

Componente formativo

**Análisis de pruebas**

**Breve descripción:**

En este componente se abordarán temas referentes a las pruebas o procesos de *testing* de *software* antes de la puesta en marcha; estas se realizan en varias etapas, las cuales determinan la usabilidad y desempeño de la aplicación, basado en la mejor experiencia de usuario, que es el enfoque y la estrategia para diseñar sistemas bien estructurados.

**Área ocupacional:**

Ciencias naturales, aplicadas y relacionadas

**Noviembre 2022**

Tabla de contenido

[Introducción 3](#_Toc142509702)

[1. Terminología relacionada con pruebas 4](#_Toc142509703)

[2. Contexto de la solución 8](#_Toc142509704)

[3. Enfoque de pruebas 11](#_Toc142509705)

[Síntesis 20](#_Toc142509706)

[Material complementario 21](#_Toc142509707)

[Glosario 22](#_Toc142509708)

[Referencias bibliográficas 23](#_Toc142509709)

[Créditos 24](#_Toc142509710)

Introducción

Apreciado aprendiz, bienvenido a este componente formativo, en el cual encontrará diferentes temas orientados hacia las pruebas realizadas al *software*, a los procesos que conlleva y a la mejor forma de hacerlas, para la puesta en marcha de los sistemas de información. Se abordan temas relacionados con la terminología asociada a los *test* de *software*, en qué ambientes se llevan a cabo, así como la relación con otras actividades realizadas en el proceso de desarrollo, los objetivos propuestos para implementarlos y cumplirlos a cabalidad.

En el siguiente video conocerá, de forma general, la temática que se estudiará a lo largo del componente formativo:

1. Introducción – Análisis de pruebas



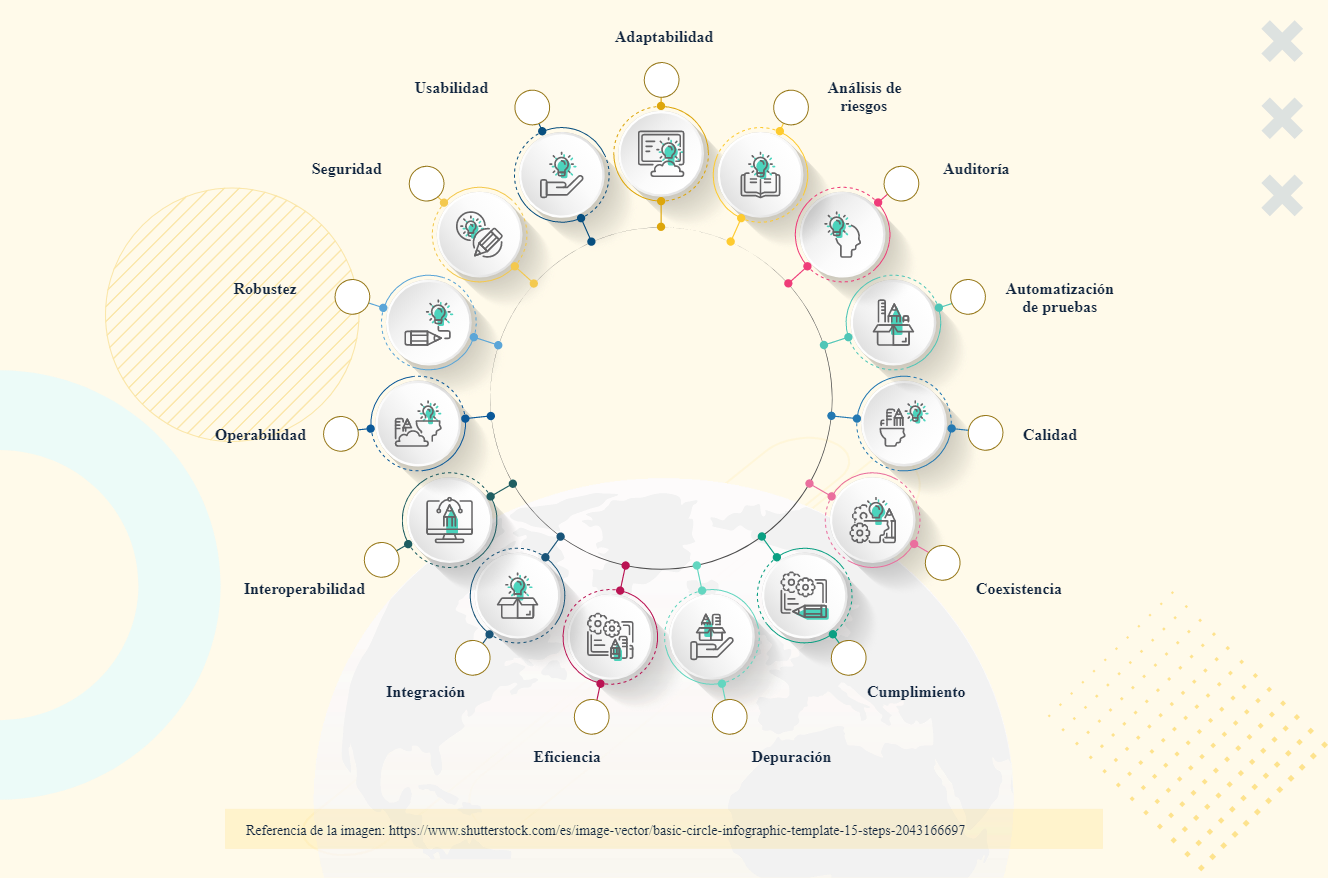
[**Enlace de reproducción del video**](https://www.youtube.com/watch?v=IFNX8ohapHE)

