UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TRABAJO DE INVESTIGACIÓN

UNA REVISIÓN DE LOS PATRONES DE DISEÑO DE SOFTWARE APLICADO A LAS APLICACIONES WEB

Para Optar el Grado de Bachiller en Ingeniería de Sistemas

Autor:
Gonzales Gonzales Christian Erick

Asesor:
Mg. Cachay Maco Junior Eugenio

Línea de Investigación:
Infraestructura, Tecnología y Medio Ambiente

Pimentel – Perú
2020

DEDICATORIA

Dirigido con mucho cariño a mis padres, por ese apoyo incondicional y esas ganas de verme por segunda vez profesional, por ser esa motivación extra para tener las fuerzas suficientes para realizarme como profesional y quienes son los pilares fundamentales que me brindan el apoyo en mi vida, con su infinito amor y sabios consejos que han sabido guiarme.
También una dedicatoria especial a mis docentes, por todas las enseñanzas recibidas durante toda la carrera universitaria.

AGRADECIMIENTO

En primer lugar, a nuestro Dios padre que me brindó la oportunidad de cumplir uno de mis objetivos, por enseñarme el camino idóneo en mi vida, por darme las fuerzas para superar los obstáculos en mi carrera estudiantil, así también agradecer a todas las personas que estuvieron involucrados de manera indirecta y directa en la formación académica que obtuve en la universidad, a mis padres por brindarme el apoyo en toda etapa de mi vida, por ser un ejemplo de superación, esfuerzo y brindarme amor y comprensión en todo el trayecto de mi vida.

RESUMEN

Las páginas web son cada vez más complejas al diseñar y construir debido a las nuevas exigencias de diseño y funcionalidad requeridas por los usuarios.
Por lo que ha dado lugar a la aparición de la Ingeniería Web como una subdisciplina de la Ingeniería de Software el cual nos permite crear sistemas interactivos web de alta calidad.
El saber más sobre la Web nos lleva a conocer como está construida o cuál es su arquitectura.

Al hablar de la arquitectura de la web nos encontraremos con los Patrones de Diseño, que son patrones reutilizables, es decir describen la experiencia comprobada y documentada de un problema repetido y poder así resolverlo en el área de desarrollo de software. Reutilizar estos patrones de diseño nos ayudara a no cometer los mismos errores que cometieron los diseñadores debido a que a estos errores ya se encontraron respuestas.

Esta revisión tiene como fin recopilar los documentos de los patrones de diseño de software, de tal manera que nos facilitara la reducción de errores en la definición, diferentes tipos de patrones de diseño, y así lograr una alta calidad al desarrollar un software web.

Este material nos ha permitido brindar aportes a los diseñadores y programadores de software al momento de realizar proyectos de software.

Palabras claves:
Ingeniería Web, Patrones de Diseño, MVC, MVP

ABSTRACT

Web pages are increasingly complex when designing and building due to the new design and functionality requirements required by users.
So it has led to the emergence of Web Engineering as a subdiscipline of Software Engineering which allows us to create interactive web systems of high quality. Knowing more about the Web leads us to know how it is built or what its architecture is.

When talking about the architecture of the web we will find the Design Patterns, which are reusable patterns, that is, they describe the proven and documented experience of a repeated problem and thus be able to solve it in the area of software development. Reusing these design patterns will help us not to make the same mistakes that designers made because these errors have already found answers.

This review aims to collect the documents of the software design patterns, in such a way that it will facilitate the reduction of errors in the definition, different types of design patterns, and thus achieve high quality when developing a web software.

This material has allowed us to provide contributions to software designers and programmers when making software projects.

KeyWords:

Web Engineering, Design Patterns, MVC, MVP

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

I. INTRODUCCIÓN

Anteriormente las webs ofrecían funcionalidades simples en el cual se podía leer información mediante descripciones e imágenes, pero actualmente existen web que no solo te brindan información, sino que además ofrecen más funcionalidades como realizar operaciones más complejas lo cual conlleva a los diseñadores construir los sistemas web con más exigencia al diseñarlo.

Es ahí la importancia que ofrece la web ya que tiene que ser construido con una arquitectura adecuada. Al centrarnos en su arquitectura nos abarcamos en la utilización de los patrones de diseño, el cual nos ayudara a resolver problemas de diseño en las que ya se han encontrado soluciones que están basados a las experiencias.

Son muchos la existencia de patrones de diseño de software en donde encontramos que han sido utilizadas al desarrollar sistemas interactivas web. Los diseñadores los reutilizan con el objetivo de intentar disminuir de manera eficiente los errores que se cometieron anteriormente otros diseñadores.

Existen varios patrones de diseño, que nos van a permitir a tener una gran variedad de patrones, el cual podamos seleccionar el adecuado y el más efectivo. Hay que tener en cuenta que también podemos agrupar los patrones de diseño de tal manera que nos ayude a solucionar los problemas al desarrollar el software.

Una de las dificultades que se presentan al seleccionar el patrón de diseño, es que encontramos demasiada información en los sitios web, libros y artículos; lo cual muchas veces confunde a los diseñadores a seleccionar el patrón adecuado.

Actualmente se han publicado investigaciones relacionadas al tema en cuestión en el que utilizan diferentes patrones de diseño, a pesar de que existen varios patrones de diseño, el diseñador sigue cometiendo errores al selecciona adecuadamente el patrón, lo que permite un atraso de tiempo y perdidas económicas al desarrollar el software. En consecuencia, nos va a permitir este documento una revisión de las propuestas planteadas al desarrollar el software.

1.1.Antecedentes de Estudio

Los autores (Khaliluzzaman & Chowdhury, 2016), en la investigación "Pre and Post Controller based MVC Architecture for Web Application", desarrollada en la 5ta Conferencia Internacional sobre Informática, Electrónica y Visión en el año 2016; identifico el problema que actualmente se tiene al implementar el Patrón de Diseño MVC en aplicaciones web debido a que es más difícil y requiere mucho tiempo de trabajo que antes. En dicha investigación se propuso un nuevo marco MVC extendido que está basado en pre y post controlador para la aplicación web. Para medir su rendimiento se usaron las herramientas el Apache Benchamark para medir su rendimiento, el cual consiste en realizar una comparación del nuevo marco MVC Extendible, Silex y Slim. Al final de la investigación se obtuvo que el patrón de diseño más efectivo fue el nuevo marco MVC Extendible, ya que ofrece un mejor rendimiento respecto a los demás. Luego de hacer la comparación entre los resultados se obtuvo con respecto a la solicitud por segundo y el tiempo por solicitud un resultado que durante 30 segundos con 30 niveles simultáneos propuestos es 115.52, 39.58 y 45.2. La relación con la investigación está dada porque han realizado la comparación mediante la herramienta Benchamark para medir su rendimiento los patrones de diseño.

Los autores (Yehia Elshater, Patrick Martin , & Ehab Hassanein, 2015) ,en la investigación "Using Design Patterns to Improve Web Service Performance", desarrollada en la Conferencia Internacional de Servicios de Información en el año 2015, identifico el problema de la demora en el tiempo de respuesta (latencia) y el rendimiento de los servicios web de búsqueda. En dicha investigación se propuso evaluar los patrones de diseño Singleton y el Patrón de Almacenamiento Caché. Para realizar la evaluación se hizo de dos modos que consiste en solicitudes que son modo secuencial que está dedicado al envió múltiple de solicitudes al servicio web de destino de forma secuencial y el modo simultaneo que proporciona más dinamismo que el modo secuencial simulando solicitudes concurrentes de consumidores de servicios, teniendo en cuenta que para simular la interoperabilidad de los servicios y el mensaje SOAP que pasa entre diferentes tecnologías, utilizan ASP.NET que se encarga construir el servicio web 'mundo' que se despliega en el lado del proveedor de servicios y además por otro lado se utiliza Java para implementar el programa cliente en el lado de servicio al consumidor. Al final de la investigación se obtuvo que el patrón de diseño que ofrece un mejor rendimiento es el Patrón Cache. Luego de hacer la comparación

entre los resultados se obtuvo que en modo secuencial los valores de desviación de estándar del enfoque tradicional, los patrones singleton y caching que son 0.39, 0.45 y 0.72 respectivamente teniendo en claro que enfoque de almacenamiento en caché es mayor que los otros dos enfoques y en el modo simultaneo que se compara el patrón caché y el servicio web tradicional que consta de tres medidas que son Tiempo medio de respuesta de consumidores que por cada 50 respuestas es de 90 y 280; tiempo medio de respuesta aproximadamente 70,5% de ahorro de tiempo de respuesta (en promedio) y solitudes por unidad de tiempo que son 6.2 y 1.9). La relación con la investigación está dada porque han realizado la comparación mediante dos modos de evaluación que consiste en modo secuencial y el modo simultaneo para medir su rendimiento de cada patrón de diseño.

