Clase 13 - DA2 Teórico
08/11/2021

Hoy:
- Prueba de actuación en clase
- Principios a nivel de paquete (acoplamiento)
- Métricas

Reglas para la prueba

- Duración **20 minutos**
- Tener **lápiz y papel** a mano o usar herramienta de **modelado** UML
- Son dos preguntas de **desarrollo**
- No se puede volver hacía **atrás**
- Deben tener prendida la **cámara**

fierrodilorenzo@gmail.com

Principios a nivel de choesión:
- REP
- CCP
- CRP

Principios a nivel de acoplamiento:

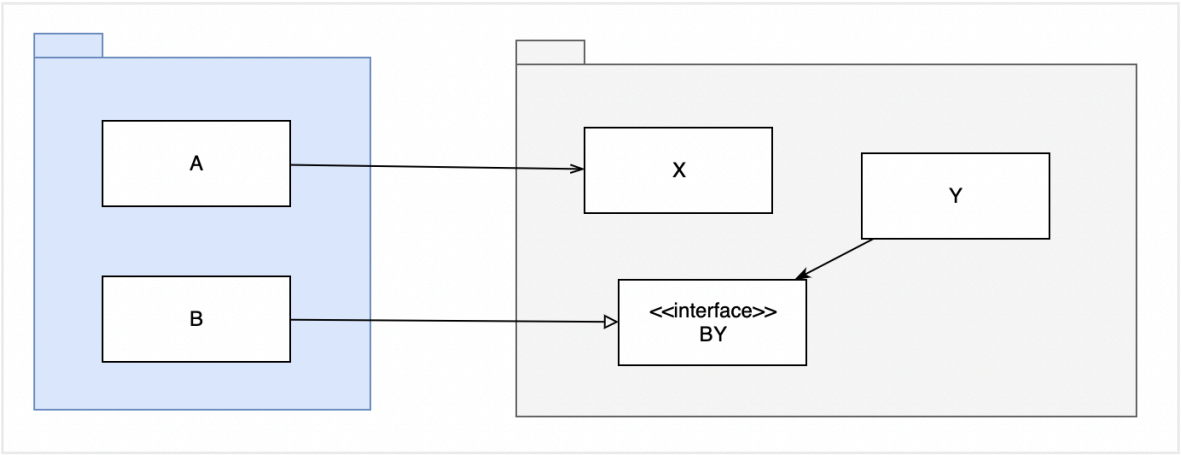
4. **ADP: Principio de dependencias aciclicas o Asyclic Dependencies Principle**

“El grafo de depedencias entre paquetes no debe tener ciclos”

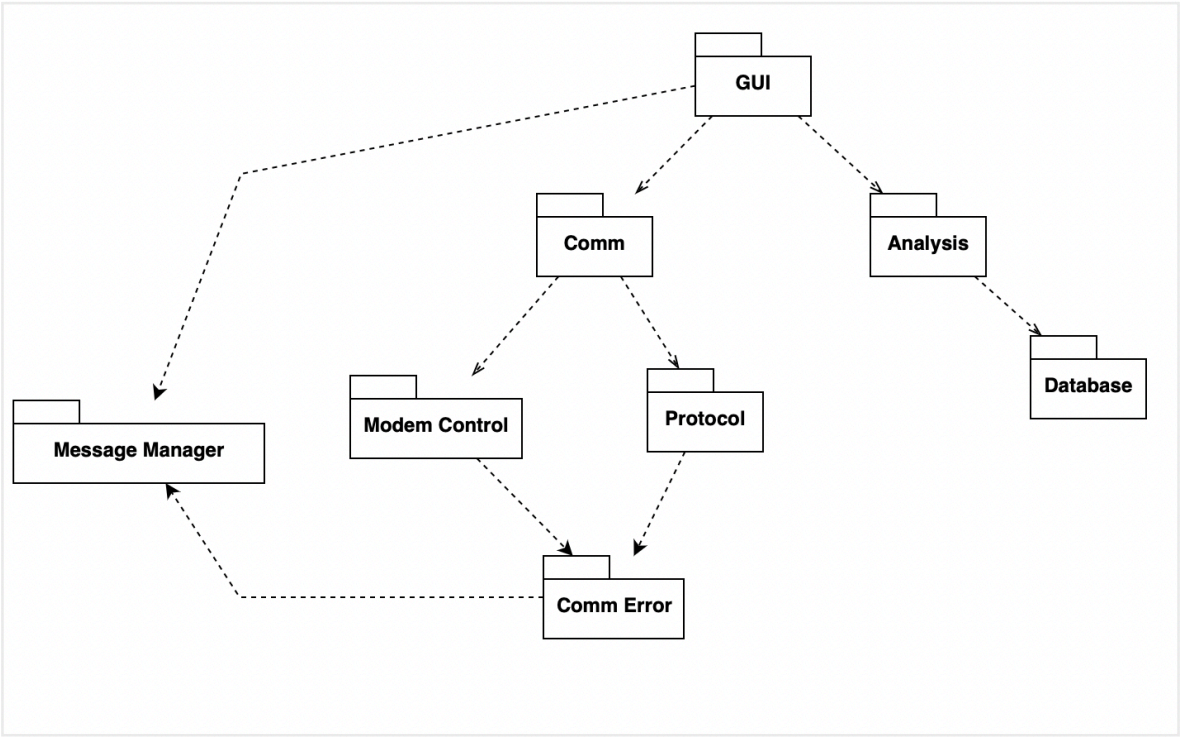
Los paquetes no deben depender indirectamente de ellos mismos. La estructura de depedencias entre paquetes debe formar un grafo dirigido y acíciclo.

