Preparador Informática

www.preparadorinformatica.com

TEMA 30. INFORMÁTICA

PRUEBA Y DOCUMENTACIÓN DE PROGRAMAS. TÉCNICAS.

TEMA 32. SAI

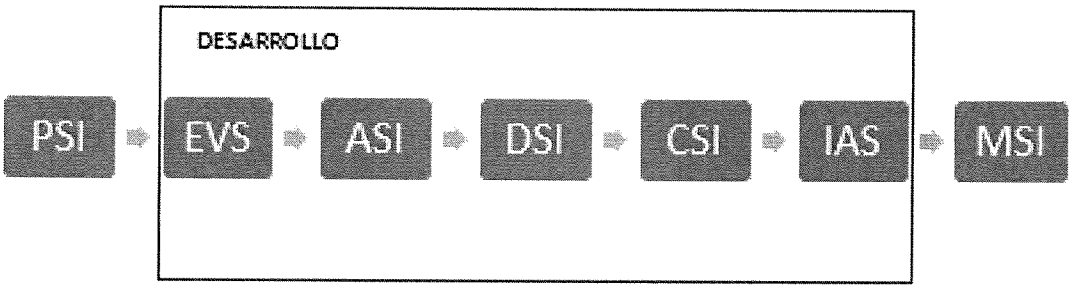
TÉCNICAS PARA LA VERIFICACIÓN, PRUEBA Y DOCUMENTACIÓN DE PROGRAMAS.

TEMA 30 INF: PRUEBA Y DOCUMENTACIÓN DE PROGRAMAS. TÉCNICAS.

TEMA 32 SAI: TÉCNICAS PARA LA VERIFICACIÓN, PRUEBA Y DOCUMENTACIÓN DE PROGRAMAS.

1. INTRODUCCIÓN

Independientemente del modelo de ciclo de vida elegido para el desarrollo de software, siempre hay una serie de etapas o fases que se deben seguir para construir software fiable y de calidad. Tomando como referencia la metodología Métrica versión 3 (desarrollada por la Administración Pública) estas fases son:



- PSI: Planificación del Sistema de Información
- EVS: Estudio de Viabilidad del Sistema
- ASI: Análisis del Sistema de Información
- DSI: Diseño del Sistema de Información
- CSI: Construcción del Sistema de Información
- IAS: Implantación y Aceptación del Sistema
- MSI: Mantenimiento del Sistema de Información

Centrándonos en las tareas de prueba, estas se concentran en las fases de CSI e IAS, mientras que la documentación es una tarea transversal a todas las fases.

En los últimos años las pruebas están cobrado un especial protagonismo, prueba de ello es el TDD (Test-Driven Development) o desarrollo guiado por pruebas de software, una técnica que está empezando a ser bastante utilizada y que se basa en desarrollar primero las pruebas, después escribir el código fuente que pase la prueba satisfactoriamente y, por último, refactorizar el código escrito.

El presente tema está dedicado a estudiar la importancia de las tareas de prueba y documentación de programas, detallando los tipos y técnicas de pruebas, fases del proceso de pruebas, así como la documentación asociada al proceso de desarrollo de software.

2. VERIFICACIÓN Y VALIDACIÓN DE PROGRAMAS

La verificación y validación es el nombre que se da a los procesos de comprobación y análisis que aseguran que el software que se desarrolla está acorde a su especificación y a las necesidades de los clientes.

- **Verificación**: es el conjunto de actividades encaminadas a asegurar que el software se ajusta a las especificaciones indicadas. El término verificación responde a la pregunta: *"¿Estamos construyendo el producto correctamente?".*

- **Validación**: es el conjunto de actividades encaminadas a asegurar que el software se ajusta a los requisitos del cliente. El término validación responde a la pregunta: *"¿Estamos construyendo el producto correcto?".*

3. PRUEBA DE PROGRAMAS

Cuando se habla de pruebas del software o de programas es común utilizar los términos "prueba" y "caso de prueba". Su definición es la siguiente:

- **Prueba**: es el proceso de ejecución de un programa con el intento deliberado de encontrar errores.

- **Caso de prueba**: conjunto de entradas, condiciones de ejecución y resultados esperados diseñados para un objetivo particular de condición de prueba.

3.1. PRINCIPIOS BÁSICOS DE LAS PRUEBAS

Según el **ISTQB** (*International Software Testing Qualifications Board*) los principios básicos que guían las pruebas del software son:

- **Principio 1**: Las pruebas demuestran la presencia de errores, pero no su ausencia.