Los autores (Thung, Ng, Thung, & Shahida Sulaiman , 2010), en la investigación "Improving a Web Application Using Design Patterns: A Case Study", presentada en Tecnología de la información en el año 2010, identifico el problema de la página web de una escuela es que tenía mala facilidad de uso y un mal rendimiento. En dicha investigación se propuso un análisis comparativo de varios patrones de diseño. Para realizar el análisis se analizó el contexto, la estructura y las consecuencias de cada patrón en los que considero dos patrones arquitectónicos que son el MVC (Modelo, Vista y Controlador) y PAC (Presentación, Abstracción y Control) y cinco patrones de navegación (Observador de Navegación, Estrategia de navegación, Noticias, Paginación y Navegación). Para medir su facilidad de uso y su rendimiento de la página web de la escuela se adoptó y combino algunos patrones de diseño. Al final de la investigación se obtuvo que al combinar unos patrones de diseño lograron que el sitio web al utilizarlo por los usuarios sea más sencillo y fácil de mantener para los diseñadores de software o desarrolladores. La relación con la investigación está dada porque han realizado la comparación o combinación de patrones de diseño al que se le realizo a un caso de estudio de una página web de una escuela para así medir su rendimiento con el uso de patrones de diseño.

Los autores (Santi Caballé, y otros, 2014), en la investigación "A Presentation Framework to Simplify the Development of Java EE Application Thin Clients", presentada en la 8va Conferencia Internacional sobre Sistemas Intensivos Inteligentes y Complejos (CISIS) en el año 2014, identifico el problema que la desventaja al usar el patrón MVC es que tiene muchas tareas repetitivas que todas las aplicaciones compatibles deben realizar, lo que hace que el desarrollo sea tedioso y complejo. En dicha investigación se propuso un marco de

software para desarrollar la aplicación Java EE que admite el nivel de presentación. Para lo cual se hizo primero la revisión de los principales marcos de software y patrones de diseño compatibles con Java EE, luego presentar el diseño del marco y finalmente la implementación para luego evaluar el marco mediante una aplicación de prueba ingenua llamada PruebaDerbyUOC y se ejecutó en el servidor GlassFish. Al final de la investigación se obtuvo que el marco propuesto logra un mejor rendimiento. Luego de hacer el diseño del marco y la implementación se obtuvo como resultado que al implementar ciertos patrones de diseño que funcionan en conjunto con los patrones Struts logran un mejor rendimiento el marco propuesto es decir el marco simplificó y facilitó mucho el desarrollo de las pruebas aplicación siguiendo las acciones de Struts 2, que proporciona un modelo claro y la implementación de muchos patrones de diseño de Java EE. La relación con la investigación está dada porque han propuesto un marco de software con mejor rendimiento que el patrón de diseño MVC.

Los autores (Giovani Guizzo, Thelma Elita Colanzi, & Silvia Regina Vergilio, 2013), en la investigación "A Feasibility Analysis for the Application of Design Patterns in Search Based Product Line Design", desarrollada en la 32va Conferencia Internacional de la Sociedad Chilena de Ciencias de la Computación (SCCC) en el año 2013, identifico el problema que los Patrones de Diseños aplicados al campo de Ingeniería de software basada en búsqueda (SBSE) no encontramos enfoques que exploran la aplicación de patrones de diseño en el software Arquitecturas de línea de producto (PLA). En dicha investigación propuso un análisis de factibilidad, realizado con patrones GoF (Gang of Four). Para lo cual se hizo en el análisis la estructura del patrón, las consecuencias del uso de acuerdo con algunas métricas arquitectónicas, flexibilidad y aspectos de automatización. Al final de la investigación se obtuvo que Cuatro patrones (Bridge, Strategy, Facade y Mediator) son factibles y dos patrones (Bridge y Strategy), de ellos tienen ámbitos de aplicación Línea de Productos de Software (SPL) específicos que implican variabilidades. Luego de realizar el análisis de factibilidad se obtuvo como resultado que los patrones de creación son inviables, ya que se aplican a un rango bajo de ámbitos. Además, los ámbitos adecuados de los patrones de comportamiento son difíciles de identificar automáticamente en los diagramas de clase, porque este tipo de diagrama es estático y muestra estructura en lugar de comportamiento, los patrones estructurales tienden a ser más factibles debido a su característica estática, que es más compatible con los diagramas de clase. La relación con la investigación está dada

porque se realizará a los patrones de diseño un análisis que me permita elegir el patrón adecuado para aplicarlos a las búsquedas.

Los autores (Alexander Katzmaier & Martin Hanneghan , 2013), en su investigación "Design pattern evaluation of mobile and web based application frameworks", desarrollada en la 6ta Conferencia Internacional sobre Desarrollos en Ingeniería de Sistemas Electrónicos en el año 2013, identifico el problema de que las técnicas para aplicaciones web de escritorio no se transfieren al entorno móvil lo suficientemente bien. En dicha investigación propuso utilizar el método empírico ya que este es un mecanismo eficaz cuando un ' cómo' o 'por qué se le está pidiendo' cuestión y el investigador no tiene control sobre las variables y la cuestión se centra en un conjunto contemporáneo de los acontecimientos. Para lo cual lo primero que se hizo fue definir y examinar que patrones de diseño que estarían incluidos y además cuales son útiles para los marcos web (ASP.NET MVC, Struts, JSF, Grails y Zend) y el marco móvil (Android, iOS, MonoCross, Appceleretor Titanium, Sencha Touch, JQuery Mobile, AppMobi, PhoneGap), los patrones que consideraron fueron MVC, Front Controller y Contextos y la Inyección de Dependencias (CDI) y así luego calificarlos con un valor 0 y 1, donde 1 es la puntuación más alta. Al final de la investigación se obtuvo que el patrón de diseño MVC es el patrón que más usan los marcos. Luego de realizar la calificación se obtuvo como resultados que la puntuación media de la MVC para el desarrollo móvil es de 0,59 en comparación con 0,16 (Front Controller) y 0,24 (CDI). La relación con la investigación está dada porque se realizará un método empírico a patrones de diseño, dando como resultado que el patrón MVC es el patrón que más usan los marcos web y móvil.

Los autores (Kazan, Cantürk, & Bastan, 2015), en su investigación "Performance Analysis of a Software Developed with and without Design Patterns: A Case Study", desarrollada en la 12va Conferencia Internacional sobre Computación Electrónica y Computación (ICECCO) en el año 2015, identifico el problema que tiene el sistema de seguimiento de Bandrol para Turkish Radio Television Corporation (TRT) – Turkey que está diseñado sin patrones de diseños y por consecuencia el sistema no cumplía con los requisitos del usuario y no podía proporcionar una cierta integración con otros sistemas. En dicha investigación se propuso rediseñar el software utilizando patrones de diseño usando modelos de Model-View-Controller, Model-View-Presenter y Proxy. Para lo cual lo primero que se hizo es diseñar la aplicación web con el enfoque orientado a objetos, se determinaron 3 casos de uso, navegar y elegir a través del catálogo de patrones de diseño, aplicar los patrones

seleccionados y por último se comparó el éxito de dos aplicaciones según los datos estadísticos mediante el software JMeter y Speedy Framework. Al final de la investigación se obtuvo que el software escrito mediante el uso de patrones de diseño es más fácil ofrecía un mejor rendimiento. Luego de realizar la comparación de las dos aplicaciones se obtuvo como resultado en estadísticas de tiempo de ejecución para el uso de 'demanda de bandrol' para 10 usuarios con la nueva aplicación con patrones de diseño en número de operaciones, promedio (milisegundo), mínimo (milisegundo), máximo (milisegundos) es de 230, 17, 1, 651 respectivamente comprado a la aplicación sin patrones de diseño es de 100, 581, 5, 7820 y para 100 usuarios con patrones es de 2500, 102, 0, 2201 a diferencia de la aplicación sin patrones de 1000, 577, 5, 8377. La relación con la investigación está dada porque se analizó un caso de estudio en el cual no fue escrito con el uso de patrones de diseño, dando como resultado que el uso de patrones de diseño proporciona un mejor rendimiento e integración con otros sistemas.

