**FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO**

| PROGRAMA DE FORMACIÓN | ANÁLISIS Y DESARROLLO DE SOFTWARE |
| --- | --- |

| COMPETENCIA | 220501096 desarrollar la solución de software  de acuerdo con el diseño y metodologías de desarrollo. | RESULTADOS DE APRENDIZAJE | 220501096-05 Realizar pruebas al software para verificar su funcionalidad |
| --- | --- | --- | --- |

| NÚMERO DEL COMPONENTE FORMATIVO | 38 |
| --- | --- |
| NOMBRE DEL COMPONENTE FORMATIVO | Las pruebas de software |
| BREVE DESCRIPCIÓN | En el presente componente formativo se conocerá el grupo de actividades enfocadas en encontrar los posibles fallos en el funcionamiento un software. |
| PALABRAS CLAVE | Calidad, diseño de prueba, herramientas de prueba, niveles de  prueba, software. |

| ÁREA OCUPACIONAL | 2 - CIENCIAS NATURALES, APLICADAS Y RELACIONADAS |
| --- | --- |
| IDIOMA | Español |

1. **TABLA DE CONTENIDOS:**

**Introducción**

**1. Generalidades de las pruebas de software**

**2. Tipos de pruebas y sus características**

**3. Los casos de prueba**

**4. Desarrollo guiado por pruebas**

1. **DESARROLLO DE CONTENIDOS:**

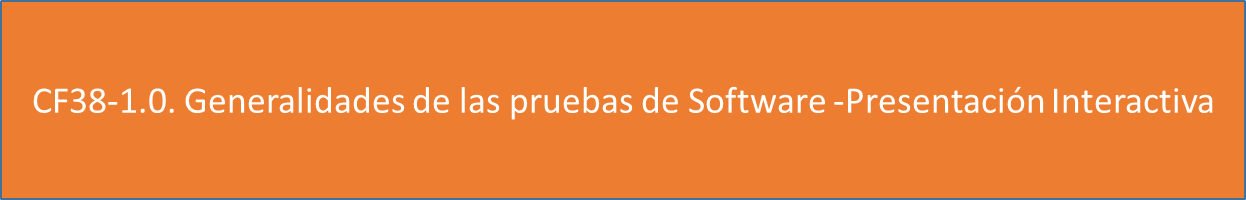
# **Introducción**

Las pruebas del software son un grupo de actividades enfocadas en encontrar los posibles fallos en el funcionamiento de todo software. Se invita a ver el siguiente video para conocer de qué tratará el componente formativo.



* + - 1. **Generalidades de las pruebas de software**

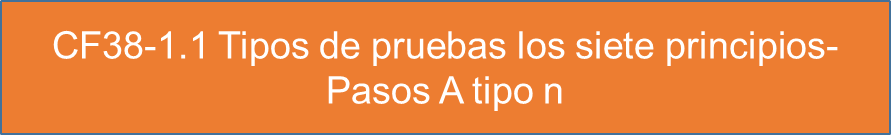
Se invita a navegar a través del siguiente recurso, el cual presentará las generalidades de las pruebas de software.



* + - 1. **Tipos de pruebas y sus características**

En los últimos años se han propuesto algunos principios con el fin de establecer pautas universales para que los programadores de software las adapten a sus procesos de pruebas.

Mera (2016), lista los siete principios:



Ahora, se pueden ver cuáles son los tipos de pruebas que se pueden realizar:

**Pruebas funcionales**

IBM (2021), plantea que estas pruebas se basan en la ejecución y revisión de las funciones y en su interoperabilidad con sistemas específicos, se llevan a cabo en todos los niveles de prueba. Buscan evaluar cada una de las opciones con las que cuenta el paquete informático, principalmente en el comportamiento externo del producto o aplicativo, también se las identifica como de tipo caja negra, ver figura 1.

**Figura 1**

*Pruebas funcionales*

**Pruebas unitarias**

Las pruebas unitarias o como también se las conocen “Unit testing”, son pequeños tests en los cuales se revisa que el comportamiento de un objeto y su lógica funcionen adecuadamente. Por lo general se realiza durante la fase de desarrollo de la aplicaciones o software, esta es ejecutada por los programadores, aunque también pueden realizarlas los aseguradores de calidad.

Las razones por las cuales se desarrollan este tipo de pruebas son:

* Son rápidas de realizar por lo que se pueden realizar en gran cantidad.
* Sirven como documentación de apoyo para el proyecto.
* Demuestran que la lógica del código funciona en todos los casos.
* Permite a los programadores a entender bien el código base, lo que permite realizar cambios oportunos.
* Se obtendrá un código de calidad.

