Departamento de Ingeniería Informática
Grado en Ingeniería Informática
Elisa Guerrero Vázquez
Esther L. Silva Ramírez

Metodología de la Programación

Tema 5 – Teoría

PRUEBA DEL SOFTWARE

Ciclo de vida del software

- **Análisis**, en esta etapa se analiza y define el problema, especificando el qué queremos. A su vez se distinguen dos fases:

  - Análisis del sistema (análisis y definición del problema).

  - Especificación de requisitos.

- **Diseño**, en esta etapa se determina cómo se van a realizar los requisitos recogidos y organizados.
- **Implementación** o codificación. En esta etapa, se traduce el algoritmo diseñado a un lenguaje de programación.
- **Pruebas**. En esta etapa se realizan pruebas del programa para detectar errores.
- **Mantenimiento.** Se modifica el programa para corregir errores o para realizar mejoras y adaptación del programa.

1. Técnicas de prueba del software

- Las pruebas deben estar bien planificadas, dado el alto coste económico que lleva consigo un fallo.
- Para las pruebas se ejecuta el programa utilizando datos similares a los datos reales, detectando si se genera error.
- Prueba <> depuración, no son lo mismo.

  Prueba = confirma la existencia de errores.

  Depuración = localiza y corrige dichos errores.

1. Técnicas de prueba del software

- La prueba de programas sólo demuestra la presencia de errores, pero no la ausencia de estos.
- Habría que probar para todos los posibles datos de entrada para confirmar corrección.
- Una alternativa consiste en obtener información “a priori” sobre el comportamiento del programa mediante el análisis de su propio texto =**Verificación**.
- Inabordable en sistemas software, por su tamaño y complejidad.
- **Diseñar un buen plan de pruebas, y que lo lleve a cabo personas distintas a los diseñadores**.

Proceso de prueba

- Las tareas de pruebas tienen éxito si con ellas se detecta la presencia de errores.
- Las pruebas consisten en:

a) Ejecutar el programa utilizando datos similares a los datos reales.

b) Observar los resultados y compararlos con los resultados deseados.

c) A partir de este estudio concluir la existencia de errores o insuficiencias del programa, para posteriormente subsanarlos en el proceso de depuración.

Proceso de prueba

- Si se obtienen errores poner en duda la calidad y fiabilidad del software.

- Si no se obtienen errores, puede ocurrir:
  a) Que realmente la calidad y fiabilidad del software sean aceptables.
  b) El plan de pruebas diseñado no sea adecuado.

Las recomendaciones de [Myers79] para las pruebas, son las siguientes:

- Cada caso de prueba debe acompañarse del resultado deseado.
- El programa debe ser probado por personas diferentes a los programadores.
- Al diseñar el plan de prueba se deben seleccionar casos de entrada válidos y no válidos.
- El plan de prueba debe diseñarse pensando en que existen errores en el programa y que hay que detectarlos.
- Es recomendable:
  - Probar si el software no hace lo que debe hacer.
  - Probar si el software hace lo que no debe hacer.

Técnicas de diseño de casos de pruebas

- Cómo es imposible probar el programa para todos los posibles datos de entradas, se eligen algunos como representativos del resto.
- Las pruebas se pueden hacer de dos formas:

  a) Pruebas de caja blanca: desarrollar pruebas que aseguren que la integración de todos los componentes del programa es correcta, es decir, que cada uno de estos componentes cumple con sus especificaciones y ha sido probado adecuadamente. Para ello se centra en la implementación del programa, eligiendo los casos de prueba que garanticen todo los posibles caminos de ejecución que pueden trazarse.

Técnicas de diseño de casos de pruebas

b) Pruebas de caja negra: se realizan pruebas para determinar que cada función realiza la tarea para la que ha sido creada. Se analiza la especificación de las funciones, su entrada y salida, y partir de esa información se seleccionan las posibles entradas al programa y se comprueba que las salidas son las esperadas.

Ambos enfoques no son excluyentes entre sí, sino complementarios.

