Contents

Codificación **2**
- Objetivo: . . . 2
- afecta . . . 2
- objetivo NO es . . . 2
- el código debe . . . 2

Principios y pautas para la programación . . . 3
- Objetivo principal del programador: . . . 3

Programación estructurada . . . 3
- Objetivo: . . . 3
- estructura estática . . . 3
- estructura dinámica . . . 3
- Objetivo de la programación estructurada: . . . 3
- simplifica . . . 3
  - son de facilitando . . . 3

Ocultamiento de la información . . . 3
- soluciones de software . . . 3
- sólo ciertas operaciones . . . 4
- la información . . . 4
- reduce . . . 4

El proceso de codificación . . . 4
- comienza . . . 4
- módulos . . . 4
- Desarrollo top-down . . . 4
- Desarrollo bottom-up . . . 4

codificación incremental . . . 4
- Proceso básico: . . . 4

Desarrollo dirigido por test . . . 5
- primero . . . 5
- luego . . . 5
- Se realiza . . . 5
- se escribe . . . 5
- responsabilidad de asegurar cobertura . . . 5
- Ayuda . . . 5
- completitud del código . . . 5
- la funcionalidad código necesitará . . . 6

Programación de a pares . . . 6
- código se escribe . . . 6
- ambos programadores diseñan . . . 6
- Una persona . . . 6
- la otra . . . 6
- roles . . . 6
- revisión de código . . . 7
- Mejor . . . 7
- más difícil . . . 7

efectividad . . . 7
Refactorización . . . 7
refactorización . . . 7
estructura interna . . . 7
comportamiento externo . . . 7
objetivo básico . . . 7
Conceptos básicos . . . 7
fin . . . 7
no debe . . . 7
no mezclar . . . 7
principal riesgo . . . 8
disminuir esta posibilidad: . . . 8
Malos olores . . . 8
son . . . 8
Posibles malos olores: . . . 8
Mejoras de métodos . . . 8
Extracción de métodos: . . . 8
Agregar/eliminar parámetros: . . . 8
Mejoras de clases . . . 9
Desplazamiento de métodos: . . . 9
Desplazamiento de atributos: . . . 9
Extracción de clases: . . . 9
Remplazar valores de datos por objetos: . . . 9
Mejoras de jerarquías . . . 9
Remplazar condicionales con polimorfismos: . . . 9
Subir métodos / atributos: . . . 9

Codificación

Objetivo:

-implementar -diseño -mejor manera posible

afecta

-testing -mantenimiento.

-mantenimiento -altos

objetivo NO es

-reducir -costos -implementación, -sino -de testing -mantenimiento,

el código debe

-ser fácil de -leer -comprender

Principios y pautas para la programación

Objetivo principal del programador:

-escribir -programas simples

-fáciles de leer -menor cantidad de errores

-rápidamente

Programación estructurada

Objetivo:

-simplificar -estructura -programas

-sea fácil razonar -sobre ellos.

estructura estática

-orden -sentencias

-código -orden lineal).

estructura dinámica

-orden -sentencias

-ejecutan.

Objetivo de la programación estructurada:

-programas -estructura -dinámica -misma que -estática,

simplifica

-flujo -estructurada de control,

son de facilitando
-comprensión

-razonamiento

Ocultamiento de la información

soluciones de software

-contienen -estructuras de datos -guardan

-información.

sólo ciertas operaciones

-se realizan -sobre -información,

la información

-sólo quede expuesta

-esas -operaciones.

reduce

-acoplamiento.

El proceso de codificación

comienza
-está disponible -especificación -diseño

-módulos.

módulos
-se asignan -programadores -individuales.

Desarrollo top-down
-módulos -niveles superiores -desarrollan

-primero.

Desarrollo bottom-up
-módulos -niveles inferiore -desarrollan

-primero.

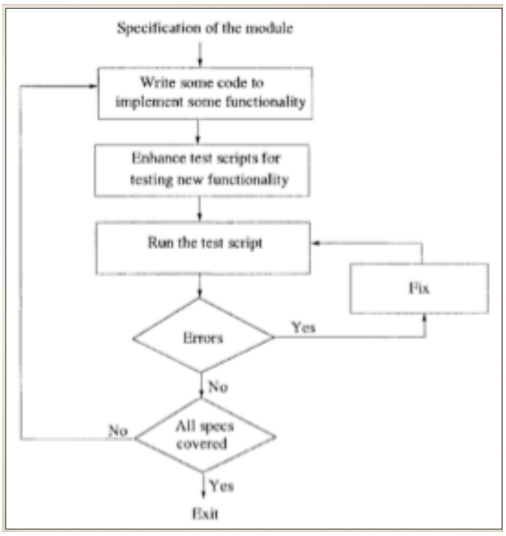
codificación incremental

Proceso básico:

-Escribir -código -módulo.

