Modelos de Calidad de Software

Tema Nº4:Introducción a Calidad de Software – Parte 4

Indicador de logro Nº4:Identifica la diferencia entre los niveles y tipos de prueba para aplicarlo según el contexto.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº4:**

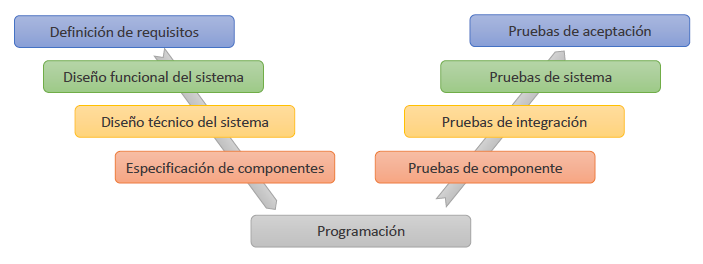
Introducción a Calidad de Software – Parte 3

**Subtema 4.1:**

Niveles de pruebas de software

**El Modelo “V” General**

El modelo “V” es el modelo de desarrollo de software más utilizado.

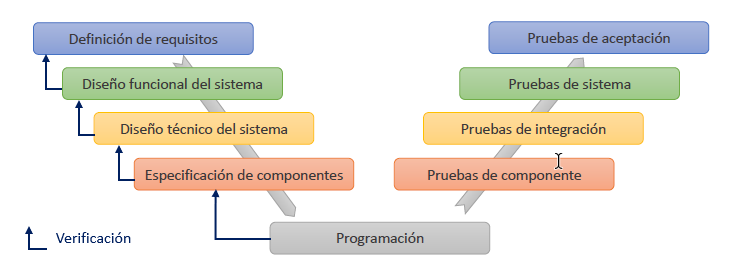


* Desarrollo y pruebas son dos ramas iguales. Cada nivel de desarrollo tiene su correspondiente nivel de pruebas.
* Las pruebas se diseñan en paralelo al desarrollo de software.
* Las actividades del proceso de pruebas tienen lugar a lo largo de todo el ciclo de vida de desarrollo de software.

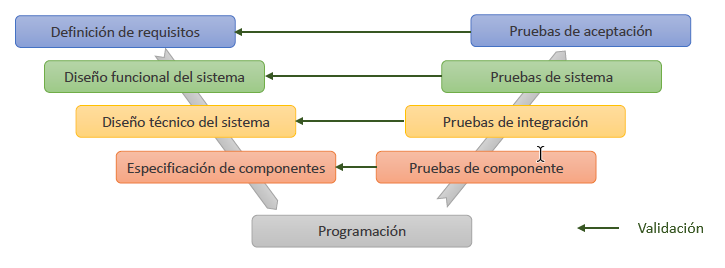
**Verificación vs Validación**

La **Verificación** es un ejercicio teórico diseñado para estar seguro que ningún requerimiento deja de cumplirse en el diseño, mientras que la **validación** es un ejercicio práctico que garantiza que el producto funcionará para cumplir con los requerimientos.

La **Verificación** responde a la pregunta **¿Se ha procedido correctamente en la construcción del sistema?**



La **Validación** responde a la pregunta **¿Hemos construido el sistema correcto?**



**Niveles de pruebas de software:**

1. **Pruebas de componente**

Las pruebas de componentes se ejecutan de forma independiente para comprobar que el resultado sea el requerido. Su objetivo es verificar las funcionalidades y/o usabilidades de los componentes, aunque no solo se limite a eso.

Sólo se prueban componentes individuales.

•Un componente puede estar constituido por un conjunto de unidades más pequeñas (por ejemplo: clase - métodos).

•Los objetos de prueba no siempre pueden ser probados en solitario de forma autónoma.

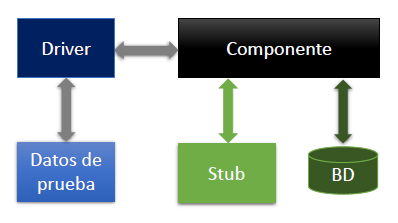
Cada componente es probado de forma independiente

•Descubriendo defectos internos

•Los efectos cruzados entre componentes quedan fuera del alcance de estas pruebas

Nota:

•En algunos casos incluso puede ser útil efectuar pruebas de estrés a nivel de componente.



La ejecución de pruebas de componente requiere frecuentemente de “drivers” y “stubs”.

•Un “driver” procesa la interfaz de un componente.

• Los “drivers” simulan datos de entrada, registran datos de salida y aportan una armadura de pruebas.