|  |
| --- |
| **Síntesis del video: Introducción – Análisis de pruebas** |
| Bienvenido a este componente formativo, se entenderá la importancia de las pruebas, al aplicar su función para la identificación de fallos en el software, las diversas herramientas que se emplean para realizar el análisis, administración y ejecución de las pruebas, dado a que, al construir un software, se cometen errores.  En las organizaciones, para solucionar los problemas originados por los errores, se utilizan las pruebas de software o testing, una serie de pasos que se realizan antes y después de construir el software, ayudan a obtener mejores resultados, ofrecer una mayor calidad del producto y, como consecuencia, los clientes estarán más satisfechos.  El proceso de prueba de software tiene tres pasos principales: el desarrollo de los casos de prueba, la ejecución de estos y el análisis de los resultados de la ejecución.  Conscientes que su disciplina y estudio de esta temática, generará mejores resultados en el cumplimiento de sus metas, se invita a abordar el estudio de estos temas. |

# Terminología relacionada con pruebas

Existen definiciones y terminologías asociadas a las pruebas de *software* que enmarcan los conocimientos, acontecimientos y prácticas que se realizan para la puesta en marcha de este, luego de la fase de producción o creación del producto que determina la solución a las necesidades del cliente, a partir de los requerimientos establecidos y solicitados. La realización de pruebas se hace con base en la búsqueda de errores generados en los sistemas desarrollados, no solo con el fin de determinar el buen funcionamiento del sistema y pensar que está listo para la puesta en marcha, sino que todo aplicativo desarrollado genera errores de cualquier índole, por lo tanto, es indispensable tenerlo presente y realizar las pruebas correspondientes.

A continuación, conozca algunas de las terminologías más utilizadas al realizar pruebas de *software*:



1. **Adaptabilidad:** El producto *software* puede adecuarse a distintos entornos a los cuales sea sometido sin la necesidad de utilizar factores externos para este fin, solo con las funcionalidades y características por las cuales fue desarrollado.
2. **Análisis de riesgos:** Tener en cuenta los riesgos en todo el proceso del desarrollo del *software* para saber afrontar y manejar los errores y fallas.
3. **Auditoría:** Evaluación realizada de forma independiente al desarrollo, para determinar los requerimientos y especificaciones del *software* a partir de los estándares y normas establecidos.
4. **Automatización de pruebas:** Ayudan al proceso de las pruebas del *software* y todo lo relacionado al diseño, puesta en marcha, gestión y comprobación de pruebas.
5. **Calidad:** Es el grado de satisfacción por el cual un sistema de información sometido a varios procesos de funcionalidad, cumple los requisitos y necesidades solicitados por el cliente.
6. **Coexistencia:** Es la capacidad del *software* de utilizar recursos y ambientes que son compartidos a través de otra plataforma tecnológica diseñada para otro fin y que son independientes.
7. **Cumplimiento:** Es la capacidad del *software* de cumplir con las especificaciones y estándares establecidos por los entes reguladores y que son claves para darle respuesta al cliente y satisfacer sus necesidades.
8. **Depuración:** Es un procedimiento en el cual se realiza análisis, se eliminan las fallas en el sistema y se solventan algunos errores generados en el proceso de desarrollo.
9. **Eficiencia:** Está relacionado con el rendimiento y uso óptimo del sistema, al igual que las capacidades generales.
10. **Integración:** Capacidad para relacionar componentes del *software* para hacerlo más complejo y funcional, con más capacidades.
11. **Interoperatividad:** Permite al *software* relacionarse o comunicarse con otros componentes o aplicaciones.
12. **Operabilidad:** Permite a los distintos usuarios su administración y uso, dependiendo el rol que tengan para tener acceso a los diferentes módulos del sistema.
13. **Robustez:** Capacidad del sistema para enfrentar entornos hostiles y ambientes distintos a los manejados de forma normal, en el cual opera de forma consistente y adecuadamente.
14. **Seguridad:** Capacidad del *software* de prevenir inconsistencias y fallos.
15. **Usabilidad:** Facilidad para usar el sistema, entendible y aprendido por todos los usuarios en distintos campos específicos.

