



UD3: DISEÑO Y DESARROLLO DE PRUEBAS DE SOFTWARE

ENTORNOS DE DESARROLLO



Arangoya
1974

Ander González Ortiz
C.E. Arangoya
ander.gonzalez@arangoya.net

INTRODUCCIÓN

Las pruebas de software consisten en **verificar y validar un producto software** antes de su puesta en marcha. Constituyen una de las etapas del desarrollo de software, y básicamente consiste en probar la aplicación construida. Se integran dentro de las diferentes fases del **ciclo de vida** del software dentro de la **ingeniería de software**.

En este Capítulo se estudian dos enfoques para el diseño de **casos de prueba** y diferentes técnicas en cada uno para probar el código de los programas.

TÉCNICAS DE DISEÑO DE CASOS DE PRUEBA

Un caso de prueba es un conjunto de entradas, condiciones de ejecución y resultados esperados, desarrollado para conseguir un **objetivo particular** o **condición de prueba**. Para llevar a cabo un caso de prueba, es necesario definir las **precondiciones y post condiciones**, identificar unos valores de entrada y conocer el comportamiento que debería tener el sistema ante dichos valores.

Tras realizar ese análisis e introducir dichos datos en el sistema, se observará si su comportamiento es el previsto o no y por qué. De esta forma se determinará si el sistema ha pasado o no la prueba.

TÉCNICAS DE DISEÑO DE CASOS DE PRUEBA

Para llevar a cabo el diseño de casos de prueba se utilizan dos técnicas o enfoques: prueba de **caja blanca** y prueba de **caja negra**.

Las primeras se centran en validar la estructura interna del programa (necesitan conocer los detalles procedimentales del código) y las segundas se centran en validar los requisitos funcionales sin fijarse en el funcionamiento interno del programa (necesitan saber la funcionalidad que el código ha de proporcionar).

Estas pruebas **no son excluyentes y se pueden (deben) combinar** para descubrir diferentes tipos de errores.

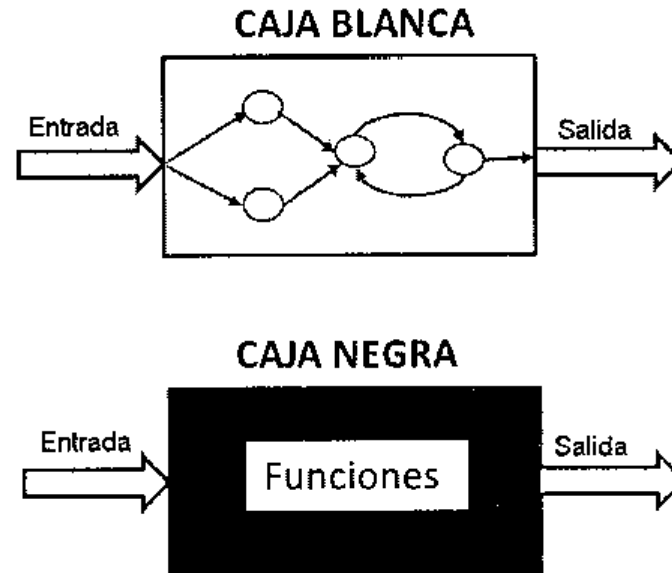
TÉCNICAS DE DISEÑO DE CASOS DE PRUEBA

Para llevar a cabo el diseño de casos de prueba se utilizan dos técnicas o enfoques: prueba de **caja blanca** y prueba de **caja negra**.

Las primeras se centran en validar la estructura interna del programa (necesitan conocer los detalles procedimentales del código) y las segundas se centran en validar los requisitos funcionales sin fijarse en el funcionamiento interno del programa (necesitan saber la funcionalidad que el código ha de proporcionar).

Estas pruebas **no son excluyentes y se pueden (deben) combinar** para descubrir diferentes tipos de errores.

TÉCNICAS DE DISEÑO DE CASOS DE PRUEBA



PRUEBAS DE CAJA BLANCA

También se las conoce como pruebas estructurales o de caja de cristal. Se basan en el minucioso examen de los detalles procedimentales del código de la aplicación. Mediante esta técnica se pueden obtener casos de prueba que:

- Garanticen que se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Ejecuten todas las sentencias al menos una vez.
- Ejecuten todas las decisiones lógicas en su parte verdadera y en su parte falsa.
- Ejecuten todos los bucles en sus límites.
- Utilicen todas las estructuras de datos internas para asegurar su validez.