- **Principio 2**: Las pruebas completas no existen.

- **Principio 3**: Las pruebas se iniciarán lo antes posible en el ciclo de vida del software.

- **Principio 4**: La mayor parte de los errores se suelen concentrar en un número reducido de módulos.

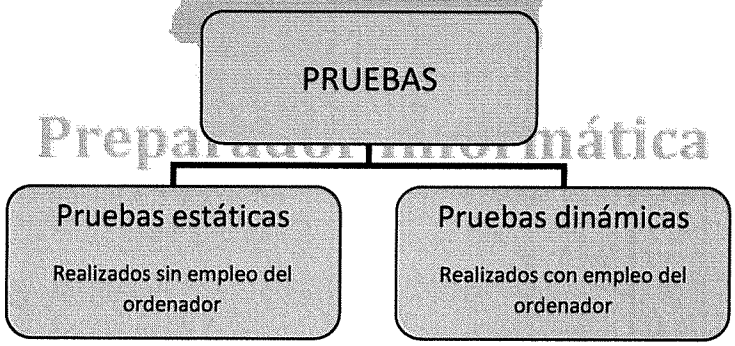
- **Principio 5**: Si las pruebas se repiten una y otra vez, con el tiempo el mismo conjunto de casos de prueba ya no encontrará nuevos errores. Los casos de prueba deben ser examinados y revisados periódicamente.

- **Principio 6**: Las pruebas deben adaptarse a las necesidades específicas del contexto.

- **Principio 7**: Un software puede haber pasado todas las fases de pruebas, pero esto no indica que no pueda tener errores que aún no se han logrado identificar.

3.2. TIPOS DE PRUEBAS

Podemos distinguir entre pruebas estáticas y pruebas dinámicas:



- **Pruebas estáticas**: son aquellas que analizan el objeto sin necesidad de ejecutarlo. Genéricamente se conocen por el nombre de revisiones y el objeto probado es la documentación asociada a las distintas fases del desarrollo.

- **Pruebas dinámicas**: son aquellas que requieren la ejecución del objeto que se está probando. Es decir, son pruebas en las que para su realización se requiere el empleo del ordenador.

3.3. FASES DEL PROCESO DE PRUEBA

A continuación, refiriéndonos a las pruebas dinámicas, se presentan las fases de que consta el proceso de pruebas.

La mayoría de autores de Ingeniería del Software consideran que las fases de que consta todo proceso de pruebas son, de manera secuencial, las siguientes:

- Pruebas unitarias
- Pruebas de integración
- Pruebas del sistema
- Pruebas de aceptación
- Pruebas de regresión

3.3.1. PRUEBAS UNITARIAS

Las pruebas unitarias o pruebas de unidad constituyen la prueba inicial de un sistema y las demás deben apoyarse sobre ellas.

Tienen por objetivo verificar la funcionalidad y estructura de cada componente (módulo) individualmente, una vez que ha sido codificado. Es decir, pretenden comprobar que el módulo, entendido este como una unidad funcional de un programa completamente independiente, está correctamente codificado.

3.3.2. PRUEBAS DE INTEGRACIÓN

La siguiente fase en el proceso de pruebas son las pruebas de integración. Estas pruebas son necesarias ya que las pruebas unitarias no aseguran que cuando se integren todos los módulos como un conjunto, el sistema completo funcione.

La forma de ensamblar los módulos introduce una complejidad adicional a estas pruebas, por lo que, para paliar en la medida de lo posible, los problemas de integración se han definido las siguientes estrategias:

a) **Integración incremental**: los módulos se integran poco a poco y a medida que se integran se van probando. Caben dos posibilidades:

- *Integración incremental descendente (top-down)*
- *Integración incremental ascendente (bottom-up)*

b) **Integración no incremental**: los módulos se integran de una vez y se prueban todos al mismo tiempo.

3.3.3. PRUEBAS DEL SISTEMA

Una vez validado el software de forma aislada mediante las pruebas unitarias y pruebas de integración, hay que combinar el software con el resto de componentes del sistema (hardware, base de datos, etc.)

Las pruebas del sistema verifican que cada elemento encaja de forma adecuada y que se alcanza la funcionalidad y el rendimiento del sistema completo. De esta forma se obtiene una visión global del sistema similar a la que se mostraría en un entorno real de producción.

