UNIVERSIDAD DE BUENOS AIRES
Facultad de Ingeniería
Departamento de Computación

7510 Técnicas de Diseño

Este es un modesto aporte para los alumnos de la fácultad de ingeniería de la UBA de las carreras de licenciatura en análsis de sistemas e ingeniería informática. De ninguna manéra pretende ser una guía de estudio, ni remplaza las clases presenciales, el material oficial de la catedra esta disponible en el web site de la máteria.
http://materias.fi.uba.ar/7510/

Autor: Isaac Edgar Camacho Ocampo
Carrera: Licenciatura en Análisis de sistemas

Buenos Aires, 2019

Contents

Chapter 1

Introducción

Toda la complejidad de la ingeniería de software surge cuando quien necesita el software ya no es quien lo construye, es decir inicialmente una persona calificada necesitaba usar la computadora para hacer algunos cálculos y demás cuestiones, entonces era el mismo quien desarrollaba el software y era el mismo quien lo usaba.

Hoy el mundo tiende a la automatización y todos los negocios están en este proceso, en general nos vamos a encontrar en la industria con la problemática de resolver algún problema, y bajo la POO el problema debe ser modelado por entidades es decir conceptos extraídos del dominio del problema.

Nosotros como analistas no somos expertos en el negocio de quien nos contrata, sin embargo debemos construir software que resuelva problemas de ese modelo de negocio, lo primero que debemos hacer es entender el negocio es decir entender las reglas de negocio que lo gobiernan.

Por ejemplo en el caso de un banco, este tendrá una linea de créditos y esta debe tener una serie de requisitos que los interesados deben cumplir, El dueño de un restaurante debe tener reglas de negocio que dirijan la política de contratación de personal, un Supermercado tendrá reglas que determinen las disposiciones de las mercaderías en góndolas, la rotación de las mismas etc.

Así todo negocio es diferente y cuando tenemos que construir software para un negocio determinado nos enfrentamos a varios problemas.

1.1 Conocimientos previos

Aqui debo poner lo que el alumno deberia conocer

Chapter 2

Modelo de dominio

Una vez que terminamos la fase de análisis de requerimientos ya sea con casos de uso o historias de usuarios, nos enfrentamos con una tarea compleja y poco clara respecto del siguiente paso en el trabajo ¿como convertir esos requisitos en el sistema funcionando es decir las lineas de código?.

Sabemos que el código surgió de un diseño previo. Pero, ¿Cómo surgió el diseño? Es decir como llego desde el modelo de casos de uso al diagrama de clases por ejemplo? La respuesta es un modelo de análisis, hay que hacer un modelo de análisis para continuar con el trabajo.

2.1 Patrónes de análisis (jill Nicola)

MODELO DE ANÁLISIS • Objetivo: entender en detalle el negocio y sus reglas • Mecanismo utilizado: Patrones de Análisis

MODELO DE DISEÑO • Objetivo: implementar una solución al problema planteado en el análisis más las restricciones impuestas por los requerimientos no funcionales. • Mecanismo utilizado: Patrones de Diseño

Grady booch dice que tenemos como herramientas tres mecanismos. • PARTICIÓN: Una funcionalidad compleja dividirla en varias • ABSTRACCIÓN: • ESTABLECIMIENTO DE JERARQUÍAS

Chapter 3

Criterios de Buen Diseño SOLID

3.1 ¿Qué es el criterio?

El criterio es nuestra herramienta para establecer diferencias y tomar decisiones, el significado de criterio es juicio o la capacidad de las personas para emitir un juicio respecto a algo o alguien de acuerdo a la información con la que cuenta. La etimología de criterio proviene del griego kriterion que significa **"norma para conocer la verdad"**.

Supongamos que queremos construir una casa, ¿como empezamos? compramos materiales, llamamos a unos amigos y les decimos !!Comencemos haber que sale!!! esto seguramente es no tener criterio.

3.2 Diseño

En el artículo de Robert C. Martín **The Dependency Inversion Principle** deja en claro lo que es un mal diseño y compara el diseño de software con un trozo de carne que se va descomponiendo conforme pasa el tiempo.

