

TECNICATURA SUPERIOR EN

Ciencia de Datos e
Inteligencia Artificial

TESTEO DE SOFTWARE

Módulo Testing

Tema: Introducción al Testing

Docente: Carolina Ahumada

Hola a todos! ¿Cómo están? En esta oportunidad estaremos adentrándonos algo más sobre la SQA, analizando un conjunto específicos de procesos que son llamado V&V, el mdelo V y los diferentes TEST.

Estos procesos tiene por finalidad asegurar que el software, que se encuentra en desarrollo, sea acorde a sus especificaciones, y que cumpla con las necesidades de los usuarios ¿les parece si comenzamos?.

¿QUÉ ES VERIFICACIÓN Y VALIDACIÓN (V&V)

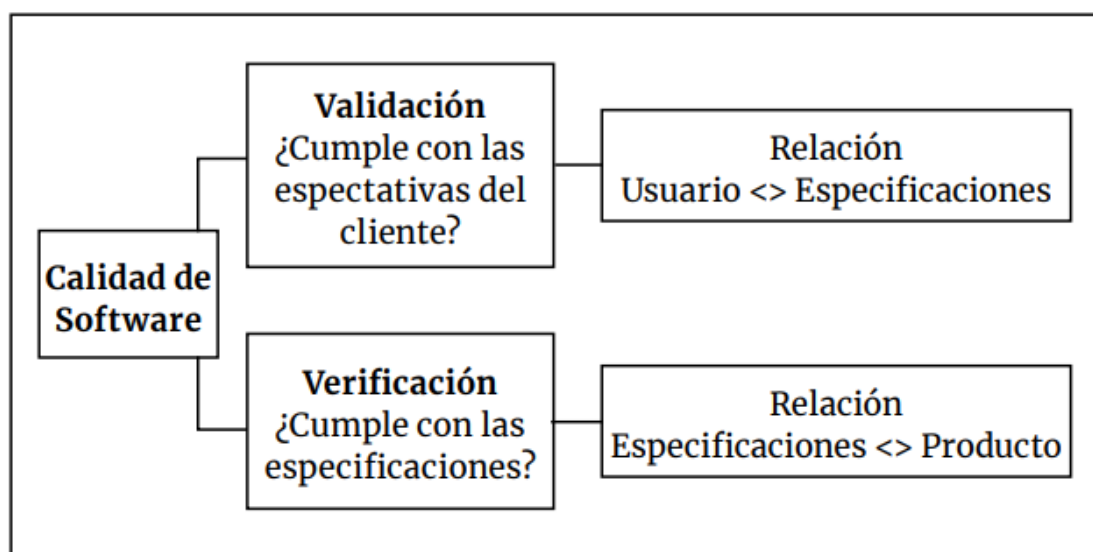
Es un conjunto de procesos de comprobación y análisis que aseguran que el software que se desarrolla está acorde a su especificación y cumple las necesidades de los clientes. Si bien los conceptos de Verificación y Validación suelen confundirse, es importante entender que no son lo mismo, ya que persiguen objetivos diferentes.

· **VERIFICACIÓN:** El papel de la verificación comprende comprobar que el software está de acuerdo con su especificación. Se comprueba que el sistema cumple los requerimientos funcionales y no funcionales que se le han especificado.

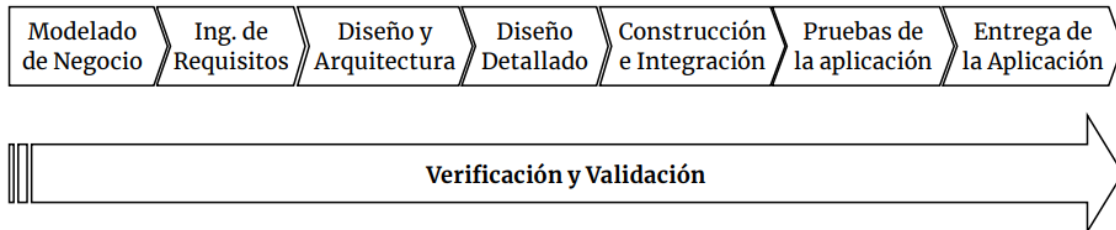
Busca responder la pregunta **¿Estamos construyendo el producto correctamente?**

· **VALIDACIÓN:** La validación es un proceso más general. Se debe asegurar que el software cumple las expectativas del cliente. Va más allá de comprobar si el sistema está acorde con su especificación, para probar que el software hace lo que el usuario espera a diferencia de lo que se ha especificado.