Ahora bien, las pruebas involucran actividades correspondientes al diseño del plan en el que se realiza el modelo y la forma como se desarrollan; así mismo, se tiene en cuenta la localización de errores al momento de validar las funcionalidades del sistema, sin olvidar que la experiencia de usuario es de suma importancia en la etapa de pruebas, así como la arquitectura de la información en la que se debe verificar la estructura, conformación y contenido de los datos que se quieren mostrar, lo cual no debe complicar la navegación. Recuerde que no debe tener mucho contenido, debe ser fácil e intuitivo, con la finalidad que el usuario encuentre lo que necesita rápidamente y se sienta en una plataforma amigable.

La siguiente figura presenta las actividades relacionadas con las pruebas.

1. Actividades relacionadas con las pruebas

La imagen nos muestra las actividades relacionadas con las pruebas:
Diseño de plan de pruebas
Errores / fallas
Experiencia de usuario
Arquitectura de información

Las pruebas que se desarrollan en cada uno de los proyectos de *software*, tienen como objetivo encontrar la mayor cantidad posible de errores en cada uno de los módulos o secciones del sistema, con el fin de realizar la retroalimentación oportuna para que se corrijan las fallas encontradas, aportando posibles soluciones.

El departamento de QA (Quality Assurance) realiza pruebas a las plataformas tecnológicas, con el fin de determinar su funcionamiento de forma eficiente y eficaz, por ende, es necesario que esta área realice los distintos planes para ejecutar pruebas y así dirigir a cabalidad cada esfuerzo de la operación en la metodología utilizada.

Además, permite:

1. Analizar la automatización.
2. Enfocarse en experiencias de usuarios con el fin de determinar la forma de utilización.
3. Hacer estudios en arquitectura de la información para mostrar de forma organizada y detallada, lo que los usuarios realmente quieren en una respectiva plataforma tecnológica.

Un sitio o web o cualquier tipo de aplicación que se desarrolle e implemente, también debe cumplir con la utilización de programas de seguimiento de errores, para asegurar el funcionamiento óptimo del *software*.

# Contexto de la solución

Las pruebas son esenciales en cualquier proyecto de *software*, permiten tener mejores resultados al desarrollar aplicaciones, y de esto depende que los clientes o usuarios estén satisfechos con la solución; hacer los ensayos respectivos implica encontrar errores y evitar pérdidas luego de la puesta en marcha. Los probadores deben regirse a partir de los requerimientos funcionales y no funcionales del *software*, para encontrar defectos y, a partir de estos, se deben elegir requisitos comprobables de las pruebas. Las aplicaciones solo están listas para las pruebas cuando se hayan terminado las fases anteriores a esta, luego se debe diligenciar la documentación respectiva de los resultados esperados con los reales, en los que se hayan detectado daños y errores.

Las pruebas deben realizarse luego de la culminación de la implementación y también después de cada corrección, porque:

* Cuando hay defectos, falencias o errores encontrados y documentados en las pruebas…
* El equipo de desarrollo debe corregir.
* Se pasa al equipo de pruebas para verificar todos los módulos y servicios e identificar que se hayan corregido los errores y que la solución de estos no haya generado otros.

Es así como la metodología utilizada en el proceso de desarrollo de *software*, requiere tener limitaciones en la determinación de cada uno de los procedimientos. Hay una fase en la cual se planifican y otra que llevan a cabo, teniendo las restricciones requeridas, es en este aspecto en el cual se visualizan los alcances del *software*, con la necesidad de hallar falencias y errores en todo el programa; esto ayuda a corregirlos antes de la etapa de producción, en la cual saldría costoso y con un riesgo enorme de pérdidas económicas y de otros tipos.

