

INGENIERÍA DEL SOFTWARE

PARCIAL II

Pregúntalo para que te tome tu hermana, tu vieja o tu tía, que no tienen ni la más puta idea de ingeniería del software

Índice:

- Codificación ----- 2
- Modelos de proceso ----- 9
de desarrollo
- Proceso de ----- 15
software
- Testing ----- 24
- Planeamiento del ----- 36
proceso de software

CODIFICACIÓN (ca)

¿Cuál es el objetivo de la codificación?
¿Cuál es su propósito?

El objetivo es implementar el diseño de la mejor manera posible. Como esta implementación afecta al testing y mantenimiento, el propósito de la codificación es reducir los costos de ambos. Para esto el código debe ser fácil de leer y comprender.

¿Para qué sirven los principios y pautas para la programación?
¿Cuáles son? Nombrarlos

Ayudan al programador a cumplir su objetivo: escribir código de alta calidad, es decir, fuertementeensible, testeable y mantenible.

Son:

- Programación estructurada
- Diseño de la información

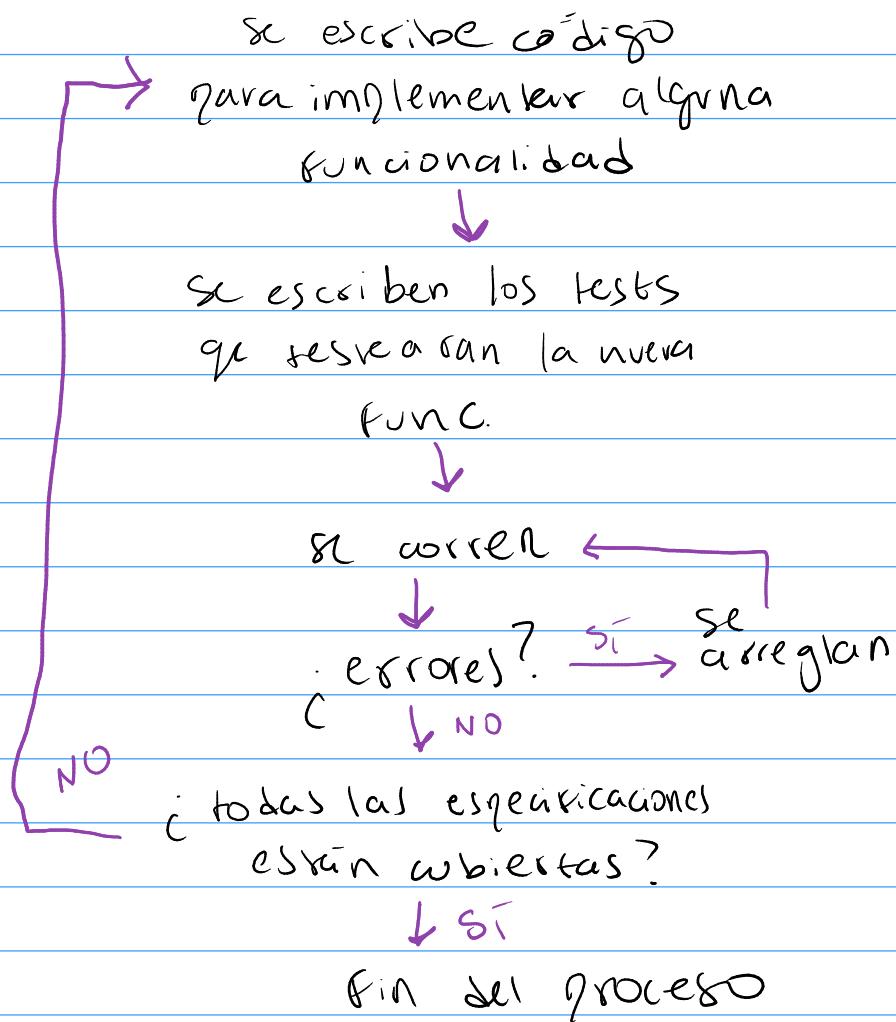
Exige el principio de programación estructurada

La PE es un principio que ayuda al programador a escribir código de alta calidad mediante la simplificación del flujo de control del programa, facilitando su comprensión y razonamiento. Para efectuar esta simplificación se aquista a escribir programas con estructura dinámica, o sea, el cómo se ejecuta el código, ya la misma ge su estructura estática, que es cómo escribir el código.