Busca responder la pregunta **¿Estamos construyendo el producto concreto?**



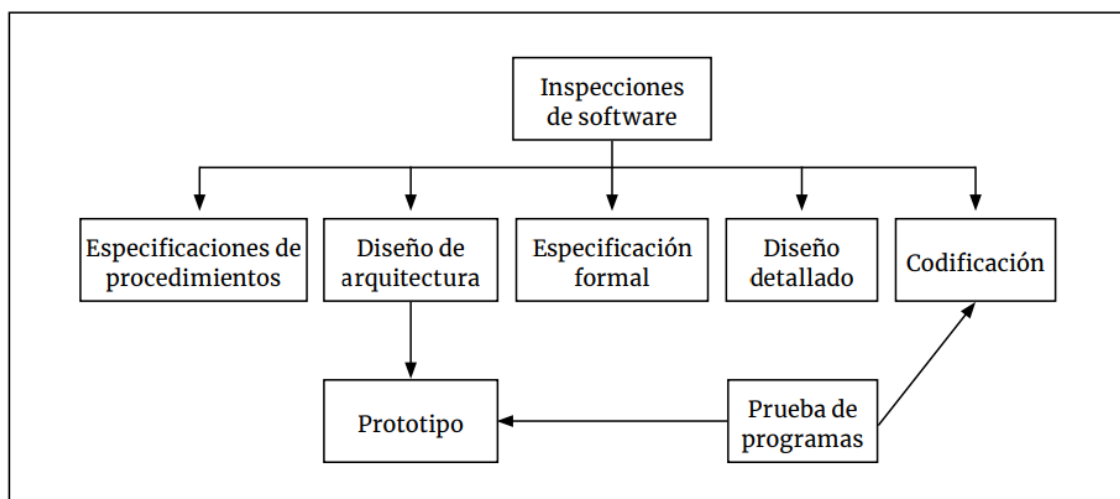
La V&V es un proceso de ciclo de vida completo, ¿A qué nos referimos con esto? El proceso Inicia con las revisiones de los requerimientos y continúa con las revisiones del diseño y las inspecciones del código hasta la prueba del producto, es decir, existen actividades de V&V en cada etapa del proceso de desarrollo del software.



Las tareas de V&V no solo se aplican a productos software terminados, sino que también a los artefactos resultantes del proceso de desarrollo, como al análisis y a la especificación de requisitos. De esta forma, se comprueba que el proyecto es viable y que las especificaciones documentadas son completas, correctas, precisas, legibles, evaluables, y que, en general, responden a las expectativas del cliente. La V&V del diseño debe garantizar que los requisitos no se encuentran incompletos o incorrectamente diseñados. En el caso de la implementación y la codificación, la V&V de software es comúnmente conocida como las pruebas de software.

¿QUE SON LAS PRUEBAS DEL SOFTWARE?

Las pruebas del software consisten en contrastar las respuestas de una implementación del software a series de datos de prueba y examinar las respuestas del software y su comportamiento operacional, para comprobar que se desempeñe conforme a lo requerido.



Cómo se ve en diagrama superior, las pruebas de software son aplicables en la etapa de diseño de arquitectura, a los fines de probar los prototipos; así como también en la etapa de codificación para probar el código ejecutable del sistema. Por otro lado la Inspección es aplicable a todas las etapas del ciclo de vida.

