Chapter 2

Chapter - 2. Literature Survey

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Name of the Sub-Title</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.</td>
<td>Quality and Performance Attributes</td>
<td>22</td>
</tr>
<tr>
<td>2.2.</td>
<td>Architectural Patterns and its Prior Works</td>
<td>24</td>
</tr>
<tr>
<td>2.3.</td>
<td>Design Patterns vs. Fault Tolerance of Web Applications</td>
<td>28</td>
</tr>
<tr>
<td>2.4.</td>
<td>Design Patterns vs. Scalability and Availability</td>
<td>34</td>
</tr>
<tr>
<td>2.5</td>
<td>Summary</td>
<td>40</td>
</tr>
</tbody>
</table>
CHAPTER 2

Literature Survey

This chapter reviews literature on prior works on improving performance of web applications. Especially it throws light into architectural and design patterns that help consistent design and implementation of web applications with high performance. David Parsons [20] studied evolving architectural patterns for web applications such as auto-completion architecture, delta management architecture, and client side buffering.

Kwon and Bang (2012) [21] proposed a design method for web application that improves scalability and flexibility. They implemented business logic in JSP file and reusable Servlet for SQL related search. They have also implemented a forward servlet which can prevent restarting web server and prevent maintenance problems. Mustafa Mamawala [22] studied on web application performance over a period of time.

They observed the deterioration of performance when it lacks performance tuning as part of the framework. QoS requirements such as security, manageability, scalability, and availability are to be considered in the design of web application for performance. Schwabe
et al. [23] studied on cohesive web application design by identifying patterns, and separating concerns.

They proposed a framework known as Object Oriented Hypermedia Design Model (OOHDM) for web application design which focuses on conceptual, navigational and interface design models. They worked on recommendation patterns, push communication for personalization. They used object oriented modeling principles in designing their architecture.

The conceptual model encapsulated classes. The navigational classes take care of dynamic navigations in web application. Interface objects are focused on abstract interface design activity while implementation integrates all these into a working solution. Carreras et al. (2010) [24] proposed bottom-up design patterns that are used for building energy web with combination of innovative communication and information technologies.

Its main focus is to have self-organizing components with continuous evaluation. Chang and Hon [25] proposed navigational structures for assessing the quality of web applications. They applied statistical testing theory to Totavia search engine web site which proved to be effective. Thung et al. [26] studied design patterns for improving performance of web applications.

They compared both navigational and architectural design patterns and proposed suitable design patterns for a school web site.
The architectural patterns they analyzed include MVC (Model View Controller), and presentation abstraction control (PAC). They also studied five navigational patterns such as set-based navigation, pagination, news, and navigation strategy and navigation observer.

The MVC pattern makes web application maintainable and scalable [27]. The PAC pattern makes it highly interactive and modular. Navigation observer records visited links to help navigation history. Thus it makes navigation easier, and decouples navigation and recording processes [28]. Navigation strategy computes link endpoints while navigating to help algorithms take best navigation paths.

It also reduces memory usage. News design pattern accommodates new information to be incorporated into web site with ease [29]. Pagination design pattern improves performance while viewing search results. Set based navigation groups related items for convenience and rich user experience [30].

Rossi et al. [31] proposed architecture for improving web information systems (WIS) with navigational patterns. The navigational patterns they studied for the purpose include node in context, active reference, shopping basket, landmark, news and set-based navigation. Landmark pattern helps users to have consistent cue about important interface.

Shopping basket pattern is similar to providing bookmarks or it is similar to shopping cart that is used in shopping malls in the real
Node in context pattern customizes object representation as per the context. Active reference pattern defines and maintains indexes in the form of active orientation tools for better navigation. Yi Li and Kevin Lü [32] proposed architecture for improving web database access.

They suggested connection pooling to improve performance of web application. Connection pooling provides a set of pre-established connections to application which eliminates scarcity of connections and promotes scalability of web applications. Lorna Uden [33] studied on design process for web applications. However, it is essential to capture right requirements for the success of web application besides using UI models like RMM (Relationship Management Methodology) and OOHDM.

Srikanth and Savithri [34] proposed a new architecture for improving quality of web applications using design patterns. Towards it, they combined many existing design patterns in an efficient manner. The set of patterns they combined along with MVC include template method, dependency injection pattern, inversion of control pattern, Database Access Objects (DAO) pattern, and aspect proxy pattern.