Exige el principio de ocultamiento de información

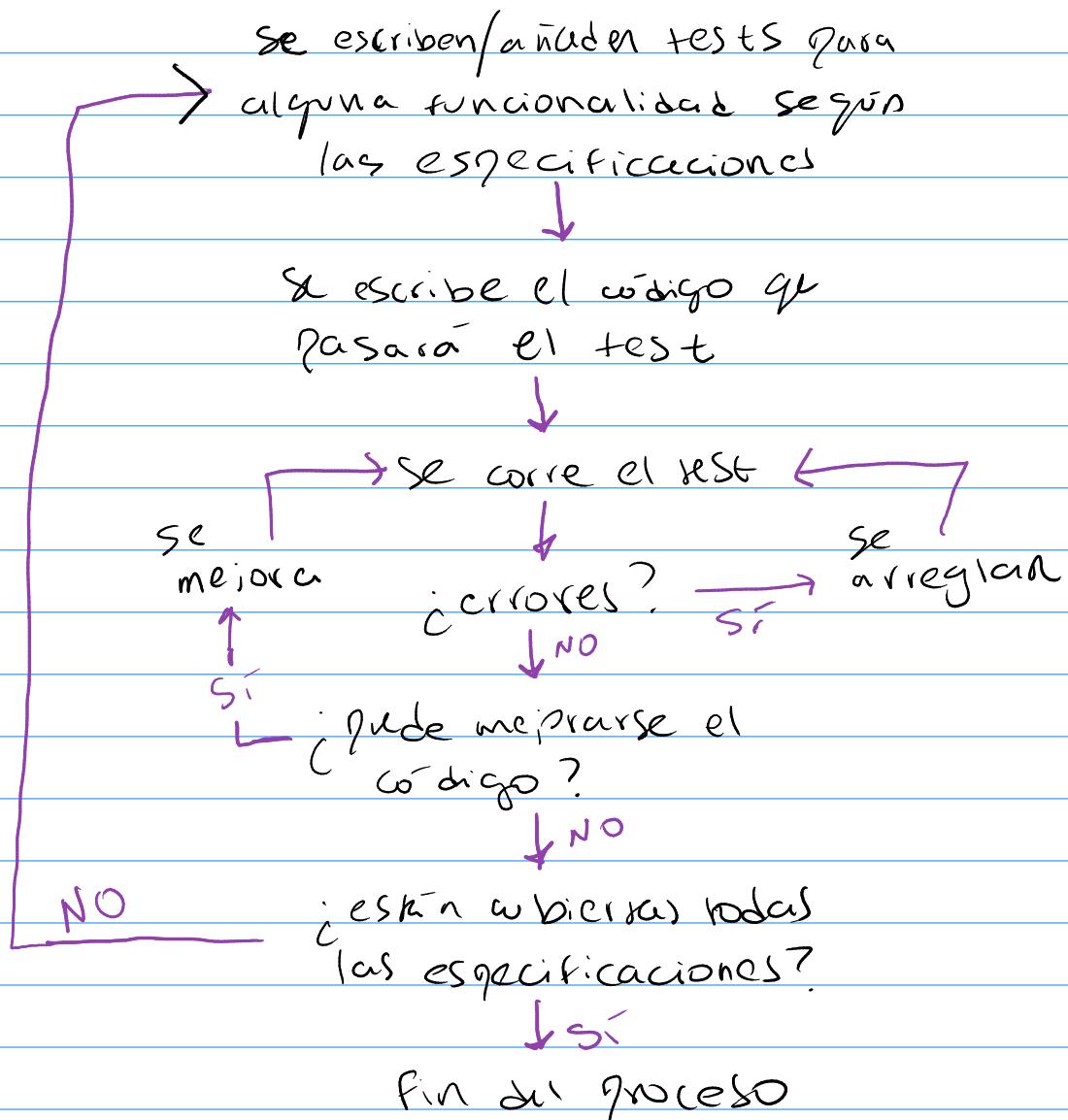
Este principio consiste en ocultar la info que guardan las estructuras de datos contenidas en el código de manera se sólo quedan expuestas a aquellas operaciones que la utilizan, reduciendo así el acoplamiento.

Explique el proceso de codificación incremental



Explique el proceso de desarrollo dirigido por tests

La responsabilidad de cubrir todas las especificaciones radica en el diseño de los tests.



Explique el proceso de programación de a pares

En este proyecto el código es escrito por programadores a la vez: ambos diseñan los algoritmos, estructuras de datos, etc. en estos roles se intercambian:

- Uno escribe el código
- El otro revisa activamente lo que se está escribiendo

Si bien con este proceso el diseño de algoritmos, estructuras y lógica mejora y es menos probable que se escapen vulnerabilidades a la hora de codificar, seuestiona su efectividad debido a la probabilidad de cierta pérdida de productividad.

¿Qué es la refactorización?

¿Qué intenta lograr? ¿Cómo debe llevarse a cabo para no "romper" la func. existente?

Es una técnica para mejorar el diseño del código existente. A la hora de llevarla a cabo decimos que la tarea que permite realizar cambios en el programa con el fin de simplificarlo y mejorar su comprensión. Se realiza durante el proceso de

codificación sobre código que ya está funcionando ya que su propósito no es cambiar el comportamiento externo del software, sino su estructura interna, mejorándola.

La refactorización intenta lograr

- Reducción del acoplamiento
- Aumento de la cohesión
- Mejora de la respuesta al principio abierto-cerrado

disminuir la posibilidad

Para evitar "romper" funcionalidades existentes podemos

- Refactorizar en pequeños pasos
- Disponer de tests automatizados para verificar la funcionalidad existente

¿Cuáles son las refactorizaciones más comunes?

Las refactorizaciones más comunes se enfocan en

mejoras en los métodos

- Extracción de métodos: Se realiza si el método es demasiado largo, el objetivo es tener métodos cortos una firma indica lo que hace el método

- Agregar/Eliminar parámetros:

Simplifica las interfaces. Se agregan si los existentes no proveen la info necesaria y se eliminan si no están siendo utilizados.

Mejoras en las clases:

- desplazamiento de método:

cuando el método interactúa mucho con
objetos de otra clase

- desplazamiento de atributos:

cuando un atributo se usa más en otra
clase que en la actual

- extracción de clases: cuando

la clase agrupa muchos conceptos, si
se separa cada uno en una nueva clase

- reemplazar valores de dentro

por objetos: se separa como una
clase una colección de atributos que se
transformaron en una entidad lógica
y se definen objetos para accederlos.

Mejoras de jerarquía:

- reemplazar condicionales con
polimorfismos: encontrar una jerarquía
de clases apropiada para aquellos análisis
por casos que dependen de indicadores
de tipo.

- Sobrir métodos/atributos: Si un
método o atributo se usa en más de
una subclase, podemos subirlo a la
superclase.

MODELOS DE PROCESO DE DESARROLLO

¿Qué es un modelo de proceso de desarrollo?

Es una estructura genérica de los procesos que siguen según los proyectos para alcanzar sus objetivos. Especifica un proceso general como un conjunto de etapas.