Los sistemas de información, por muy sencillos que sean, tienen una gran cantidad de niveles o casos de pruebas, las cuales deben ser planificadas, administradas, ejecutadas y sobre todo, utilizar tecnologías y herramientas para realizar pruebas automatizadas. Estas tecnologías utilizadas para llevar a cabo las pruebas, ayudan a que el *software* obtenga la calidad deseada y los recursos sean manejados eficientemente.

Existe gran variedad de herramientas que se utilizan para la gestión de estas tareas, conozca algunas de ellas:

1. **Herramientas para pruebas estáticas:** Permiten la detección de fallas y defectos al inicio del proyecto. Algunos tipos son:
   * **Herramientas de revisión:** utilizadas para informar o dar a conocer comentarios sobre las revisiones e informes sobre fallas encontradas.
   * **Análisis estático:** permiten distinguir y encontrar errores en el código antes de llevar a cabo las pruebas dinámicas; ejecutan estructuras para la codificación, análisis y dependencias.
   * **Herramientas de modelado:** permiten verificar y contemplar los modelos del *software* como la información de una base de datos relacional en la consecución de inestabilidades, inconsistencias y defectos.

Existen varias herramientas que permiten realizar la automatización de pruebas estáticas como son: PMD, ChekStyle, SONAR, Simian, FindBugs, MacCaabeTest.

1. **Herramientas para la planificación y gestión:** Están relacionadas con la documentación de las pruebas, mantenimiento y gestión de resultados de estas tareas, en este caso, permiten documentar, analizar y gestionar las debilidades e incidencias en el proyecto de desarrollo.

Las herramientas utilizadas en la planificación y gestión son: Teslink, Redmine, Trello, Mantis, HP Quality Center, IBM Rational quiality mánager, Testopia, BugZilla.

1. **Herramientas para pruebas de automatización:** Funcionan a partir de scripts que se pueden realizar en distintos lenguajes de programación, para ejecutar herramientas de pruebas funcionales de forma automática. Las herramientas utilizadas son: Selenium, QTP, SoapUI, Cucumber, IBM Rational Automation Framework.
2. **Herramientas de pruebas de carga y rendimiento:** Permiten llevar a cabo pruebas en las cuales se intensifique la sobrecarga de la aplicación, y así validar hasta qué nivel puede responder, se realiza sobre todo a sistemas que trabajan en tiempo real para establecer los límites de este. Las herramientas utilizadas son: Imeter, LoadUI, FunkLoad, HP Load runner.
3. ***Drivers* *y* *Stubs*:** Son miniaplicaciones o programas pequeños, diseñados para la realización de pruebas en la integración de los módulos del sistema. Los *drivers* permiten brindar información a los módulos que están debajo de él, es decir, funciona a un alto nivel del *software* y los *stubs* reciben información de módulos que están más arriba; por lo tanto, el trabajo realizado, funciona a un nivel bajo del sistema.

Por otra parte, cuando se realizan pruebas de *software* para identificar los riesgos del producto, la idea es anticipar las fallas de las funcionalidades y módulos del sistema para afrontar su impacto ante el cliente.

Enfocarse en los riesgos facilita el trabajo para los controles de calidad, diseño y ejecución de pruebas, debido a que se priorizan con respecto a las fallas, también permite realizar una evaluación, e identificar áreas propensas a fallas y de alto riesgo dentro de la plataforma tecnológica.

Trabajar en los riesgos y tenerlos definidos con anticipación va a ayudar en muchos aspectos, como por ejemplo:

1. Optimización de pruebas.
2. Reducción de costos por la implementación de tareas optimizadas y sencillas.
3. Permite dirigir y enfatizar los casos de pruebas hacia esos elementos de alto riesgo, para determinar las fallas y resolverlas rápidamente.
4. Permite claridad para delimitar el alcance de las pruebas.
5. Aumenta la calidad del *software*, porque se implementan herramientas menos costosas que determinan defectos en la aplicación.

El diseño del plan de pruebas para mantener una buena calidad del *software*, debe prepararse con anticipación y objetividad hacia los errores o fallas que se puedan encontrar, para así contar con la solución y el tiempo requerido. Al no contar con esta planificación, es complejo determinar los tiempos y soluciones a las inconsistencias encontradas, además la inversión y recursos es mayor y más prolongada. Por lo tanto, es necesario que se tenga en cuenta, aún desde la concepción del proyecto.