HABLEMOS DE TESTING

¡Llegamos al punto de hablar del famoso Testing!... y para eso vamos a comenzar por ver algunas definiciones: “Es el proceso de ejecutar software para verificar que cumple con los requisitos establecidos, y para detectar errores” ...BS7925-1

“Testing es el proceso de ejecutar un programa con la intención de encontrar errores” ...Glen Myers “Es el proceso de ejecutar un producto para: verificar que satisface los requerimientos e identificar diferencias entre el comportamiento real y el comportamiento esperado.” (IEEE Standard for Software Test Documentation, 1983)

El proceso que consiste en todas las actividades del ciclo de vida, tanto estáticas como dinámicas relacionadas con la planificación, preparación y evaluación de productos de software y productos relacionados con el trabajo para determinar que cumplen los requisitos especificados, para demostrar que son aptos para el propósito y para detectar defectos. (ISTQB) En primer lugar, echemos un vistazo a las pruebas como un proceso:

- Proceso: La prueba es un proceso más que una sola actividad. Hay una serie de actividades involucradas
- Todas las actividades del ciclo de vida: se lleva a cabo durante todo el ciclo de desarrollo de software tratando de encontrar y corregir defectos desde el inicio del ciclo de vida a fin de reducir el coste total.
- Tanto estáticos como dinámicos: Así como hay pruebas donde se ejecuta el código del software para demostrar los resultados, también pueden probar y encontrarse defectos sin ejecutar código (revisión de documentos y el código).
- Planificación, preparación, evaluación se llevará a cabo antes y después de la ejecución de pruebas. Tenemos que administrar la planificación de las pruebas, el seguimiento y la presentación de informes. También tenemos que identificar lo que vamos a hacer, mediante la selección de las condiciones de prueba y el diseño de casos de prueba, así como evaluar el sistema bajo prueba los criterios de finalización, lo que ayuda a decidir si hemos terminado nuestras pruebas. Productos de software y productos relacionados con el trabajo: Probamos muchas otras cosas

además de un código como los requisitos, especificaciones de diseño, manuales de usuario, etc.

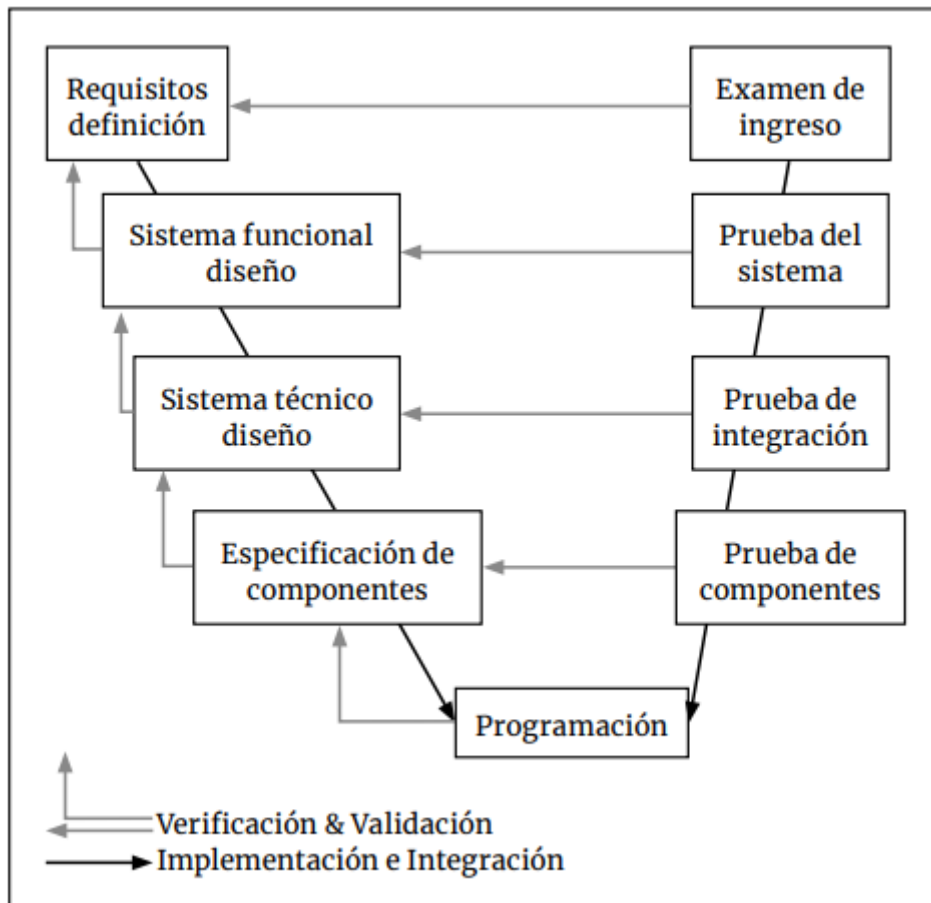
La segunda parte de la definición se refiere a las razones por que lo hacemos:

