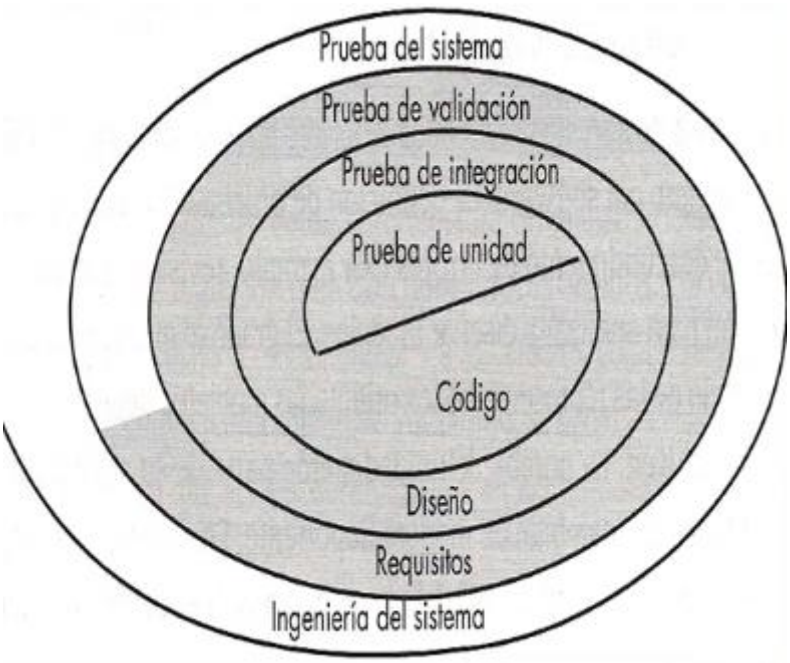
Ingeniería de Software 2 : Clase 9 – Estrategias de pruebas

Enfoque estratégico de pruebas

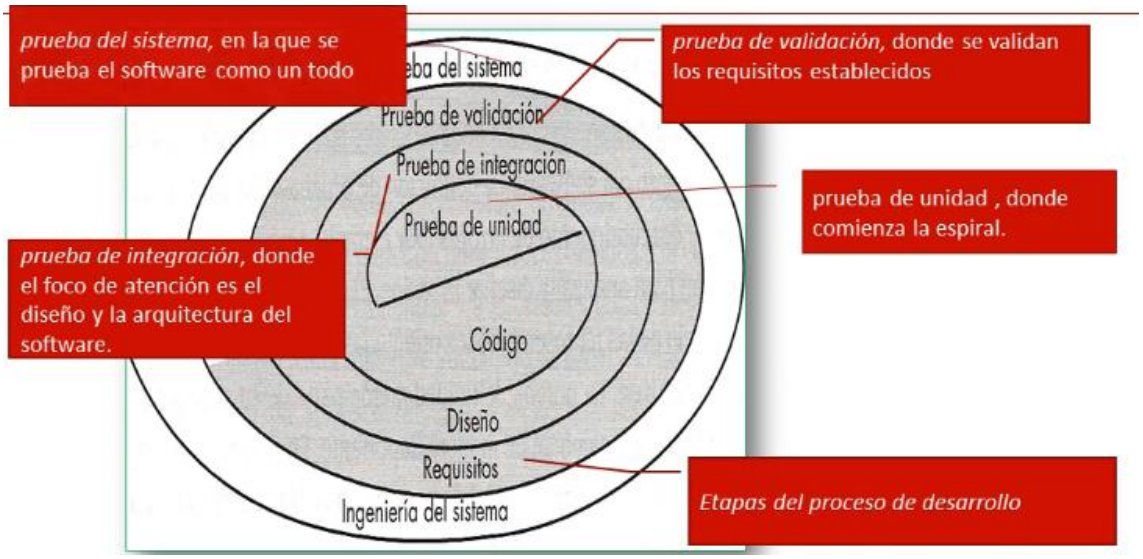
- Una estrategia de pruebas del software nos proporciona una guía que describe pasos a seguir, cuándo son planeados y llevados a cabo, el esfuerzo que implican, tiempo, y recursos requeridos.
    - Proporciona
        - Planificación de pruebas.
        - Diseño de casos para probar.
        - Ejecución de las pruebas.
        - Recolección y evaluación de datos resultado.
- Es un conjunto de actividades planeadas con anticipación, se realizan de forma sistemática.
- Conjunto de pasos con técnicas y métodos específicos de diseño de casos de prueba.
- Incluye pruebas de bajo nivel y alto nivel.
- Las actividades de las pruebas forman parte de la verificación y validación del aseguramiento de la calidad del software.