La gestión de la calidad es un componente de solución de *software* que junto a las distintas actividades que se realizan en ella, permiten la detección temprana y la corrección de errores y fallas en la construcción de este. Para determinar una solución, en la etapa del plan de pruebas, es necesario diseñar y desarrollar casos y tareas que abarque todo lo concerniente en el *software* como iteraciones, combinaciones, permutaciones, con el fin de determinar cuáles benefician o afectan los módulos o subsistemas. En ese orden de ideas, se indagan las operaciones que realiza el producto, se determinan los casos, ponerlos en primera medida y finalmente, realizar programas automatizados para gestionar las tareas de pruebas.

# Enfoque de pruebas

Anteriormente, se ha visualizado la importancia de realizar pruebas de *software* y los riesgos que tiene cada proceso del ciclo de su desarrollo, sin embargo, existen alternativas y opciones para determinar el plan de pruebas, es un punto crucial en el que es necesario abordar todos los aspectos de inicio a fin y, la forma como se realizan en todos los componentes del sistema. De igual forma, se deben tener en cuenta los recursos y, sobre todo, la documentación del proyecto y del proceso como tal. En el enfoque de pruebas se tienen en cuenta las distintas técnicas usadas, características y procesos de cada una, así como la metodología, cobertura y métricas aplicadas en este ciclo.

En el ciclo de vida de desarrollo de *software*, se contemplan métodos que llevan a la definición y estructura de las estrategias para el plan de pruebas, existen alternativas y métodos para el desarrollo de programas como, por ejemplo, el modelo en cascada, el modelo general en V y otros como metodologías ágiles, que son implementadas en muchas empresas. Todos ellos usan su propia estructura para la implementación de aplicaciones y la puesta en marcha. En el modelo V se contemplan igualdades en los procesos de desarrollo (lado izquierdo) y actividades de prueba, (lado derecho), como se presenta en la siguiente figura.

1. Modelo V

la imagen nos muestra el modelo V, teniendo al lado izquierdo:
Definición de requerimientos
análisis del sistema 
diseño de software
diseño de módulos
Al lado derecho:
pruebas de aceptación 
pruebas de sistema
pruebas de componentes
pruebas unitarias
y en la parte de abajo: Código.

En los niveles de desarrollo deben realizar una prueba por cada nivel, en este sentido, el encargado de realizarlas debe constatar que se cumpla con la calidad del *software*, para lo cual existe verificación y validación en todo el proceso.

Se amplía la información sobre algunas de estas pruebas:

1. **Verificación y validación (VyV):** La validación consiste en que los requerimientos solicitados por el cliente se cumplan a cabalidad, por su parte, la verificación es la que determina si los procesos de implementación del *software* cumplen con las funcionalidades o funcionalidad específica. En los procesos VyV existen una serie de actividades que permiten dar calidad; además de las pruebas, se tienen en cuenta las revisiones técnicas, simulación, revisiones periódicas y documentación, entre otras.
2. **Pruebas unitarias o de componentes:** Son pruebas que se realizan de forma individual a cada componente del *software*, funcionalidad o fragmento de código; cada unidad es invocada desde una clase o componente externo para formar el *software*, por lo tanto, cada una debe ser funcionalmente independiente. El desarrollador es quien debe realizar estas pruebas, porque es el más idóneo e implementó el módulo o componente del sistema, por lo tanto, conoce la estructura generada.
3. **Pruebas de integración:** Estas pruebas son necesarias para constatar que los componentes o módulos del sistema, pueden funcionar correctamente acoplados e integrados, debido a que pueden ocurrir imprevistos o errores no contemplados. Este tipo de pruebas verifican el funcionamiento, consolidación e interacción entre partes del sistema, ya sea con el sistema operativo, el *hardware* en el cual se está corriendo, y las interfaces en el sistema.
4. **Pruebas de validación o aceptación:** Estas pruebas se realizan por parte del cliente y son llevadas a cabo antes de la puesta en marcha o etapa de producción.

Se complementa un poco más la información sobre las pruebas de integración y de validación o aceptación.

