Estado del arte

Patrones de diseño versus antipatrones

Modelos de programación I

Docente

Miguel Alfonso Feijóo

Integrantes

Miguel Nicolás Díaz Vargas - 20202020054
José Miguel Londoño - 20202020064
Daniel Felipe Páez Mosquera - 20202020070

BOGOTÁ D.C

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Facultad de ingeniería

Ingeniería de sistemas

2022 – 1

- ***INTRODUCCIÓN***

La mayoría de nuestro mundo actualmente se encuentra asistido por herramientas tecnológicas, las cuales buscan por medio de innovación y adaptación, facilitar la vida de nosotros los seres humanos en nuestro día a día, ejemplos de esto que se está mencionando hay muchos fáciles de identificar en este mismo momento.

Un ejemplo muy cotidiano y fácil de comprender es como nos comunicamos ahora, actualmente disponemos de una gran cantidad de plataformas que nos permiten interactuar con muchas personas de diferentes formas sin necesidad de estar cerca de todas ellas, así podríamos realizar una interminable lista donde enumeramos las diferentes aplicaciones que nos permiten realizar distintas actividades o nos ayudan a tener un rendimiento más eficaz en nuestro trabajo, estudio, quehaceres caseros, etc. Pero a pesar de que utilizamos muchas de estas cosas no somos conscientes de lo que hay detrás de todo esto.

¿Qué hay detrás de todo esto?

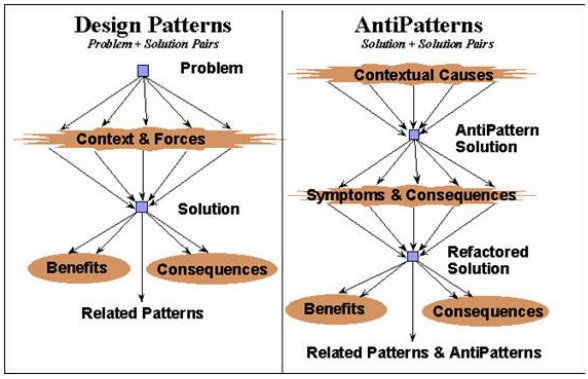
La respuesta a esta pregunta se puede dividir en varias partes, sin embargo, durante el desarrollo de este texto nos concentraremos en la programación y sus buenas prácticas, en el desarrollo de este proyecto utilizamos los llamados patrones de diseño, que son soluciones ya estructuradas para resolver problemas comunes.

En este orden de ideas, tenemos que los patrones son como una especie de plantilla que nosotros como programadores tenemos para resolver problemas que son recurrentes, de esta forma se evita crear o inventar algo que ya se había hecho.

Así como existen los patrones de diseño también existen los conocidos “Anti patrones”, estos hacen parte de lo que se conoce como malas prácticas de programación, ya que son poco prácticos y no implementan los principios básicos que permiten tener un código estructurado, de tal

forma que sea fácil de entender para cualquier persona con el conocimiento adecuado desee conocerlo, modificarlo o reordenarlo.

Para que sea más clara la diferencia entre ambos, haremos una confrontación entre ambos y conceptualizaremos tanto los patrones como los anti patrones.



PATRONES VS ANTI PATRONES

Para iniciar esta confrontación vamos a explicar un poco más a fondo que es la alta cohesión y bajo acoplamiento de forma que se esclarezca porque son tan importantes al momento de desarrollar un proyecto de programación como este.

1. Alta cohesión

   La alta cohesión consiste en que una clase o método solo desempeña la tarea para la cual fue creado, de forma que lo que hace no involucra ninguna otra acción ni intervención de otra clase u método.

2. Bajo acoplamiento

   El bajo acoplamiento lo vemos cuando es evidente la independencia que existe entre los componentes del programa, ayudando a evitar un efecto domino si se llega a presentar un error, ya que, al tener dependencias entre las clases y métodos, un solo error podría causar que las otras clases dependientes de la que tuvo el error se vean afectadas de forma que se volvería ineficaz la modificación de un código con alto acoplamiento.

Teniendo claro estos dos conceptos podemos proceder a confrontar los patrones y anti patrones, principalmente para explicar porque su uso es de carácter retorico al

momento de desempeñarnos en un ambiente profesional o en el desarrollo de algún proyecto académico de forma avanzada.

- **Patrones de diseño**

  Tenemos diferentes tipos de patrones, estos se dividen en distintas familias, puesto que tienen diferentes aplicabilidades y no todos son adecuados para resolver todos los problemas, las familias son las siguientes.

  1. Creacionales:
     Estos patrones creacionales nos ofrecen una amplia variedad de mecanismos para crear objetos, facilitando la reutilización de código y su versatilidad.

  2. Estructurales:
     Los estructurales consisten en cómo podemos organizar las clases y objetos que hacen parte de una estructura más grandes, de tal forma que se mantenga su flexibilidad y eficiencia.

  3. Comportamentales:
     Estos son muy importantes en el momento que entramos a hablar de la responsabilidad de una clase u objeto, ya que estos patrones trabajan con algoritmos que nos facilitan la asignación de responsabilidades entre objetos.

  Ahora que conocemos las diferentes familias de los patrones, podemos inferir para qué sirven y cómo podemos utilizar cada uno de los que se encuentra dentro de estos grupos, seguramente también sería sencillo identificar los casos en los que no deberíamos aplicar determinados patrones.