Verificación y Validación

- Verificación: conjunto de actividades que asegura que el software implemente de forma correcta una función específica.
    - Construimos el producto correctamente?
- Validación: conjunto de actividades que aseguran que el software corresponde con lo que pide el cliente.
    - Construimos el producto correcto? -> contra la especificación de requerimientos.



Ingeniería de Software 2 : Clase 9 – Estrategias de pruebas



Tipos de pruebas de software convencionales

- Prueba de unidad:
  - Verificación de que el componente funciona de forma correcta ante el ingreso de diferentes casos de prueba.
- Prueba de integración:
  - Se verifica que los componentes trabajan correctamente en conjunto.
- Prueba de validación:
  - Nos dará una seguridad final de que el software satisface todos los requisitos funcionales/no funcionales.
- Prueba del sistema:
  - Se verifica que cada elemento encaje de forma adecuada y que se alcanza la funcionalidad y el rendimiento del sistema en su totalidad.

Pruebas de Unidad

- Se prueba la interfaz del módulo para asegurar que la información viaja adecuadamente.
- Se examinan estructuras de datos locales.
- Se prueban condiciones límite para asegurar el correcto funcionamiento.
- Se ejercitan los caminos de ejecución independientes.
- Errores comunes detectados en este tipo de pruebas:
  - Calculos incorrectos.
    - Predecesores aritméticos incorrectamente aplicados.
    - Operaciones mezcladas.
    - Inicialización incorrecta.
    - Falta de precisión.

        - Representación simbólica errónea.
    - Comparaciones erróneas.
    - Flujos de control inapropiados
- Los casos de prueba para este tipo deben descubrir estos errores:
    - Comparaciones entre diferentes tipos de datos
    - Operadores lógicos aplicados incorrectamente
    - Expectativas de igualdad con grado de precisión
    - Comparación incorrecta de variables
    - Terminación inapropiada o inexistente de bucles
    - Falla en la salida cuando se encuentre una iteración divergente
    - Variables de bucle modificadas inapropiadamente

- Procedimiento de las pruebas de unidad
    - Los componentes a probar no son programas independientes, se debe desarrollar para cada prueba de unidad un software que controle y/o resguarde la prueba.
    - Un controlador es un programa principal que acepta los datos del caso de prueba, pasa estos datos al módulo a ser probado, mostrando los resultados.
    - Un resguardo nos sirve para reemplazar a módulos subordinados al componente que hay que probar.
    - Los controladores y resguardos son trabajo extra.
    - Entonces hay que realizar controladores y resguardos sencillos, para que el trabajo extra sea poco.
    - La prueba de unidad es simplificada cuando se diseña un módulo altamente cohesivo.
        - Una única tarea para un módulo, sin mezclar tareas que no tienen relación entre sí.

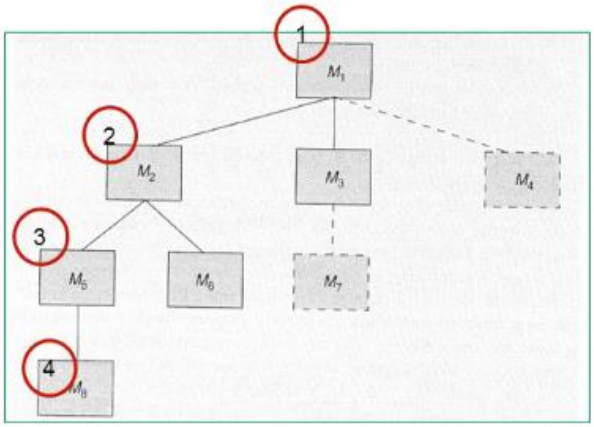
Pruebas de integración

- Se toman los componentes que pasaron las pruebas de unidad y se combina dichos componentes según el diseño que establecemos.
- En esta combinación puede ocurrir:
    - Los datos se pueden perder en una interfaz.
    - Un módulo puede tener un efecto adverso e inadvertido sobre otro.