¿Como rompemos un ciclos?

1. Aplicamos DIP (basicamente invertimos la dependencia a través de introducir una interfaz



2. Extraigo aquellas funcionalidades que generan un ciclo hacia un nuevo paquete. Creamos un nuevo paquete que agrupa las clases común que son las que nos estaban introduciendo el ciclo.



5. **SDP (Principio de dependencias estables o Stable dependencies principle)**

“Las dependencias entre paquetes deben ir en el sentido de la estabilidad”

Un paquete debe depender solamente de paquetes que sean más estables que él, en otras palabras, las relaciones entre paquetes

tienen que ir desde paquetes menos estables y deben llegar a paquetes más estables.

¿Qué es la estabilidad de un paquete?

- Es la medida en que tan complicado es cambiar un paquete. Esto podemos calcularlo de varias formas:
  - Tamaño
  - Cantidad de clases
  - Calidad de código
  - La cantidad de paquetes que dependen de él y la cantidad de paquetes de los cuales él depende.
    - Esto es lo que vamos a tener en cuenta para medir la estabilidad de un paquete.

Paquete estable

Los paquetes estables son aquellos que tienen muchos paquetes que dependen de él (muchas dependencias entrantes) pero el depende de nadie o muy pocos.

En este caso, “X” es un paquete que tiene tres dependencias entrantes y cero dependencias salientes. Esto significa **que tiene tres buenas razones por las que no cambiar. Decimos entonces que X es:**

- Responsable: porque tiene tres paquetes que dependen de él, y si el cambia, dichos paquetes se verán obligados a cambiar. “X” es responsable de esos tres paquetes
- Independiente: porque “X” no depende de nadie, no hay nadie que lo pueda hacer cambiar mas que si mismo

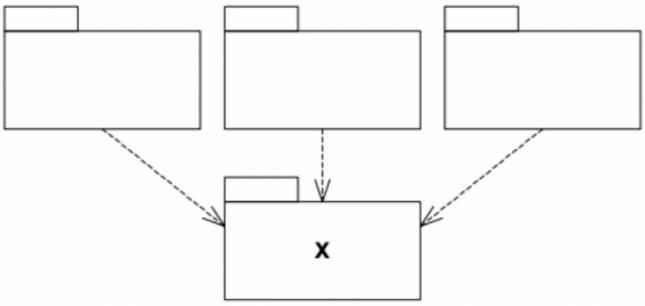


Figure 2-25
X is a stable package

Paquete inestable

Los paquetes inestables dependen de muchos otros paquetes y nadie depende de él. Decimos entonces que este paquete es:

- Irresponsable: porque nadie depende de él
- Dependiente: porque él depende de muchos

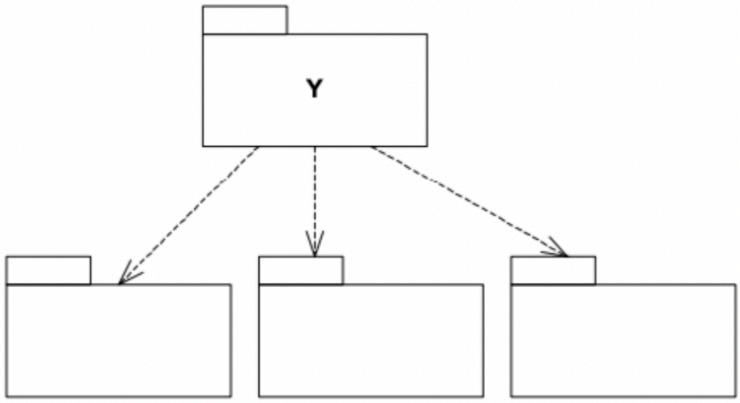


Figure 2-26
Y is instable.