Los autores (XiaoLong Li, y otros, 2015), en la investigación “Application of MVVM Design Pattern in MES”, presentada en la Conferencia Internacional sobre tecnología cibernética en automatización, control y sistemas inteligentes en el año 2015, identifico el problema que al desarrollar diferentes requerimientos en diferentes empresas el Sistema de Ejecución de Manufactura (MES) tiene que desarrollar a medida de la empresa es por eso que al desarrollar el sistema es importante la reducción de códigos modificados y disminución de acoplamiento entre la página de visualización y datos. En dicha investigación se propuso el patrón de diseño MVVM ya que separa la estructura del software para ver, modelar y ver así el modelo mediante XAML y Data Binding de la tecnología WPF. Para realizar el análisis se analizó el contexto, la estructura y las consecuencias del patrón de diseño MVVM. Se desarrolla con .Net Frame Work y el uso generalizado de WPF. Al final de la investigación se obtuvo que al implementar el patrón de diseño MVVM tiene atributos de alta cohesión y bajo acoplamiento; además de reducir el código de desarrollo secundario lo que permite resolver el problema de MES. La relación con la investigación está dada porque han realizado la implementación del patrón de diseño MVVM para diseñar el MES para una industria discreta y así medir su rendimiento ante la efluencia de la modificación del sistema debido a los cambios de requisitos del cliente.

1.2. Planteamiento del problema de investigación

¿Cuál es el estado de conocimiento respecto a los patrones de diseño de software aplicado a las aplicaciones web?

1.3. Objetivos

1.3.1. Objetivo General

Realizar una revisión de los patrones de diseño de software aplicado a las aplicaciones web

1.3.2. Objetivo Especifico

a) Establecer preguntas de investigación.
b) Realizar proceso de búsqueda.
c) Seleccionar y registrar artículos.

1.4. Marco Teórico conceptual

Los fundamentos teóricos tienen el objetivo de entender la arquitectura que tiene el software, siendo los patrones de diseño lo que más utilizan los diseñadores al construir al software.

1.4.1. Software

Son un conjunto de instrucciones detalladas en el cual realizara una tarea que se le asigno, es por eso que espera recibir órdenes para así realizar dicha tarea. (Diéguez, 2014)

Se le define como el componente lógico que se encarga de relacionar al usuario y la computadora, es decir el usuario envía órdenes mediante periféricos de entrada y la computadora los traduce mediante periféricos de salida. (Pérez Carvajal, 2014)

Son programas que se ejecutan en las maquinas, también se describe como un medio que ofrece un servicio es decir realiza funcionalidades complejas. La elaboración del software ofrece una gran ayuda a muchas personas por lo que se le considera como una actividad económica. (Chaos García, Gómez Palomo, & León Molina, 2017)

Se llama software a los programas, sistemas operativos o la elaboración de un producto que se pone a la venta. Existen varios tipos de software, uno de ellos es el tipo de software de propósito general, el cual está diseñado para realizar una tarea específica, dentro de este tipo encontramos a las aplicaciones web. (Cardador Cabello, 2014)

1.4.2. Ingeniería de Software

Comprende la producción del software en dos aspectos, el primero está basado en teorías y herramientas que tienen como fin brindar solución a los problemas que se encuentren y como segundo son los técnicos de desarrollo de software, teorías de apoyo, la gestión de proyectos y el desarrollo de herramientas y métodos. Estos aspectos abarcan la producción del software desde las etapas iniciales hasta el mantenimiento después de haber utilizado el software. Para los ingenieros de software esta disciplina tiene de gran importancia para que adopten organizado trabajo y un enfoque sistemático. (Gómez Ruedas, 2016)

Tiene como finalidad brindar un software de calidad, en aspectos como la eficiencia, usabilidad y más. No solo el software debe funcionar, sino que debe tener calidad por lo que se utiliza una serie de técnicas, métodos y más de tal manera que nos permita obtener un software de calidad. Hay que tener en cuenta que la Ingeniería de Software abarca todas las etapas de desarrollo el software y no solo una etapa en el desarrollo del software. (Cabot Sagrera, 2013)

1.4.2.1. Ingeniería Web

Está basado en las tecnologías web del desarrollo del software, utilizando métodos que brinden una mejor calidad al software web.

1.4.2.1.1 Aplicaciones Web

Las aplicaciones web son codificados por un lenguaje de programación y son construidos para que sean soportados por los navegadores web y así poder interactuar entre si el usuario y el servidor. (Cardador Cabello, 2014)

Son aplicaciones que están escritas mediante un tipo de lenguaje y que puede ser ejecutado en cualquier sistema operativo; se le denomina también aplicación web a la comunicación que utiliza mediante protocolos HTTP. Existen muchos ejemplos de aplicaciones como son las tiendas en línea y más. (Hernández Diaz, André Ampuero, & Martínez Prieto, 2012)

Su arquitectura está constituida de dos lados, el lado del cliente en el cual podemos utilizar la aplicación mediante navegadores web (Google Chrome, Firefox), y así de esta manera el cliente interactuar con la aplicación y el otro lado que es el servidor, que es en donde reside los datos, reglas y lógica de la aplicación. No solamente puedes acceder a estas aplicaciones a través de la computadora, sino que además se pueden utilizar en dispositivos móviles, por lo que su utilidad no limita el lugar en donde se encuentre el usuario mientras se tenga internet. (Ferrer Martínez, 2014)

1.4.2.2. Desarrollo de software

De acuerdo con (Juan Carlos Moreno Pérez & Arturo Francisco Ramos Pérez, 2014), para desarrollar un software de calidad existen varios tipos de modelo de desarrollo, en el cual encontramos etapas o fases para el desarrollo del Software, los cuales son:

a) **Análisis**

   Está basado en recopilar, examinar y formular los requerimientos que quiere el cliente, por lo que se usan técnicas que ayuden a recopilar los requisitos del software, como son las entrevistas en el cual interactúa los clientes y los analistas.

b) **Diseño**

   Se divide en dos fases que son el diseño general, que se encarga de los requisitos generales de la arquitectura de la aplicación y el diseño en detalle que son el subconjunto de dicha aplicación.

c) **Codificación**

   Luego de las funciones que se definieron en la fase de diseño, se implementa el software mediante un lenguaje de programación.

d) **Pruebas**

   Se realizan dos pruebas en esta fase, fase de unidad que son pruebas individuales para cada subconjunto de la aplicación y la otra prueba de integridad que garantiza que los módulos diferentes se integren con la aplicación.

e) **Mantenimiento**

   Hace referencia a las actualizaciones secundarias y procedimientos correctivos del software.

Llegamos a la conclusión que, en la fase de diseño, se habla de la arquitectura de la aplicación, es por eso que hablaremos de los Patrones de Diseño ya que son aplicables para la arquitectura del software final.

1.4.2.3. Patrones de diseño

Los autores (Honrubia López, 2014), dicen que los patrones de diseño no son códigos, sino que son la idea abstracta que proporciona una plantilla de tal manera que al adaptarlo al desarrollo del software nos brinde una solución a un problema determinado del diseño. Para considerarlo patrón debe cumplir con dos requerimientos indispensables:

a) La primera es que la solución que resuelva el problema debe brindar una solución muy efectiva y la mejor opción en comparación a otras posibles que se tenga.
b) La segunda es la reutilización de la solución para cuando se tenga el problema.

La utilidad de los patrones tiene como propósito:

a) Al desarrollar el software debemos tener un grupo de pares problema-solución de tal manera que puedan ser utilizados.
b) Evitar una búsqueda de una solución en la que ya se probó su efectividad.
c) Al realizar el diseño del software se busca que se imponga estándares.

Según los autores (Montero, Zarraonadía, & Díaz, 2011), los patrones es la forma de documentar las mejores prácticas y lecciones que se han aprendido al momento de resolver un determinado problema complejo que se encuentra dentro de un dominio de diseño concreto en el desarrollo del software, la arquitectura o el diseño; por lo que esos patrones tienen la capacidad de brindar soluciones reutilizables de tal manera que ayude a las personas que no tienen experiencia.

Estos patrones redactan a los lectores su historia de cómo debe ser aplicado a un determinado problema y los motivos que llevaron para que crear dicho patrón. Al diseñador novatos estas soluciones nos proporcionan alternativas y problemas adicionales.
La manera de utilizar los patrones es nunca separarlos ya que al unirlo nos brindan soluciones a problemas más complejos, por lo que deben estar interconectados entre sí.

A los patrones lo encontramos en catálogos y lenguajes de patrones, los cuales pueden ser complementarios.

a. Lo clasificamos haciendo uso de criterios para que nos ayuden a realizar una búsqueda de una solución a un problema concreto. Para utilizarlo primero se tiene que saber qué problema se va a solucionar, para luego ir al catálogo y seleccionar que patrones se ajustan mejor a la necesidad del problema.
b. Esta clasificación de lenguaje de patrones es más compleja ya que se aplica un conjunto de patrones y en el orden en que se aplicó debido a que nos ayuda a que tener un proceso más efectivo. Estos patrones no son completamente independientes porque tienen como objetivo resolver un problema complejo y es por eso que deben estar interconectados entre sí.

Su estructura del patrón de diseño son las siguientes:

1) **Nombre**
   Este nombre nos va a permitir identificar al patrón por lo que debe ser único.

2) **Problema**
   Describe en un resumen de una o dos frases los objetivos que tiene que alcanzar como patrón.