1. **Integración descendente:** Consiste en que el componente de mayor nivel integra otros componentes del sistema, por lo tanto, esta prueba es incremental para definir la arquitectura del sistema.
2. **Integración ascendente:** Se realizan pruebas con los componentes más simples y sencillos de la aplicación y luego se van integrando a los de mayor envergadura o a los subsistemas.
3. **Integración *ad-hoc*:** En la medida que se termina la implementación de cada componente, estos se van probando y posteriormente integrando con otro componente en iguales condiciones, es decir, ya probado.
4. **Integración de esqueleto:** Se realiza integración de componentes sobre un componente o subsistema con una complejidad moderada o avanzada, los componentes más pequeños se van agregando a este.

Se finaliza con las pruebas de validación y aceptación.

Se encuentran las siguientes pruebas:



1. **Pruebas de aceptación del contrato:** Se validan los datos e información del contrato y los requerimientos ahí establecidos, pueden ocurrir casos en que no se cumpla esas condiciones y al realizar las pruebas por parte del cliente, surjan errores en el *software*.
2. **Pruebas de aceptación del usuario:** Consiste en realizar pruebas con el usuario final de la plataforma tecnológica, porque ocurre que, en muchos casos, cuando se lleva a cabo la implantación del *software*, algunos usuarios no contemplan una o varias funcionalidades, mientras que otros sí dan la aceptación.
3. **Pruebas operativas:** Se realizan con el administrador de tecnologías de la información -TI-, para realizar distintos procesos como creación de usuarios, copias de seguridad, mantenimientos, vulnerabilidades en la infraestructura, entre otros.

Existen diversas metodologías para realizar pruebas de *software* con la calidad deseada, pueden ser estáticas, dinámicas o basadas en la experiencia.

**Pruebas estáticas**

En este tipo de pruebas se tienen en cuenta los documentos de especificación de requisitos para analizar y revisar el código fuente de la aplicación, con el fin de detectar errores al inicio del desarrollo.

En ellas se encuentra:

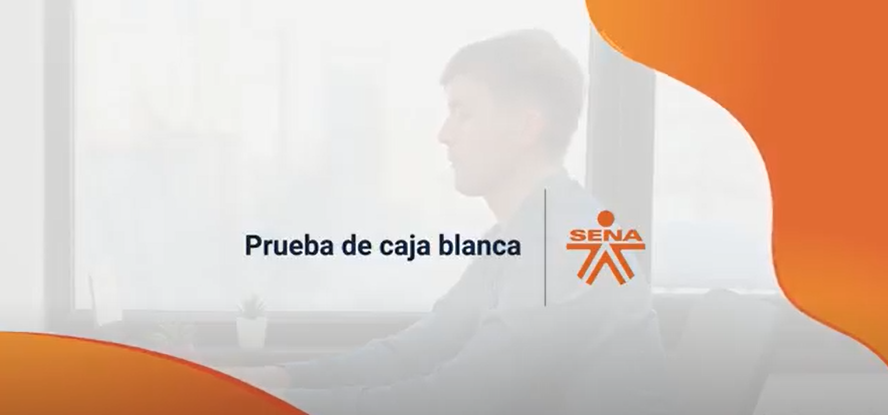
1. **Análisis estático:** No se realiza ejecución del *software*, si no, para encontrar fallas en el código fuente del sistema, por lo tanto, se hace necesario hacer uso de herramientas para analizar el código de la aplicación, con el fin de encontrar errores en la implementación. Los defectos más comunes hallados son relacionados con interfaces inconsistentes entre módulos, código inaccesible y sin lógica, entre otros.
2. **Revisiones:** Se realizan con el fin de establecer correcciones a las fallas encontradas de una forma temprana, lo que implica reducción en costes de tiempo y dinero. En este ámbito se pueden encontrar los defectos en requisitos, en el diseño e interfaces incorrectas.
3. **Inspección:** Se realiza sobre los documentos del proyecto para validar el cumplimiento de estándares de calidad del *software*, es realizado por varios entes como revisor, escriba y moderador, personas encargadas y responsables de las revisiones o inspecciones del producto.

**Pruebas dinámicas**

Se realizan con la ejecución de la plataforma tecnológica, en la cual se encuentran dos formas de probar el funcionamiento, la internamente conocida como técnica de caja blanca, en la cual se validan los módulos y la relación entre ellos con sus diferentes funcionalidades y, las pruebas que se realizan de forma externa o técnicas de caja negra, en la que se determinan las funcionalidades operativas y las fallas ocurridas en el entorno de la aplicación.