- La combinación de subfunciones no produzca el resultado que esperamos.

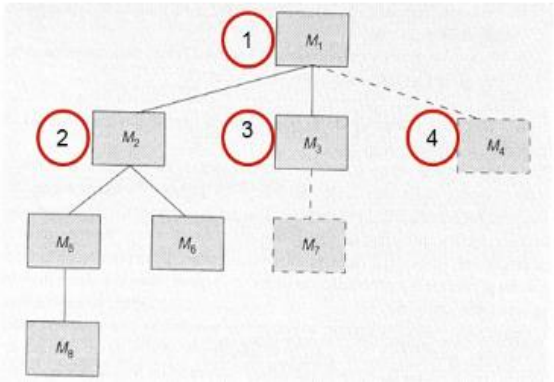
- El programa es construido y probado en pequeños segmentos en los que los errores son más fáciles de aislar y corregir.
- La integración se puede dar de dos formas distintas:
  - Descendente: los módulos se integran al descender por la jerarquía de control, iniciando por el programa principal.
    - En profundidad: primero en profundidad integra todos los módulos de un camino de control principal de la estructura.

Pasos:
1. Conductor: Módulo principal
   Resguardos: Para los módulos subordinados
2. Sustituir resguardos por módulos 1 a 1
3. Probar
4. Reemplazar otro resguardo
5. Pruebas de regresión



    - En anchura: primero en anchura incorpora todos los módulos directamente subordinados a cada nivel de la jerarquía.

Pasos:
1. Conductor: Módulo principal
   Resguardos: Para los módulos subordinados
2. Sustituir resguardos por módulos 1 a 1
3. Probar
4. Reemplazar otro resguardo
5. Pruebas de regresión



  - Ascendente:
    - La prueba arranca con módulos atómicos (módulos en el nivel más bajo de la estructura jerarquica).
    - El proceso requerido de los módulos subordinados siempre está disponible y se elimina la necesidad de resguardos,

pero no de módulos conductores.

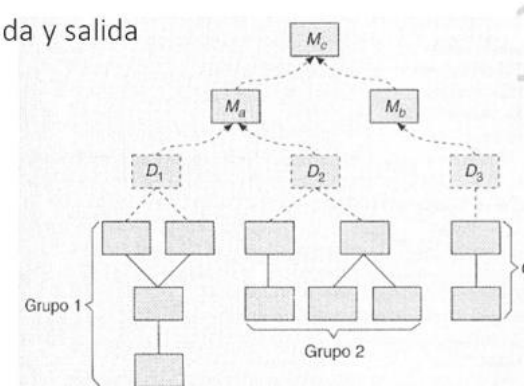
Pasos :

»1.Combinar módulos de bajo nivel

»2.Hacer conductor para coordinar entrada y salida

»3. Probar el grupo

»4.Eliminar conductores



- Hay una discusión sobre las ventajas y desventajas de los enfoques
  - Descendente: la desventaja es necesitar resguardos.
  - Ascendente: la desventaja es que el programa como entidad no existe hasta que se agrega el último módulo.
- La elección de enfoque depende de las características del software.
- Un enfoque que combine ambos (prueba sandwich) puede ser la mejor opción.

Pruebas de Regresión

- Cuando agregamos un nuevo módulo como parte de una prueba de integración, el software cambia.
- Surgen nuevos caminos, que pueden derivar a nuevas E/S y se invoca una nueva lógica de control.
- Los cambios surgidos pueden llevarnos a problemas con funciones que antes trabajaban de forma perfecta.
- **La prueba de regresión consiste en volver a ejecutar un subconjunto de pruebas que se han llevado a cabo anteriormente para asegurarse de que los cambios no han propagado efectos colaterales no deseados.**
- Las pruebas de regresión pueden ser realizadas de forma manual volviendo a realizar el subconjunto de casos de prueba o utilizando herramientas automáticas.
- El conjunto de pruebas de regresión contiene tres clases diferentes de casos de prueba:
  - Una muestra de pruebas que ejercite todas las funciones del software.
  - Pruebas adicionales que se centren en las funciones del software que son probablemente afectadas por el cambio.
  - Pruebas que se centren en los componentes del software que han cambiado.