3) **Contexto**
   Aplicación del patrón a un problema recurrente

4) **Fuerzas**
   Describe los objetivos, las fuerzas y restricciones relevantes para ese patrón, además de la comunicación que hay entre ellas o con las metas a alcanzar.

5) **Solución**

   Son un conjunto de instrucciones en el que describe la construcción del producto resultante, al ser descrito puede estar acompañado de dibujos, diagramas o esquemas de solución.

6) **Ejemplo**

   Los ejemplos pueden ser visuales de tal manera que ayude al lector a entender su utilidad y en qué momento se aplica el patrón.

7) **Contexto resultante**

   Indica cual es el estado del sistema después de que se aplicó el patrón, en la que incluye sus consecuencias ya sea positivas o negativas.

8) **Exposición razonada**

   Expone cómo funciona el patrón y el porqué de su utilidad, es decir explica sus mecanismos subyacentes.

9) **Patrones relacionados**

   Son patrones en el cual se pueden agrupar con este o la posibilidad de aplicarlos a partir del contexto resultante o también representa las soluciones alternativas.

A pesar de todas las ventajas que tenemos al utilizar los patrones, existen desventajas que nos permite tener limitaciones para que su adaptación sea los más fructífera posible.

a) La gran variedad de patrones de diseño, confunde a la propia comunidad que va dirigida, por lo que ocasiona una serie de problemas. Hay potencialmente varios patrones disponibles que mayormente están publicados en libros y artículos, pero también están disponibles en diferentes sitios web; lo que permite los lectores realicen una búsqueda manual para identificar al patrón.

b) Una vez encontrado el patrón, el siguiente obstáculo es lo difícil de aplicarlo debido que, si no entiendes a profunda el problema a resolver, no sabrás en que momento aplicarlo. Por lo que una de las dificultades que se presenta es determinar cómo, cuándo y donde tienen que ser aplicados los patrones de diseño. Este problema ocurre debido a que no se detalla el nivel de implementación del patrón, y lo que si detalla en si son una enumeración de elementos, descripciones abstractas y sus relaciones para solucionar un problema.

c) Existe la posibilidad de hacer uso de otros patrones para completar la solución, por lo que se tendría que repetir los pasos anteriores.

d) Otro de los problemas es el uso de patrones que no está soportado metodológicamente, salvo en algún dominio concreto.

e) El encontrar patrones de diferentes nombres que aborden el mismo problema o el estilo literario que usa el autor, muestra una dificultad de entendimiento por parte de los diseñadores.

1.4.2.4. Tipos de Patrones de diseño

1.4.2.4.1. Patrones de diseño orientado a objetos

Según (Gamma, Helm, Johnson, & Vlissides, 1994), organiza a estos patrones por dos criterios que son: propósito (de creación, estructurales y de comportamientos) y ámbito (clase y objeto).

Propósito				
		De Creación	**Estructurales**	**De Comportamiento**
Ámbito	**Clase**	Factory Method	Adpater (de clases)	Interpreter Template Method
	Objeto	Abstract Factory Builder Prototype Singleton	Adapter (de objetos) Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Figura 1: Patrones de Diseño orientado a objetos Gof.

Fuente: (Gamma, Helm, Johnson, & Vlissides, 1994)

A. Patrones de creación

Estos patrones nos ayudan a que el sistema sea independiente de cómo está representado, creado, compuesto sus objetos. Un patrón de creación de clases hace uso de la herencia para cambiar de la instancia a crear, en cambio un patrón de creación de objetos delega la creación de la instancia en otro objeto. En esta clasificación encontramos a estos patrones.

a. Abstract Factory (Fábrica abstracta)
b. Factory Method (Método de fabricación)
c. Singleton (Único)
d. Builder (Constructor)
e. Prototype (Prototipo)

B. Patrones Estructurales

Estos patrones forman estructuras grandes mediante combinación de clases y objetos. Estos patrones estructurales de clases utlizan la herencia para componer interfaces o implementaciones. Se encuentran en esta clasificación los siguientes patrones.

a. Adapter (Adaptador) (de clases y de objetos)
b. Facade (Fachada)
c. Flyweight (Peso ligero)
d. Decorator (Decorador)
e. Bridge (Puente)
f. Composite (Compuesto)
g. Proxy (Apoderado)

C. Patrones de Comportamiento

Estos patrones tienen que ver con algoritmos y con la asignación de responsabilidades de objetos. Además de describir patrones de clases y objetos, también describen los de comunicación entre ellos. Los patrones son los siguientes.

a. Chain of Responsibilitity (Cadena de Responsabilidad)
b. Comand (Orden)
c. Interpreter (Intepreter)
d. Mediator (Mediador)
e. Memento (Recuerdo)
f. Observer (Observador)
g. State (Estado)
h. Strategy (Estrategia)
i. Template Method (Método de Plantilla)
j. Visitor (Visitante)

1.4.2.4.2. Patrones de diseño de arquitectura

El autor (Eslava Muñoz, 2018), nos dice que la arquitectura de software es la organización de los componentes del sistema, que están relacionados entre sí y con el contexto, de tal manera que aplique principios y normas de diseño y calidad que fortalecen y fomentan la usabilidad a la vez del sistema. La arquitectura hace uso de diseño del software para responder criterios tanto en el proceso de desarrollo como de ejecución.

1) Patrón de diseño Modelo, Vista y Presentador (MVP)

Según el autor (Prabowo, Hatma Suryotrisongko , & Aris Tjahyanto , 2018), el patrón de diseño MVP es más flexible en su estructura al implementar una función en la clase específica y también es fácil realizar muchos escenarios de prueba ya que mejora la capacidad de mantenimiento y es así que facilita a crear y ejecutar escenarios de prueba. Este patrón de diseño MVP tiene el mismo concepto del patrón de diseño MVC, pero lo hace diferente el paradigma moderno que eventualmente produce un patrón de diseño que separa cada componente.

Según (Robledo & Robledo, 2013), nos dice que este patrón de diseño se implementa en aplicaciones de grandes proyectos en donde hay varios programadores trabajando simultáneamente debido a la complejidad que presenta.

El esquema que presenta este patrón de diseño está formado mediante tres componentes que son:

a. Modelo (Model)

Este componente se encarga de la lógica de la aplicación. Si nos referimos a las aplicaciones web, son a lo que llamamos servelts, servicio web o cualquier otro tipo de software que resida en el servidor. Es el componente que se comunica con el Presentador para así enviar la información y recibir de este actualizaciones o consultas.

b. Vista (View)

En este componente se incluye todos los widgets necesarios para que así el usuario pueda interactuar con la aplicación. Normalmente este reside del lado del cliente. Cuando el usuario realiza una acción cualquiera sobre la interfaz del usuario, este componente que es la vista informa al Presentador de lo que ocurre en el cierto evento y este a su vez puede indicar a la Vista que actualice lo que ve el usuario.

c. Presentador (Presenter)

Este componente es la pieza clave de todo el esquema, debido a que es el puente de conexión entre el Modelo y la Vista. En respuesta a los eventos del usuario se comunica con el Modelo y así dependiendo de su respuesta pueda enviar a la Vista comandos de actualización.