Para conocer la técnica de prueba de caja blanca, ver el siguiente video:

1. Prueba de caja blanca



[**Enlace de reproducción del video**](https://www.youtube.com/watch?v=6cnuU7u86Xw)

|  |
| --- |
| **Síntesis del video: Prueba de caja blanca** |
| A partir de estructuras de control if, else, for, while, se obtienen casos de pruebas, en sistemas de información. Se validan estructuras de datos, condiciones lógicas aplicadas y rutas del código del programa.  Dentro de las pruebas de caja blanca, se tiene:  Prueba de ruta básica: consiste en probar todos los caminos de ejecución de una aplicación o programa, y realizar una prueba por cada uno de los caminos independientes del programa. Es necesario tener presente la representación de grafo.  Prueba de ciclo o bucle: se puede mencionar que los bucles son fundamentales en la codificación del software. Se tienen cuatro categorías: Bucles simples. Bucles anidados. Bucles concatenados. Bucles no estructurados. |

Continuando con las pruebas de caja negra.

Se realizan al finalizar todas las pruebas, estas técnicas están basadas en la especificación funcional y no funcional del sistema y tienen en cuenta la estructura de la aplicación que se desarrolla. En las pruebas de caja negra se perciben funciones incorrectas o faltantes, errores de inicio y fin, fallas en la interfaz y también en estructuras de datos.



En esta técnica se observan varios métodos:

1. **Partición equivalente:** Se aplican a datos válidos y no válidos de una clase, e igualmente a valores de salida, internos y relativos a las interfaces. En este caso se tienen en cuenta las condiciones de entrada si hacen referencia a rangos o valores específicos, se define una clase con equivalencia válida y dos inválidas. Si las condiciones de entrada hacen referencia a conjuntos o entradas booleanas, se define una clase válida y otra inválida.
2. **Análisis del valor límite:** Tiene en cuenta los valores extremos o límites de una clase y determina el mayor número de errores, se realiza posterior a la participación equivalente como complemento a esa técnica.
3. **Pruebas de tabla de decisión:** Se basa en decisiones y condiciones lógicas como verdadero y falso, valores booleanos, además identifican los requerimientos basados en las condiciones lógicas en el sistema
4. **Pruebas de caso de uso:** Permiten definir pruebas de aceptación con la colaboración de los usuarios, teniendo en cuenta que los casos de uso muestran la forma de interactuar de ellos con el sistema.

**Pruebas basadas en la experiencia**

Hacen referencia a las técnicas, conocimientos y experiencias del encargado de realizar las pruebas, además, de acuerdo con las tecnologías y la relación con aplicaciones de características semejantes, haciéndolas idóneas para el probador.

Existen dos tipos de pruebas basadas en experiencia:

1. **Predicción de error:** Se definen los errores del sistema, componentes, módulos o funciones, a partir de experiencia o habilidades del encargado de realizarlas.
2. **Pruebas exploratorias:** Se basan en la autonomía y responsabilidad del probador para realizarlas conforme al aprendizaje, diseño y ejecución de manera simultánea de las pruebas.

**Cobertura**

Se conoce como medida para la calidad de las pruebas e indica la cantidad de código del programa que se ha probado. La cobertura indica lo que abarca cada una de las tareas, se basa en criterios de los cuales se contiene uno al otro y así, se generaliza con respecto a los casos probados, componentes o funcionalidades del *software*. La cobertura permite conocer las medidas por las cuales se examina un sistema de información, con las pruebas que se realizan de acuerdo con criterios de calidad establecidos.

**Métricas y recursos de pruebas**

Realizar mediciones sobre las pruebas a los módulos o componentes de software son determinantes en la definición de la calidad del sistema que se construye, con esto se estiman los errores y fallos encontrados en la fase de pruebas del programa. Si no se cuenta con las métricas en los procesos de casos de pruebas, no es específico ni objetivo el proceso, porque no se determinan ni se cuantifican las tareas realizadas. Por el contrario, al contar con datos cuando se llevan a cabo, se establecen lineamientos y cifras exactas a través de cada proceso que se puso en marcha en esa fase, como por ejemplo, cantidad de pruebas realizadas sobre los distintos módulos, cantidad de errores encontrados por funcionalidad, faltantes por llevarse a cabo, cantidad de resultados nulos, fallas o completas.

Se cuenta con dos tipos de métricas para pruebas, las cuales determinan las mediciones al realizar los casos de pruebas en los ambientes, componentes y módulos del sistema.