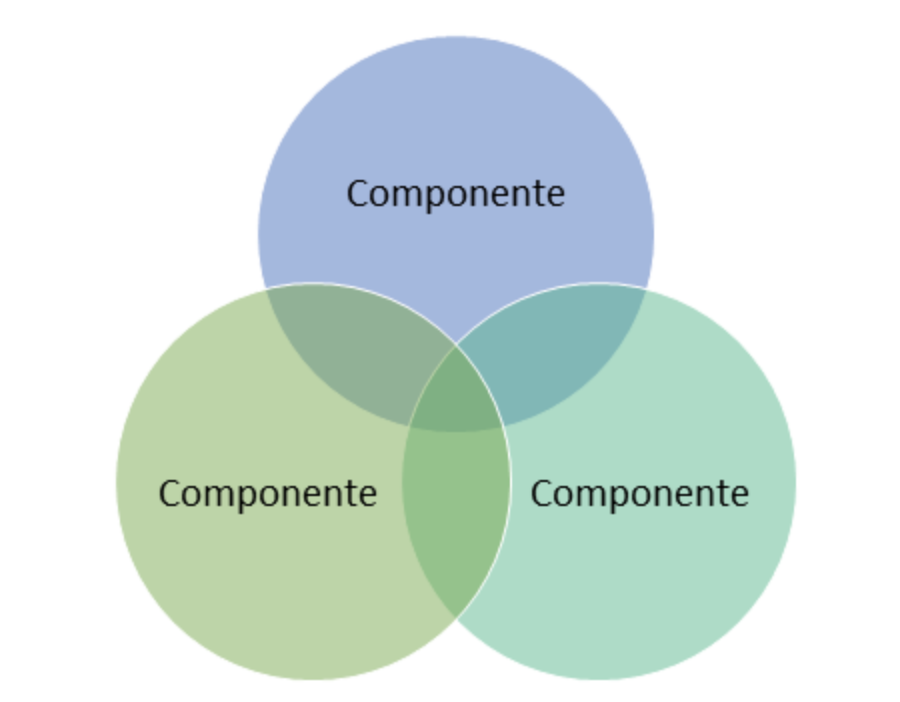
•Los “drivers” utilizan herramientas de programación.

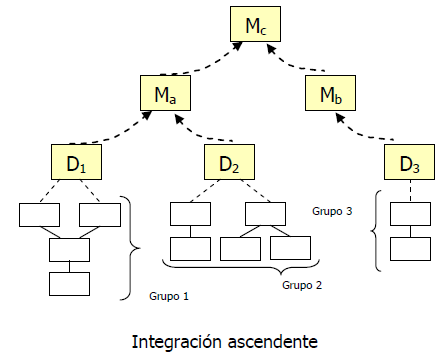
•Un “stub” reemplaza o simula un componente que aún no se encuentra disponible.

1. **Pruebas de integración**

Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias y lo que prueban es que todos los elementos unitarios que componen el software, funcionan juntos correctamente probándolos en grupo.

Este tipo de pruebas son ejecutas por el equipo de desarrollo y consisten en la comprobación de que elementos del software que interactúan entre sí, funcionan de manera correcta.





1. **Pruebas de sistema**

Las pruebas de sistema tienen como objetivo ejercitar profundamente el sistema comprobando la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica.

Este tipo de pruebas deben ser ejecutadas idealmente por un equipo de pruebas ajeno al equipo de desarrollo, una buena práctica en este punto corresponde a la tercerización de esta responsabilidad.

La obligación de este equipo, consiste en la ejecución de actividades de prueba en donde se debe verificar que la funcionalidad total de un sistema fue implementada de acuerdo a los documentos de especificación definidos en el proyecto.

Los casos de prueba a diseñar en este nivel de pruebas, deben cubrir los aspectos funcionales y no funcionales del sistema.

Para el diseño de los casos de prueba en este nivel, el equipo debe utilizar como bases de prueba entregables tales como: requerimientos iniciales, casos de uso, historias de usuario, diseños, manuales técnicos y de usuario final, etc.

Por último, es importante que los tipos de pruebas ejecutados en este nivel se desplieguen en un ambiente de pruebas / ambiente de pre-producción cuya infraestructura y arquitectura sea similar al ambiente de producción, evitando en todos los casos utilizar el ambiente real del cliente, debido principalmente, a que pueda ocasionar fallos en los servidores, lo que ocasionaría indisponibilidad en otros servicios alojados en este ambiente.

