



T8: Recodificación

¿Qué es Recodificar?

Recodificación

Cambiar la estructura interna del SW

- más fácil entender
- más barato de modificar

sin cambiar su comportamiento

mejorar, Mejorar diseño, POC =

El SW tiende a Podrías...

- ① complicado entender estos
- ② cambia requisitos con el O
- ③ chapuzas → foto B y d.

- ④ Bugs
- ⑤ A) le "pertenece" el código

"Deuda técnica" de Ward Cunningham

Hipotéticamente la APP si no hacemos refactoring
y lo pagaremos en el futuro

¿Por qué Hacer Refactor?

• continua mejora del diseño

• más fácil de entender

↳ limpia los errores

• Ayuda a encontrar Bugs

• Ayuda a codificar más rápido

↳ A largo plazo → Δ productividad

↳ ciclos cortos → ④ productividad, Mejorar diseño

Solo es útil si usas UNITTEST

• se errores
• f conexión, e atropellami
condición Necesaria

Malos olores en el código

Comments: Si para q se entienda tienen que comentarlos - Alai

Duplicated Code

- ① Long Parameter List
- ② Divergent Change
- ③ Large Class
- ④ Shotgun Surgery

Feature Envy

- ⑤ Data Clumps
- ⑥ Parallel Inheritance Hierarchies
- ⑦ Primitive Obsession
- ⑧ Lazy Class

Switch Statement

- ⑨ Speculative Generality
- ⑩ Temporary Field
- ⑪ Message Chains
- ⑫ Middle Man

Inappropriate Intimacy

- ⑬ Alternative class w/ interfaces
- ⑭ Incomplete library class
- ⑮ Data Class

Duplicated Code:

Si la estructura del código es más repetida:

- Extrae el Método
- Mueve a un Padre común
- Légete las Reglas → ② Template Method (más bucles)
- Elige el Algoritmo más claro
- Extrae la clase → crea una nueva con herencia

Long Method

Si el Método ocupa más de una página...

- Extrae el Método
- Reemplaza elem temporales por Queries, elimina bucle, cuando ocurren signifadito
- Introducir un Obj. Parámetro
- Usa parám. comunes de los en Obj.
- Reemplaza métodos por Obj. Método, muchos parámetros
- Descomponer los condicionales

Large Class

Si una clase tiene muchas < métodos

- Extrae la clase → agrupar ↗

Long Parameter List

Un método no necesita muchos parámetros, solo los suficientes → ② para recuperar lo necesario

- Reemplaza parámetro ↔ Método
- Reemplaza N parámetros → Un Objeto

Divergent Change:

Si names q estan cambiando mucho (q = clase → problema)

- Extrae la clase
- Reglas: resultados q cambian mucho en una clase

Shotgun Surgery

Si hace muchos cambios pequeños para cada cambio separado

- Mueve el Método / campo a una sola clase
- clase en línea: agrupa en conjunto controllablemente

Feature Envy

Si un método parece más interesado en otra clase q en él

- Mueve el Método / otra clase
- Extrae el Método

Data Clumps

datos suelen estar juntos q ser llamados a la vez

- Convertir campos relacionados → 1 clase
- Meter un Obj. Parámetro

Switch Statement

si lleva a tu código + inhibir cambios

- Borrar Método
- mover Método donde se pueda aplicar Polimorfismo

Alej herencias ↗

- Reemplaza condiciones
- Polimorfismo

Primitive Obsession

TIPOS PRIMITIVOS (String, INT,...) inhiben el cambio

DNI: String

DNI: Obj. DNI

- ↳ letra: verifica num
- ↳ si hay un cambio (A otro país)

- Sustituye TIPOS P → objetos
- Mueve Método / campo → sola clase
- Mueve Obj. Parámetro
- Reemplaza Array → Objeto

Parallel Inheritance Hierarchies

2 jerarquias de herencia // . Si cada vez q queremos cambiar algo en una jerarq q está dentro en la otra → da error

- Mueve el Método / campo
- una jerarquía negra ref a la otra

Lazy Class

una clase q no lleva mucho → eliminada

Speculative Generality

Si una clase tiene cosas q solo usa en los test → eliminada

Temporary Field

Si una clase tiene campos q solo se usan en excepciones especiales → extraílos

Message Chains

largos cadenas de ↓ para regar un valor

↳ largos

- Elimina un enlace cadena
- Extrae el Método

Middle Man:

Si hay un intermedio claudio que solo encapsula valores → elimínalo / conviértelo en una subclase del obj real

Inappropriate Intimacy:

clases q llaman mucho a los attr de la otra, y no a los suyos → igual no sé donde debe

- Mueve Método / campo → separar piezas
- Extrae la clase
- Reemplaza attr en q → comp osición

Alternative Classes with ≠ Interfaces

clases q hacen lo mismo cu ≠ interfaces

- Renombra el Método / campo → lo =
- Mueve el Método a otras clases
- Extrae la Superclase q no morde redundanteamente el código para el refactor

Incomplete Library Class

Si la librería es insuficiente para sus necesidades, tiene ser ambiguo q tipo q se

- Foreign Methods → uno quiere q se sustituya tener
- Mueve Extensiones Locales

Data Class

clases q tienen: attr, setters, getters y ya q "dumb data Holders"

- Encapsula el campo / colección
- setters
- Mueve el Método
- Extrae el Método

Duplicated Code

Long Method

Large Class

Long Parameter list

necesita tantos Param?
Param → obj / Método

Divergent Change

si tienes q cambiar mucho = close → problema

Shotgun Surgery

muchos cambios pequeños por cada cambio

Feature Envy

si un método ❤ otra clase

Data Clumps → equipos

Datos → estar juntos
llamados a la vez

Switch Statement

Lleva a x2 código + inhibir cambios

Primitive Obsession

string, int, ... → inhiben el cambio

DNI : String / Obj

Nº de letras → verificar
cambio forma DNI es fácil

MALOS OLORES

Temporary Field

campos q solo se usan x veces
↓
especiales

Message Chains

Largas cadenas P para → valor

Middle Name

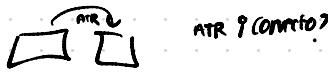
O → O → solo encapsula
valores

Parallel Inheritance Hierarchies



Inappropriate Intimacy

close llama atr de la otra



Lazy Class

Speculative Generality

caract q solo usa test

Alternative Classes w/ ≠ interfaces

clase/
obj