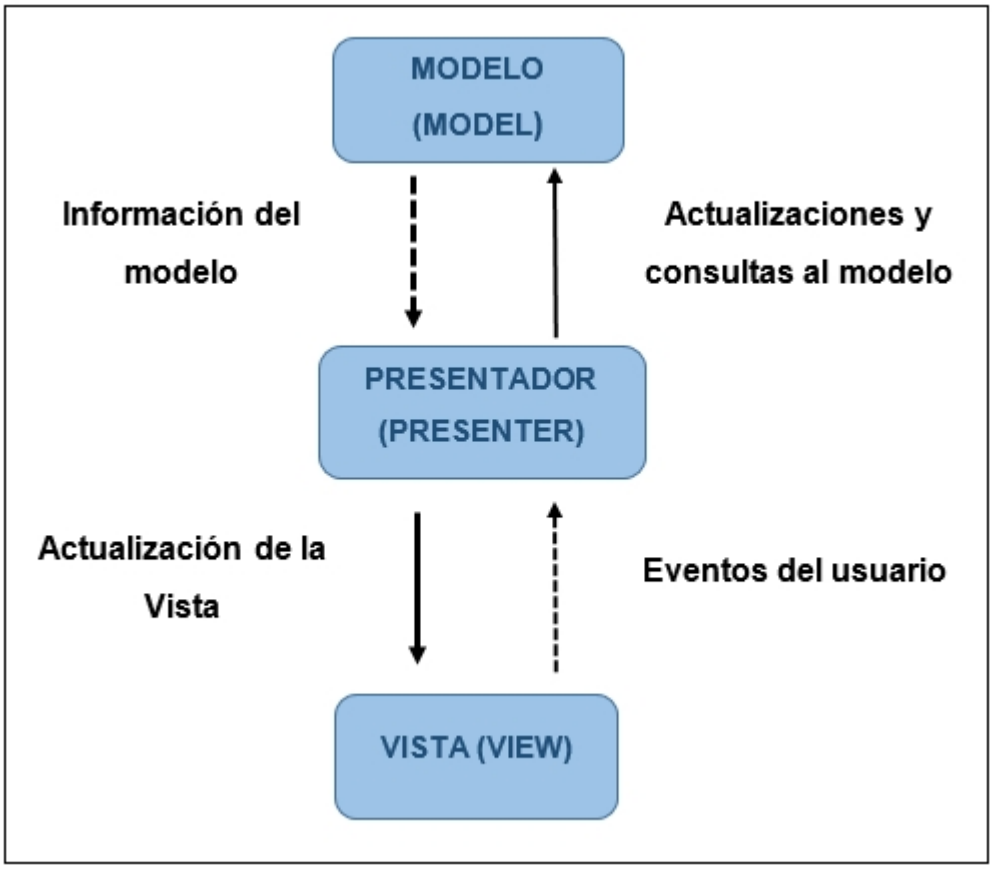

Figura 2: Esquema del patrón de diseño Modelo, Vista y Presentador (MVP).

Fuente: (Robledo & Robledo, 2013)

2) Patrón de diseño Modelo, Vista y Controlador (MVC)

El autor (Eslava Muñoz, 2018), este patrón separa los datos de la lógica de negocio de una aplicación de la interfaz de usuario y del módulo que se encarga de gestionar las comunicaciones y los eventos. Está basado en las ideas de reutilizar el código y de separar de conceptos, que son características que facilitan la búsqueda en la tarea de desarrollo de aplicaciones y de un mantenimiento posteriormente.

El esquema de este patrón está formado mediante tres componentes que son:

a. Modelo (Model)

El autor (Eslava Muñoz, 2018), nos dice que este componente es la representación de la información con el cual está operando el sistema, es por eso que gestiona todos los accesos a dicha información, tanto como las consultas y actualizaciones, también implementa los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Es el componente que envía a la Vista la parte de la información que se solicita a cada momento para que se muestre (típicamente al usuario). Estas peticiones de acceso o la manipulación de información son llevados al modelo mediante el Controlador.

Según (Fernández Romero Yenisleidy, 2012), este componente debe ser independiente al sistema de almacenamiento, también lo define como el encargado de notificar a las vistas el cambio de datos que pueda producir un agente externo, como por ejemplo una actualización de datos, un temporizador que desencadena una inserción o más.

b. Controlador (Controller)

En este componente es el que se encarga de responder a los eventos (usualmente son las acciones del usuario) e invoca las peticiones al Modelo cuando se realiza alguna solicitud sobre la información (Ejemplo: la edición de un documento o un registro en una base de datos). Puede también enviar peticiones a la Vista que está asociada, si se solicita un cambio en la forma que se presenta el Modelo (Ejemplo: el desplazamiento o scroll por un documento o los diferentes registros de una base de datos), es por eso que se podría decir que el Controlador hace de intermediario entre la Vista y el Modelo.

Según (Fernández Romero Yenisleidy, 2012), este componente recibirá los eventos de entrada como por ejemplo al realizar un click, un cambio en un campo de texto o más. Estas acciones suponen peticiones al modelo o la vista, una de estas peticiones a la vista puede ser una llamada al método "Actualizar ()".

c. Vista (View)

Este componente se encarga de presentar al Modelo en un formato adecuado en el que se pueda interactuar (es la interfaz de usuario mayormente), es por eso que se necesita del Modelo la información que se debe representar como salida.

Según (Fernández Romero Yenisleidy, 2012), este componente recibe los datos que son enviados por el controlador o del modelo, para así luego mostrarlo a los usuarios.

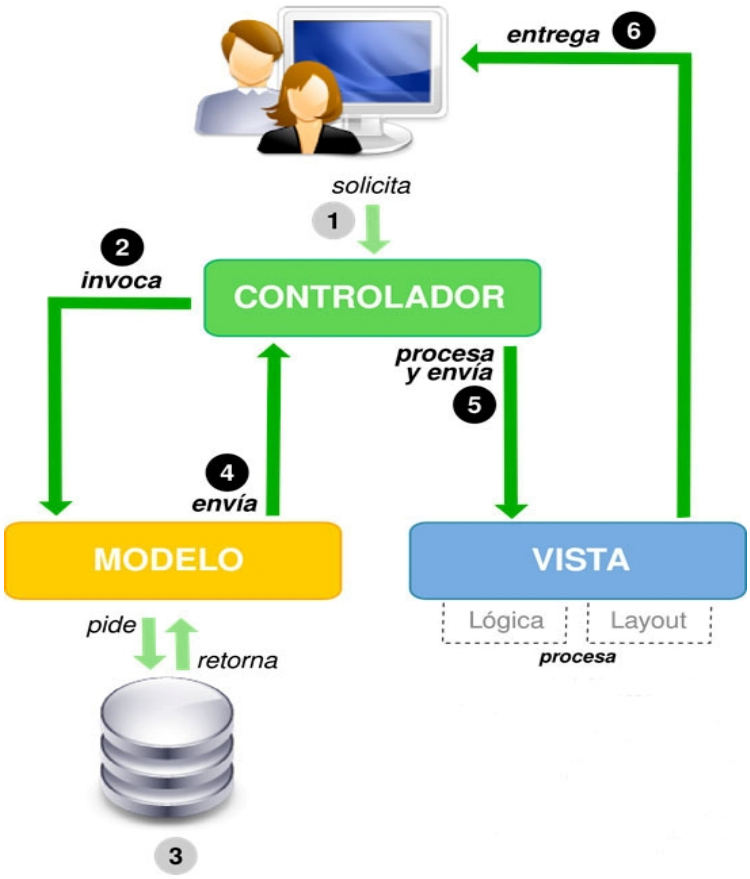

Figura 3: Esquema del patrón de diseño Modelo, Vista y Controlador (MVC).

Fuente: (Eslava Muñoz, 2018)

3) Patrón de diseño Precontrolador, Controlador, Modelo, vvista y Postcontrolador (MVC Extendible)

Según el autor (Chowdhury, 2016), nos dice que a diferencia del MVC (Modelo, Vista y Controlador) tradicional que tiene tres, este marco se divide en cinco partes principales que son el precontrolador, controlador, modelo, vista y postcontrolador, logrando como objetivo la funcionalidad de controlador sin problemas. Hablaremos de cada uno de sus componentes:

a. Precontrolador

Se encarga de procesar las solicitudes de los usuarios, primero acepta la solitud del usuario para obtener datos URI, crea la instancia de la clase de aplicación, esta clase llama al verificador de tipo solicitud para obtener el tipo de método, luego el tipo de método y URI generado envían al método de envío de los componentes de la ruta para activar el controlador con el método registrado, si la ruta coincide envía un filtro de solicitud pero sino coincide se envía un mensaje de "ruta no encontrada" al usuario; y por último el despachador de ruta analiza el "Controlador @ acción" a "Controlador" como clase de controlador y "acción" como el método de acción del "Controlador", este despachador ejecuta el método de "acción" de la clase "Controlador", antes de invocar el método registrado; el componente de ruta inyecta todas las dependencias de la clase "Controlador" a través del método constructor de la clase "Controlador" y esperando los resultados que provienen del controlador, por lo que cuando sale del controlador, el, la salida se envía a la clase de aplicación.

b. Controlador

Realiza tres pasos para realizar el proceso de una solitud de usuario filtrada que se proporciona desde el componente de ruta, se empieza al momento que el método de "acción" se comunica con modelo para tener los datos del modelo, luego este método proporciona los datos con un archivo de vista de plantilla al método de renderización de la clase de vista para generar la vista html y finalmente la clase del controlador devuelve la vista html al componente de ruta.

c. Modelo

Abarca todo tipo de concepto y la información que se encuentra relacionada con el negocio, este componente se comunica con la base de datos y recupera estos datos de la base de datos para que así llevar a cabo las necesidades de lógica comercial o necesidades del usuario. Solo realiza un paso este componente, pero si se tiene una lógica empresarial compleja, este componente puede realizar más pasos.

d. Vista

Es la capa de presentación que toma los datos suministrados por el controlador y genera resultados html en forma de una cadena, haciendo uso del archivo de plantilla de vista y los datos que se obtienen del controlador. Realiza un paso que es el método de renderización de la clase de vista que toma los datos que provienen del controlador, y así después generara la vista html haciendo uso de estos datos con la plantilla que se proporciona cuando se realiza la llamada al método.

e. Postcontrolador

Se encarga de aceptar la salida html que es generada de la clase de aplicación y la envía al componente de respuesta para imprimir esta salida html a la página web del usuario, para realizar este proceso primero la clase de aplicación distribuye todos los servicios que han sido registrados, luego esta clase envía vista html que proviene de los componentes de ruta al método de representación html del componente de respuesta y por último los métodos de representación html de los componentes de respuesta imprimen la vista html en el punto de entrada a través de la clase de aplicación y así el usuario puede observar la vista html en su página web.