Una de las técnicas utilizadas para desarrollar los casos de **prueba de caja blanca** es la prueba del **camino básico** que se estudiará más adelante.

PRUEBAS DE CAJA NEGRA

Estas pruebas se llevan a cabo sobre la interfaz del software, no hace falta conocer la estructura interna del programa ni su funcionamiento. Se pretende obtener casos de prueba que demuestren que las funciones del software son operativas, es decir, que las salidas que devuelve la aplicación son las esperadas en función de las entradas que se proporcionen.

A este tipo de pruebas también se les llama **prueba de comportamiento**. El sistema se considera como una caja negra cuyo comportamiento sólo se puede determinar estudiando las entradas y las salidas que devuelve en función de las entradas suministradas.

PRUEBAS DE CAJA NEGRA

Con este tipo de pruebas se intenta encontrar errores de las siguientes categorías:

- Funcionalidades incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y finalización.

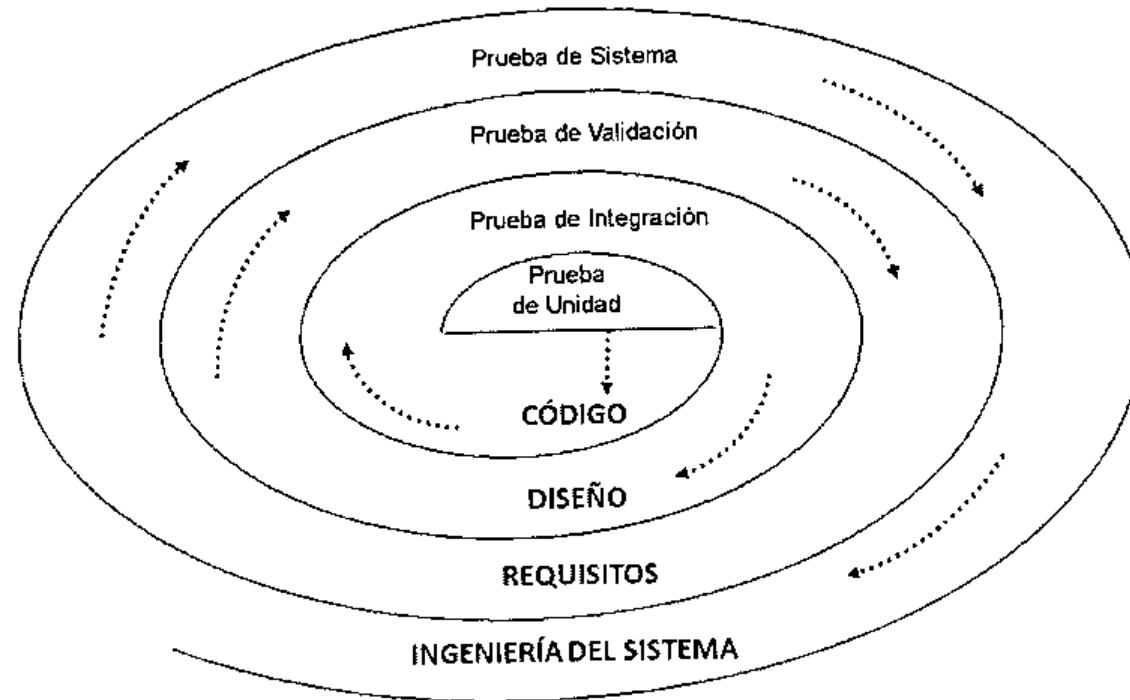
Existen diferentes técnicas para confeccionar los casos de prueba de caja negra, algunos son: clases de equivalencia, análisis de valores límite, métodos basados en grafos, pruebas de comparación, etc. En este capítulo se estudiarán algunas de estas técnicas.