Lo que dice el tio bob es que comparar la ingeniería del Software con la ingeniería civil es una mala comparación porque cada proyecto civil se diseña para no cambiar y la ingeniería del Software trata de modelar una porción de la realidad en la computadora, si queremos que la porción de la realidad modelada sea virtualmente parecida al mundo real, debe necesariamente cambiar, porque la naturaleza del mundo es cambiante, desde ese punto de vista la comparación con el trozo de carne que se va degradando es una buena metáfora.

3.2.1 ¿Que es un mal diseño?

Estamos en presencia de un mal diseño cuando esté tiene tres cualidades.

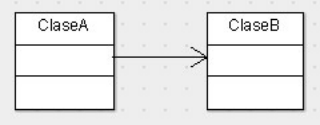
- El diseño es rígido.
- El diseño es frágil .
- Y por último el diseño es inmóvil .

Es decir las tres características de un mal diseño es la rigidez la fragilidad y la inmovilidad

3.3 Diseño

3.4 Dependencia

En el campo del software una dependencia es una aplicación o una biblioteca requerida por otro programa para poder funcionar correctamente. Por ello se dice que dicho programa depende de tal aplicación o biblioteca.
En UML se dice la clase A depende de la clase B.



En el siguiente programa en C++ existe una dependencia con la libreria iostream.

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

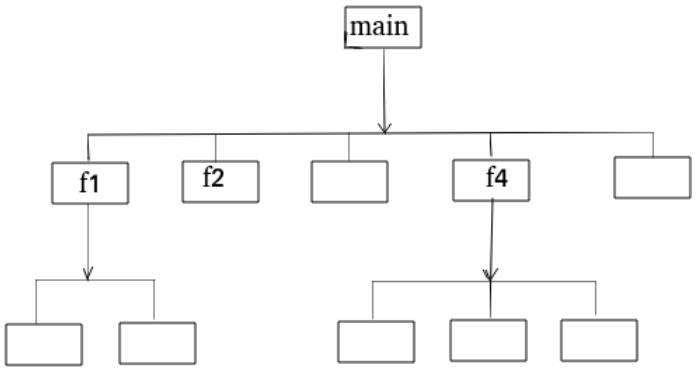
3.4.1 ¿Por qué preocuparse por las dependencias?

Una dependencia es un riesgo. Por ejemplo, si mi sistema requiere la instalación de un Java Runtime Environment (JRE) y no está instalado, mi sistema no funcionará, Si un prográma en lenguaje C utiliza la libreria ”stdio.h” eso tambien es una dependencia y si esa libreria no esta presente ni siquiera va a compilar.
Las dependencias representan riesgo y manejar ese riesgo tiene algún costo que hoy se resuelve a través de la experiencia, prueba y error, o la sabiduría colectiva de un equipo.

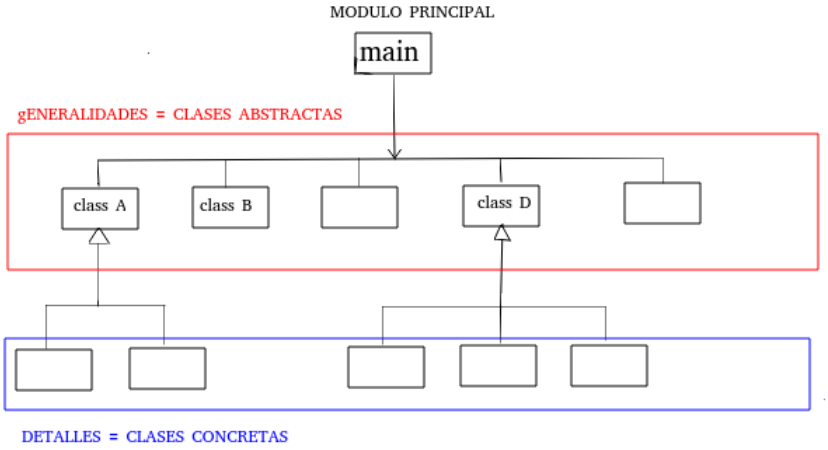
3.4.2 ¿Por que se dice inversion de las dependencias?

En el análisis y diseño estructurados usabamos la regla de **divide y vencerás**, comenzamos con un problema de alto nivel y lo dividimos en partes más pequeñas. Para cualquiera de esas partes más pequeñas que todavía son ”demasiado grandes”, continuamos separándolas. El concepto / requisito / problema de alto nivel se divide en partes cada vez más pequeñas. El diseño de alto nivel se describe en términos de estas partes cada vez más pequeñas y, por lo tanto, depende directamente de las partes más pequeñas y más detalladas. Esto también se conoce como diseño de arriba hacia abajo.