Dong et al. (2007) [35] studied the process of identification of design patterns in UML models. They developed a tool that will visualize design patterns existed in UML model of a web application.
An Liu et al. (2010) [36] observed that fault-tolerance is very important in web services composition. Web applications might interact with web applications in enterprise applications.

They proposed a fault tolerant framework for transactional web services. The framework includes high level strategies for exception handling and transaction techniques. The attributes of the framework include reusability, portability, convenience, and economy. Smith [37] emphasized the use of AJAX technology to be part of web application design to provide rich user experience.

Cane [38] studied the problem of measuring performance of web applications. His main focus was on Response Time as a measure. According to Garrido et al. [39] a layered approach using design patterns can improve the usability of web applications. Such layered approach also makes the applications maintainable. Taleb et al. [40] presented a patterns oriented design approach for web application development.

The categories of design patterns they focused include information architecture design patterns, page layout and user interface design patterns, navigation design patterns, interaction design patterns, information visualization design patterns, interoperability and system design patterns. They have identified that developers have no in-depth knowledge about the type of patterns to be used to solve various design problems.
Leff and Rayfield [41] proposed a flexible web application partitioning based on MVC architecture for improving performance. They also studied various deployment scenarios that can be supported without the need for changing source code. As per their design repartitioning web application also does not incur cost. Lucca et al. (2010) [42] proposed a mechanism to identify the design patterns implemented in a web application automatically.

They used reverse engineering techniques to discover patterns in web application. Their work enabled to know whether typical design patterns are used in application and take necessary decisions to redesign it. Tammet et al. [43] proposed a rule-based approach for systematic application development with web based interface. They separated user interface and business logic so as to make the application maintainable.

They implemented a rule server which manages business rules while the application’s BL layer adapts to the latest business rules without the need for modifying source code.

2.1 Quality and Performance Attributes

Jeff Offutt [44] studied on quality of web applications and observed that reliability, usability and security are the most important quality attributes for the success of web applications. Other quality attributes include availability, scalability, maintainability, and time-to-market.
Design patterns are proven industry best practices that improve the quality attributes of applications making them more maintainable [45]. Bernardi et al. (2010) [46] observed that pattern based development of web applications can improve quality attributes resulting in reusability and maintainability. Especially they proposed mechanisms for semi-automatic redesign to improve quality of existing web applications.

To redesign applications they used Java Server Faces (JSF) framework along with MVC architectural pattern. Ricca [47] studied web application development closely and opined that the web application quality is a multidimensional attribute which includes performance, accessibility, usability, maintainability, reliability, correctness and conformance to standards.

Yand and Chen [48] presented a framework by name ISPWAD which enables security of web applications besides improving performance. Design issues with regard to security are to be considered while developing web applications. The security issues to be considered include access control, SQL injection, Session hijacking, Information disclosure, Hidden field tampering.

Design issues with respect to performance include time taken for accessing web page, time taken for making DML (Data Manipulation Language) operations on database. Menasce and Almeida (2001) [49] define parameters for performance of web application. They include
response time, throughput, availability, and cost. Response time is the time taken to respond to queries.

Throughput is the rate of processing requests. Availability refers to the time in which services of web applications are available. Cost refers to some kind of performance such as price performance ratio. Besides these attributes some other attributes like security, reliability, scalability, extensibility can characterize Quality of Service (QoS) of a web application [49], [50], [51], [52].

2.2 Architectural Patterns and its Prior Works:

This section reviews literature on prior architectural patterns for improving quality and performance of web applications. Muñoz-Arteaga et al. [53] studied design patterns for interactive web application from the perspective of classification of security feedback. Their effort was to find equilibrium between security and usability of web application development.

YONG-JUN and LI KE-XI (2010) [54] proposed a framework for web application development which provides basic building blocks that make use of best practices such as design patterns to reuse software components. Their framework is known as JEMSF which is characterized by DTO (Data Transfer Object) for parameter passing, (DAOs)Database Access Objects, connection pool, and SQL processor.

All the patterns are used within the confines of MVC architectural pattern. The framework has three layers. The first layer
makes certain events and gets responses. The second layer is the controller that performs XML (2004) [55] interpretation, request processing and exception handling. The third layer is responsible to interact with relational database known as model. To process data from user and to the database we need this MVC architectural pattern.

