“Tidy First?”: Diseño de Software con Kent Beck

Una conversación con Kent Beck sobre su nuevo libro, Tidy First?, y su enfoque al diseño de software.

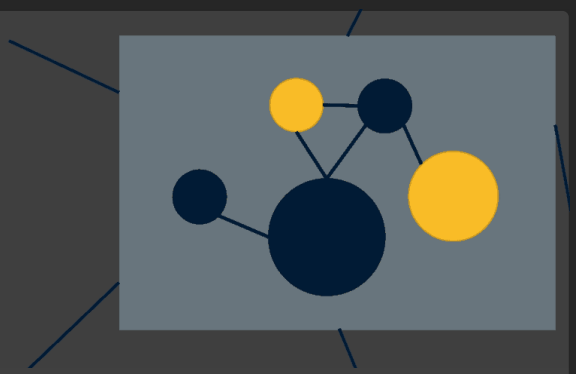
Conceptos clave

Acoplamiento

- Cambios en un elemento producen cambios en otro elemento
- Cambios en cascada
- Coste elevado

Cohesión

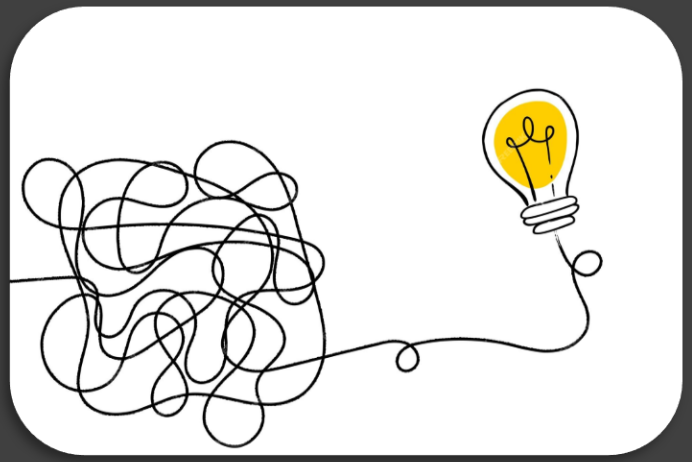
- Un módulo es cohesivo si sus funciones están estrechamente relacionadas



Tidy first?

Pequeños cambios mejoran nuestro software

- Renombrar variables, funciones, etc...
- Extraer funciones
- Cambiando la estructura, no el comportamiento
- Cambios fáciles para hacer fácil el cambio



Tradicionalmente

Los desarrolladores tienden a escribir código desordenado con la intención de mejorarlo más tarde. Sin embargo, Beck argumenta que esta estrategia puede ser contraproducente.

La Pregunta

La frase "Tidy First?" no es una afirmación, sino una pregunta que invita a los desarrolladores a reflexionar sobre cuándo es mejor limpiar el código.

El Costo del Desorden

Dificultad

Un código desordenado y difícil de modificar puede generar fricción en el desarrollo, aumentando el tiempo necesario para realizar cambios y propiciando errores.

Costo Oculto

Un código mal estructurado no solo dificulta la adición de nuevas características, sino que también genera costos ocultos en términos de mantenimiento, corrección de errores y carga de los desarrolladores.

Estrategias para "Limpiar Primero"

1. Identificar las partes más confusas y susceptibles al cambio en el código antes de implementar un cambio.
2. Dividir el proceso en pequeños pasos para no interrumpir el flujo del desarrollo.
3. Adoptar el hábito de refactorizar constantemente como parte del proceso continuo de desarrollo.

Refactorización con Confianza

- Una de las preocupaciones al refactorizar es romper funcionalidades existentes. Beck resalta la importancia de tener pruebas automatizadas que permitan hacer cambios con confianza.
- Si los cambios son reversibles porque darle tanta importancia en analizarlo

Impacto en Colaboración y Productividad

Código limpio reduce la carga cognitiva y facilita la comprensión

Mejora la eficiencia y la colaboración entre desarrolladores

Ahorro de costos ocultos en mantenimiento y corrección de errores

Relación con el Diseño

Diseño Fractal

Los principios de diseño se aplican a todos los niveles.

Diseño Previo

Minimizar las decisiones de diseño en condiciones de incertidumbre.

Software Real

El software que funciona es a menudo desordenado.

Cuándo Limpiar el Código

1

Costo

El costo de la limpieza versus el costo de los cambios.

2

Ingresos

El valor del tiempo y la necesidad de generar ingresos.

3

Acoplamiento y Cohesión

El impacto de la limpieza en el acoplamiento y la cohesión.

El Equivalente de Constantine

1

Costo del Software

El costo del software es principalmente el costo de los cambios.

2

Acoplamiento

El acoplamiento es la causa principal de los cambios costosos.

3

Desacoplar

El costo de desacoplar el sistema.

El Futuro de la Programación

IA en el Desarrollo

La IA como una herramienta para el desarrollo de software.

Programación Humana

La IA no reemplazará a los programadores humanos.

Habilidades

La programación sigue siendo una disciplina que requiere habilidades.