

República Bolivariana de Venezuela

Ministerio del Poder Popular para la Educación Superior

Universidad Nacional Experimental de la Gran Caracas "UNEXCA"

Trayecto II, semestre II

Unidad Curricular: Ing. Del Software I

Metodologías de Prueba

Docente: Yoly Arrechedera Estudiantes: Oriana Marín V30.709.244

Brando Martínez V27.295.116

Bruno Palacios V30.150.650

Ronaldo Rivero V31.169.960

Caracas 10 de noviembre del 2025

Introducción

En la Ingeniería del Software moderna, la calidad no es un atributo opcional, sino un requisito indispensable para la supervivencia de cualquier producto. La metodología de pruebas constituye el conjunto de estrategias, procesos y herramientas empleados sistemáticamente para asegurar que el software desarrollado sea robusto, confiable y satisfaga plenamente las expectativas. Esta unidad profundiza en la disciplina del testing, que va mucho más allá de la simple búsqueda de errores, enfocándose en la prevención de defectos desde las fases tempranas del ciclo de vida del desarrollo de software (SDLC).

El rol central de las pruebas es proporcionar información objetiva y confiable sobre la calidad del producto a los stakeholders. Un enfoque estructurado permite mitigar los riesgos operativos y financieros asociados a fallos en producción, garantizando la conformidad con los requisitos funcionales y no funcionales establecidos. A través de la aplicación metódica de los distintos niveles de prueba, se busca la cobertura máxima y la detección temprana de las desviaciones respecto al comportamiento esperado.

Verificación y Validación del Software

La Verificación y la Validación (V&V;) representan el marco conceptual que sustenta la garantía de calidad en el software, a menudo esquematizado en el Modelo V, donde las actividades de prueba se alinean directamente con las actividades de desarrollo correspondencia.

**Verificación ("Are we building the product right?")**

La verificación se enfoca en el proceso interno del desarrollo. Su propósito es asegurar que cada artefacto de trabajo (documentos de requisitos, diseños, código fuente, etc.) cumpla con las condiciones impuestas en la etapa anterior. Es una actividad estática que no requiere la ejecución del código.

Alcance: Se aplica desde las especificaciones de requisitos hasta el código individual.

Técnicas: Incluye la revisión formal (inspecciones, walkthroughs), auditorías de código, análisis estático (herramientas que examinan el código sin ejecutarlo) y revisión de la documentación para asegurar la consistencia y el cumplimiento de estándares.

Resultado Clave: Confirmación de que el producto se está construyendo siguiendo el plan y el diseño establecidos.

## **Validación ("Are we building the right product?")**

La validación se centra en el producto final y su adecuación al propósito del negocio. Su objetivo es confirmar, mediante la ejecución del software, que el sistema cumple con las necesidades y expectativas reales del usuario en un entorno operativo.

Alcance: Se aplica al código ejecutable, desde módulos integrados hasta el sistema completo.

Técnicas: Consiste en la ejecución dinámica del software a través de casos de prueba predefinidos, simulando el comportamiento del usuario y la interacción con datos reales.

Resultado Clave: Aceptación del cliente y confirmación de que el software resuelve el problema para el cual fue diseñado.

### **El Proceso de Prueba**

El proceso de prueba es cíclico y estructurado, buscando integrar las actividades de aseguramiento de la calidad de manera temprana y continua en el SDLC (el concepto de Shift Left). Un ciclo de vida de prueba bien definido consta de las siguientes fases principales:

#### **Planificación y Control de Pruebas:**

Se define la estrategia general, los objetivos de las pruebas y los criterios de entrada y salida.

Se identifican los riesgos del producto y del proyecto que las pruebas deben mitigar.

El entregable principal es el Plan de Pruebas, que detalla el alcance, el enfoque, los recursos necesarios y el cronograma.

#### **Análisis y Diseño de Pruebas:**

Se analizan los requisitos de negocio y funcionales para determinar las condiciones de prueba.

Se diseñan los casos de prueba, identificando los datos de entrada, los resultados esperados y los procedimientos de ejecución. Se aplican técnicas de diseño como la partición de equivalencia y el análisis de valor límite.

#### Implementación y Preparación del Entorno:

Se desarrollan y priorizan los procedimientos de prueba, combinando los casos de prueba en secuencias ejecutables (scripts de prueba).

Se configura y verifica el ambiente de prueba (hardware, software, red y datos de prueba) para que sea lo más parecido posible al entorno de producción.

#### Ejecución de Pruebas:

Los scripts de prueba se ejecutan secuencialmente.

Los resultados reales se comparan con los resultados esperados.

Los incidentes (defectos) se documentan, se rastrean y se asignan a los desarrolladores para su corrección.

#### Evaluación de Criterios de Salida y Reporte:

Se verifica si se han cumplido los criterios de salida del ciclo de prueba (ej. porcentaje de casos de prueba aprobados, cobertura de requisitos alcanzada).

Se elaboran métricas de calidad y el Informe Final de Pruebas, que documenta la calidad del sistema, los riesgos residuales y la recomendación de lanzamiento.

#### Prueba de Unidad (Unit Testing)

La Prueba de Unidad es el nivel más granular y se centra en el código fuente de los componentes individuales.

Objetivo Principal: Aislamiento y validación de la funcionalidad de los módulos de software más pequeños, asegurando que cada bloque de código se comporte según lo especificado.

Alcance: Métodos, funciones, clases o procedimientos. Cada unidad se prueba por separado, aislada de las dependencias externas.

#### Técnicas y Herramientas:

Se utiliza el Testing de Caja Blanca, donde el conocimiento de la estructura interna del código guía el diseño de las pruebas.