Las pruebas de sistema se refieren a:

Requisitos funcionales:

* Funcionalidad

Requisitos no funcionales:

* Fiabilidad
* Usabilidad
* Eficiencia
* Mantenibilidad
* Portabilidad

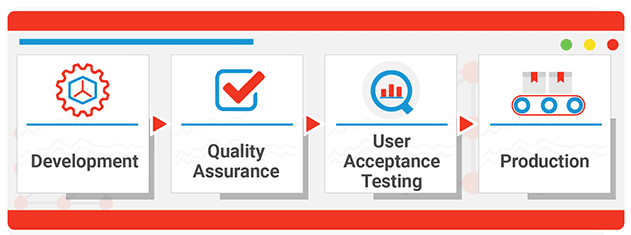


1. **Pruebas de aceptación**

La prueba de aceptación del usuario es la última acción de prueba antes de desplegar el software. El objetivo de la prueba de aceptación es comprobar si el software está preparado y lo pueden utilizar los usuarios para realizar las funciones y tareas para las que se diseñó.

¿El software satisface todos los requisitos contractuales?

•Con la aceptación formal se cumplen hitos legales: comienzo de fase de garantía, hitos de pago/abono, acuerdos de mantenimiento, etc.



**Subtema 4.2:**

Tipos de pruebas de software

Tipos de pruebas:

1.Pruebas funcionales. Objetivo: probar la función del objeto de prueba.

2.Pruebas no funcionales. Objetivo: probar las características del producto software.

3.Pruebas estructurales. Objetivo: probar la cobertura de la estructura del producto.

4.Pruebas de confirmación/regresión. Objetivo: probar el objeto de prueba después de cambios.

5.Pruebas de mantenimiento. Objetivo: probar después de aceptado el producto.

1. **Pruebas funcionales**

Objetivo: probar la función del objeto de prueba.

* La funcionalidad puede ser vinculada a los datos de entrada y salida de un objeto de prueba.
* Los métodos de Caja Negra (Black box) se utilizan en el diseño de casos de prueba relevantes.
* Las pruebas se desarrollan teniendo en cuenta los requisitos funcionales (establecidas en las especificaciones, conceptos, casos de estudio, reglas de negocio o documentos relacionados).

Ámbito de aplicación:

* Las pruebas funcionales se pueden llevar a cabo en todos los niveles de pruebas.

Ejecución:

* El objeto de prueba es ejecutado utilizando combinaciones de datos de prueba derivados/generados a partir de los casos de prueba.
* Los resultados de la ejecución de la prueba son comparados con los resultados esperados.

1. **Pruebas no funcionales**

Objetivo: probar las características del producto software.

* ¿De qué forma el software lleva a cabo la función?
* Las características de calidad no funcionales (fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad) a menudo son vagas, incompletas o inexistentes, dificultando las pruebas asociadas a las mismas.

Ámbito de aplicación:

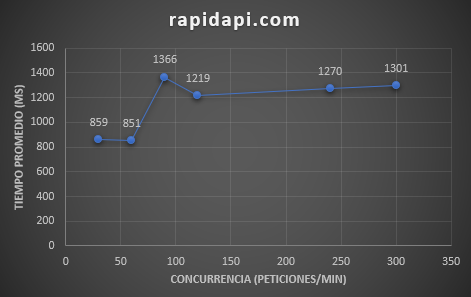
* Las pruebas no funcionales se pueden llevar a cabo en todos los niveles de pruebas.

Ejecución:

* La conformidad con los requisitos no funcionales se mide utilizando requisitos funcionales (seleccionados).

**Prueba de carga (Load test)**

* Sistema bajo carga (carga mínima, más usuarios/transacciones).



**Prueba de rendimiento (Performance test)**

* Rapidez con la cual un sistema ejecuta una determinada función.

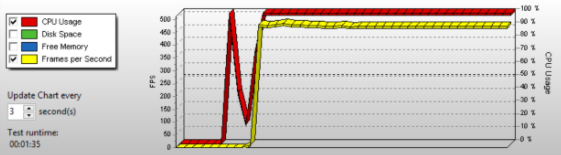


**Prueba de volumen (Volume test)**

* Procesamiento de grandes cantidades de datos/archivos.

**Prueba de estrés (Stress test)**

* Reacción a la sobrecarga/recuperación tras el retorno a una carga normal.



**Prueba de estabilidad (stability test)**

* Rendimiento en “modo de operación continua” (tiempo de operación sin reporte de incidencias).