- Determinar que (los productos de software) cumplir con los requisitos especificados: Si el producto cumple con sus especificaciones, podemos proporcionar esa información para ayudar a los stakeholders a juzgar la calidad del producto y decidir si está listo para su uso.
- Demostrar que (los productos de software) son aptos para el propósito: Hace referencia a si el software hace lo suficiente para ayudar a los usuarios a llevar a cabo sus tareas o no.
- Detectar defectos: Esto nos ayuda a entender los riesgos asociados con poner el software en uso operacional, así como mejorar el desarrollo de procesos para cometer menos errores en el futuro.

EL MODELO - V

Modelo-V es el marco de trabajo para describir las actividades del ciclo de vida de desarrollo software desde la especificación de requisitos hasta el mantenimiento. El modelo-V ilustra cómo las actividades del proceso de pruebas pueden ser integradas dentro de cada fase del ciclo de vida de desarrollo software.

El modelo en V se suele entender como una metodología de testing, sin embargo, se trata en realidad de una adaptación del ciclo de vida clásico o en cascada en la cual se pretende aliviar algunos de sus problemas, como por ejemplo la validación tardía que se realizaba del producto realizando validaciones y verificaciones en paralelo al proceso de desarrollo, las cuales están adaptadas a la fase del proyecto en la que nos encontremos. La V del modelo representa a dos secuencias de fases, la primera se corresponde con la secuencia de fases de desarrollo del proyecto y la segunda con la secuencia de fases de testing del proyecto. Las fases de un mismo nivel se realizan en paralelo. Estas dos cascadas se muestran más alejadas cuanto más abstracta es la visión del producto, es decir, en las etapas más tempranas y se acercan hasta tener como punto de unión la codificación o punto más concreto del desarrollo de software.



Las actividades de la rama izquierda son las conocidas del modelo cascada:

- **DEFINICIÓN DE REQUERIMIENTOS:** Se reúnen las necesidades y requisitos del cliente o futuro usuario. Se especifican y aprueban las características deseadas.
- **DISEÑO FUNCIONAL:** Se mapean los requerimientos en funcionalidades y mensajes.
- **DISEÑO TÉCNICO:** Se diseña la implementación del sistema. Esto incluye las interfaces, y la descomposición del sistema en módulos o sub sistemas (arquitectura). Cada subsistema deberá de ser desarrollado lo más independientemente posible.
- **ESPECIFICACIÓN DE COMPONENTES:** En este paso se definirán las tareas, comportamiento interno, interfaces y estructura de cada subsistema definido en el diseño técnico.
- **PROGRAMACIÓN:** En esta fase se codifica cada módulo especificado.