1. **Métrica base:** Hacen referencia a los datos medidos por el probador en el proceso de desarrollo y ejecución de los casos de pruebas, a estos datos continuamente se les realiza seguimiento en todo el ciclo de vida de las pruebas.
2. **Métrica calculada:** Nacen de los datos recolectados de las métricas base, son realizadas por el líder de pruebas a partir de los informes de pruebas.

Recuerde explorar los demás recursos que se encuentran disponibles en este componente formativo; para ello, diríjase al menú principal, en el cual encontrará la síntesis, una actividad didáctica para reforzar los conceptos estudiados, material complementario, entre otros.

Síntesis

El siguiente mapa integra los criterios y especificidades de los conocimientos expuestos en el presente componente formativo.

La imagen nos muestra un mapa conceptual con todo el contenido estudiado en este componente.

Análisis de pruebas dividido en tres componentes: 

1. Alcance: 
-Objetivos: Medibles, alcanzables. 
-Escenarios: Caja negra, caja blanca, otros. 

2. Escenarios: Descripción, condiciones, criterios de finalización. 

3. Evaluación mediante pruebas: Plan de pruebas 

[Anexo. Síntesis](sintesis.pdf)

Material complementario

|  |  |  |  |
| --- | --- | --- | --- |
| Tema | Referencia APA del Material | Tipo de material | Enlace del Recurso o  Archivo del documento material |
| Terminología relacionada con pruebas | Engineering Lifecycle Management – IBM. (2021). *Evaluación de riesgos*. | Página web. | <https://www.ibm.com/docs/es/elm/6.0.2?topic=artifacts-assessing-risk> |
| Contexto de la solución | Software Testing Bureau. (2022). *Crear un buen Plan de Pruebas.* | Página web. | [https://www.softwaretestingbureau.com/crear-un-buen-plan-de-pruebas/#:~:text=El%20plan%20de%20pruebas%20tiene,cambios%20que%20tiene%20el%20proyecto](https://www.softwaretestingbureau.com/crear-un-buen-plan-de-pruebas/#:~:text=El plan de pruebas tiene,cambios que tiene el proyecto) |
| Enfoque de pruebas | Myservername.com. (2022). *Tutorial de Mantis Bug Tracker: Guía completa de MantisBT Issue Tracker*. | Página web. | <https://spa.myservername.com/mantis-bug-tracker-tutorial> |
| Enfoque de pruebas | Turrado, J. (2020). *Qué son las pruebas de software.* | Artículo de divulgación. | <https://www.campusmvp.es/recursos/post/que-son-las-pruebas-de-software.aspx> |
| Enfoque de pruebas | Trans-TI. (2021). *Software testing: Conocé el ciclo de vida de las pruebas de Software.* | Página web. | <https://trans-ti.com/2021/01/11/software-testing-conoce-el-ciclo-de-vida-de-las-pruebas-de-software/> |

Glosario

**Bucle:** procesos repetitivos, son instrucciones de código que se pueden ejecutar muchas veces.

**Calidad:** cualidades que caracterizan al software y a su vez identifican su utilidad y existencia en las que cumple y supera las expectativas de los usuarios.

**Cobertura:** cantidad o porcentaje que abarca una actividad.

**Error:** inconsistencia o problema que devuelve un resultado no deseado.

**Falla:** en la ejecución de una aplicación, se manifiesta como un defecto visible dentro de las pruebas de software.

**Inspección:** verificación que se realiza de un componente de software con el fin de que se cumplan las condiciones requeridas.

**Prueba:** proceso de evaluación y verificación de los componentes o módulos del sistema, con el fin de encontrar fallos y establecer parámetros que definan calidad en el producto.

**Revisión:** comprobación que una funcionalidad de un componente de software se ejecute correctamente.

**Software:** conjunto de métodos u operaciones que interactúan entre sí para llevar a cabo funcionalidades que automatizan una tarea en específico.

**Técnica:** procedimientos que se llevan a cabo para obtener un resultado deseado con respecto a una funcionalidad o componente de software.

**TI:** hace referencia a tecnologías de la información, está enfocado en las aplicaciones y el hardware de computadoras.

Referencias bibliográficas

Gómez, M., Cervantes, J. & González, P. P. (2019). *Fundamentos de Ingeniería de Software*. <https://www.academia.edu/66057456/Fundamentos_Ing_SW_VF_1_>

Toledo., F. (2018). *¿Qué es la cobertura de pruebas?* <https://www.federico-toledo.com/que-es-la-cobertura-de-pruebas/>

Créditos

Elaborado por Innovative Education