Una vez se elige un modelo es necesario adecuarlo al proyecto

Explique el modelo de cascada, sus ventajas, desventajas y aplicación

Es el modelo más simple. La secuencia inicial de las fases es

- Análisis de requerimientos
- Diseño de alto nivel
- Diseño detallado
- Codificación
- Testing
- Instalación

Estas siguen un orden lineal, es decir, una comienza sólo cuando la anterior finaliza. Cada una se encarga de distintas incumbencias y debe producir una salida que debe ser evaluada y certificada (producto de trabajo)

Ventajas:

- Conceptualmente simple; se divide el problema en fases qd qeden realizarse independientemente
- Fácil de ejecutar, las fronteras entre las fases están bien definidas
- intuitivo y lógico

desventajas:

- riesgoso ya qd es a "todo o nada", se desarrolla el software todo "de una"
- no hay feedback qd parte de los usuarios entre fases
- requerimientos se congelan muy temprano
- no permite cambios

aplicación:

- problemas conocidos
- proyectos cortos
- automatización de procesos manuales existentes

Explica el modelo de proceso prototípico

En este modelo, durante la fase de análisis de requerimientos, se construye un prototipo del proyecto qd permitir comqrender dichos requerimientos. El desarrollo de este prototipo comienza con una versión preliminar de los

requerimientos), incluyendo sólo características claves se requieren comprensión.

El cliente usa el prototipo y provee feedback. Este feedback se tiene en cuenta para la modificación del prototipo y de los requerimientos iniciales.

El prototipo debe seguir modificándose hasta que los costos y tiempo superen el beneficio del proceso.

Cuando esto sucede o una vez que los requerimientos fueron comprendidos, el prototipo debe descarrarse y se pasa a la siguiente fase.

Ventajas:

- Ayuda a la recolección de requerimientos
- Reduce el riesgo
- Mayor estabilidad en los req., se construyen más tarde
 - La experiencia en la construcción del prototipo ayuda al desarrollo principal, haciendo que los sistemas finales sean más estable).

Desventajas:

- Puede causar aumento de costo y tiempo
- Ambiente desordenado
- No permite cambios tardíos

Aplicación:

- Requerimientos no se comprendieron
- Sistemas c/ usuarios novatos
- Interfaces c/ el usuario son muy importantes

Explique el proceso de desarrollo iterativo

Este modelo consiste en desarrollar y entregar el software incrementalmente, es una "secuencia de cascadas". Cada incremento es completo en sí mismo, se prueban y se recibe feedback luego de que se completa.

Dentro de él tenemos dos tipos:

Modelo con mejora iterativa:

Primero se hace una implementación simple para un subconjunto del problema completo

Luego se crea una lista de control del proyecto (LCP) que contiene, en orden, todas las tareas a realizar para lograr la implementación final.

en cada paso se elimina una tarea de esta lista haciendo diseño, implementación y análisis del sistema parcial. Luego se actualiza la lista y se pasa a la siguiente tarea

ESTO SE REpite hasta vaciar la lista.

Modelo en espiral:

Se identifican objetivos, alternativas y restricciones se evalúan las alternativas, considerando riesgos.

se desarrollan estrategias de resolverlos

se desarrolla y verifica el SW

se plantea el próximo paso

Ventajas:

- entregas regulares y rápidas
- reduce riesgos
- acorta cambios naturalmente
- permite feedback del usuario
- permite priorizar requisitos

Desventajas:

- sobrecarga de planteamiento en c/u iteración
- posible incremento del costo total (trabajo de una iteración puede deshacerse en otra)
- diseño y arquitectura suelen ser afectados c/ tantos cambios

Aplicación:

- empresas donde el tiempo de respuesta es esencial
- no quede enfriarse el riesgo de proyectos largos
- los requerimientos son desconocidos y se comprenden con el tiempo

Exige el modelo de proceso de desarrollo timeboxing.

Consiste en una secuencia lineal de iteraciones, en donde c/u es una minicascada. Cada fase de esta minicascada tiene la misma duración, puede desarrollarse independientemente y tiene su propio equipo de trabajo.

Cuando un equipo finaliza, pasa la salida a la siguiente etapa y comienza a trabajar en la nueva iteración.

Las sprints son cada una de esas iteraciones, y también tienen todas la misma duración. Se utiliza zigzaging para que se ejecuten in paralelo. El cronograma de este modelo no es negociable.

Ventajas:

- las del iterativo
- facilita el planeamiento y la negociación
- ciclo de entrega muy corto

Desventajas:

- se complejiza la admin. del proyecto
- potencial incremento de los costos
- equipos de trabajo muy grandes

Aplicación:

- tiempos de entrega muy cortos
- flexibilidad para agregar características

EL PROCESO DE SOFTWARE

(cap-8)

¿Qué es el proceso de software?
¿Por qué lo dividimos en fases?

El proceso de software es lo que se ejecuta para llegar al producto terminado. Es un conjunto de fases, donde cada una realiza una tarea bien definida y produce una salida. Estas salidas son los productos de trabajo, entidades formales y tangibles capaces de ser verificadas.

Esta división en fases sirve para modularizar el problema, tratarlo de a partes más pequeñas y manejables. Además, el hecho de que el c/u tenga una salida verificable ayuda a hacer el seguimiento y control del mismo, quedando detectar errores o fallas en cada una.

¿Qué es el enfoque ETUX?

Es el enfoque que sigue cada una de las fases del proceso de SW, este sirve para resolver el problema de cuándo una fase empieza y termina. Para ello especifica/define criterios de entrada

y salir de cada fase.