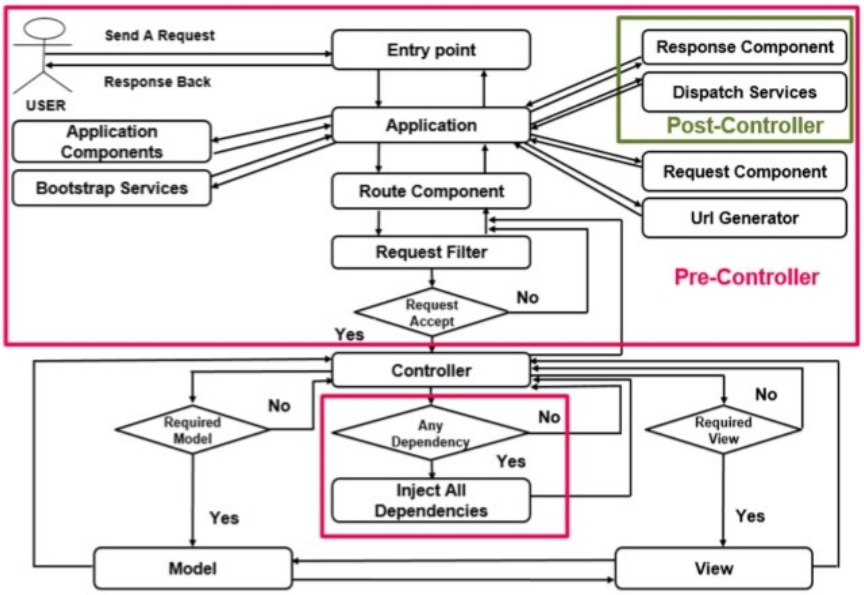

Figura 4: Esquema del patrón de diseño Precontrolador, Controlador, Modelo, Vista y Postcontrolador (MVC Extendible).

Fuente: (Chowdhury, 2016)

4) Patrón de diseño Modelo, Vista y Vista Modelo (MVVM)

Según el autor (Yehia Elshater, Patrick Martin , & Ehab Hassanein, 2015), este patrón de diseño es la extensión del patrón MVC; y se comunica primero el componente de la Vista con la Vista Modelo mediante enlace de datos y el enlace de comando. La VistaModelo lee los datos del Modelo y así traducir esos datos a la Vista para su visualización. Su esquema de este patrón es la siguiente.

a. Modelo (Model)

Este componente se encarga principalmente de empaquetar los datos e información, de tal manera que se puedan leer mediante el componente de VistaModelo.

b. Vista (View)

Se encarga de estar comunicado frecuentemente con el cliente y así mostrarle toda la información al cliente, de tal manera que los usuarios puedan obtener una experiencia favorable mediante páginas de comunicación. Estas páginas o ventanas pueden ser diseñadas con un simple acuerdo de vista según los requisitos del usuario. Este componente es la que más frecuentemente se modifica.

c. Vista Modelo (View Model)

La lógica de la visualización de la Vista se empaqueta en el Vista Modelo. Este componente se encarga de coordinar los controles de la Vista y los objetos en el Modelo; se podría decir que la Vista Modelo es el puente de la Vista con el Modelo, de tal manera que conecta todo el sistema y separa la Vista y el Modelo. Por lo tanto, este componente es principal para todo el rendimiento del sistema.

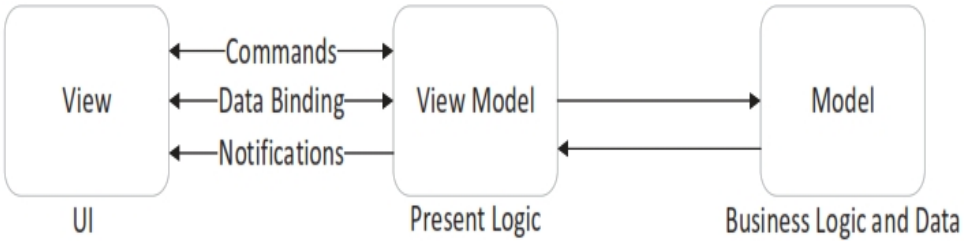

Figura 5: Esquema del patrón de diseño Modelo, Vista y Vista Modelo (MVVM).

Fuente: (Yehia Elshater, Patrick Martin , & Ehab Hassanein, 2015)

5) Patrón de Diseño Presentación, Abstracción y Control (PAC)

El autor (Phek Lan Thung, Chu Jian Ng, Swee Jing Thung, & Shahida Sulaiman , 2010), nos dice que este patrón de diseño tiene una estructura adecuada para los sistemas de software interactivo en la jerarquía de agentes cooperativos.

Su estructura es de tres componentes los cuales son la presentación, abstracción y control. Las subdivisiones lo dividen en aspectos de la interfaz de usuario del agente de su función principal y la comunicación con otros agentes. El agente actúa como un sistema interactivo que genera vistas como un conjunto de agentes de cooperación del cual se desarrolla de una forma independiente.

Para el desarrollo de aplicaciones interactivas los agentes son los adecuados.

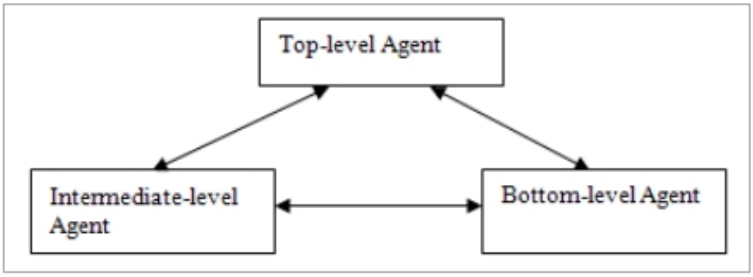

Figura 6: Esquema del patrón de diseño Presentación, Abstracción y Presentación (PAC).

Fuente: (Phek Lan Thung, Chu Jian Ng, Swee Jing Thung, & Shahida Sulaiman , 2010)

Esta figura muestra como colaboran entre si los componentes de este patrón de diseño, el cual consta de agentes separados lo cual permite que el desarrollo del modelo de datos y la interfaz de usuario para trabajar de manera independiente. Este patrón admite cambios dentro los componentes de presentación o abstracción de un agente de PAC que no afectan a los otros agentes en el sistema y también admite múltiples tareas debido a que los agentes del PAC se pueden distribuir fácilmente a diferentes subprocesos, procesos o maquinas. Esta función de multitarea nos facilita la aplicación es de multiusuario e interactivas lo cual nos permite que el PAC sea adecuado para aplicaciones multiusuario pero la desventaja es las largas cadenas de comunicación en la aplicación que pueden generar ineficiencia, ya que será complicado obtener el control mientras se mantienen demasiados agentes independientes. Es por eso que para superar las responsabilidades de PAC, hay algunos trabajos actuales que integran diferentes patrones de diseño en PAC para así ampliar sus funcionalidades.

II. MÉTODO DE INVESTIGACIÓN

En este procedimiento se realiza un estudio de revisión, de tal manera que nos ayude a recopilar trabajos de investigación en el que define a los patrones de diseño para hacer aplicados en aplicaciones web. Y de esta manera poder conocer el actual estado del conocimiento.

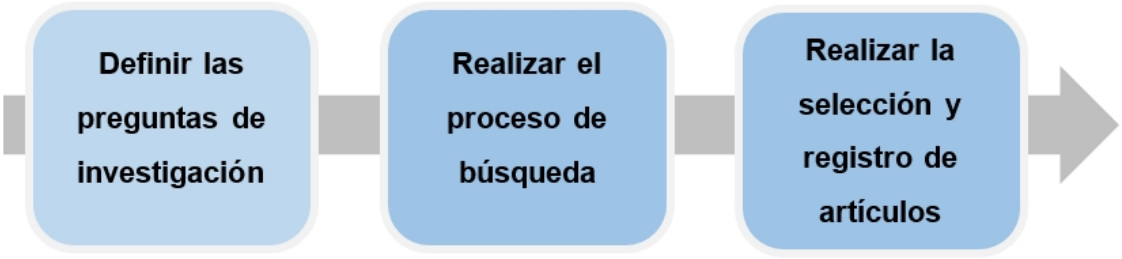

Figura 7: Método de Investigación.