3.3.4. PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación son las que realiza el cliente para comprobar si el sistema cumple los requisitos expresados por él, de modo que el sistema esté listo para el uso operativo en el entorno en el que se va a explotar el sistema. Podemos distinguir:

- **Pruebas alfa**: los clientes prueban el programa en el lugar de desarrollo con la presencia del equipo desarrollador, mientras éste registra errores. Es decir, las pruebas alfa se llevan a cabo en un entorno controlado.
- **Pruebas beta**: los clientes prueban el programa en su lugar de trabajo y sin la presencia del equipo de desarrollo. El cliente registra errores durante la prueba y va informando al equipo de desarrollo.

3.3.5. PRUEBAS DE REGRESIÓN

Las pruebas de regresión están asociadas a la fase de mantenimiento del software y su objetivo es comprobar que los cambios efectuados no producen un comportamiento no deseado en el programa.

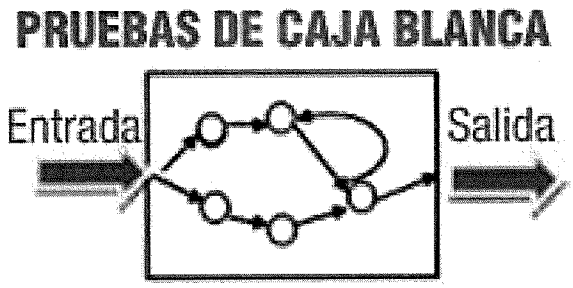
Las pruebas de regresión se deben realizar cada vez que se efectúa un cambio en el sistema, ya sea para corregir un error o para introducir una mejora, a fin de controlar que las modificaciones no producen efectos negativos sobre el mismo u otros componentes. Normalmente implican la repetición de las pruebas que ya se han realizado previamente (unitarias, de integración, de aceptación, etc.).

3.4. TÉCNICAS DE PRUEBA DEL SOFTWARE

En la ingeniería del software, nos encontramos con dos enfoques fundamentales:

- **Pruebas de Caja Negra**: cuando una aplicación es probada usando su interfaz externa, sin preocuparnos de la implementación de la misma.

- **Pruebas de Caja Blanca**: en este caso, se prueba la aplicación desde dentro, usando su lógica de aplicación.



3.4.1. PRUEBAS DE CAJA NEGRA

Las pruebas de caja negra se centran en probar el programa usando su interfaz externa, sin preocuparnos de la implementación del mismo. Lo fundamental es comprobar que los resultados de la ejecución del programa son los esperados, en función de las entradas que recibe. Es decir, solo interesa si realiza las funciones que se esperan de él.

Las técnicas más usuales que siguen el método de caja negra son:

- **Particiones de equivalencia**: Consiste en dividir el dominio de entrada de un programa en clases de equivalencia de los que se pueden derivar casos de prueba, donde la prueba de un valor representativo de la misma sea extrapolable al que se conseguiría probando cualquier valor de la clase.

- **Análisis de valores límite**: En este caso, a la hora de implementar un caso de prueba, se van a elegir como valores de entrada, aquellos que se encuentra en el límite de las clases de equivalencia. Se justifica en la constatación de que para una condición de entrada que admite un rango de valores es más fácil que existan errores en los límites que en el centro. Ejemplo: un campo numérico que requiera una cifra de 3 dígitos, tendrá como valores límites inferiores 99 y 100, y como valores límites superiores 999 y 1000.

- **Valores típicos de error**: desarrolla casos de prueba con ciertos valores susceptibles de causar problemas, esto es, valores típicos de error, y valores especificados como no posibles. La determinación de los valores típicos de error se realiza en función de la naturaleza y funcionalidad del programa a probar, por lo que depende en buena medida de la experiencia del diseñador de la prueba.

3.4.2. PRUEBAS DE CAJA BLANCA

Las pruebas de caja blanca se centran en probar el comportamiento interno y la estructura del programa, examinando la lógica interna del mismo.

Este tipo de pruebas, se basan en unos criterios de cobertura lógica, cuyo cumplimiento determina la mayor o menor seguridad en la detección de errores.