ETUX es por

Entry (criterio de entrada): se definen qué condiciones deben cumplirse para iniciar la fase

Task (tarea): define qué se realizará en cada fase

Verification (verificación): qué controles o inspecciones deben realizarse al producir de trabajo

Exit (criterio de salida): qué condiciones deben cumplirse para finalizar la fase

ii) El criterio de entrada de una fase debe ser consistente con el criterio de salida de la fase anterior.

Explique el proceso de administración del proyecto

Es aquél que se encarga de asegurar la ejecución eficiente de las fases del proceso de desarrollo. → fases:

Planeamiento: Se produce, antes de comenzar el proyecto, el plan que será la base del seguimiento.
tareas claves:

- estimación de costos y tiempos
- Selección de personal
- Planeamiento del seguimiento y control de calidad

Seguimiento y control: se revisa activamente el proceso de desarrollo para asegurar de que se está llevando a cabo en tiempo y forma correctas.

- Seguir y observar parámetros clave como costos, tiempo, riesgos y factores que los afecten
- tomar acciones correctivas si sea necesario

Las métricas del proceso de desarrollo son las que brindan la info necesaria para ser efectiva

Análisis de terminación: se realiza al terminar el proceso de desarrollo. Su propósito es analizar el desempeño del proceso e identificar lo aprendido. En procesos iterativos se realiza luego de cada iteración.

Exige el proceso de inspección

Es un proceso estructurado cuyo objetivo es detectar defectos en los productos de trabajo, mejorando así la calidad y productividad.

Este proceso tiene roles definidos:

- MODERADOR: tiene la responsabilidad general. Asegura se el foco permanece sobre la identificación de defectos y garantiza se la reunión se ejecute ordenada y amigablemente.
- AUTOR: quien realizó el producto de trabajo
- REVISOR: quien identifica los defectos
- LECTOR: lee linea a linea el producto de trabajo para enfocar el progreso de la reunión
- ESCRIBA: registra las observaciones

Las fases de este proceso son

Planeamiento:

- Se selecciona el equipo de revisión
- Se identifican al moderador
- Se prepara el trabajo q/la distribución
 - Producto a revisar
 - Sus especs.
 - Lista de control estándares)

Puesta en marcha (Preparation)

Cada miembro del equipo revisa el producto de trabajo individualmente para identificar defectos usando listas de control y estándares).

Opcional: breve reunión previa donde se entregar el trabajo, se explicar el propósito de la revisión y se introducen las áreas de cuidado.

Reunión de revisión grupal:

propósito: definir la lista final de defectos
criterio de entrada: revisiones individuales adecuadas, revisadas por el moderador

reunión:

- se lee línea a línea el producto de trabajo
- si efectúa cualquier observación se hable
- si discute para identificar el defecto
- el escritor registra la decisión final de la reunión:

- escritor presenta la lista final
- pocos defectos => se acepta el producto
- muchos " " => se somete a otra revisión
- resumen de inspecciones y analizar la ejecución de la revisión

Corrección y seguimiento

• entregar el producto al autor junto con la lista de defectos/observaciones para que haga las correcciones necesarias.
Una vez corregido, si el moderador lo agradece, finaliza el proceso, sino se somete a otra revisión.

Explique el proceso de administración de configuración (SCM)

Este proceso se encarga de controlar sistemáticamente los cambios en los ítems generados en el proyecto (documentos, programas, datos, etc.). A medida que estos ítems van cambiando generan nuevas versiones, la SCM debe asegurar que estas se combinen sin pérdidas.

Los mecanismos principales para llevarlo a cabo son

- Control de versiones
- Control de acceso
- Identificación de la configuración

Tiene 3 fases

Planeamiento:

- identificar ítems
- definir la estructura del repositorio de referencia, reconciliación, procs.

- definir procedimientos de publicación.

Ejecución:

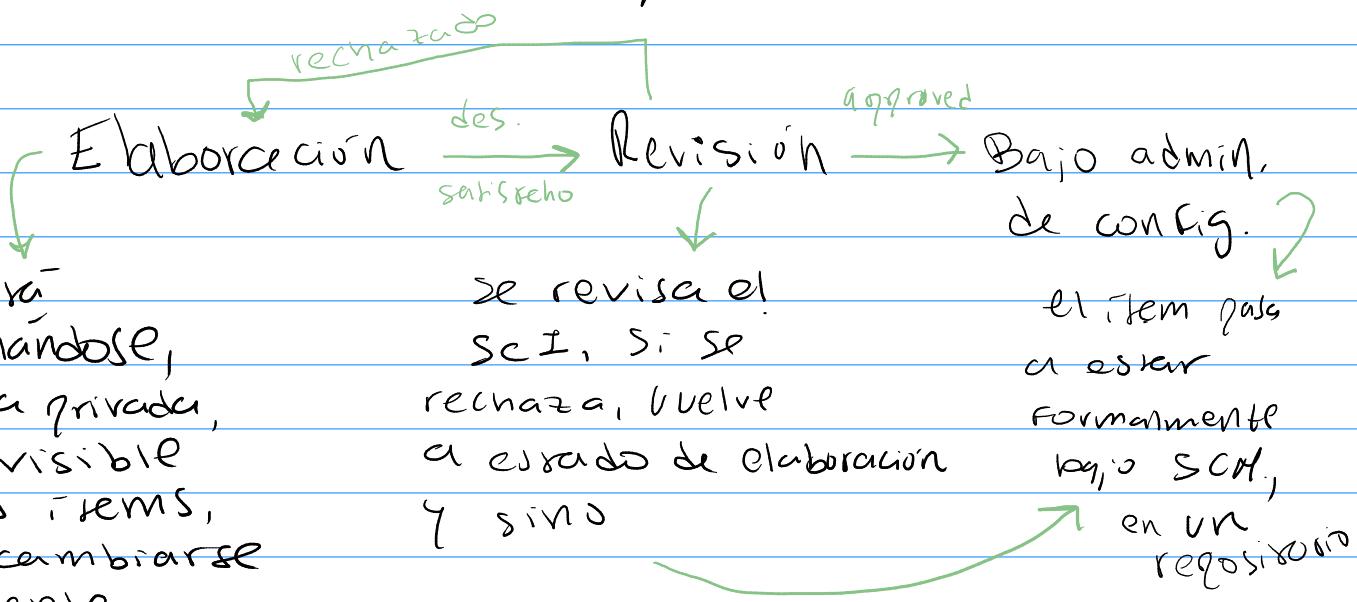
- realizan los procedimientos según lo establecido en el planeamiento

Auditoría:

- Se verifica que no se cometieron errores.