ESTRATEGIAS DE PRUEBAS DEL SOFTWARE

La estrategia de prueba del software se puede ver en el contexto de una espiral :

- En el vértice de la espiral comienza la **prueba de unidad**. Se centra en la unidad más pequeña de software, el módulo tal como está implementado en código fuente.
- La prueba avanza para llegar a la **prueba de integración**. Se toman los módulos probados mediante la prueba de unidad y se construye una estructura de programa que esté de acuerdo con lo que dicta el diseño. El foco de atención es el diseño.

ESTRATEGIAS DE PRUEBAS DEL SOFTWARE



ESTRATEGIAS DE PRUEBAS DEL SOFTWARE

- La espiral avanza llegando a la **prueba de validación** (o de aceptación). Prueba del software en el entorno real de trabajo con intervención del usuario final. Se validan los requisitos establecidos como parte del análisis de requisitos del software, comparándolos con el sistema que ha sido construido.
- Finalmente se llega a la **prueba del sistema**. Verifica que cada elemento encaja de forma adecuada y se alcanza la funcionalidad y rendimiento total. Se prueba como un todo el software y otros elementos del sistema.

PRUEBA DE UNIDAD

En este nivel se prueba cada unidad o módulo con el objetivo de eliminar errores en la interfaz y en la lógica interna. Esta actividad utiliza técnicas de caja negra y caja blanca, según convenga para lo que se desea probar. Se realizan pruebas sobre:

- La interfaz del módulo, para asegurar que la información fluye adecuadamente.
- Las estructuras de datos locales, para asegurar que mantienen su integridad durante todos los pasos de ejecución del programa.
- Las condiciones límite, para asegurar que funciona correctamente en los límites establecidos durante el proceso.
- Todos los caminos independientes de la estructura de control, con el fin de asegurar que todas las sentencias se ejecutan al menos una vez.
- Todos los caminos de manejo de errores.

PRUEBA DE INTEGRACIÓN (I)

En este tipo de prueba se observa como interaccionan los distintos módulos. Se podría pensar que esta prueba no es necesaria, ya que, si todos los módulos funcionan por separado, también deberían funcionar juntos. Realmente el problema está aquí, en comprobar si funcionan juntos.

Dos enfoques para llevar a cabo las pruebas:

- **Integración no incremental o *big bang*.** Se prueba cada módulo por separado y luego se combinan todos de una vez y se prueba todo el programa completo. En este enfoque se encuentran gran cantidad de errores y la corrección se hace difícil.

PRUEBA DE INTEGRACIÓN (II)

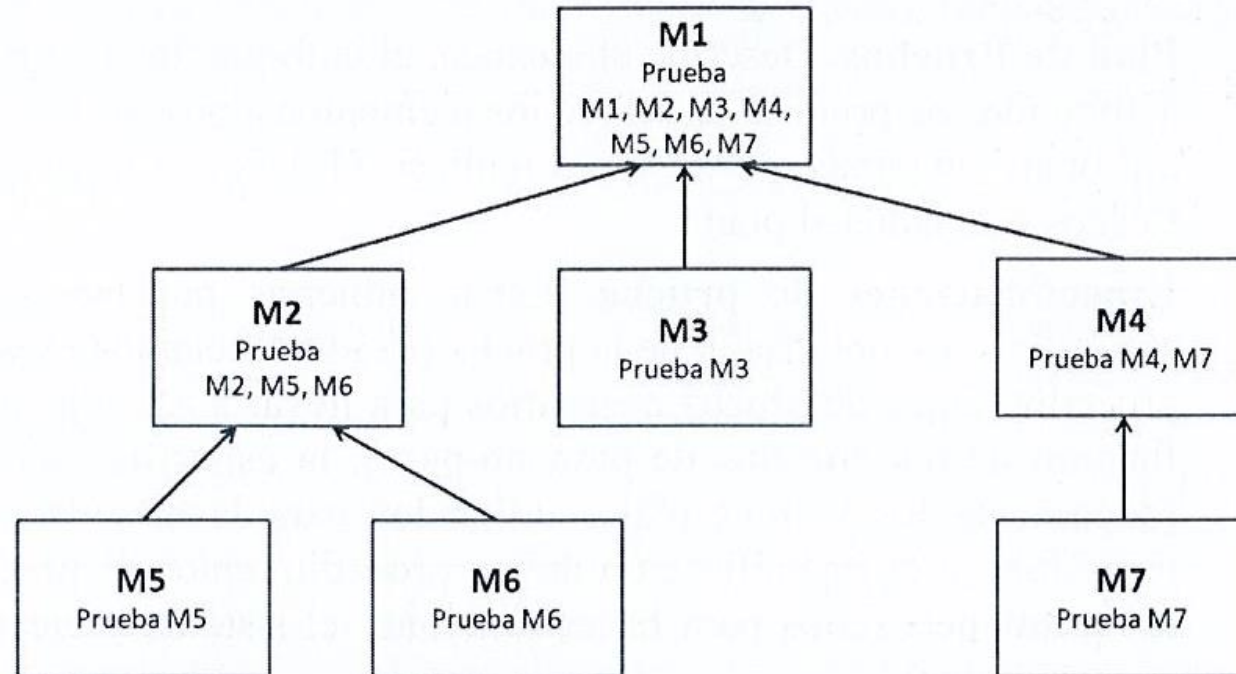
- **Integración incremental.** El programa completo se va construyendo y probando en pequeños segmentos, en este caso los errores son más fáciles de localizar.