-Realizar -test -unidad.

-Si error: -arreglar bugs -repetir tests.



Desarrollo dirigido por test

primero
-escribe -tests

luego

-código -para que -pasen -los -tests

Se realiza
-incrementalmente.

se escribe
-código -suficiente -pasar -test

responsabilidad de asegurar cobertura
-funcionalidad -radica -diseño

-test -no -codificación.

Ayuda
-asegurar -código -testeable.

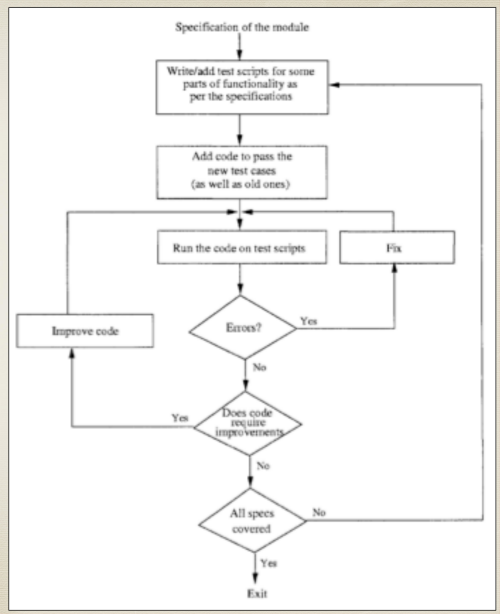
completitud del código
-depende

-toda la cuan exhaustivo -el test.

la funcionalidad código necesitará
-radica en el refactorización

-mejorar -código

-confuso



Programación de a pares

código se escribe
-dos programadores

ambos programadores diseñan
-algoritmos,

-estructuras de datos, -estrategias, -etcétera.

Una persona
-tipea -código,

la otra
-revisa -código

roles
-alternan -periodicamente.

revisión de código
-continua.

Mejor
-diseño -algoritmos/estructuras de datos/lógica/...

más difícil
-escapen -condiciones particulares.

efectividad
-no es -bien sabida -(perdida de productividad?)

Refactorización

refactorización

-tarea -permite -cambios -programa -con el

-simplificarlo -mejorar -comprensión -sin cambiar -comportamiento observacional

estructura interna

-cambia.

comportamiento externo

-permanece igual.

objetivo básico

-mejorar -diseño -plasmado en el -código

Conceptos básicos

fin

-reducir -acoplamiento, -incrementar -cohesión, -mejorar -principio abierto-cerrado.

no debe

-cambiar la funcionalidad.

no mezclar
-codificación normal -refactorización.

principal riesgo
-“romper”

-funcionalidad existente.

disminuir esta posibilidad:

-Refactorizar -en -pequeños pasos.

-tests -para -funcionalidad existente.

Malos olores

son
-signos -indican -posible -necesidad -refactorización.

Posibles malos olores:

-Código duplicado:

-Método largo:

-Clase grande:

-Lista larga de parámetros:

-subclase es la misma que la superclase;

-Demasiada comunicación

-Encadenamiento de mensajes:

Mejoras de métodos

Extracción de métodos:

-separar en -métodos cortos -cuya -signatura -indique -lo -que

-hace.

Agregar/eliminar parámetros:

-simplificar -interfaces

Mejoras de clases

Desplazamiento de métodos:

-método -actúa demasiado -objetos -otra clase.

Desplazamiento de atributos:

-atributo -se usa más -otra clase, -moverlo

-Mejora -cohesión -acoplamiento.

Extracción de clases:

-clase -agrupa -múltiples conceptos,

-Mejora cohesión.

Remplazar valores de datos por objetos:

-Algunas veces, -colección de atributos -transforma -entidad lógica.

-Separarlos -como -clase

Mejoras de jerarquías

Remplazar condicionales con polimorfismos:

-no se está explotando

-OO.

-Reemplazar -casos -a través -jerarquía de clases -apropiada.

Subir métodos / atributos:

-elementos comunes -pertenecer -superclase.

-funcionalidad -atributo -duplicado -subclases, -subirse

-superclase.