Explica el mecanismo de control de acceso del proceso de administración de configuración

Es uno de los mecanismos principales del proceso de administración de configuración del SW. Consiste en herramientas que limitan el acceso a personal específico mediante procedimientos check in y check out. Podemos verlo en el ciclo de vida de un ítem de configuración.



Una vez que el ítem está en un repositorio, cualquier modificación debe controlarse, para ello, el acceso a estos ítems se controla. Si quiero modificarlo, debo tomarlo del repositorio (check-out), modificarlo, enviarlo a revisión y cuando se aprueba, con check-in mandarlo de nuevo al repositorio.

Explique el proceso de administración de cambios de requisitos

Este proceso busca controlar/administrar los cambios en los requerimientos ya que estos tienen impacto en los productos de trabajo e ítems de configuración.

Proceso:

- registrar los cambios
- realizar el análisis de impacto sobre los productos de trabajo y los ítems (identificar cambios necesarios en c/u y su naturaleza).
- estimar el impacto en esfuerzo y cronograma
- analizar el impacto con las personas involucradas
- reevaluar los productos de trabajo y los ítems

El impacto del cambio es analizado para decidir si hacerlo efectivo o no.

Exige el proceso de administración de procesos

Este proceso se enfoca en la evaluación y mejora del proceso de desarrollo. Estas mejoras deben hacerse incrementalmente, en pequeños pasos y deben seleccionarse cuidadosamente.

Existen marcos que sugieren formas de proceder en esta mejora del proceso, uno es CMM.

Exige CMM

Es un marco que sugiere formas de proceder en la mejora del proceso durante la administración de procesos.

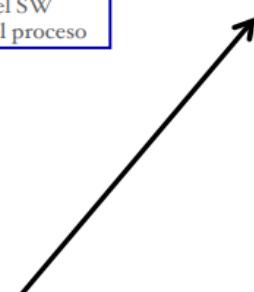
Plantea 5 niveles del proceso de software, el 1º es ad-hoc. En cada nivel el proceso tiene ciertas capacidades y establece bases para pasar al siguiente nivel, esto son, áreas en las cuales enfocarse para mejorar el proceso.

Nivel 5 - Optimizado:
- Admin. de cambio de proceso
- Admin. de cambio de Tecnología
- Prevención de defectos

Nivel 4 - Administrado:
- Admin. de la calidad del SW
- Admin. cuantitativa del proceso

Nivel 3 - Definido:
- Organización en definición del proceso
- Programa de entrenamiento
- Revisión de pares
- Admin. integrada del SW

Nivel 2 - Repetible:
- Admin. de requerimientos del SW
- Admin. de configuración del SW
- Planeamiento del proyecto
- Control y monitoreo del proyecto



Para someter a un proceso a este marco, primero se evalúa en qué nivel está y luego se procede a mejorar en las áreas que es necesario para ir subiéndolo de nivel hasta llegar al último.

TESTING (Cap. 10)

¿Qué es un test? - Para qué sirve?
¿Qué actores intervienen?

Un test es un programa que se ejecuta con el objetivo de detectar defectos en el software, para así poder corregirlos, aumentando la calidad del producto entregado. Estos se llevan a cabo por los mismos programadores presentes en el proceso de codificación o por otros nuevos. Los encargados de escribir estos tests son llamados testers.

¿Cuál es la diferencia entre un desfase y un defecto?

Un desfase ocurre cuando el comportamiento del software no es el esperado, mientras que un defecto es aquello que causa el desfase.

Desfase \Rightarrow presencia de un defecto
implica

Defecto no necesariamente implica la ocurrencia de un desfectoro, pero sí tiene potencial para causarlo.

¿Cuál es el rol del testing?
¿Cómo se lleva a cabo?

Rol \Rightarrow garantizar calidad en el SW mediante la identificación de defectos.

Estos se identifican ejecutando un programa y si se sigue un conjunto de casos de test, si hay desfectoros en la ejecución \Rightarrow defectos en el SW.
Durante el testing debemos causar estos desfectoros ya que así detectaremos defectos. Luego, para identificar estos defectos recurrimos al debugging.

¿Qué es un oráculo de test?
¿Para qué sirve?

Aquello que conoce el comportamiento correcto para algún caso de test.

Sirve para saber si nuestro test genera algún desfectoro o no y así proceder a identificar o no el defecto.

Se considera, idealmente, que el oráculo siempre da el resultado correcto, por eso no puede ser humano, ya que cometemos errores. En algunos casos los oráculos quedan generados directo de la especificación.

¿Qué es un caso de test?
¿Qué es un criterio de selección de test? ¿Para qué existen?

Un caso de test son los datos de entrada con los que se va a correr un test, generalmente son valores de variables. Como la ejecución de estos casos de test debe implicar la ausencia de defectos y a su vez debe ser un conjunto de datos y estas dos ideas son contradictorias, para seleccionarlos usamos criterios de selección de test. Estos criterios especifican las condiciones que debe satisfacer un conjunto de casos de test respecto al programa y/o la especificación. Deben ser confiables y válidos.