Se dan dos estrategias Ascendente y Descendente.

- En la integración **Ascendente** la construcción y prueba del programa empieza desde los módulos de los niveles más bajos de la estructura del programa.
- En la **Descendente** la integración comienza en el módulo principal (programa principal) moviéndose hacia abajo por la jerarquía de control.

PRUEBA DE INTEGRACIÓN (III)

La figura muestra una estrategia de integración Ascendente. El módulo principal es el que está en la raíz, M1. Se empieza probando los módulos de más bajo nivel en la jerarquía modular del sistema y se procede a probar la integración de abajo hacia arriba hasta llegar al programa principal M1.



PRUEBA DE VALIDACIÓN

La validación se consigue cuando el software funciona de acuerdo con las expectativas razonables del cliente definidas en el documento de especificación de requisitos del software o ERS. Se llevan a cabo una serie de pruebas de caja negra que demuestran la conformidad con los requisitos. Las técnicas que utilizaremos son:

- **Prueba Alfa.** Se lleva a cabo por el cliente o usuario en el lugar de desarrollo. El cliente utiliza el software de forma natural bajo la observación del desarrollador que irá registrando los errores y problemas
- **Prueba Beta.** Se lleva a cabo por los usuarios finales del software en su lugar de trabajo. El desarrollador no está presente. El usuario registra todos los problemas que encuentra, reales y/o imaginarios, e informa al desarrollador en los intervalos definidos en el plan de prueba. Como resultado de los problemas informados el desarrollador lleva a cabo las modificaciones y prepara una nueva versión del producto de uso.

PRUEBA DEL SISTEMA

La prueba del sistema está formada por un conjunto de pruebas cuya misión es ejercitar profundamente el software. Son las siguientes:

- **Prueba de recuperación.** En este tipo de prueba se fuerza el fallo del software y se verifica que la recuperación se lleva a cabo apropiadamente.
- **Prueba de seguridad.** Esta prueba intenta verificar que el sistema está protegido contra accesos ilegales.
- **Prueba de resistencia (Stress).** Trata de enfrentar el sistema con situaciones que demandan gran cantidad de recursos, por ejemplo, diseñando casos de prueba que requieran el máximo de memoria, incrementando la frecuencia de datos de entrada, que den problemas en un sistema operativo virtual, etc.

DOCUMENTACIÓN PARA LAS PRUEBAS

El estándar IEEE 829-1998 describe el conjunto de documentos que pueden producirse durante el proceso de prueba. Son los siguientes:

- **Plan de Pruebas.** Describe el alcance, el enfoque, los recursos y el calendario de las actividades de prueba. Identifica los elementos a probar, las características que se van a probar, las tareas que se van a realizar, el personal responsable de cada tarea y los riesgos asociados al plan.