6. **SAP (Principio de Abstracciones estables o Stable Abstractions Principle)**

“Los paquetes deben ser tan estables como abstractos son”

Los paquetes deben apuntar a ser lo más abstractos posible si su estabilidad es alta. Y viceversa.

¿Qué es la abstracción de un paquete?

Un paquete es tan **abstracto** como la cantidad de clases abstractas e interfaces que él posee en comparación con sus clases concretas. Es decir, un paquete tiende a ser más abstracto cuando tiene más interfaces y clases abstractas que clases concretas. Y viceversa para paquetes **concretos**.

Lo que nos dice este principio es:

1. Los paquetes **estables** (aquellos que muchos dependen de ellos y ellos no dependen de nadie o dependen pocos) deben tender a ser **abstractos** (con muchas interfaces y clases abstractas).
2. Los paquetes **inestables** (aquellos donde estos depdende de muchos y nadie depende de ellos) deben tender a ser **concretos** (con muchas clases concretas).

- Un paquete **estable y concreto** es peligroso porque al ser estable muchos dependen de él y al ser concreto tiene muchas clases concretas que cambian frecuentemente. Por ende esto es doloroso (impacto de cambio alto). Decimos que estos paquetes están ubicados en la **Zona de Dolor.**
- Un paquete **inestable y abstracto** es inútil, porque al ser inestable nadie depende de él y al ser abstracto tiene muchas interfaces y clases abstractas, que nadie estaría usando. Decimos que estos paquetes están ubicados en la **Zona de Inutilidad**.

The I vs A graph. The SAP can now be restated in terms of the I and A metrics: I should increase as A decreases. That is, concrete packages should be instable while abstract packages should be stable. We can plot this graphically on the A vs I graph. See Figure 2-28.

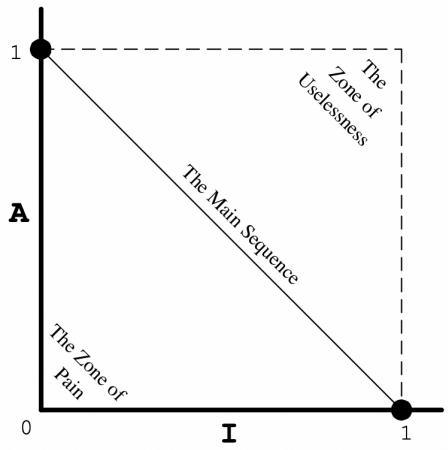


Figure 2-28
The A vs I graph.

Métricas

Aclaración: Obligatorio - Métricas.
No se espera que calculemos las métricas a mano. Se debe utilizar una herramienta externo como NDepend. Las métricas a cauclar son: H, A, I, D'.
Lo más importante de esta sección del obligatorio, no es el calculo de las métricas en sí, si no el analisis realizado sobre los resultados de las mismas. Dicutir posibles mejoras o ideas que se desprendan del resultado de las métricos.

Vamos a tener presente algunas definiciones que ya vimos la clase pasada y esta:

- **Estabilidad:** es la medida de qué tan complicado es cambiar un paquete. Lo vamos a terminar reduciendo a la cantidad de paquetes que dependen de él y la cantidad de paquetes de los cuales él dependen.
- **Abstracción:** qué tan abstracto es un paquete (comparando

cantidad de clases abstractas e interfaces vs clases concretas, dentro de un paquete).
- **Cohesión:** qué tan cohesivo es un paquete (que tan relacionadas o aisladas están las clases de un paquete)
- **Distancia:** cuando traficamos A vs I, ubicamos los paquetes en la gráfica. Entendemos la distancia como qué tan cerca o qué tan lejos están entre ellos.

Cohesión relacional (H):

- Mide qué tan cohesivo es un paquete. Lo hace a través de medir la relaci´øn entre las clases del paquete
- Tiene en cuenta dos valores:
  - R = cantidad de relaciones entre clases internas al paquete
  - N = cantidad de clases e interfaces dentro del paquete
- H = (R + 1) / N
- La métrica H es buena si el valor este entre 1,5 y 4,0
- Si existen múltples relaciones entre dos clases de un mismo paquete, se cuantifica cada una de ellas.

Vamos a considerar como relaciones a cuantificar las siguientes:

- Si C tiene un atributo del tipo D.
- Si C tiene una operación con un parámetro del tipo D
- Si C tiene una asociación, agregación, o composición con navegabilidad hacia D
- C es hijo de D
- C implementa la interfaz D
- Las asociaciones bidireccionales se cuentan dos veces.