- Las pruebas de caja blanca se deben centrar en cubrir el control de flujo del programa. Una técnica de prueba de caja blanca es la Prueba de la Ruta Básica. Esta técnica permite definir un conjunto básico de rutas de ejecución, obteniendo un conjunto de casos de prueba que garanticen la ejecución de cada instrucción al menos una vez durante la prueba.

- Se usa el Diagrama de Control de Flujo (CFD) para buscar las distintas rutas independientes. Para encontrar cuántas rutas existen se calcula la complejidad ciclomática del CFD:
  - NA-NN+2
  - NNP+1

  Siendo NA – número de aristas, NN – número de nodos y NNP – número de nodos predicados (contiene una condición).

- Pasos a seguir para realizar la prueba de ruta básica:
  - Generar el CFD.
  - Determinar la complejidad ciclomática.
  - Determinar el conjunto básico de rutas linealmente independientes.
  - Preparar los casos de prueba para forzar la ejecución de cada ruta del conjunto básico.

- La prueba de bucles es fundamental.
- Considerando *n* el número máximo de iteraciones:
  - Probar que no se cumpla la condición del bucle.
  - Probar una única iteración.
  - Probar dos iteraciones.
  - Probar $m<n$ iteraciones.
  - Probar $n-1$, n, $n+1$ iteraciones.
  - Si se trata de bucles anidados comenzar realizando las pruebas descritas en 1 a 5 comenzando en el más interno.

- Las pruebas de caja negra se deben centrar en buscar errores de tipo funciones incorrectas o ausentes, de interfaz, en estructuras de datos, de funcionamiento, etc. Para ello podemos crear casos de prueba en función de:
  - Si un dato de entrada está en un rango de valores entre A y B, diseñar un caso de prueba para cada uno de estos valores extremos, además de para un valor justo antes de A y justo después de B.
  - Si un dato de entrada puede contener varios valores, diseñar casos de prueba para el máximo, para el mínimo y para los valores justo por encima y por debajo de ambos.
  - Si se trata de un valor lógico, diseñar casos de prueba para valor falso y valor verdadero.
  - Si el valor de entrada representa un conjunto, diseñar casos de prueba para un valor que pertenezca y otro que no pertenezca al conjunto.
  - Diseñar de las misma forma los casos de prueba para los valores de salida.

2. Estrategias de prueba del software

- Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultados una correcta construcción del software.
- Proporciona una guía tanto para el programador como para la organización de control de calidad y el cliente.
- Una estrategia para la prueba del software puede seguir las siguientes etapas:

2. Estrategias de prueba del software

- Prueba de unidad. Se prueba cada componente y se comprueba su funcionamiento.
- Prueba de integración. Una vez probados todos los elementos y dando por supuesto que son correctos, se integran para comprobar las interfaces entre los componentes del software.
- Prueba funcional o de validación. El software completamente ensamblado se prueba como un todo, se comprueba si cumple o no los requisitos funcionales, requisitos de rendimiento, de comportamiento, de seguridad, etc. En resumen, se comprueba que no haya diferencias entre el software y los requisitos especificados para su funcionamiento.

- Prueba del sistema. El software ya validado se integra con el resto de los componentes del sistema (elementos mecánicos, interfaces electrónicas, etc.), se prueba que funciona el software y el resto de elementos del sistema en su conjunto, alcanzando la funcionalidad y rendimiento del sistema.

- Prueba de aceptación. Esta prueba la efectúa el usuario en su propio entorno, con datos reales, y es quien decide si el software cumple los requisitos fijados por él, y en definitiva, si lo acepta o no.

Bibliografía

- Boehm, B. W. *Software Engineering Economics.* Prentice-Hall, 1981.
- Myers, Glenn J. *The Art of Software Testing*, John Wiley and Sons, New York,1979.
- Piattini, Mario G., Calvo-Manzano, José A.,Cervera, Joaquín, Fernández, Luis. *Análsis y Diseño detallado de Aplicaciones Informáticas de Gestión*. RA-MA, España, 1996
- Pressman, Roger S. *Ingeniería del Software*. Un enfoque práctico. McGraw-Hill, España, 3ª Edición, 1996.
- Sommerville, Ian. *Ingeniería del Software*. Addison-Wesley Iberoamericana, México, 2ª Edición, 1988.