitude when compared with the one without XWADF.

TABLE 6- Availability of LMS

Without XWADF			With XWADF		
Downtime Per Year	**Downtime Per Week**	**Availability Measure**	**Downtime Per Day**	**Downtime Per Week**	**Availability Measure**
7.6 Days	245 minutes	92%	99.93%	3.98 hours	4.02 minutes

As can be seen in Table 6, it is evident that the LMS with enhanced XWADF availability is more in the orders of magnitude when compared with the one without XWADF.

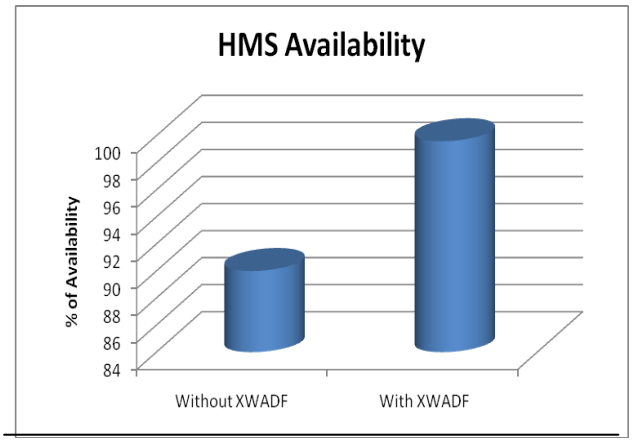

Figure 7 – Availability measure of HMS

As shown in Figure 7 our architectural approach to web application development could yield high availability of web applications. This is evident in case study applications such as HMS.

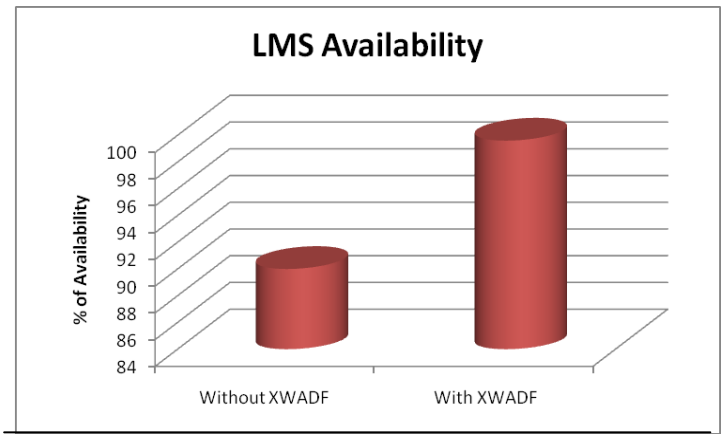

Figure 8 – Availability measure of LMS

As shown in Figure 8 our architectural approach to web application development could yield high availability of web applications. This is evident in case study applications such as LMS.

7. CONCLUSIONS AND FUTURE WORK

In this paper we identified design patterns that can improve scalability and availability of web applications. We enhanced our architectural pattern XWADF [16] to incorporate the identified design patterns. The enhanced architecture is applied to the existing case study applications such as LMS and HMS. Latency and throughput attributes are used to measure reliability in terms of scalability while MTTR metric is used to measure the availability of applications. The throughput and latency of applications before and after adapting our architectural pattern were tested with number of virtual users using the testing tool LoadUIWeb 2. The empirical results revealed that our architectural pattern can improve reliability of web applications and make them robust in terms of scalability and availability. Our future work is to enhance our architectural pattern XWADF which is based on MVC further to improve reliability in terms of fault tolerance and maintainability.

REFERENCES

[1] Weibin Zhao, Henning Schulzrinne. DotSlash: Providing Dynamic alability to Web Applications with On-demand Distributed Query Result Caching National Science Foundation , p1-13.

[2] W. Zhao and H. Schulzrinne. DotSlash: A selfcon _guring and scalable rescue system for handling web hotspots effectively. In International Workshop on Web Caching and Content Distribution (WCW), Beijing,China, October 2004.

[3] W. Zhao and H. Schulzrinne. DotSlash: Handling web hotspots at dynamic content web sites. In IEEE GlobalInternet Symposium, Miami, Florida, March 2005.

[4] S. Sivasubramanian, G. Pierre, M. van Steen, and G. Alonso. GlobeCBC: Content-blind result caching fordynamic web applications. Submitted for publication,Vrije Universiteit, June 2005.

[5] ZHOU WEI (2012). SCALABLE DATA MANAGEMENT FOR WEB APPLICATIONS. China: geboren te Zhejiang Province. p1-134.

[6] Amit Manjhi?, Anastassia Ailamaki?, Bruce M. Maggs?, Todd C. Mowry?†, Christopher Olston?, Anthony Tomasic?. (2006). Simultaneous Scalability and Security for DataIntensive Web Applications. ACM ,p1-12.

[7] Omer Tripp1, Marco Pistoia2, Patrick Cousot3, Radhia Cousot4, and Salvatore Guarnieri5. ANDROMEDA: Accurate and Scalable Security Analysis of Web Applications. p1-16.

[8] Dan Sanderson (2012). Building Scalable Web Applications with Google App Engine. 2nd ed. google press. p1-50.

[9] Charles Garrod,Amit Manjhi,Anastasia Ailamaki,Bruce Maggs,Todd Mowry,Christopher lston,Anthony Tomasic. (2008). Scalable Query Result Caching for Web Applications. ACM. ,p1-12.