Fuente: *Elaboración propia*

2.1. Preguntas de investigación

Tiene como objetivo brindar respuesta a las interrogantes; en este trabajo de investigación las preguntas son:

1. ¿Cuántos artículos sobre los patrones de diseño que son aplicados a aplicaciones web se han publicado entre el 2013 al 2018?

2. ¿Cuáles son los patrones de diseño que son aplicados a las aplicaciones web?

3. ¿Qué patrones de diseño se usan frecuentemente para ser aplicados en las aplicaciones web?

4. ¿Qué tipo de problemas abordan los estudios de patrones de diseño para ser aplicados en las aplicaciones web?

2.2. Proceso de búsqueda

Se realiza búsqueda de los artículos, que se llevan a cabo haciendo uso de base de datos de artículos científicos más reconocidos como son: SienceDirect, Scielo, IEEE Xplore. También se hace uso de herramientas de colaboración como ResearchGate y el buscador académico de Google Scholar.

2.3. Seleccionar y registrar artículos

Se realizó un análisis de cada artículo que se ha encontrado, para seleccionar cual es relevante o no para este trabajo de investigación. Al analizarlo se tomó en cuenta cada uno de sus elementos como: el título, el resumen, la introducción, los antecedentes, el método, los resultados y las conclusiones. Siendo un aspecto indispensable para ser seleccionado el tratar de los patrones de diseño que son aplicados a las aplicaciones web.

Los artículos seleccionados que cumplían con el criterio establecido fueron registrados considerando la siguiente información: País, año, autores, journal, tema que aborda, problema que enfrento, ¿Qué se hizo?, ¿Cómo se hizo?, ¿Qué resultados se obtuvo?, y a que conclusiones se llegó.

III. RESULTADOS

El análisis de resultados se realiza teniendo en consideración la revisión de los artículos científicos encontrados, que están basados en las preguntas planteadas en el capítulo anterior.

1. Pregunta Nº 1, ¿Cuantos artículos sobre los patrones de diseño que son aplicados a aplicaciones web se han publicado entre el 2013 al 2018?

La cantidad de artículos que hace referencia a los patrones de diseño que son aplicados a aplicaciones web es variada y a la vez un poco limitada en los últimos años. Esto a pesar de que al aplicar los patrones de diseño se considera como una característica importante en el desarrollo de software.

Tabla 1: *Cantidad de artículos por año*

Año	Cantidad
2013	**2**
2014	**1**
2015	**3**
2016	**1**

Fuente: *Elaboración propia*

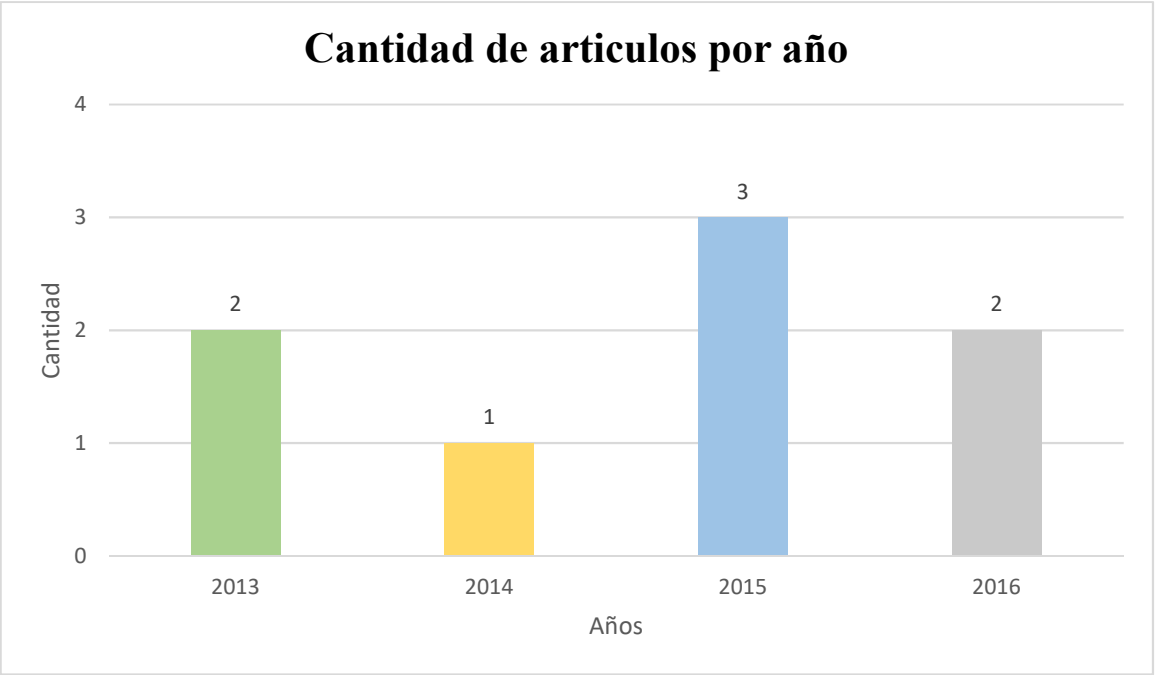

Figura 8: Cantidad de artículos por año.

Fuente: *Elaboración propia*

Se observa en esta figura la tendencia acerca de la cantidad de artículos sobre los patrones de diseño que son aplicados a las aplicaciones web en los últimos seis años. A pesar de que los patrones de diseño son importantes al desarrollar el software, no se han realizado muchas investigaciones al respecto.

2. **Pregunta Nº 2, ¿Cuáles son los patrones de diseño que son aplicados a las aplicaciones web?**

   Realizando un análisis del material literario se ha encontrado los patrones de diseño que se aplicaron en el desarrollo del software que utilizan los distintos autores en sus respectivos trabajos de investigación.

Tabla 2: *Patrones de diseño que son utilizados por autores*

Nº	Autores	Patrones de diseño
01	Alexander Katzmaier & Martin Hanneghan	Modelo, Vista y Controlador (MVC)
02	Kazan, Cantürk, & Bastan	Modelo, Vista y Presentador (MVP)
03	Phek Lan Thung, Chu Jian Ng, Swee Jing Thung, & Shahida Sulaiman	Presentacion, Abstraccion y Control (PAC)
04	Yehia Elshater, Patrick Martin , & Ehab Hassanein	Modelo, Vista y VistaModelo (MVVM)
05	Khaliluzzaman & Chowdhury	PreControlador, Controlador, Modelo, Vista y PostControlador (MVC Extendible)
06	Yehia Elshater, Patrick Martin , & Ehab Hassanein	Patrones orientado a objetos

Fuente: *Elaboración propia*

Como se puede apreciar en la tabla anterior, en la mayoría de los artículos los autores utilizan diferentes patrones de diseño o realizan agrupamiento entre los patrones como, por ejemplo: Modelo, vista y controlador (MVC); Modelo, Vista y Presentador (MVP); Modelo, Vista y VistaModelo (MVVM); PreControlador, Controlador, Modelo, Vista y PostControlador (MVC Extendible) y agrupar los patrones como el Modelo, vista y controlador (MVC), Modelo, Vista y Presentador (MVP) y proxy.

3. **Pregunta Nº 3, ¿Qué patrones de diseño se usan frecuentemente para ser aplicados en las aplicaciones web?**

Tabla 3: *Patrones de diseño con mayor frecuencia*

	Cantidad
Modelo, Vista y Controlador (MVC)	**3**
Modelo, Vista y Presentador (MVP)	**2**
Patrones orientado a objetos	**2**
Agrupamiento de patrones de diseño	**2**

Fuente: *Elaboración propia*

Como se puede observar en la tabla anterior, los patrones de diseño que se han usado con mayor frecuencia, en base a la revisión de los artículos encontrados, los métodos más utilizados son: Modelo, Vista y Controlador (MVC), Modelo, Vista y Presentador (MVP).