**Prueba de robustez**

* Reacción a entradas erróneas o datos no especificados.
* Reacción a fallos de hardware/recuperación ante situaciones de desastre.

**Pruebas de seguridad para los datos del software (Test for data security)**

* Protección contra accesos no autorizados.
* Protección contra el robo y daño de datos.



**Pruebas de portabilidad/compatibilidad (Compatibility test)**

* Reacción a distintos entornos (HW, OS, etc.)

**Pruebas de usabilidad (test for usability)**

* Estructurado, comprensible, fácil de aprender para el usuario.

**Pruebas de mantenibilidad (Maintainability test)**

* Capacidad de ser analizado y modificado.

1. **Pruebas estructurales**

Objetivo: probar la cobertura de la estructura del producto:

* Análisis de la estructura de un objeto de prueba (enfoque: caja blanca).
* La finalidad de las pruebas es medir el grado en el cual la estructura del objeto de prueba ha sido cubierta por los casos de prueba.



GLOSARIO ISTQB

**Cobertura de pruebas**: Véase cobertura.

**Cobertura**: El grado, expresado como porcentaje, hasta el cuál un ítem de cobertura especificado ha sido ejercido por un juego de pruebas. Tenga en cuenta que este término no ha sido referenciado específicamente en esta sección, pero se lo incluye aquí, porque es un sinónimo de cobertura de pruebas.

Ámbito de aplicación:

* Las pruebas estructurales son posibles en todos los niveles de pruebas, se realizan de forma conjunta a las pruebas de componente y de integración mediante el uso de herramientas.
* El diseño de pruebas estructurales se finaliza tras haber sido diseñadas las pruebas funcionales, con el propósito de obtener un alto grado de cobertura.

Ejecución:

* Se probará la estructura interna de un objeto de prueba (por ejemplo, el control de flujo en el interior de un componente, el flujo a través de la estructura de un menú).
* Todos los elementos estructurales identificados deben estar cubiertos por casos de prueba.

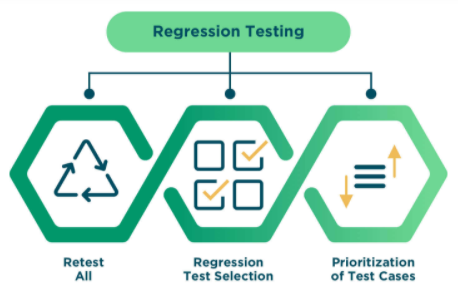
1. **Pruebas de confirmación/regresión**

Objetivo: probar el objeto de prueba después de cambios.

* Después de que un objeto de pruebas o el entorno del sistema ha sido objeto de modificación todos los resultados de pruebas anteriores resultan inválidos, por lo que las pruebas deben ser repetidas.
* Es necesario volver a probar el elemento modificado y las zonas adyacentes debido a efectos colaterales no deseados de la funcionalidad cambiada.
* El alcance de las pruebas de regresión depende del riesgo que la nueva funcionalidad implementada (extensión o corrección de errores) impone al sistema.

Ámbito de aplicación:

* Las pruebas de confirmación/regresión pueden ser realizadas en todos los niveles de pruebas.
* Las pruebas típicas tras una modificación son las siguientes:
  + Pruebas de confirmación (Retest). Pruebas para confirmar que el defecto ha sido corregido.
  + Pruebas de regresión. Pruebas para descubrir nuevos defectos introducidos en funcionalidades previamente sin fallos.



Ejecución:

* Básicamente la ejecución tiene lugar de la misma forma en la cual se han ejecutado las pruebas en iteraciones previas.
* En la mayoría de los casos, una prueba de regresión completa no es viable dados sus altos costos y duración.
* Criterios para la selección de casos de prueba de regresión:
  + Casos de prueba de prioridad alta (rutas o flujos funcionales críticos).
  + Probar solamente la funcionalidad estándar, saltarse casos y variaciones especiales.
  + Probar solamente la configuración utilizada con mayor frecuencia.
  + Probar solamente subsistemas/zonas seleccionadas del objeto de pruebas que tienen el riesgo de sufrir alteraciones.

**Si durante las fases tempranas del proyecto resulta evidente que ciertas pruebas son adecuadas para las pruebas de regresión, se deberá considerar la automatización de estas pruebas.**

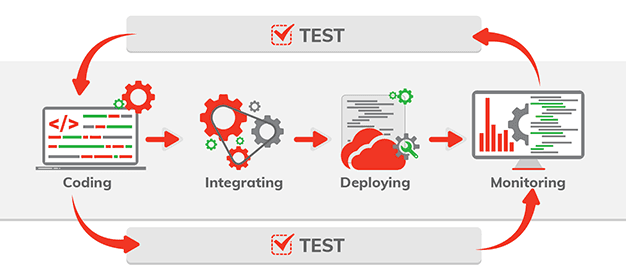
1. **Pruebas de mantenimiento**

El cliente ha probado el producto y es puesto en producción.

