Autores: Jorge Barrera Ibarra, Jaime Gálvez Silva, Ariel López Caris.
Correo : j.barreraibarra@gmail.com jaimegalvezsilva@gmail.com ariel.lopez.caris@gmail.com

## Resumen: Patrones de Diseño.

Son aquellas soluciones comunes en el desarrollo del software, además se ha demostrado que funcionan resolviendo problemas similares en el diseño, por lo tanto son reutilizables y se aplican en el caso que correspondan.

Dentro de un diseño encontramos muchos principios y reglas, los cuáles no bastan para el desarrollo, es por esto que los Ingenieros tienen estructuras y esquemas de solución que reutilizan según el contexto en que se encuentren diseñando.

El concepto de Patrón de diseño nace en 1979, el arquitecto Christopher Alexander publica el libro The Timeless Way of Building; donde propone una serie de buenas prácticas y esquemas para construir edificios de buena calidad. Dentro de lo que se destaca en el libro están las siguientes líneas "Cada patrón describe un problema que ocurre infinidad de veces en nuestro entorno, así como la solución al mismo, de tal modo que podemos utilizar esta solución un millón de veces más adelante sin tener que volver a pensarla otra vez."

Este concepto busca plasmar estructuras ya diseñadas y probadas que puedan solucionar problemáticas que habitualmente suceden en el desarrollo del Software y para esto debemos tener en cuenta los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios).

## Introducción.

El presente informe dará a conocer el concepto de Patrones de Diseños, utilizados por Ingenieros en el desarrollo del Software y veremos la necesidad de usarlos para obtener una buena calidad del producto.

También profundizaremos en las clasificaciones populares de patrones utilizados, donde encontramos los de tipo Creacionales, Estructurales y de Comportamiento.

Será muy interesante estudiar que un Patrón como estructura de diseño tiene aplicaciones semejantes dependiendo de la problemática que se esté tratando.

# Patrones de Diseño (Design Patterns)

Se define Patrón como una solución recurrente a un problema situado en un contexto diferente. En esta ocasión situaremos el concepto de patrón en el diseño del software.

Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos y se basan en la experiencia que se ha demostrado que funcionan.

Debemos conocer en primera instancia las ventajas del uso de patrones:

- Ante un problema reiterado ofrece una solución contrastada que lo resuelve.
- Describe el problema en forma sencilla.
- Describe el contexto en que ocurre.
- Describe los pasos a seguir.
- Describe los puntos fuertes y débiles de la solución.
- Describe otros patrones asociados.

## Tipos de Patrones.

**Patrones de creación**

Solucionan problemas de creación de instancias. Nos ayudan a encapsular y abstraer dicha creación.
Entre los Patrones de Creación destacan:

- **Factory Method**. Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.
- **Singleton**. Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.
- **Abstract Factory**. Proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.

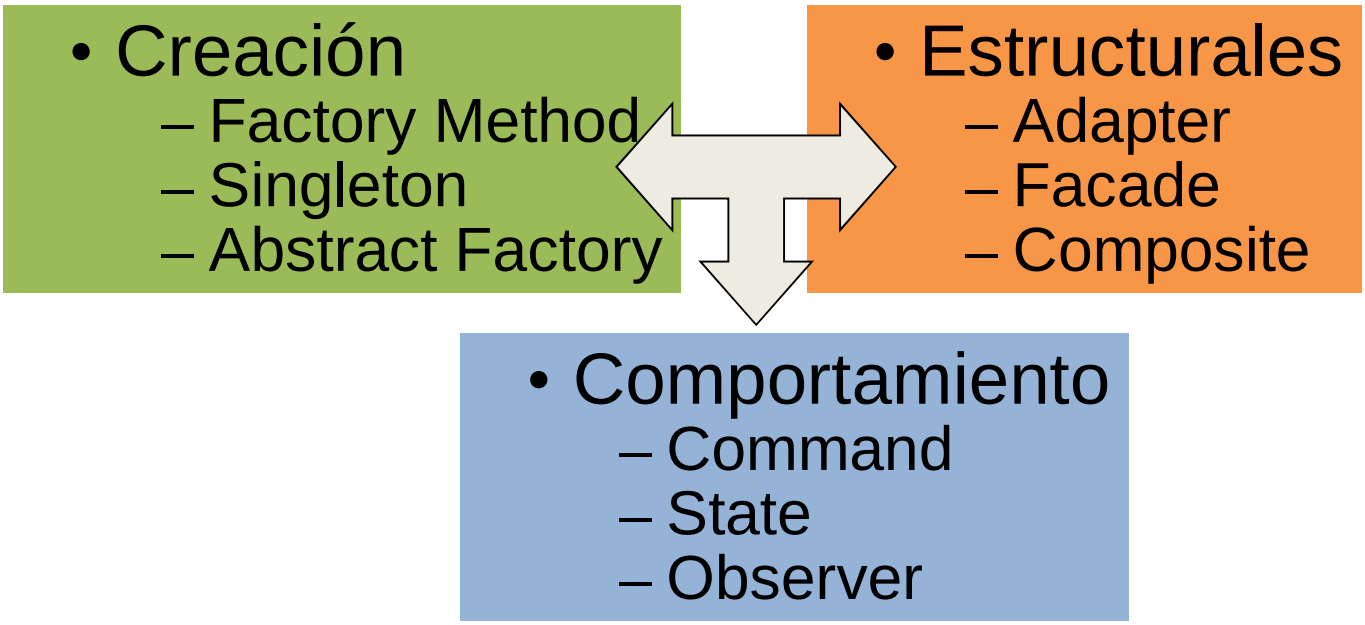
**Patrones Estructurales**

Solucionan problemas de composición (agregación) de clases y objetos.
Entre los Patrones Estructurales destacan:

- **Adapter**. Convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.
- **Facade**. Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema se más fácil de usar.
- **Composite**. Combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.

**Patrones de comportamiento**
Soluciones respecto a la interacción y responsabilidades entre clases y objetos, así como los algoritmos que encapsulan.
Entre los Patrones de Comportamiento destacan:

- **Command**. Encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer la operaciones.
- **State**. Permite que un objeto modifique su comportamiento cada vez que cambia su estado interno. Parecerá que cambia la clase del objeto.
- **Observer**. Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.



## Conclusiones.

La presente investigación tuvo como objetivo conocer el concepto de Patrón de diseño, las diferentes categorías y tipos, a la vez lo útil que resultan en el desarrollo de Software.

Luego saber qué ventajas tenemos al usar estos patrones, los cuales se encuentran testeados que al utilizarlos obtenemos la alta calidad esperada en las distintas aplicaciones que nos encontremos.

Por otro lado se definen cada tipo de patrones donde destacamos los más ejemplares tanto para los creacionales, estructurales y de comportamiento, que en su conjunto podremos reutilizarlos para lograr resolver problemáticas en el desarrollo de Software.

## Bibliografía.


http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php

http://msdn.microsoft.com/es-es/library/bb972240.aspx

http://www.proactiva-calidad.com/java/patrones/index.html

http://chuidiang.blogspot.com/2005/12/qu-son-los-patrones-de-diseo-el-patrn.html

http://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o