A través de estos niveles de construcción, el sistema se describirá cada vez con más detalle, siendo lo óptimo encontrar los errores en el nivel de abstracción de donde

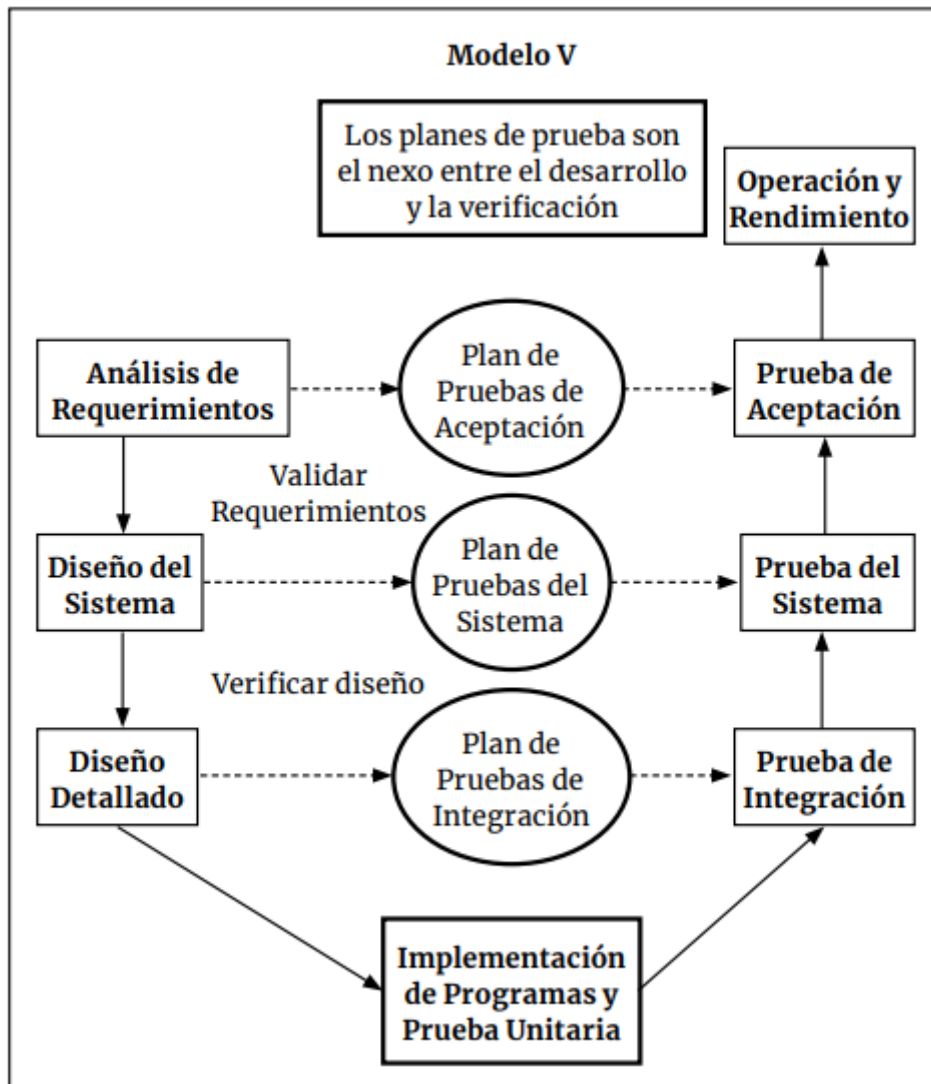
se originaron. Para lograr esto, se define un nivel de prueba equivalente a cada nivel de construcción.

Los NIVELES DE TESTING¹ resultantes son:

- TEST DE COMPONENTES O UNITARIO: Verificará que cada componente de software cumpla con la especificación.
- TEST DE INTEGRACIÓN: Comprobará que los componentes interactúen entre sí, según lo especificado en el diseño técnico.
- TEST DE SISTEMA: Verificará que el sistema se comporte como un todo, y que, como una unidad, actúe según lo especificado.
- TEST DE ACEPTACIÓN: Verificará que el sistema cumpla con los requisitos del cliente, tal y como se especifica por contrato. El sistema deberá de cumplir con las necesidades y expectativas del usuario. Dentro de cada nivel de prueba, el tester debe de asegurarse que los resultados del desarrollo cumplan con los requisitos especificados para ese nivel de abstracción. Este proceso de verificar los resultados de desarrollo de acuerdo con sus requisitos originales se llama validación.