It has provision for pooling objects for reuse and SQL processor component that will interact with database through database objects obtained from pool. Kerji (2011) [57] proposed the best use of decorator design pattern to adapt to changing requirements of end users who need different kinds of information in the form of XML. Their approach promoted reusability of code and besides reducing maintenance costs.

Tune et al. (2004) [56] proposed a simultaneous multithreading (SMTH) processor for increasing throughput of applications. Thus the SMTH can make use of maximum utilization of resources. Broadwell (2004) [58] did experiments on response time [59] as performance metric to evaluate the performance of internet services. Patil et al. [60] opined that modern web application developers should consider achieving high Response Time of web applications to maximize user experience.

They proposed proxy cache concept for enterprise application development. As explored in [61] caching is best used when same payload is carried by multiple queries. Even the commercial user agents like Firefox, Chrome etc. also have cache-friendly features [62].
Feng and Yang (2007) [63] presented an improved connection pool model that can be used in modern web applications for improving Response Time and throughput.

They proved that improving connection pool performance can lead to web application performance. According to Li and Lu [64], performance of a web database has its influence in the performance of a web application. They considered response time to measure the performance of web applications.

Maciaszek and Liong (2004) [65] proposed an architectural pattern for web application development. They followed a layered architecture and named it as “PCMEF”. It has four layers namely presentation, control, domain and foundation. The presentation layer is to provide user interface. The control layer is responsible for request processing.

The domain layer has two packages such as entity and mediator for encapsulating business objects and to mediate between control, entry package and foundation layer respectively. Foundation layer provides communication with data sources. Madeyski and Stochmialek [66] proposed a new architectural known as extensible Web Architecture (XWA) framework for web application development.

This framework is based on both MVC and PCMEF. It also has four layers namely presentation, control, domain and foundation.
However, these layers are organized into MVC so as to mix the MVC and PCMEF architectures.

There was no comprehensive research on proposing design patterns for improving quality and performance of web applications. However, attempts were made by various researchers to work in this direction. MVC (Model View Controller) has been used widely for enterprise web applications. It proved to be scalable and maintainable [111].

This is the main architectural pattern which provides basic layers to achieve division of labor. David Parsons [112] proposed client side buffering, delta management architecture, and auto-completion architecture for web applications. Design method for web application’s scalability and flexibility was proposed by Kwon and Bang [113].

Schwabe et al. [114] proposed an architectural pattern named “Object Oriented Hypermedia Design Model (OOHDM)”. It makes use of design models such as interface, conceptual, navigational. These patterns were used to develop push communication and recommendation patterns. Quality of web applications was assessed by Chang and Hon [115]. Navigational and architectural design patterns were explored by Thung et al. [116] using a school web site as case study.

Their study includes navigational patterns such as set-based navigation, pagination, news, and navigation strategy and navigation
observer besides architectures such as MVC and PAC (Presentation Abstraction Control). Rossi et al. [117] focused on navigational patterns for web information systems. The concept of connection pooling was used by Yi Li and Kevin Lü [118] for improving performance of web applications that access relational databases.

Lorna Uden [119] explored UI models like RMM (Relationship Management Methodology) and OOHDM. Leff and Rayfield [120] and Lucca et al. [121] tried out patterns in web application development based on MVC architecture. Based on security Muñoz-Arteaga et al. [122] proposed web application development that is coupled with usability.

A framework for web application development was given by YONG-JUN and LI KE-XI [123] which has basic building blocks. Jeff Offutt [124] observed that reliability, usability and security are the most important quality attributes for the success of web applications. Other quality attributes include availability, scalability, maintainability, and time-to-market. In [125] it is observed that design patterns are proven industry best practices that improve the quality attributes of applications making them more maintainable.

2.3 Design Patterns vs. Fault Tolerance of Web Applications

Web applications are widely used by businesses to reach global audience. E-commerce and other commercial applications that leverage business performance need to be given highest priority since
their performance leads success. Having said this it is essential to make such applications fault tolerant. Fault is a deviation from the normal behavior of the application.

It is also known as a malfunction. Modern enterprise web applications need to have reliable functionality. Fault tolerance is one of the reliability factors which can be achieved using modern structuring techniques blended with design patterns. Fault tolerance is highly desirable feature of any software product. Enterprise web applications are no exception.