¿Cuál es la diferencia entre testing de caja blanca y testing de caja negra?

El testing de caja negra, también conocido como testing funcional, no utiliza la estructura interna del código para seleccionar los casos de test, estos se seleccionan a partir del comportamiento esperado del sistema. En cambio, en el testing de

caja blanca o testing estructural si se utiliza la estructura interna del código, se enfoca en ella y los casos de test se derivan de ahí.

Mientras que el testing funcional es útil a un alto nivel y es bueno para detectar errores de entrada/salida (errores funcionales), el estructural es útil a bajo nivel y es bueno para detectar errores en la lógica del programa, o sea errores estructurales.

Estas técnicas son complementarias, deben usarse ambas.

Explique particionado de clases de equivalencia

Es un enfoque para seleccionar casos de test en el testing de caja negra. Consiste en dividir el espacio de entrada en clases de equivalencia, si el test funciona para un caso de test dentro de una clase, entonces funcionará de la misma manera para todos los elementos de la misma clase.

Para particionar, debemos identificar las **clases** para las cuales se especifican **distintos comportamientos**. También deben crearse clases de equivalencia para entradas inválidas. Una vez particionadas las clases tenemos dos criterios de selección de casos de test:

1- Cada caso cubre tantas clases de equivalencia como sea posible

2- Cada caso cubre a lo sumo una clase

A estos casos se les suma un caso más por cada clase de equivalencia para las entradas inválidas.

Exige análisis de valores límites.

Es uno de los enfoques para seleccionar casos de test en el testing de caja negra.

Un caso de test de valores límites es un conjunto de datos de entrada que están al borde de las clases de equivalencia.

Para cada clase se seleccionan valores en los límites, valores al límite justo fuera de los límites, valores justo dentro de los límites y un valor normal.

A la hora de seleccionar los casos de test tenemos, para los programas de una entrada con rango definido, un

criterio: Se seleccionan

2 valores límite

2 valores justo fuera

2 valores justo dentro

1 valor normal

} 7 casos

Para programas de múltiples entradas tenemos dos criterios:

- todas las combinaciones posibles de

las distintas entradas = $7^{n \rightarrow n}$ de entradas
(inversible)

- Valores límite en una variable y valores normales en las otras + un caso con valores normales para todas
= $6 \cdot n + 1$

Explique el grafo de causa-efecto

Enfoque q/ seleccionar tests en el testing de caja negra. Ayuda a seleccionar las combinaciones de las clases de equivalencia como condiciones de entrada.

Primero debemos identificar causas y efectos q/ hego q/ causas producen q/ efectos.

Los nodos serán estos dos y las aristas determinarán dependencias entre estos dos.

A partir de este grafo quede armarse una tabla de decisión q/ listar las combinaciones de causas q/ hacen efectivo c/ efecto.

Eso quede usarse q/ seleccionar cuantos de test.

Exige el testing de a pares

Enfoque q) seleccionar casos de test en el testing de caja negra. El testing de a pares surge del descubrimiento de que, la mayoría de los defectos en sistemas cuyo comportamiento dependía de múltiples parámetros, se revelaban con la interacción de pares de parámetros. En resumen, este enfoque, consiste en ejercer cada uno de esos pares.

El menor conjunto de casos de test se obtiene cuando cada par es cubierto solo una vez. Para encontrarlo y garantizar una cobertura completa existen eficientes algoritmos.

Exige testing basado en estados

Enfoque q) seleccionar casos de test en el testing de caja negra. Los casos de test se seleccionarán usando el modelo de estados del sistema, esto es una representación gráfica de los estados lógicos del mismo. Tiene cuatro componentes: el conjunto de estados, el conjunto de transiciones (cambios en respuesta a algún evento de entrada), conjunto de eventos (entradas), conjunto de acciones (salidas en respuesta a los eventos y al estado actual).

Una vez tenemos el modelo de estados, hay varios criterios para generar los casos de test:

- Cobertura de transiciones: deben cubrirse todas las transiciones
- Cobertura de par de transiciones: deben cubrirse todos los pares de transiciones adyacentes que entran y salen de un estado
- Cobertura del árbol de transiciones: deben cubrirse todos los caminos simples, del estado inicial al final o a uno visitado.

Exige el criterio basado en el flujo de control

Es uno de los criterios de selección de casos de test del testing de caja blanca. Se considera al programa como un grafo de flujo de control donde los nodos son bloques de código (sentencias) y las aristas son posibles transiciones de control entre nodos. Podemos generar casos de test a partir de los siguientes criterios:

- Criterio de cobertura de sentencias: Cada sentencia debe ejecutarse al menos una vez
- Criterio de cobertura de ramificaciones: Cada rama del grafo debe recorrerse al menos una vez, las decisiones deben ejecutarse como V y F.

Cob.
Ramificaciones \Rightarrow Sentencias
implícitas

Sin embargo puede haber test suites que cubran ramificaciones que no detecten un error mientras que haya test suites que cubran sentencias que sí lo detecten.

Ejemplo: $\log(x) : l = 0$
 $\text{if } (x > 0) \text{ then } l = \log(x);$
 $\text{return } (l)$

Ramificaciones = $\{(x=1, l=0), (x=-1, l=0)\}$

Sentencia = $\{x=0, l=0\}$

\hookrightarrow falla en tratar de hacer $\log(0)$

• Criterio de cobertura de caminos:
todos los posibles caminos del nodo inicial al nodo final.

Problema: cant. de caminos puede ser infinita.

Explique el criterio basado en el flujo de datos

Es uno de los criterios de selección de casos de test del testing de caja blanca. Para poder seleccionar los test suites, se construye un grafo de definición-uso eligiendo el grafo de flujo de control.

