INGENIERA DE SOFTWARE 2

Medicion de complejidad:

Tamaño: mientras mas grande, mas complejo es  
una forma de medir el tamaño es la cantidad de instrucciones, mientras mas instrucciones tiene, mas complejo es.  
una app de tipo financiera es mucho menos compleja que una app de tipo industrial  
una línea de código varia dependiendo la complejidad en los tipos de lenguaje de programación   
Una empresa con pocas herramientas, puede realizar su propia medición

Halstead es un programa de medición midiendo el n de operadores   
las métricas son vocabulario, largo, volumen, complejidad, esfuerzo, tiempo y defectos.  
su misión es la de responder a consideración teóricas de un programa

McCabe: El programa se hace mas complejo si encuentra mas caminos por tomar  
Es una medida de tipo secundaria con resultados particularmente buenos

Relaciones sistémicas: el costo esta impulsado x el esfuerzo  
El costo es una relación lineal del esfuerzo de los profesionales.  
Costo y esfuerzo son indistintos.

Con mediciones puedo hacer estimaciones de cuanto va a ser el costo total del proyecto, y permite saber el presupuesto final del mismo.  
  
La estimación mas complicada es la genesis del proyecto para hacerlo (define tiempo, personal y $$ en el proyecto)

Vista de diseño, consiste en iterar entre las vistas de diseño conceptual y diseño técnico.  
  
Acoplamiento: no hay un indid¿ce de acoplamiento, es un concepto general, se transforma en criterios de información.  
el acoplamiento produce una modificación menor y hace explotar otra porque esta acoplada, permite técnicas de programación defensivas, en la que el modulo no sabe quien lo llama per sabe que lo llaman de una interfaz defensiva.

Tipos de acoplamiento:   
de datos a través de un repositorio común  
estructuras: no tengo estructuras de tipo formal, sino definida,  
control: como llamo a un modulo, como retorna, que argumentos le paso  
externo: programa modificando al código ejecutable  
común: estructura global  
contenido: el peor de todos, muchos lenguajes no lo permiten, es un programa que se modifica a sí mismo,

Cohesion: tipos  
función: encapsula todo  
Secuencia: cascada de funciones  
Comunicación: similar entradas y salidas  
Procedimiento: aséctp tecmoco marshalling   
temporal: ocurren al igual tiempo  
Logica: consecuencia logica

27/03/2024

Tipos de arquitectura  
orientada al rest: explora el código y se encapsula de forma que por cada agregado no se necesita testear la totalidad (flexibilidad)  
orientada al dominio: la construcción de código comienza por establecer un modelo de reglas de negocio que lo forman (robustez)  
orientada a los microservicios:   
  
encapsulado  
la cohesion (alta) y acoplamiento (bajo)

Polimorfismo:   
permite abstraer la naturaleza del argumento y concentrarse en el método  
se usa en patrones ya que permite usar patrones que son distintos entre si

Arquitectura en capas:  
3 grandes grupos  
como presento, es la piel del aplicativo, una vez que se completo un ciclo se destruye,   
reglas de dominio, es el conocimiento de como funciona el aplicativo, es la lógica relacionada a los casos de uso, no se guarda registro  
de datos, una vvez que se aceptan los datos se los dan a uno que da de alta al cliente, los datos se guardan en una tercera conjunto de funciones que es la capa de datos, se dan la persistencia de los negocios, se guardan registro.

03/04

Patrones de creación:   
Singleton: patron que nos permite asegurarnos que una clase tenga una única instancia y proporciona un punto de acceso global a esa instancia.  
Es una clase que tiene dos elementos.  
su problema es que tiene poca cohesion (usa dos funciones en una clase)

Factory: patron que proporciona una interfaz para crear objetos en una superclase y permite a las subclases alterar tipos de objetos que se crearan.

8/04/2024

Tipos de patrones de diseño:

Prototype: permite agregar un método obteniendo una copia sin preocuparme de como está implementado.  
te permite poder repetir un código prefabricado y de cierta complejidad.  
  
  
TIPOS DE PATRONES ESTRUCTURALES

De creación: abstraen la forma en la que se crean los objetos, permiten tratar las clases de forma genérica dejando para después la decisión de que clases crear.

Estructurales: son fundamentales en la relación del uso entre los objetos, y están determinadas por las interfaces que soportan los objetos.

De comportamiento: estudian relaciones entre llamadas de los diferentes objetos, ligados con la dimensión temporal.

Adapter: relacionada a tareas con integración de sistemas, un lado del adaptador ve de forma distinta a lo que tiene que hacer el código adaptador, se encarg de hacer las conversiones para ambos lados, lo usamos cuando tenemos problemas de adaptación.

Proxy: lo que hace es actuar en nombre de una interfaz, es como un abogado, habla por vos

Bridge:

Composite: le digo al patron que es un generador de arboles con clases de cualq índole, hace estructura los punteros a los objetos en una estructura interna de árbol, su uso es limitado a que todos los objetos que están en el árbol tengan una interfaz común

Facade: intenta simplificar un acceso complejo tratando de aislar la complejidad del código, establece

Flyweight: permite reducir requerimientos de memoria, es como una caja fuerte donde det estados se ponen en un solo lado para muchos objetos (es una verga)