El modelo V puede dar la impresión de que las pruebas comienzan relativamente tarde, después de la implementación del sistema, pero este no es el caso. Los niveles de prueba en la rama derecha del modelo debe interpretarse como niveles de ejecución de la prueba. La preparación de la prueba (planificación de la prueba, análisis de prueba y diseño) comienza antes y se realiza en paralelo a las fases de desarrollo en la rama izquierda (no explícitamente mostrado en el modelo V).

Cada nivel de prueba tendrá un objetivo diferente, y por lo tanto, se utilizarán diferentes métodos, herramientas, y perfiles profesionales. Con lo visto, podemos completar el gráfico del modelo V, para el desarrollo de software con metodologías tradicionales, con algunos conceptos vistos en otras materias (como análisis y diseño de sistemas) y otros que veremos en esta materia:



Ahora que tenemos todo el panorama, veamos cada uno de los niveles de pruebas que se hacen en las distintas etapas del desarrollo del software.

TEST DE COMPONENTES O UNITARIO

Las pruebas de componente (también conocidas como pruebas de unidad, módulo o programa) tienen por objetivo localizar defectos y comprobar el funcionamiento de módulos de software, programas, objetos, clases, etc.. que puedan probarse por separado. Es decir, independientemente del resto del sistema. Este aislamiento es necesario para evitar influencias externas, de esta manera si la prueba detecta un fallo se sabrá que el mismo proviene del mismo componente en sí.

En general las pruebas de componentes se llevan a cabo mediante el acceso al código del objeto, y con el soporte de un entorno de desarrollo, como por ejemplo una herramienta de depuración, o un marco de pruebas unitarias. En la práctica, estas pruebas cuentan con la participación del programador que escribió el código que se está testeando. Los errores detectados se corrigen en el momento sin gestionarlos de manera formal.

- **BASES DE PRUEBA:** Las pruebas unitarias o de componentes se basarán en los requisitos de componentes, el diseño detallado y código fuente.
- **OBJETOS DE PRUEBA:** Los objetos de prueba típicos son módulos de programa / unidades o clases, (base de datos) scripts y otros componentes de software.
- **EN LA PRUEBA UNITARIA SE DEBE DE VERIFICAR:**
 - Correspondencia entre parámetros y argumentos en las llamadas de funciones o módulos.
 - Tipos de datos inconsistentes.
 - Valores de inicialización o/y default incorrectos.
 - Precedencia aritmética o lógica incorrecta o mal entendida.
 - Precisión en las funciones de cálculo.
 - Comparación entre tipos de datos diferentes.
 - Errores en comparación por diferente precisión en los datos que intervienen.
 - Terminaciones de ciclo/loop impropias o no existentes.
 - Variables de ciclo/loop modificadas erróneamente.
 - Manejo de errores inadecuados o inentendibles.
 - Procesamiento de condiciones excepcionales incorrectas

Para ejemplificar los casos de prueba nos vamos a concentrar en el enunciado siguiente:

Una biblioteca universitaria permite que los estudiantes saquen libros en préstamo. Cada estudiante puede retirar hasta **3 libros a la vez** y debe devolverlos dentro del plazo de **7 días**.

Si un estudiante no devuelve el libro en la fecha límite, la biblioteca le cobra una **multa de \$10 por cada día de retraso**. La multa se calcula por cada libro vencido.

Por ejemplo:

- Si un estudiante devuelve 1 libro con 3 días de retraso, deberá abonar \$30.
- Si devuelve 2 libros con 5 días de retraso cada uno, deberá abonar \$100.

El sistema de la biblioteca debe:

1. Registrar los préstamos realizados a cada estudiante.
2. Calcular la fecha límite de devolución (7 días desde el retiro).
3. Verificar la devolución de los libros.
4. Calcular automáticamente la multa en caso de retraso.

Ejemplo Caso de Test de Componentes o Unitario

☞ Se prueba un componente aislado.

Ejemplo:

- El método `calcularMulta(díasRetraso)` debe devolver **\$10** por cada día de retraso.
- Caso de prueba: `calcularMulta(3)` → se espera **30**.
- Si devuelve 30, el **componente funciona bien**.