DOCUMENTACIÓN PARA LAS PRUEBAS

- Especificaciones de prueba. Están cubiertas por tres tipos de documentos:
 - La especificación del diseño de la prueba: se identifican los requisitos, casos de prueba y procedimientos de prueba necesarios para llevar a cabo las pruebas y se especifica la función de los criterios de pasa no-pasa.
 - La especificación de los casos de prueba: documenta los valores reales utilizados para la entrada, junto con los resultados previstos.
 - La especificación de los procedimientos de prueba: donde se identifican los pasos necesarios para hacer funcionar el sistema y ejecutar los casos de prueba especificados.

DOCUMENTACIÓN PARA LAS PRUEBAS

- Informes de pruebas. Se definen cuatro tipos de documentos:
 - un informe que identifica los elementos que están siendo probados
 - un registro de las pruebas: donde se registra lo que ocurre durante la ejecución de la prueba
 - un informe de incidentes de prueba: describe cualquier evento que se produce durante la ejecución de la prueba que requiere mayor investigación
 - un informe resumen de las actividades de prueba

PRUEBAS DE CÓDIGO

La prueba del código consiste en la ejecución del programa (o parte de él) con el objetivo de encontrar errores. Se parte para su ejecución de un conjunto de entradas y una serie de condiciones de ejecución; se observan y registran los resultados y se comparan con los resultados esperados. Se observará si el comportamiento del programa es el previsto o no y por qué.

Para las pruebas de código se van a mostrar **diferentes técnicas que dependerán del tipo de enfoque utilizado**: de caja blanca, se centran en la estructura interna del programa; o de caja negra, más centrado en las funciones, entradas y salidas del programa.


PRUEBA DEL CAMINO BÁSICO

La prueba del camino básico es una técnica de prueba de caja blanca que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba **se ejecuta por lo menos una vez cada sentencia** del programa.

Para la obtención de la **medida de la complejidad lógica** (o complejidad ciclomática) emplearemos una representación del flujo de control denominada **grafo de flujo o grafo del programa**.

NOTACIÓN DE GRAFO DE FLUJO

El grafo de flujo de las estructuras de control se representa de la siguiente forma:

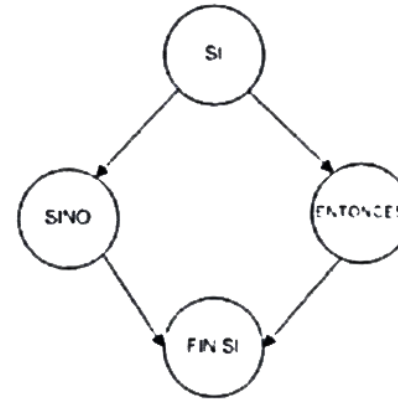
ESTRUCTURA	GRAFO DE FLUJO
SECUENCIAL Instrucción 1 Instrucción 2 Instrucción n	

NOTACIÓN DE GRAFO DE FLUJO

El grafo de flujo de las estructuras de control se representa de la siguiente forma:

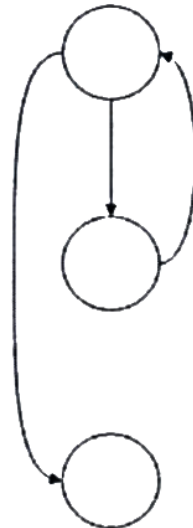
CONDICIONAL

Si <condición> **Entonces**
 <Instrucciones>
Si no
 <Instrucciones>
Fin si



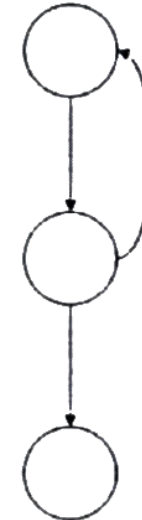
HACER MIENTRAS

Mientras <condición> **Hacer**
 <instrucciones>
Fin mientras



REPETIR HASTA

Repetir
 <instrucciones>
Hasta que <condición>



NOTACIÓN DE GRAFO DE FLUJO

El grafo de flujo de las estructuras de control se representa de la siguiente forma:

CONDICIONAL MÚLTIPLE

Según sea <variable> **Hacer**
 Caso opción 1:
 <Instrucciones>
 Caso opción 2:
 <Instrucciones>
 Caso opción 3:
 <Instrucciones>
 Otro caso:
 <Instrucciones>
Fin según