Ingeniería de Software 2 : Clase 9 – Estrategias de pruebas

- Se deben identificar módulos críticos:
    - Abordan muchos requisitos de software.
    - Tienen un alto nivel de control.
    - Es complejo o proclive a errores.
    - Tiene requerimientos de rendimientos definidos.
    - Los módulos críticos deben probarse lo antes posibles. Las pruebas de regresión ponen la lupa sobre ellos.

Pruebas de Unidad e Integración para software creado en paradigma OO

- Prueba de unidad:
    - Generalmente una clase encapsulada es el foco de la prueba de unidad.
    - Los métodos son las unidades comprobables más pequeñas.
    - La prueba de clase es el equivalente en este caso, la cual debe ser dirigida a las operaciones encapsuladas por la clase y el comportamiento de estado de ésta.
- Prueba de integración:
    - El software OO no tiene estructura de contro jerárquica obvia.
    - La prueba basada en hebra integra el conjunto de clases requeridas para responder a una entrada/evento.
    - La prueba basada en uso comienza con las clases independientes, luego las dependientes.

Ingeniería de Software 2 : Clase 9 – Estrategias de pruebas

Pruebas del Sistema

- Se constituye por una serie de pruebas diferentes.
- Cada prueba tiene su propósito, todas trabajan para verificar la correcta integración de todos los componentes del sistema.
- Pruebas de recuperación:
  - Se controla la recuperación de fallas (reacción ante fallas) y el modo de reanudación del procesamiento en un tiempo determinado.
  - Se fuerza el fallo para comprobarlo.
- Pruebas de seguridad:
  - Se comprueban los mecanismos de protección integrados.
- Pruebas de resistencia (Stress):
  - Se diseñan para enfrentar a los programas a situaciones anormales.
- Pruebas de rendimiento:
  - Se prueba el sistema en tiempo de ejecución. A veces va emparejada con la prueba stress.

Pruebas de Validación

- La validación del software es conseguida mediante una serie de pruebas que demuestren la conformidad del cliente con los requisitos.
- Cuando se procede con cada caso de prueba de validación:
  - Las características de funcionamiento o de rendimiento están de acuerdo con las especificaciones y son aceptables;
  - O
  - Se descubre una desviación de las especificaciones y se crea una lista de deficiencias.
- Comienzan cuando terminan las pruebas de integración.
- Se revisa la configuración:
  - Se asegura que todos los elementos de la configuración del software se desarrollaron correctamente, estén catalogados y contengan detalle suficiente para reforzar el soporte.
- Pruebas de Aceptación (ALFA y Beta)
  - Las realiza el usuario final en lugar de los desarrolladores. Puede ser desde algo informal hasta algo sistemático de pruebas bien planificadas.
  - Dentro de las pruebas de aceptación podemos hallar:
    - Pruebas ALFA: desarrolladores con clientes antes de lanzar el producto.

        - Pruebas BETA: seleccionando los clientes que efectuarán la prueba. El desarrollador no se encuentra presente.

- Aceptación ALFA:
    - Se llevan a cabo por un cliente en el lugar donde se desarrolla el sistema.
    - Se usa el software de forma natural con el desarrollador como observador, registrando errores y problemas de uso.
    - Se hacen en un entorno controlado.
    - Se realizan después de que todos los procedimientos de prueba básicos, pruebas unitarias e integración se han completado, y se produce después de las pruebas del sistema.
    - Esta no es la versión final de software y cierta funcionalidad puede ser añadido al software incluso después de las pruebas alfa.
- Aceptación BETA:
    - Se llevan a cabo por usuarios finales del software en lugares de trabajo del cliente.
    - El desarrollador no se encuentra presente en esta prueba. Se prueba en un entorno no controlado por el programador.
    - El cliente registra los problemas que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador.
    - La prueba beta es la última fase de las fases de prueba y se hace utilizando técnicas de caja negra.
    - A veces la versión beta también se libera al mercado, y en base a los comentarios de los usuarios se hacen modificaciones.