Podemos ver como el módulo principal main se divide en mas funciones
de menor nivel y a su vez estas se vuelven a dividir, de esta manera podemos ver que
los modulos mas generales dependen de los detalles de implementacion.



En el Diseño orientado a objetos tanto los modulos de alto nivel,
como los de bajo nivel deben depender de las abstracciones.


MODULO PRINCIPAL
main
gENERALIDADES = CLASES ABSTRACTAS
class A
class B
class D
DETALLES = CLASES CONCRETAS

Chapter 4

Paradígmas de Programación

4.1 Paradigma Funcional

Chapter 5

Diseño de código

Chapter 6

Arquitectura de Software

6.1 Introducción

6.1.1 Patrones de arquitectura

Chapter 7

Temas Relacionados

7.1 Docker

7.2 API Rest

7.3 TDD

Chapter 8

Patrones de Diseño

8.1 Creacioanles

8.2 De Comportamiento

Chapter 9

Métricas

CONCEPTOS PREVIOS

1. **Eficiencia:** Producir mas con menos recursos o productividad.
2. **Eficacia:** Lograr el resultado aunque se consuman muchos recursos.
3. textbfCosto de oportunidad: Es el costo de producir una unidad de un bien.
4. **Contribucion marginal:** Ganancia neta.

Chapter 10

Conclusiones

CONCEPTOS PREVIOS

1. **Eficiencia:** Producir mas con menos recursos o productividad.
2. **Eficacia:** Lograr el resultado aunque se consuman muchos recursos.
3. textbfCosto de oportunidad: Es el costo de producir una unidad de un bien.
4. **Contribucion marginal:** Ganancia neta.

¿Que es modelizar? Es hacer una simplificacion de la realidad y nosotros trabajamos con esa simplificacion ya que la realidad es muy compleja.
¿Para que hacer un modelo?

- **Economia de recursos:** Al igual que en la ingenieria civil cuando se hacen planos para representar una obra (ya que no seria lógico hacer el edificio y ante un error rehacerlo), cuando modelamos un problema usando programacion lineal y dada la escazes de recursos, es mas eficiente trabajar sobre modelos.
- **Eficiencia**: de nuevo so no tengo recursos limitantes entonces trabajo sobre la realidad y no modelo nada.
- **simplicidad**: puedo mediante abstracción lograr un modelo mas sencillo y eliminar la complejidad inherente del problema.
- **En resumen es mejor que hacer multiples ensayos.**

Los modélos se aplican a problemas de desición y este existe cuando existen formas alternativas de actuar, con distintos resultados y diferentes eficiencias para lograr el objetivo es decir existen dudas respecto del curso alternativo a utilizar.

Elementos de un modelo

Hipótesis y supuestos: Para simplificar el modelo se delimita el sistema en estudio a través de las hipótesis y supuestos simplificativos. Así se comienza a transformar el sistema físico en un modelo simbólico. Las hipótesis deben ser probadas científicamente. Los supuestos son hipótesis que no pueden probarse.
Ejemplos: Si estamos modelando una panaderia, y el recurso agua no es limitante y por otro no se dice nada de la venta de lo producido podemos agregar las hipotesis

1. *Hay agua suficiente para todos los procesos*

2. *Se vende todo lo que se produce*

Objetivo: Mide la eficiencia de nuestro sistema y lo que buscamos es hallar la merjo solucion. El objetivo surge como respuesta a tres preguntas:
¿Qué hacer? es decir que es lo que queremos determinar.
¿Cuándo? (período de tiempo) puede ser un mes o año o un periodo t si no se especifica.
¿Para qué? para maximizar ganancias, o minimizar costos, nunca ambos a la vez.

Actividad Proceso unitario que se realiza en el sistema físico caracterizado por consumir recursos y/o generar un resultado económico y/o indicar un estado, por ejemplo producir un bien o indicar si se finalizó un proceso.

Variables Son las que miden o indican el estado de una actividad. Las que miden pueden ser continuas o enteras. Las que indican son, generalmente, variables (0,1) o bivalentes