The basics of fault tolerance system includes fault detection, fault containment which is essential to prevent its propagation, fault isolation, no single point of failure and the availability of contingency plans in place as reversible models [67]. At hardware level it is possible to overcome faults using replication, redundancy, and diversity.

Replication refers to have multiple systems which are identical. Redundancy of identical systems can help in switching to alternative systems in case of failure. Diversity refers to have multiple implementations that can tolerate fault [68]. Troger [69] explored object oriented design patterns for fault tolerant application development.

Fault tolerant systems employ patterns such as observers and monitors. Such patterns are useful to both tasteful and stateless systems. The fault tolerance and security are non-functional
requirements of application. According to Troger fault tolerant patterns are architectural patterns, detection patterns, error recovery patterns, error mitigation patterns and fault treatment patterns.

Error detection patterns can detect the faults or failures while the error recovery patterns help in operation continuity. In other words, error recovery patterns are responsible to ensure that system continues functioning in spite of some faults and getting rid of the effects of faults if any by creating new state. Error mitigation patterns are responsible to mask errors and compensate the system in other way while the fault treatment patterns prevent recurring of such faults and take corrective measures [70].

Web applications are prone to runtime faults and need fault tolerant mechanisms at application design level. Harrison et al. (2010) [71] show the modern structures that can be employed to build integrated, fault-tolerant applications in the real world. They employed a multi-level exception handling to make the applications fault tolerant.

They also applied coordinated atomic actions in order to handle abnormal situations. They modeled travel agency web site for demonstrating the modern structures for fault tolerance. In their general implementation structure it is evident that the application has pre-defined structured along with legacy components. The wrappers created and strategically placed can handle abnormal behavior at
Garcia and Toledo [73] presented architecture for web services to be tolerant. However, in this paper we assume no usage of web services and built an architectural pattern that demonstrates fault tolerance with the help of design patterns. From the experiments it is understood that fault tolerance has its toll on system performance. Therefore it has to be used sparingly. As witnessed in the architecture with fault tolerance and with faults in the system caused more response time.

Gawand, Mundada and Swaminathan (2011) [74] explored software architectural design for fault tolerance followed by fault tolerance redundancy patterns, reflective state pattern, Tri-Module redundancy pattern and control-monitor safety pattern. The safety measures they considered in their research include modularity, fault avoidance, fault tolerance, technology transparency and ease of modifications.

They identified system failures as of two types namely random and systematic. The former is the failure caused by some degradation in the system while the latter represents a fault due to the flow in the underlying system [75]. The hierarchy of safety tactics presented in [76]. Failure avoidance is achieved through substitution. Failure
detection is achieved using time outs, timestamp, condition check and comparison.

Failure containment is achieved using redundancy, recovery, masking and barrier. Finally failure is either avoided or handled with the help of this hierarchy of safety tactics. In this paper we are inspired by this hierarchy of safety measures. Richar III and Tu (1998) studied fault tolerance practically in Java applications. They observed that fault tolerance is indispensable in both sequential and distributed software systems.

They described Java design patterns for achieving fault tolerance. The patterns include Idempotent for stateful applications, pattern for fault tolerant C/S applications [77] and a pattern for bag-of-tasks applications. Garcia, Beder and Rubira (2012) [78] explored fault tolerant object oriented systems. They advocate design decisions pertaining to exception handling for making applications fault tolerant.

They explored exception handling in sequential system and also concurrent systems with a banking application as case study. The components in their fault tolerant architectures include exception, handler, exception handling strategy, concurrent exception handling action. Their software architecture for exception handling has a collection of design patterns such as exception pattern, handler pattern, strategy pattern and concurrent exception handling pattern are used for making software application fault tolerant.
The names of patterns reflect the purpose of patterns as they are intuitive in nature. In this paper we use a design pattern using AOP (Aspect Oriented Programming) [79] paradigm for complete exception handling thus we separate concerns in the application. Duncan and Pullum [80] employed three level class frameworks as pattern for making applications fault tolerant.

By separating their classes into base classes, data type dependent classes and data type specific utilities as three layers they achieved fault tolerance. Again in each type of classes there are two types of components employed namely executive components and building block components. The executive components make use of fault tolerant techniques which are independent of data types.

The building block components bestow utilities pertaining to fault tolerance besides giving application specific code libraries. This approach is not used in this paper as our framework needs other design patterns. A generic approach is possible to have complex fault tolerant applications. Xu, Bandell and Romanovsky [81] focused on building a generic approach that could handle the residual faults in complex software systems.

