Universidad Autónoma de Nuevo León.

Facultad de Ciencias Físico Matemática.

Licenciatura en Seguridad en Tecnologías de Información

Profesor: Lic. Miguel Angel Salazar Santillán

Alumna: Sofía Esmeralda Pecina Rojas.

Matricula: 1667093

Unidad de aprendizaje: Diseño orientado a objetos

tema: Antipatrones

¿Qué es?

Los antipatrones son soluciones negativas que presentan más problemas que los que solucionan. Son una extensión natural a los patrones de diseño. Comprender los antipatrones provee el conocimiento para intentar evitarlos o recuperarse de ellos. El estudio de los antipatrones permite conocer los errores más comunes relacionados con la industria del software.

Hay 3 tipos de antipatrones:

Desarrollo de Software AntiPatrones

Un objetivo clave del desarrollo de AntiPatrones es describir formas útiles de refactorización de software. La refactorización de software es una forma de modificación de código, utilizada para mejorar la estructura del software en apoyo de la subsiguiente extensión y mantenimiento a largo plazo. En la mayoría de los casos, el objetivo es transformar el código sin afectar la corrección.

Arquitectura de software AntiPatrones

Arquitectura AntiPatrones, se centran en la estructura a nivel de sistema y de nivel de empresa de aplicaciones y componentes.

Gestión de proyectos de software AntiPatrones

En la profesión de ingeniería moderna, más de la mitad del trabajo implica la comunicación humana y la resolución de problemas de personas. La administración de AntiPatrones, identifica algunos de los escenarios clave en los que estos problemas son destructivos para los procesos de software.

Ejemplos:

Lava Flow

es comúnmente encontrado en sistemas que fueron originalmente una investigación, pero terminaron en el producto final. Se caracteriza porque al igual que la lava, "fluye" de las versiones anteriores de desarrollo esparcidos a lo largo del código, el cual ahora se ha endurecido como el basalto, inamovible, generalmente una masa inútil de código de la que nadie puede recordar mucho.

Este es el resultado de los primeros tiempos de desarrollo cuando, mientras se investigaba, los desarrolladores intentaban muchas formas de cumplir con los

requerimientos, típicamente en una carrera para algún tipo de demostración, echando al viento las buenas prácticas de diseño y sacrificando la documentación.

Functional decomposition

Este antipatrón, es bueno en un entorno de desarrollo procedural. Incluso resulta útil para comprender la naturaleza modular de una aplicación a gran escala. Una causa es cuando la falta de aplicación de la arquitectura. Cuando los implementadores no tienen ni idea sobre la orientación a objetos, no importa lo bien diseñada que esté la arquitectura; ellos simplemente no entienden lo que están haciendo. Y sin la supervisión adecuada, por lo general, encontrarán la manera de esquivar algo utilizando las técnicas que ellos conocen.

Poltergeists

Se trata de clases que juegan roles y responsabilidades limitadas dentro del sistema; por lo tanto, su ciclo de vida efectivo es bastante breve. Polstergeist desordena el diseño del software, creando abstracciones innecesarias; son excesivamente complejas, difíciles de comprender y difíciles de mantener. Este antipatrón es típico en casos donde los diseñadores están familiarizados con el proceso de modelado, pero son nuevos en definición de diseños orientados a objetos y definición de arquitecturas.

Golden Hammer

Un martillo de oro es una tecnología o concepto familiar aplicado obsesivamente a muchos problemas de software. La solución consiste en ampliar el conocimiento de los desarrolladores a través de grupos de educación, formación y estudio de libros para exponer a los desarrolladores a tecnologías y enfoques alternativos.

Spaghetti code

La estructura de software ad hoc hace que sea difícil extender y optimizar el código. La refactorización de código frecuente puede mejorar la estructura del software, el mantenimiento del software de soporte y permitir el desarrollo iterativo.