Las principales técnicas de caja blanca son:

- **Prueba de interfaz:** este tipo de prueba debe ser la primera en realizar. Se basa en analizar el flujo de datos que pasa a través de la interfaz del módulo, tanto externa como interna, para asegurar que la información fluye de forma adecuada tanto hacia el interior como hacia el exterior del módulo que se está probando.
- **Pruebas de estructuras de los datos locales**: estas pruebas tienen como objetivo asegurar la integridad de los datos durante todos los pasos de la ejecución del módulo.
- **Prueba del camino básico**: permite obtener una medida de la complejidad lógica de un programa (complejidad ciclomática) y utilizar esa medida como guía para definir un conjunto básico de caminos de ejecución. Es decir, la prueba del camino básico se orienta a cubrir la ejecución de cada una de las sentencias, cada una de las decisiones y cada una de las condiciones en las decisiones, tanto en su vertiente verdadero como falsa.
- **Prueba de bucles**: comprueban la validez de las construcciones de los bucles. Determinar la validez de las construcciones de los bucles es fundamental para garantizar que es correcto el módulo que se está probando.

4. DOCUMENTACIÓN DE PROGRAMAS

Todo el proceso de desarrollo de software debe quedar perfectamente documentado, para pasar así de una fase a otra de forma clara y definida. Además, es importante también para poder acometer futuras revisiones del proyecto.

Se distinguen tres grandes documentos en el desarrollo de software:

A) Guía técnica

a. Información que contiene:
   - El diseño de la aplicación.
   - La codificación de los programas.
   - Las pruebas realizadas.

b. Dirigido a:
   - Personal técnico en informática (analistas y programadores)

c. Objetivo:
   - Facilitar un correcto desarrollo, realizar correcciones en los programas y permitir un mantenimiento futuro.

B) Guía de uso

a. Información que contiene:
   - Descripción de la funcionalidad de la aplicación.
   - Forma de comenzar a ejecutar la aplicación.
   - Ejemplos de uso del programa.
   - Requerimientos software de la aplicación.
   - Soluciones de los posibles problemas que se pueden presentar.

b. Dirigido a:
   - Usuarios que van a utilizar la aplicación (clientes).

c. Objetivo:
   - Dar a los usuarios finales toda la información necesaria para utilizar la aplicación.

C) Guía de instalación

a. Información que contiene:

- Toda la información necesaria sobre puesta en marcha, explotación y seguridad del sistema.

b. Dirigido a:

- Personal informático responsable de la instalación, en colaboración con los usuarios que van a usar la aplicación (clientes).

c. Objetivo:

- Dar toda la información necesaria para garantizar que la implantación de la aplicación se realice de forma segura, confiable y precisa.

5. HERRAMIENTAS SOFTWARE PARA LA REALIZACIÓN DE PRUEBAS

A continuación, se presenta una tabla con varias herramientas que permiten la automatización de ciertos tipos de pruebas de software.

Tipo de prueba	Herramienta	Lenguaje
UNITARIA	JUnit	Java
	Simple Test	PHP
	PHP Unit	PHP
	Jtiger	Java
FUNCIONAL Y DE REGRESIÓN	Selenium	Java, PHP, Python, Ruby, etc.
	Watir	Ruby
	Watij	Java
	JWebUnit	Java
	NUnit	C#, J#, VB y C++
	Http Unit	Java

Tipo de prueba	Herramienta	Lenguaje
INTEGRACIÓN	TestNG	Java
	Hudson	Java
	Continuum	Java
CARGA Y RENDIMIENTO	JMeter	Java
	Jcrawler	Java
ACEPTACIÓN	FitNesse	Java, PHP, Ruby, .NET
	Concordion	Java, Python, Ruby, .NET
	Nessus	NASL,XML,HTML

6. CONCLUSIÓN

En este tema se ha presentado una visión global de las tareas de prueba y documentación, ya que son de vital importancia durante el desarrollo de software. Las pruebas permiten verificar que el software que se está creando es correcto y cumple con las especificaciones impuestas por el usuario, garantizando así la calidad del producto entregado. Y por otra parte, una correcta documentación durante el desarrollo de software influirá también positivamente en la calidad del software.

7. BIBLIOGRAFÍA

- Pressman, R. S. **Ingeniería del Software: Un enfoque práctico**. Editorial McGraw-Hill
- Sommerville, I. **Ingeniería del software**. Editorial Addison Wesley
- **Norma ISO/IEC/IEEE 29119** (Estándar para pruebas del software).
- **Métrica versión 3**. Metodología de planificación, desarrollo y mantenimiento de sistemas de información. Ministerio para las Administraciones Públicas.
- www.istqb.org (International Software Testing Qualifications Board)