TEST DE INTEGRACIÓN

Después de la prueba del componente, el segundo nivel de prueba en el modelo V es la prueba de integración. Una condición previa para las pruebas de integración es que los objetos de prueba sometidos a ella (es decir, componentes) ya han sido probados. De esta manera, los defectos propios de cada componente ya deberían de haber sido solucionados. La prueba de integración está orientada a verificar que las partes de un sistema que funcionan correctamente de manera aislada, también lo hacen en conjunto. Para ello se hace foco en testear la interacción, es decir, la comunicación entre los componentes, y no cada componente en sí. El foco de esta tarea estará en encontrar problemas en la colaboración y comunicación entre los componentes.

Las pruebas de integración se ocuparán de probar las interfaces entre los componentes, las interacciones con distintas partes de un mismo sistema, como el sistema operativo, el sistema de archivos y el hardware, así como las interfaces entre varios sistemas.

Existe más de un nivel de prueba de integración, y pueden llevarse a cabo en objetos de prueba de diferentes tamaños:

- 1 - Pruebas de integración de componentes: se ocuparán de probar interacciones entre los componentes del software y se realizarán como paso siguiente a las pruebas de componentes.
 - 2 - Pruebas de integración de sistema: Probarán la interacción entre distintos sistemas, o entre hardware y software. Estas se realizarán a continuación de las pruebas de sistemas.
- BASES DE PRUEBAS: Las pruebas se desarrollarán en base al Diseño del Software y sistema, la arquitectura, los flujos de trabajo y casos de uso.
 - OBJETOS DE PRUEBA: Los objetos de prueba típicos son la Implementación de base de datos, la infraestructura, las interfaces, la configuración del sistema y sus datos.

Ejemplo Caso de Test de Integración

☞ Se comprueba que los componentes funcionen **en conjunto**.

Ejemplo:

- El **módulo de préstamos** llama al **módulo de cálculo de multas**.
- Se prueba que al devolver un libro con 3 días de retraso:
 - El sistema registre la devolución ✓
 - Y genere automáticamente la multa de **30** ✓
- Aquí no se prueba cada módulo aislado, sino la **comunicación entre ellos**.

TEST DE SISTEMAS

Una vez completada la prueba de integración, el tercer y siguiente nivel de prueba es la prueba del sistema. La prueba del sistema verifica si el producto integrado cumple con los requisitos especificados. ¿Por qué es esto aún necesario después de ejecutar el componente y pruebas de integración?

Las razones para esto son las siguientes:

- En los niveles de prueba más bajos, la prueba se realizó en función de especificaciones técnicas, es decir, desde la perspectiva técnica del productor de software.

Las prueba del sistema, sin embargo, miran el sistema desde la perspectiva del cliente y el futuro usuario. Los probadores validarán si los requisitos se implementan de manera completa y adecuada. · Muchas funciones y características del sistema resultan de la interacción de todos los componentes del sistema; en consecuencia, son visibles solo cuando el todo el sistema está presente y puede ser observado y probado.

El objetivo de la prueba del sistema es validar si el sistema completo cumple con los requisitos funcionales y no funcionales especificados y qué tan bien lo hace.

- Es en realidad como un test de unidad, de una unidad formada por otras unidades ya integradas.
 - La diferencia con el test de unidad es que puede ser encarada por QA y las funcionalidades tienen visibilidad para el cliente.
 - La diferencia entre sistema y módulo puede estar en el foco del test y el alcance funcional (servicio).
 - Es la prueba realizada cuando una aplicación está funcionando como un todo (Prueba de la construcción Final).
-
- BASES DE PRUEBAS: Las pruebas de sistemas pueden incluir pruebas basadas en Riesgos, especificaciones de requerimientos, procesos de negocio, casos de prueba, escenarios, descripciones de alto nivel sobre el comportamiento del sistema, interacción con sistemas operativos y recursos, etc.
 - OBJETOS DE PRUEBAS: Manuales de sistema, usuario y funciona miento, configuración del sistema, datos de configuración, el sistema.