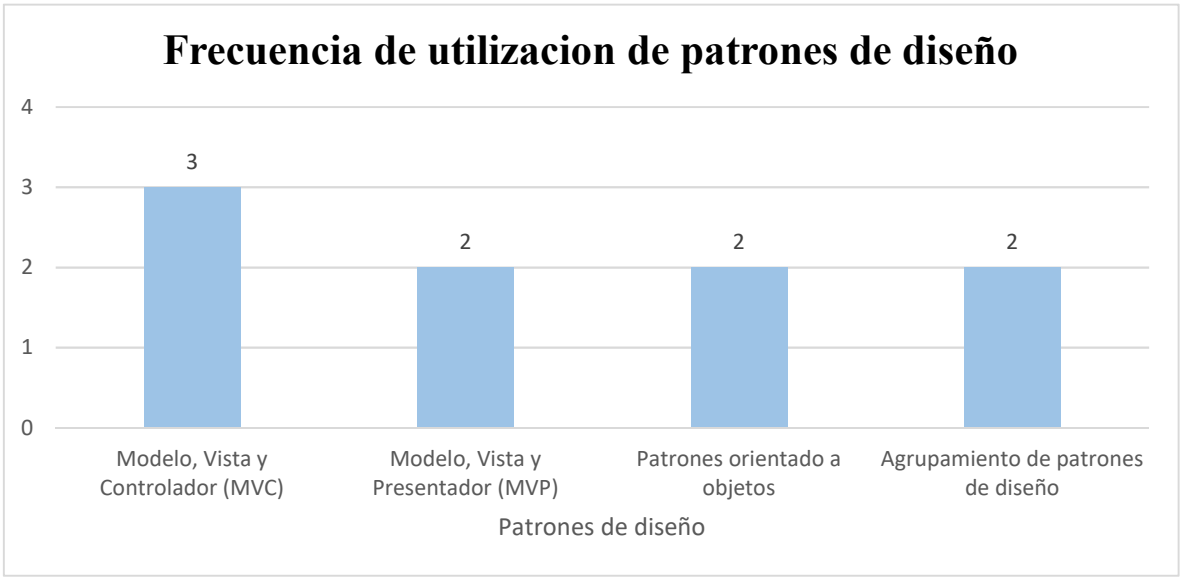

Figura 9: Frecuencia de utilización de patrones de diseño.

Fuente: *Elaboración propia*

4. **Pregunta Nº 4, ¿Qué tipo de problemas abordan los estudios de patrones de diseño para ser aplicados en las aplicaciones web?**

   Realizando un análisis de los trabajos de investigación encontrados, los distintos autores señalan que existen determinados problemas, siendo algunos de ellos los mostrados a continuación.

Tabla 4: *Problemas abordados por artículos*

Nº	Autores	Problemas
01	Alexander Katzmaier & Martin Hanneghan	Evaluó patrones de diseño demostrar cual es el patrón de diseño que más se usan en los marcos web y móvil
02	Kazan, Cantürk, & Bastan	Rediseñar un sistema que no estaba construida con patrones de diseño
03	Phek Lan Thung, Chu Jian Ng, Swee Jing Thung, & Shahida Sulaiman	Mejorar el rendimiento de una aplicación web mediante el uso de patrones de diseño
04	Yehia Elshater, Patrick Martin , & Ehab Hassanein	Implementación de patrones de diseño en Sistema de Ejecución de Manufactura (MES)
05	Khaliluzzaman & Chowdhury	Proponer un nuevo patrón de diseño y compararlo para demostrar que patrón ofrece un mejor rendimiento
06	Yehia Elshater, Patrick Martin , & Ehab Hassanein	Analizar y comparar patrones de diseño para medir quien ofrece un mejor rendimiento en los servicios web de búsqueda

Fuente: *Elaboración propia*

Según lo mostrado en la tabla anterior, uno de los problemas en común que enfrentan estos estudios es que existe una dificultad para determinar cuál es el patrón de diseño más adecuado que deben utilizar los diseñadores al desarrollar el software. Esto debido a la existencia de varios patrones de diseño, y a la poca evidencia de la literatura sobre cuál es el patrón de diseño a utilizar. Además, otro de los problemas es que no solo puedes seleccionar un patrón de diseño, sino que también puedes agrupar los patrones según el problema que deseas solucionar, por lo que debes saber a profundidad el problema. Finalmente se señalan otros problemas como: demasiada información sobre los patrones de diseño que se encuentran en sitios web, libros y más, lo que confunde a la comunidad de diseñadores de poder seleccionar el patrón adecuado.

IV. CONCLUSIONES

Basado en la revisión bibliográfica, se presenta en este documento una revisión actualizada de patrones de diseño. Cabe mencionar que el uso de patrones de diseño sirve como un modelo que ayuda a todos los forman el equipo de desarrollo de software a tener una idea clara de lo que harán al realizar el software, a tener mejor usabilidad de aplicaciones web, además en un posible mantenimiento del software es más rápido y eficiente realizarlo.

El rendimiento es un factor muy importante en los productos de software por lo que se debe cuidar celosamente este aspecto y los patrones de diseño ofrecen una gran alternativa para lograrlo así mismo es de mucha importancia que los equipos de desarrollo de software en gestiones adecuadamente el proceso de desarrollo siendo también los patrones de diseño de gran utilidad. Para realizar un diseño en el proceso de implementación existen diferentes patrones de diseño, que al tener varios patrones nos llevan a la confusión, por lo que se tiene que tener claro el problema que desea solucionar para seleccionar el patrón adecuado.

Según la revisión bibliográfica efectuada se ha podido determinar que los patrones de diseño que se han usado con mayor frecuencia son: Modelo, Vista y Controlador (MVC); y Modelo, Vista y Presentador (MVP).

Se requiere emplear los patrones de diseño revisados a través de nuevos casos de estudios, que nos permita generalizar ciertos resultados.

V. REFERENCIAS

Alexander Katzmaier , & Martin Hanneghan . (2013). Design pattern evaluation of mobile and web based application frameworks. *IEEE Xplore*.

Cabot Sagrera, J. (1 de Enero de 2013). Ingeniería del software.

Cardador Cabello, A. L. (7 de Julio de 2014). Implantación de aplicaciones web en entornos internet, intranet y extranet (MF0493_3).

Chaos García, D., Gómez Palomo, S. R., & Letón Molina, E. (1 de Enero de 2017). Introducción a la Informática básica.

Diéguez, F. R. (1 de Enero de 2014). Integración de componentes software en páginas web.

Eslava Muñoz, V. J. (1 de Enero de 2018). El nuevo PHP.

Ferrer Martínez, J. (1 de Enero de 2014). Implantación de aplicaciones Web.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Patrones de Diseño Elementos de software orientado a objetos reutilizable.* España.

Giovani Guizzo, Thelma Elita Colanzi, & Silvia Regina Vergilio. (2013). A Feasibility Analysis for the Application of Design Patterns in Search Based Product Line Design. *IEEE Xplore*.

Gómez Ruedas, J. (1 de Enero de 2016). Dirección y gestión de proyectos de tecnologías de la información en la empresa.

Hernández Diaz, L. R., André Ampuero, M., & Martínez Prieto, J. P. (1 de Enero de 2012). Un modelo para la implementación de la seguridad de una aplicación Web con el uso de la programación orientada a aspectos.

Honrubia López, F. (1 de Enero de 2014). Programación de aplicaciones para Iphone y Ipad.

Juan Carlos Moreno Pérez , & Arturo Francisco Ramos Pérez. (01 de Enero de 2014). Administracion de Software de un Sistema Informatico.

Kazan, E., Cantürk, M., & Bastan, M. (2015). Performance Analysis of a Software Developed with and without Design Patterns: A Case Study. *IEEE Xplore*.

Khaliluzzaman, M., & Chowdhury, I. I. (2016). Pre and Post Controller based MVC Architecture for Web Application . *IEEE Xplore*.

Montero, S., Zarraonadía, T., & Díaz, P. (1 de Enero de 2011). Patrones de diseño aplicados al desarrollo de Objetos Digitales Educativos (ODE).

Pérez Carvajal, R. J. (01 de Julio de 2014). Mantenimiento del software (UF1894).

Phek Lan Thung, Chu Jian Ng, Swee Jing Thung, & Shahida Sulaiman . (2010). Improving a Web Application Using Design Patterns: A Case Study. *IEEE Xplore*.

Prabowo, G., Hatma Suryotrisongko , & Aris Tjahyanto . (2018). A Tale of Two Development Approach: Empirical Study on The Maintainability and Modularity of Android Mobile Application with Anti-Pattern and Model-View-Presenter Design Pattern . *IEEE Xplore*.

Robledo, C., & Robledo, D. (1 de Enero de 2013). Google Web Toolkit.

Santi Caballé, Jose-Arturo Ortega, Josep-Maria Camps, Leonard Barolli, Elis Kulla, & Evjola Spaho. (2014). A Presentation Framework to Simplify the Development of Java EE Application Thin Clients . *IEEE Xplore*.

Thung, P. L., Ng, C. J., Thung, S. J., & Shahida Sulaiman . (2010). Improving a Web Application Using Design Patterns: A Case Study. *IEEE Xplore*.

XiaoLong Li, DaLiang Chang, HuiPen, XiaoYu Zhang, YuanXin Liu, & YaXian Yao. (2015). Application of MVVM Design Pattern in MES. *IEEE Xplore*.

Yang Zhang , & Yanjing Luo. (2010). An Architecture and Implement Model for Model-View-Presenter Pattern. *IEEE Xplore*.

Yehia Elshater, Patrick Martin , & Ehab Hassanein. (2015). Using Design Patterns to Improve Web Service Performance. *IEEE Xplore*.