The generic model is actually a multi-layered reference architecture which comprises of two things namely configuration method and an architectural pattern. Their work is based on [82]. The FT component used has both normal activity handling and exception
handling component. The interface exceptions are propagated to the exception handling part of the FT component where it has many handlers for exceptional responses, abort exceptions and failure exceptions.

This architecture for making applications FT is known as multi-level reference architecture. Pre-defined classes and runtime libraries are used to implement the FT. Ermagan et al. (2008) [83] proposed fault tolerant software oriented architectures which are model driven by nature. They applied the architectures to a trading system. The pattern implemented was named as “rich service” pattern. The design patterns in fault tolerance makes fault tolerant.

The FT approach used includes defining failure hypothesis, failure detection and failure mitigation. The interaction pattern along with deadlines proposed in [84] is used while the mitigation strategies explored in [85], [86] are used in the FT approach. In this paper we follow design patterns based approach to implement fault tolerant architecture.

2.4 Design Patterns vs. Scalability and Availability

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. 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 world. 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. Due to the rapid growth of potential users of web applications, scalability and availability has become cornerstone of web application design.

Zhao and Schulzrinne [87] built an architecture that promotes dynamic scalability of web applications. Their architecture was named DotSlash [88], [89] that has been improved further [90] to increase scalability. The design goals of the system are scalability, transparency [91] and self-configuration. They employed query result caching [92] 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. Caching feature helps in speeding up the request processing thus promoting scalability. Cache [93] 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 chapter three.

Wei (2012) [94] 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 [95] and fault tolerance. Unlike our framework explored in chapter 3, this work has thrown light into backend side improvements for promoting scalability.
Manjhi et al. (2006) [96] studied the tradeoff between security and scalability of web applications. They employed a third party product known as Database Scalability Service Provider (DSSP) [97] 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. Data scalability services are provided by DSSP architecture. This is achieved by DSSP 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 [98] without compromising scalability.

This will help administrators to get relieved from the burden of managing tradeoff. The work in [99] 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 [100] 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) (2008) [101]. It focuses on scalable security analysis rather than promoting

However, in this paper our focus is on building web applications using original J2EE [103] technologies such as Servlets and JSP along with design patterns. Garrod and Manjhi (2006) [104] presented a proxy query result cache named “Ferdinand”. It maintains high cache hit rate using both local cache and distributed query cache. 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 (2005) [105] introduced a middleware named “Ganymed” which takes care of database replication for improving scalability of web applications. Oracle (2011) [106] presents architecture of MySQL database server which has out of the box replication features that ensures high availability and scalability of web applications.

MySQL [107] 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 (2011) [108] 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 and ForwardServlet 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 [109] studied GOF [110] 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, understandability and reusability respectively.
2.5 Summary

The architectural patterns that have been found in the literature are useful in certain areas for improving quality and performance of web applications. These patterns include MVC, JEMSF, OOHDM, PAC, SMT, PCMEF, and ISPWAD and so on. The interesting thread in all these frameworks are that each framework is focused on certain aspects of design of web applications.

For instance MVC has division of labor with many advantages. However, it does give freedom in all the layers for implementation. PAC throws light into presentation part only. SMT is best used for optimum resource utilization. JEMSF uses design patterns such as DTO and DAO besides connection pooling. It left scope for many other design patterns that cater to scalability, availability and fault tolerant web application development.

OOHDM mainly focuses on hypermedia utilization in object oriented fashion for making navigations dynamic and interactive. It is the mix of conceptual model, navigational classes, and interface objects while ISPWAD throws light into security aspects of web applications.

From the study of these frameworks and other ways and means in the literature with respect to improving quality and performance of web applications it was felt that there was no comprehensive architectural pattern with underlying design patterns for improving
scalability in terms of response time and throughput, availability, fault
tolerance, and maintainability of web applications.

These attributes are desired by any enterprise web application. This is the motivation behind taking up this work. Our architectural pattern proposed in the coming chapter discusses many design patterns that can improve the quality and performance of web applications. Moreover it discusses the refactoring model which helps converting existing web applications to take advantage of our architecture.

The architectural pattern is further improved gradually to improve the attributes specified above with respect to quality and performance of web applications.