Se emplean drivers (código que llama a la unidad bajo prueba) y stubs o mocks (objetos simulados) para reemplazar los módulos dependientes que aún no existen o que no se quieren involucrar en la prueba.

Las métricas de cobertura de código (cobertura de sentencias, de decisiones o de caminos) son esenciales para determinar la exhaustividad de las pruebas unitarias.

Responsabilidad: Es primordialmente una tarea del desarrollador y se realiza de forma continua, a menudo integrada en el proceso de integración continua (CI/CD).

### Prueba de Integración (Integration Testing)

La Prueba de Integración tiene como objetivo verificar la comunicación, el flujo de datos y las interfaces entre los diferentes módulos que componen el sistema. Un fallo a este nivel a menudo se debe a una mala interpretación de las interfaces o a errores en el manejo de datos compartidos.

Objetivo Principal: Exponer defectos en las interfaces y la interacción entre las unidades ya probadas.

Enfoque: Se realiza un Testing de Caja Gris, donde se utiliza un conocimiento parcial de la estructura interna (las interfaces) pero se diseña la prueba desde una perspectiva funcional (cómo se comunican).

Estrategias de Integración:

Integración Incremental:

Top-Down (Descendente): Los módulos de control superior se prueban primero; los módulos inferiores se reemplazan por stubs.

Bottom-Up (Ascendente): Los módulos de nivel más bajo (utilidades) se prueban primero y se combinan. Se utilizan drivers para simular los módulos superiores.

Integración Híbrida/Sándwich: Combina enfoques, probando simultáneamente la capa superior e inferior y dejando la capa intermedia para el final.

Integración Big Bang: Todos los módulos se integran a la vez. Es menos eficaz para la localización de fallos, pero se usa en sistemas pequeños o cuando el tiempo es crítico.

Responsabilidad: Generalmente es compartida entre desarrolladores y el equipo de QA.

### Prueba del Sistema (System Testing)

La Prueba del Sistema trata el software como una caja negra y evalúa el comportamiento del sistema completo e integrado, en un entorno que simula la producción.

Objetivo Principal: Verificar que el sistema en su totalidad cumple con todos los requisitos funcionales y no funcionales del documento de especificaciones.

Enfoque: Es un Testing de Caja Negra puro, ya que no se tiene en cuenta la estructura interna del código; solo se valida el comportamiento externo.

Tipos de Pruebas Clave:

Pruebas Funcionales: Verificación de todas las características y flujos de trabajo de negocio.

Pruebas No Funcionales:

Rendimiento: Velocidad, tiempo de respuesta y tasas de transferencia bajo carga específica.

Estrés: Comportamiento del sistema bajo condiciones extremas (ej. alto volumen de usuarios o datos).

Seguridad: Autenticación, autorización, integridad de los datos y resistencia a ataques.

Usabilidad: Facilidad de uso y la experiencia del usuario (UX).

Recuperación: Capacidad del sistema para recuperarse de fallos de hardware o software.

Responsabilidad: Exclusiva del equipo de QA. Es el primer punto donde el sistema se prueba de forma exhaustiva con todas sus dependencias externas (bases de datos, otros sistemas, redes).

Prueba de Aceptación (Acceptance Testing - AT)

La Prueba de Aceptación es el nivel final de prueba, actuando como un puente entre el desarrollo y el despliegue. Su propósito no es encontrar fallos de código, sino confirmar la conformidad del producto con los requisitos de negocio y la preparación para el lanzamiento.

Objetivo Principal: Obtener la aprobación formal del cliente o patrocinador del proyecto de que el sistema está listo para ser liberado.

Enfoque: Se basa en los escenarios de negocio críticos. Los casos de prueba suelen ser flujos de usuario de alto nivel.

Tipos Fundamentales:

Prueba de Aceptación del Usuario (UAT): Realizada por usuarios finales o expertos en el dominio del negocio. Se enfoca en la utilidad y operatividad del software en el contexto del día a día del usuario.

Prueba de Aceptación Operacional (OAT): Se centra en los aspectos operativos, como la preparación para el lanzamiento, la configuración del entorno, la formación del personal de soporte y los procedimientos de respaldo y restauración.

Prueba Alpha y Beta: La prueba Alpha se realiza internamente por usuarios simulados, mientras que la prueba Beta se lleva a cabo con un grupo selecto de usuarios reales en un entorno real antes del lanzamiento general, a menudo denominado "lanzamiento controlado" o pilot.

Criterio de Éxito: La firma del cliente o la satisfacción de los criterios de salida definidos en el contrato o plan de negocio.

## Conclusión

Esta Unidad pone de manifiesto que la calidad del software es el resultado directo de una metodología de pruebas bien definida y aplicada. La distinción entre Verificación (construir bien el producto) y Validación (construir el producto correcto) establece el marco conceptual necesario para abordar la calidad desde una perspectiva tanto interna como externa.

La progresión jerárquica de los niveles de prueba es vital: se comienza en el nivel atómico con la Prueba de Unidad, garantizando que los bloques de construcción funcionen correctamente. Luego, la Prueba de Integración asegura la comunicación entre estos bloques. La Prueba del Sistema valida el funcionamiento completo del software en un entorno real, incluyendo aspectos no funcionales críticos. Finalmente, la Prueba de Aceptación sirve como el umbral de lanzamiento, confirmando que el producto satisface las necesidades de negocio del cliente. En conjunto, estos niveles y el proceso estructurado de prueba no solo detectan defectos, sino que, de manera proactiva, aseguran la confianza y la sostenibilidad del producto de software a largo plazo.