* El ciclo de desarrollo inicial, incluidas las pruebas asociadas, ha sido completado.

El mismo software se encuentra al comienzo del ciclo de vida.

* Será utilizado por muchos años, será ampliado.
* Es muy probable que aún contenga defectos, por lo tanto, será modificado y corregido.
* Necesitará adaptarse a nuevas condiciones y deberá integrarse a nuevos entornos.



Pruebas de regresión.

* Las pruebas de regresión son necesarias y realizadas una vez realizado algún cambio en el sistema.
* El alcance de las pruebas de regresión puede variar en función de las circunstancias (condiciones del entorno).

**Cualquier nueva versión del producto, cada nueva actualización y cada cambio del software requiere pruebas adicionales**

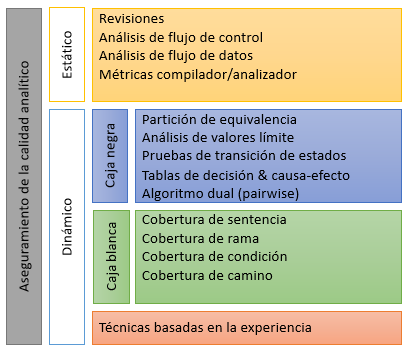
**Subtema 4.3:**

Técnica de Pruebas de Caja Negra y Caja Blanca

GLOSARIO ISTQB

**Pruebas de Caja Negra**: Pruebas ya sea funcionales o no funcionales, sin referencia a la estructura interna del componente o sistema.

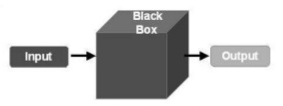
**Pruebas de Caja Blanca**: Pruebas basadas en la estructura interna del componente o sistema.



* Las pruebas dinámicas se dividen en 3 categorías.
* La agrupación se realiza en función del carácter básico del método utilizado para obtener los casos de prueba.
* Cada grupo tiene sus propios métodos para diseñar sus casos de prueba.

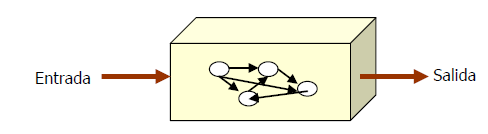
**Técnicas basadas en la especificación o de Caja Negra (Black Box)**

* El probador observa el objeto de prueba como una caja negra. La estructura interna del objeto de pruebas es irrelevante o desconocida.
* Los casos de prueba se obtienen a partir del análisis de la especificación (funcional y no funcional) de un componente o sistema.
* La funcionalidad es el foco de atención. La técnica de caja negra también se denomina prueba funcional o **prueba orientada a la especificación**.
* El objeto de prueba ha sido seleccionado de acuerdo con el modelo funcional del software.
* Se ejecuta el objeto de pruebas para hacer las evaluaciones que indican los casos de prueba.



**Técnicas basadas en la estructura o de Caja Blanca (White Box)**

* El probador conoce la estructura interna del programa (código fuente).
* Los casos de prueba son seleccionados en base a la estructura interna del programa.
* La estructura del programa es el foco de atención. La técnica de caja blanca también es conocida como prueba estructural o **pruebas basadas en el flujo de control**.
* El porcentaje de cobertura es medido y utilizado como fuente para la creación de casos de prueba adicionales.
* Se pueden utilizar herramientas para evaluar el código.



**Técnicas basadas en la experiencia**

* El conocimiento y la experiencia respecto de los objetos de prueba y su entorno son las fuentes para el diseño de casos de prueba.
* El conocimiento y la experiencia respecto de posibles puntos débiles, posibles errores y errores previos son utilizados para determinar y definir casos de prueba.

**Actividad:**

Revisa y analiza el enlace sugerido\* en la plataforma virtual luego realiza la actividad propuesta

1. ¿Cuál es la diferencia entre niveles y tipos de pruebas?
2. ¿Qué niveles de prueba efectúa el desarrollador?
3. ¿Qué niveles de prueba efectúa el analista de calidad?
4. Brindar ejemplos prácticos de Caja Negra y Caja blanca.

\*Cada docente deberá proporcionar el enlace que considere apropiado para la ejecución de la actividad.