  La buena práctica evidencia como nosotros

somos capaces de orientar, ordenar y desarrollar una solución apropiada a cualquier necesidad que llegue a surgir al momento de desarrollar cualquier software, teniendo en cuenta que en la mayoría de ocasiones no trabajaremos solos, por lo tanto, debemos trabajar de tal forma que todos los integrantes del proyecto se vean en la capacidad de entender, modificar y cambiar nuestro código, ya que todos seguimos los mismos lineamientos de desarrollo, permitiendo una buena fluidez de trabajo.

- **Anti patrones**

Por otro lado, los anti patrones son lo que debemos evitar a toda costa para no tener deficiencias en el desarrollo de software, ya que suelen verse como soluciones muy complejas e ineficaces para problemas sencillos, sin embargo, su estudio es de suma importancia, puesto que nos permite identificarlos fácilmente para corregir su implementación.

Estos también se clasifican de diferentes formas las cuales son:

1. The Blob:
   En este se encuentran las clases “toderas” son aquellas que poseen muchos métodos y atributos, además realizan muchas operaciones, rompiendo el principio de alta cohesión y bajo acoplamiento.

2. Lava Flow:
   La principal característica de este es el código no óptimo o inservible, lo que se conoce como código muerto.

3. Poltergeists:
   Se conoce porque hay clases que no tienen funciones evidentes y no se sabe muy bien cuál es su función dentro del programa.

4. Golden Hammer:
   Es la implementación innecesaria de los patrones como si todo tuviera que

encajar sin importar que no sea la herramienta adecuada.

5. Spaghetti code:
   Es el más conocido, consiste en la implementación de muchos condicionales anidados, provocando bucles muy grnades y propensos a fallas con efecto domino.

CONCLUSIÓN

Es necesario tener conocimiento sobre las malas y las buenas prácticas, ya que esto nos puede ayudar a evitar las malas y hacer un mejor uso de las buenas, también es crucial que el programador tenga una capacidad analítica destacable, de forma que pueda implementar correctamente los patrones al momento de desarrollar software.

Por último, es importante recalcar que lo que hagamos debemos hacerlo pensando en que hay otras personas que trabajan en lo mismo y por respeto a ellos debemos trabajar de forma que todos nos entendamos y evitemos caer en las malas prácticas de programación.

REFERENCIAS

[1] Medium.com. [Online]. Available: https://medium.com/all-you-need-is-clean-code/patrones-de-diseño-b7a99b8525e. [Accessed: 14-Aug-2022].

[2] “Qué son los patrones de diseño,” Openwebinars.net, 05-Nov-2019. [Online]. Available: https://openwebinars.net/blog/que-son-los-patrones-de-diseno/. [Accessed: 14-Aug-2022].

[3] T. Tech, “Alta cohesión y bajo acoplamiento en diseño de software,” Tech And Solve, 01-Feb-2019. [Online]. Available: https://techandsolve.com/alta-c

ohesion-y-bajo-acoplamiento-en-diseno-de-software/. [Accessed: 15-Aug-2022].

[4] F. Wikipedia, Patrones de Diseno: Patron de Diseno, Command, Singleton, Modelo Vista Controlador, Decorator, State, Lenguaje de Patron, Grasp, Bridge. Books LLC, Wiki Series, 2011.

[5] “Anti-patrones: la mejor forma de hacer un pésimo sistema de software,” SG Buzz. [Online]. Available: https://sg.com.mx/revista/11/anti-patrones-la-mejor-forma-hacer-un-pesimo-sistema-software. [Accessed: 15-Aug-2022].

[6] “Especificación de los Antipatrones de diseño de Desarrollo,” Juntadeandalucia.es. [Online]. Available: https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/204. [Accessed: 15-Aug-2022].

[7] V. M. C. de Rubén, V. M. C. de Antonio José, and V. M. C. de Daniel, TDD. 2019.

[8] P. P. Originals, “Malas Prácticas en la Programación,” Blogspot.com. [Online]. Available: http://moztrodev.blogspot.com/2013/03/malas-practicas-en-la-programacion.html. [Accessed: 15-Aug-2022].

[9] Ehu.es. [Online]. Available: http://siul02.si.ehu.es/~alfredo/iso/06Patrones.pdf. [Accessed: 15-Aug-2022].

[10] M. M. Canelo, “Qué son los Patrones de Diseño de software / Design Patterns,” Profile Software Services, 24-Jun-2020. [Online]. Available: https://profile.es/blog/patrones-de-diseno-de-software/. [Accessed: 15-Aug-2022].

[11] Autentia.com. [Online]. Available:

https://www.autentia.com/wp-content/uploads/2021/05/Fichas_SoftwareDesign.pdf. [Accessed: 15-Aug-2022].

[12] “¿Qué son los patrones de diseño? Conoce para qué te sirven,” Platzi, 06-Jul-2015. [Online]. Available: https://platzi.com/blog/patrones-de-diseno/. [Accessed: 15-Aug-2022].

[13] “15 Malas prácticas más comunes en la programación,” Com.mx, 25-Mar-2018. .

[14] “Antipatrón de diseño,” Los diccionarios y las enciclopedias sobre el Académico. [Online]. Available: https://es-academic.com/dic.nsf/eswiki/90215. [Accessed: 15-Aug-2022].

[15] “Patrones de Diseño Software,” Informaticapc.com. [Online]. Available: https://informaticapc.com/patrones-de-diseno/. [Accessed: 15-Aug-2022]