Ejemplo Caso de Test de Sistema

☞ Se prueba el sistema **completo como una sola unidad.**

Ejemplo:

- Un usuario inicia sesión, busca un libro, lo reserva, lo retira y luego lo devuelve con retraso.

- El flujo completo debe:
 - Registrar la reserva ✓
 - Permitir el préstamo ✓
 - Al devolver, aplicar la multa correctamente ✓
 - Generar recibo de pago ✓
- Acá se valida que **todo el sistema** respete las especificaciones técnicas.

TEST DE ACEPTACIÓN

Todos los niveles de prueba descritos hasta ahora representan actividades de prueba que son responsabilidad del equipo de desarrollo. Estas pruebas se ejecutan antes que el software se presente al cliente o al usuario. Sin embargo, antes de instalar y utilizar el software en la vida real (especialmente para el software desarrollado individualmente para un cliente), otro último nivel de prueba debe ser ejecutado: la prueba de aceptación. Aquí, el foco está en el cliente y la perspectiva del usuario. Suele ser la única prueba en la que los clientes realmente están involucrados o que pueden entender.

El objetivo de las pruebas en este caso no es encontrar errores, si no, establecer confianza en el sistema que se está entregando. La finalidad es que el usuario conozca el producto, evalúe su usabilidad, y sugiera mejoras. Los tipos más comunes de pruebas de aceptación son:

PRUEBAS DE ACEPTACIÓN DE USUARIO: En general se verifica la idoneidad del uso del sistema por parte de los usuarios comerciales.

PRUEBAS OPERATIVAS: Es la aceptación por parte de los administradores del sistema, en estas se evaluará: funcionamiento de backUp, Recuperación de desastres, gestión de usuarios, tareas de mantenimiento, cargas de datos y migraciones, seguridad, etc.



PRUEBAS DE ACEPTACIÓN CONTRACTUAL O NORMATIVA: En esta se toman como base los criterios de aceptación previstos en el contrato de desarrollo del software a medida. **PRUEBAS ALFA Y BETA (DE CAMPO):** Cuando se desarrolla un software para lanzar al mercado, los desarrolladores buscan obtener feedback de los potenciales clientes antes de poner a la venta el producto. Para ello, implementan el producto con

la finalidad de que estos potenciales usuarios lo prueben. Las llamadas pruebas ALFA se llevan a cabo en los establecimientos de los equipos de desarrollo, es decir, los potenciales usuarios se dirigen al establecimiento donde está el equipo de desarrollo para probar el sistema. Las pruebas BETA en cambio, son llevadas a cabo en los establecimientos de los clientes

Ejemplo Caso de Test de Aceptación

☞ Se valida contra los **requisitos del cliente** y la **expectativa del usuario**.

Ejemplo:

- El cliente pidió:
 - Que las multas se calculen automáticamente.
 - Que el usuario pueda pagarlas online.
 - Que reciba un correo confirmando el pago.
- El test verifica:
 - El usuario efectivamente puede pagar su multa online 
 - Y recibe el correo de confirmación 
- Si esto se cumple, el sistema está **listo para la aceptación del cliente**.

A PENSAR Y REFLEXIONAR!

A continuación, te invito a responder el siguiente cuestionario con respecto a los temas de la clase, es de repaso, pero es importante para pensar qué nos llevamos de la clase, no copies y pegues, sacá tus propias conclusiones! No lleva calificación, pero son parte de tu aprendizaje. No debes subirlo a la plataforma.

1. ¿Qué es el modelo V? ¿Para qué sirve?
2. ¿Qué son los niveles de testing?
3. ¿Cuáles son los niveles de testing según este modelo?
4. ¿Qué es lo que se prueba en cada nivel de testing

REFERENCIAS BIBLIOGRÁFICAS

Pressman, R. S. (2010). *Ingeniería del software: Un enfoque práctico* (7.ª ed.). McGraw-Hill.

Toledo, F. (2016). *Introducción a las pruebas de sistemas de información*. Universidad ORT Uruguay.