Una sentencia en el grafo de flujo de control queda ser:

- en nodos def : definición de una variable (la var está a la izq. de la asignación)
- en aristas uso-c : la var se usa para cónputo (derecha de la asig.).
- en aristas uso-q : la var se utiliza en un predicado de transferencia de control.

• $\text{def}(i)$: conjunto de variables para el cual hay una definición en el nodo i

• $c\text{-use}(i)$: conjunto de variables para el cual hay un uso-c en el nodo i

• $q\text{-use}(i,j)$: conjunto de variables para el cual hay un q-use en la arista i, j

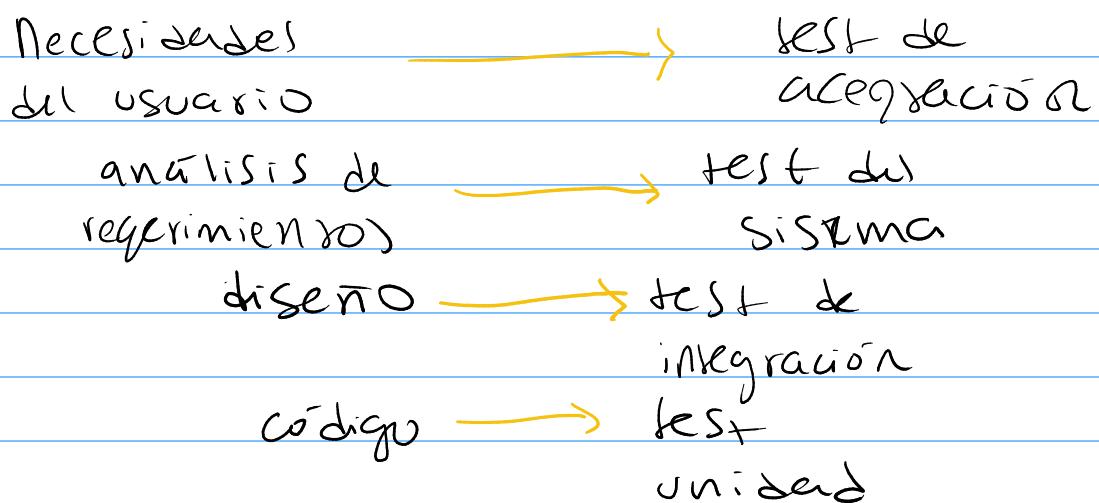
Una vez terminado el grafo de def-use podemos seleccionar los testswives usando estos criterios:

• todos las definiciones: por cada nodo i y cada x en $\text{def}(i)$ hay un camino libre de definiciones respetando hasta un uso-c o uso-q; debo recorrer todos los caminos libres de definiciones para una variable indicada, es decir aquellos donde la variable sólo se define una vez antes de usarse para cálculo o para transferencia de control.

- todos los usos-? : todos los usos-? de todas las definiciones
- todos los usos-c; algunos usos-p; algunos usos-c.

¿Qué son los niveles de testing?
¿Para qué sirven? Nombralos

Los niveles de testing son distintos tipos de testing que revelan distintos tipos de defectos. La naturaleza de estos cambia en cada fase del proyecto, por eso en cada una se realizan distintos tipos ó niveles de testing



Explique test de regresión

Es otro nivel/tipo de testing, se realiza cuando se introduce algún cambio al SW para verificar que la funcionalidad existente continúa funcionando correctamente. Si no puede realizarse el test suite completo (vez que se introduce un cambio, deben priorizarse los casos de test necesarios).

¿Por qué es importante registrar los defectos y hacerles seguimiento?

Porque los muchos defectos que tiene el SW son generalmente encontrados por muchas personas distintas, por lo que es necesario tener registros de ellos para no pasar por alto ninguno. Una vez registrados, hacerles seguimiento es importante para poder rastrearlos hasta que se cierren.

¿Cómo es el ciclo de vida de un defecto?

Los defectos son, primero, encontrados y registrados, aquí el defecto es un defecto remitido. Luego se asigna su corrección a alguien, este hace debugging y lo corrige, aquí el defecto es un defecto corregido. Esta corrección es revisada por quien lo encontró, si se agradece, el defecto pasa a ser un defecto cerrado.

¿Cuáles son las posibles categorizaciones para un defecto?

Otra forma de clasificación, según el tipo de defecto: funcional, lógicos, asignación de interés de usuario, asignación de componentes, etc.

Otra es según la severidad en términos de impacto en el SW

- **Criticó**: puede demorar el proceso; afecta a muchos usuarios
- **Mayor**: mucho impacto pero pocas soluciones provisorias; mucho esfuerzo corregirlo pero tiene menor impacto en el cronograma.
- **Menor**: defecto aislado se manifiesta raramente y tiene poco impacto
- **Cosmético**: pequeños errores sin impacto en el funcionamiento correcto del sistema

Idealmente todos deben cerrarse, hay que saber cuál de terminar cuando un producto es entregable.

PLANEAMIENTO DEL PROCESO DE SOFTWARE (Cap. 5)

¿Cuáles son los sógicos más importantes dentro del planeamiento del proc. de software?

- 1- Planeamiento del proceso de sw
- 2- Estimación del esfuerzo
- 3- Planificación de recursos humanos
- 4- Planeamiento de la admin. de config. del sw
- 5- Planeamiento del control de calidad
- 6- Administración de riesgos
- 7- Planificación del seguimiento del proyecto

Exige el planeamiento del proceso

Es uno de los tópicos más importantes dentro del planeamiento del proyecto de SW. Aquí se plantea cómo se estructurará el proyecto:

- Se determina el modelo de proceso a seguir
- Se lo adapta al proyecto
- Se definen las etapas y criterios de entrada y salida de C/U así como las actividades de verificación
- Se definen las metas principales

Exige el modelo COCOMO

Este modelo sirve para estimar el esfuerzo del proyecto durante el

✓ planeamiento del mismo. Tiene un enfoque top-down, es decir, primero se estima el esfuerzo total y en base a eso se estima el de cada parte del proyecto.