Ingeniería de Software 2 : Clase 9 – Estrategias de pruebas

Depuración

- La depuración de programas, es el proceso de identificar y corregir errores en programas informáticos.
- La depuración **no** es una prueba, pero ocurre siempre como consecuencia de la prueba efectiva.
    - Se descubre un error, depurando elimino dicho error.
- El proceso de depurar tiene dos resultados posibles:
    - Hallamos la causa, la corregimos y se elimina el error.
    - No hallamos la causa. La persona que realiza la depuración debe sospechar la causa, en ese caso es necesario diseñar un caso de prueba que ayude a confirmar las sospechas, el trabajo en este caso vuelve atrás a la corrección del error en una forma iterativa.
- Características de los errores que atentan contra la facilidad de la depuración:
    1. Síntoma lejano (geográficamente) de la causa
    2. Síntoma desaparece temporalmente al corregir otro error
    3. Síntoma producido por error
    4. Síntoma causado por error humano
    5. Síntoma causado por problemas de tiempo
    6. Condiciones de entrada difíciles de reproducir
    7. Síntoma intermitente (especialmente en desarrollos hardware-software)
    8. El síntoma se debe a causas distribuidas entre varias tareas que se ejecutan en diferentes procesadores

Enfoques de la depuración

- Diseñar programas de prueba adicionales que repitan la falla original de forma que ayude a descubrir la fuente de la falla en el programa.
- Rastrear el programa de forma manual y simular ejecución.
- Usar herramientas interactivas.
- Una vez que se corrige el error se debe reevaluar el sistema: volver a hacer las inspecciones y repetir las pruebas (pruebas de regresión).

Prueba de entornos especializados

- A medida que el software se vuelve complejo, crece también la necesidad de enfocar pruebas especializadas.
- Pruebas de interfaces gráficas.
- Pruebas de arquitecturas cliente-servidor.
- Pruebas de la documentación y las ayudas.

Ingeniería de Software 2 : Clase 9 – Estrategias de pruebas

- Pruebas de sistema en tiempo real.

Prueba de arquitectura cliente-servidor

- Prueba de funcionalidades de la aplicación.
- Prueba de servidor:
  - Probar las funciones de coordinación y manejo de datos del servidor.
  - Probar como se desempeña el servidor (tiempo de respuesta y procesamiento total de datos).
- Prueba de BD:
  - Probar exactitud e integridad de datos, examinar transacciones, asegurar que se almacenan, actualizan y recuperan los datos.
- Prueba de transacciones:
  - Se crea una serie de pruebas para asegurar que cada transacción se procede de acuerdo a los requisitos.
- Prueba de comunicación de red:
  - Se verifica la comunicación entre los nodos, que el paso de mensajes, transacciones y tráfico de la red sea realizado sin errores.

Pruebas de documentación y funciones de ayuda

- Es importante para la aceptación del programa por parte del cliente.
- Se revisa la guía del usuario o funciones de ayuda en línea.
- Prueba de documentación en dos fases:
  - Revisar e inspeccionar:
    - Examinar la claridad del documento.
  - Prueba en vivo:
    - Usar documentación junto al programa real.

Pruebas de sistemas de tiempo real

- El diseño de casos de prueba, además de los convencionales deben incluir manejo de eventos (interrupciones), temporización de datos, paralelismo entre tareas, etc.
  - Pruebas de tareas
    Probar tareas de forma independiente, en búsqueda de errores lógicos y funcionamiento.
  - Pruebas de comportamiento
    Simular el comportamiento del sistema de tiempo real y examinarlo como consecuencia de eventos.

- Pruebas inter-tareas
  Se prueban las tareas asincrónicas entre las cuales se sabe que hay comunicación.
- Pruebas de sistemas
  Se prueba el software y hardware integrados.