[10] M. Taleb,A. Seffah,A. Abran. (2007). Patterns-Oriented Design Applied to Cross-Platform Web-based Interactive Systems. IEEE. p1-6.

[11] Adrian Chadd. Writing Scalable Web Applications, p1-22.

[12] Christian Plattner and Gustavo Alonso. Ganymed: Scalable Replication for Transactional Web Applications. p1-20.

[13] oracle (2011). MySQL Reference Architectures for Massively Scalable Web Infrastructure. oracle. p1-29.

[14] OhSoo Kwon and HyeJa Bang. (2011). Design Approaches of Web Application with Efficient Performance in JAVA. International Journal of Computer Science and Network Security. 11 (7), p1-7.

[15] Jeff Offutt. (Web Software Applications Quality Attributes. Management of the failure correction process. ,p1-11.

[16] Umar Khan1,T.V. Rao2. (2014). XWADF: Architectural Pattern for Improving Performance of Web Applications. p1-9.

[17] williamson. (2004). Buliding j2EE application for performance and scalablity. Available: http://www2.sys-con.com/itsg/virtualcd/java/archives/0604/davidson/index.html. Last accessed 11th april 2014.

[18] Akiva Leert,Pomegranate. The CRUD Methodology. p1-3.

[19] David M. Fishman. (2000). Application Availability: An Approach to Measurement. SUN MICROSYSTEMS INC,p1-19.

[20] Susan Stanley,. (2011). MTBF, MTTR, MTTF & FIT Explanation of Terms. IMC Network. ,p1-6.

[21] bobby chandra,tasuya koizumi. (2012). A study on Network Reliablity Evualtion for Developing Countries. *IEEE*.,p1-5.

[22] Muhammad Bilal Munir,Arif Mushtaq. A Framework for Extending Usability Engineering. p1-6.

[23] Hyun Jung La and Soo Dong Kim. (2010). Balanced MVC Architecture for Developing Service-based Mobile Applications. *IEEE*. ,p1-8.

[24] Hao Jianping,Yu Yongli,. (1998). Computer aidede Maintainblity in china. *IEEE*. ,p1-5002E

[25] Jo˜ao de Sousa Saraiva, Alberto Rodrigues da Silva. (2008). TheWebComfort Framework: an Extensible Platform for the Development of Web Applications. *IEEE*. 0 (0), p1-8.

[26] Minh Tu Ton That,Salah Sadou,Flavio Oquendo. (2012). Using Architectural Patterns to Define Architectural Decisions. *IEEE*. ,p1-5.

[27] Ilana Nisky,Ferdinando A.,Mussa-Ivaldi,Amir Karniel,. (2013). Analytical Study of Perceptual and Motor Transparency in Bilateral Teleoperation. *IEEE*. 43 (6), p1-13.

[28] Kashif Ali,Mokhtar Aboelaze,Suprakash Datta. (2006). Modified Hotspot Cache Architecture: A Low Energy Fast Cache for Embedded Processors. *IEEE*. ,p1-8.

[29] Pengfei Song, Matthew W. Urban, Armando Manduca, Heng Zhao, James F. Greenleaf, and Shigao Chen. (2012). Comb-push Ultrasound Shear Elastography (CUSE): A Novel and Fast Technique for Shear Elasticity Imaging. . p1-4.

[30] Amit Manjhi?, Anastassia Ailamaki Bruce M. Maggs. (2006). Simultaneous Scalability and Security for DataIntensive Web Applications. *ACM*. ,p1-12.

[31] Steven Zittrower and Cliff C. Zou. (2012). Encrypted Phrase Searching in the Cloud. *IEEE*. ,p1-7.

[32] Siddharth Tiwari, Richa Bansal, Divya Bansal. (2008). Optimized Client Side Solution for Cross Site Scripting. *IEEE*. ,p1-4.

[33] Jiang Guo, Yuehong Liao, Behzad Parviz. (2006). A Performance Validation Tool for J2EE Applications. *IEEE*. ,p1-10.

[34] Ivan Zoratti,,. (0). MYSQL Security Best Practices. *IET*,p1-15.

[35] Noura El Maghawry, Ahmed R.Dawood. ASPECT ORIENTED GoF DESIGN PATTERNS. ,p1-7.

Authors

Md.Umar Khan received his B.E degree in Civil Engineering from Madras University, Tamil Nadu, and India, his M.Tech degree in Computer Science from Jawaharlal Nehru Technological University, Hyderabad, and A.P. India. He is now pursuing his PhD degree at JNTU Ananthapur University, Andhra Pradesh. His research interests include Web Engineering, especially the Design Patterns. He is currently working as Associate Professor in the Department of Computer Science & Engineering of Prakasam Engineering College, Kandukur, Prakasam (district) A.P. India.

Dr. T.Venkateswara Rao received his B.E. Degree in Electronics and Communication Engineering from Andhra University, Visakapatnam, India and his M.E degree in Computer Science from University of Madras India. He received his Ph.D., degree in computer engineering from Wayne State University, Detroit, U.S.A. He is currently working as Professor and HOD in Computer Science and Engineering department at PVP Siddhartha Institute of Technology, Vijayawada A.P. India. Dr. T.V. Rao has published more than 25 papers in various national and international journals/conferences. His main research interests include multiprocessor systems.