El procedimiento se plantea este modelo para estimar el esfuerzo el el siguiente:

A) Obtenir el estimado inicial usando el tamaño del proyecto en KLOC para ello usamos

$$E_i = a * \text{tamaño}^b$$

a y b son constantes que dependen del tipo de proyecto. Hay una tabla

en P/M
(persona)
mcl)

de dependiendo si el proyecto es orgánico (relativamente simple, realizable por equipos pequeños), semi-rígido o rígido (más ambicioso y novedoso, con restricciones estrictas impuestas por el entorno y otros requerimientos en cuanto a interfaz y confiabilidad) asigna valores fijos para a y b .

B) Obtener el conjunto de 15 factores multiplicativos que representan distintos atributos.

Estos atributos son: de software, como la confiabilidad (RELY); de hardware, como la volatilidad de la máquina virtual (VIRT); de personal, como la certificación de los programadores (PAB) y de proyecto, como (a) herramientas de desarrollo del sw (TOOL).

A cada uno de estos 15 atributos se le asignará un factor multiplicativo según su escala de certificaciones (muy bajo, bajo, normal, alto, muy alto y extra alto, no todos cuentan con la escala completa).

C) Ajustar el estimador de esfuerzo

Una vez que el atributo tiene su factor multiplicativo podemos ajustar nuestra estimación inicial

$$\text{Estimación del esfuerzo final} = E_i \cdot \prod_{i=1}^{18} f_i \rightarrow \text{factores multiplicativos de los 18 atributos}$$

D) Calcular la estimación del esfuerzo para cada fase

Según el tamaño en KLOC del proyecto, el estimador de esfuerzo de cada fase será un porcentaje del esfuerzo final.

Exige planificación y RRHH

Es uno de los tópicos más importantes dentro del planeamiento del proyecto del SW. Tiene dos niveles

Planeación global: abarca meses parciales y fecha final.

↓ ↗ estimar el tiempo programado del proyecto en meses (M)
duracion se determina con la distribución de los RRHH (curva de Rayleigh) $M = 2,5 * E^{0,38}$ (cociente) y la distribución de esfuerzos

Planificación detallada: Se deciden y asignan las tareas de bajo nivel (2/3 días). Para ello es necesario un equipo de trabajo estructurado.

algunas estructuras

- **Org. jerárquica:**

- Admin del proyecto o desglobo global tiene programadores, testers y admins. de config p/ ejecutar las tareas

- **Equipos democráticos:**

- Pequeños grupos

- se rota el liderazgo

- **Alternativa:**

- Tres tareas principales: desarrollo, testing y administración del programa.

- cada una tiene su equipo y un líder.

- Todos responden a un líder gral.

- Desarrollo y testing se esperan independientes

- El admin provee especificaciones y asegura que wo y test estén coordinados.

Exige planeamiento de control de calidad

Es uno de los fórmicos más importantes dentro del planeamiento del proyecto del SW. Su objetivo es especificar las actividades a realizarse para identificar y corregir/eliminar defectos, así como las herramientas y métodos qe se usarán a la ejecución.

Estas actividades son actividades QC.

Hay varios enfoques para planear el QC

- Enfoque ad-hoc: test y revisiones

cuando y como se necesite

- Enfoque de procedimiento: el plan define las tareas a realizarse, este provee los 7 herramientas para llevarlas a cabo.

- Enfoque cuantitativo: analiza los datos recopilados & los recursos para poder predecirlos en un futuro

Exige administración de riesgos

Es uno de los tópicos más importantes dentro del planeamiento del proyecto del SW. Su objetivo es minimizar el impacto de la materialización de cualquier condición o evento de ocurrencia inesperada y no común que pueda causar la caída del proyecto, es decir, la materialización de los riesgos.

Tiene dos grandes etapas:

- Evaluación de los riesgos: se realiza durante el planeamiento, aquí se identifican los posibles riesgos, mediante listas de control, experencias pasadas, brainstorming, etc. Algunos factores de riesgo son: personal insuficiente o inapropiadamente entrenado; tiempo y costos irrealistas y software legado.

hat
a

Luego se hace el análisis de riesgos y definición de las prioridades.

Los riesgos son muchos, deben priorizarse aquellos de mayor probabilidad de ocurrencia e impacto. Dos formas de ordenarlos

según el valor de exposición al riesgo (RE); eso es $RE = \text{prob. ocurrencia} * \text{impacto}$.

Deberán priorizarse los de mayor RE

Clasificando la probabilidad de ocurrencia y el impacto de la misma en Alta, Media y Baja. Deberán priorizarse los riesgos con clasificación AA y AM/MA

Control de riesgos: se plantea un plan de mitigación de riesgos que incluye los pasos a realizar para reducir la probabilidad o el impacto negativo de un riesgo. Tmb se definen las acciones a realizar en caso de materializarse.

Exige planificación del seguimiento del proyecto

Es uno de los tópicos más importantes dentro del manejo del proyecto del SW. Su objetivo es hacer visible la ejecución del proyecto de manera de realizar acciones correctivas cuando sea necesario para asegurar el éxito del proyecto.

Se plantean qué medir, cómo y cuándo.
Principales medidas: tiempo, costos, recursos
y tamaño.

Hay distintos niveles de seguimiento

- Nivel de actividad: se supervisa las actas. Se realizan en tiempo y forma.
- Reporte de estado: resumen de las actas complejas y tendencias des de el último seguimiento.
- Análisis de metas garantiales: se revisan las metas garantiales.