Inestabilidad (I):

Esto lo vemos analizando la cantidad de dependencias entrantes vs la cantidad de dependencias salientes. Los paquetes con más dependencias entrantes deberán exhibir un alto grado de estabilidad.

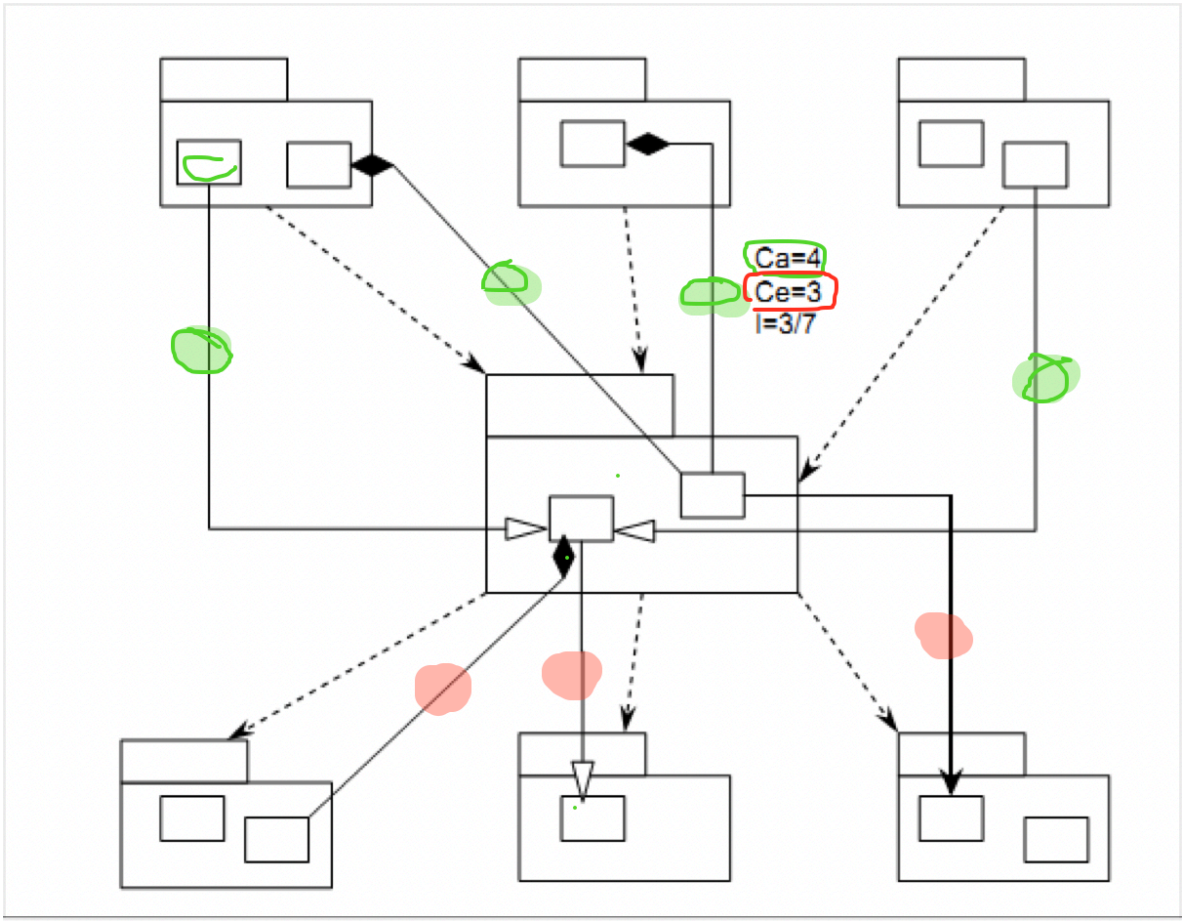
I = Ce / (Ce + Ca)

- Ca = Acoplamiento aferente (dependencias entrantes). **Numero de clases o interfaces** fuera del paquete que dependen de clases o interfaces dentro del paquete
- Ce = Acoplamiento eferente (dependencias salientes). **Numero de clases o interfaces** fuera del paquete de las cuales clases o interfaces dentro del paquete dependen

El valor de I oscila en el rango [0,1].

- Valor cercano a 0 el paquete esta en su máxima estabilidad (no depende de nadie). Se debe intentar extender este paquete llevandolo a ser abstracto.
- Valor cercano a 1 el paquete esta en sun maxima inestabilidad porque depdende de muchos paquetes y nadie depende de el. Este paquete cambia de forma muy sencilla dado que nadie depende de el y por ende su impacto de cambio sera muy bajo.`

Ejemplo:


Ca=4
Ce=3
I=3/7