En algunos casos, las pruebas pueden hacerse de manera manual pero lo mejor es usar herramientas que permitan ejecutar el servicio de la mejor manera posible. Existen muchas herramientas en el mercado y estas varían en función del lenguaje de programación que se esté utilizando; aquí se mencionarán algunas de las más conocidas:

Interfaz de usuario gráfica

Descripción generada automáticamente

Para realizar las pruebas unitarias se deben tener en cuenta los siguientes criterios:

* Se debe probar un sólo código a la vez.
* Hay que realizar pruebas frecuentemente mientras programas.
* Se debe corrige los bugs identificados en las pruebas antes de continuar.
* Cualquier cambio que ser realice también debe pasar la prueba.

**Pruebas de integración**

De acuerdo con Mera (2016), estas pruebas son las encargadas de verificar el óptimo acople entre los componentes y de probar las interfaces entre los módulos, como el sistema de archivos en integración con el hardware y el componente de validación de usuario con el sistema operativo (ver figura 2). Se debe tener en cuenta los objetos típicos de prueba, los cuales son:

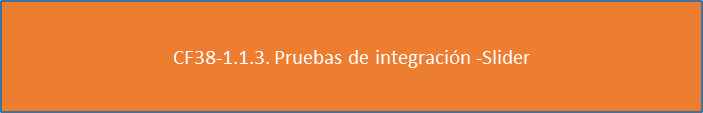
* Bases de datos de subsistemas.
* Datos de configuración.
* Configuración del sistema.
* Infraestructura.
* Interfaces.

**Figura 2**

*Pruebas de integración*

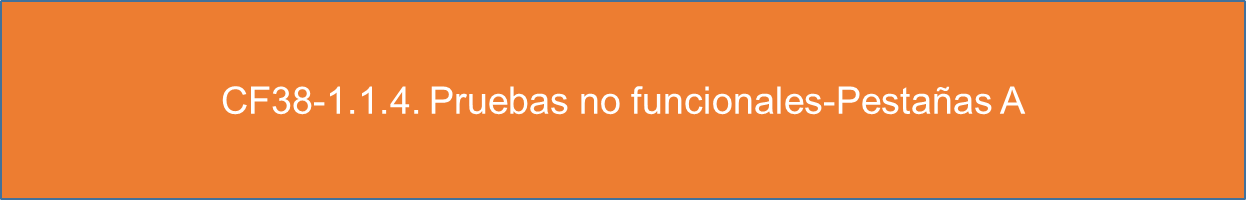


El mismo autor argumenta que en las pruebas de integración se revisan las interfaces entre componentes o subsistemas, para cerciorarse que son llamados cuando se necesite y que los datos o mensajes que se transmiten son los adecuados. Existen los siguientes tipos de pruebas de integración:



**Pruebas no funcionales**

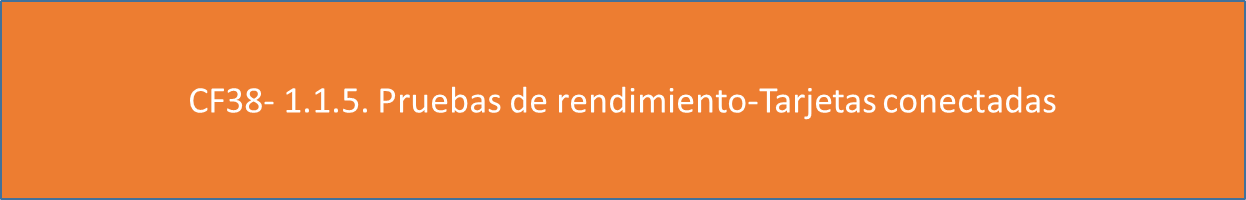
En este tipo de pruebas se comprueban los requisitos basados en la operación de un software, no en su funcionalidad. Estas pruebas pueden ayudar a establecer la carga que resiste el producto, si su rendimiento es el adecuado o si es estable a nivel de contacto con el servidor, en otras palabras, no dice si tienen un bajo desempeño o rendimiento en la producción. Aquí se encuentra:



**Pruebas de rendimiento**

Según IBM (2021), estas pruebas sirven para evaluar lo rápido que el programa realiza una tarea en condiciones específicas de trabajo, también evalúa la [escalabilidad](https://es.wikipedia.org/wiki/Escalabilidad), [fiabilidad](https://es.wikipedia.org/wiki/Fiabilidad_de_sistemas) y uso de los recursos.

Los objetivos de estas pruebas son:



**Ventajas y desventajas de los tipos de pruebas**

Todo proceso tiene sus ventajas y desventajas, a continuación se presentan cada una de ellas.

Ventajas de estos tipos de pruebas:

****

Desventajas de estos tipos de pruebas:

Por lo general, todas las pruebas de software traen beneficios al desarrollador, dependiendo del tipo de la prueba los beneficios pueden ser más altos y enfocados en un fin o propósito especifico, dando como resultado una mejor calidad de producto, mejorando su funcionalidad, reduciendo costos, tiempos, recursos humanos y mejorando la experiencia del usuario.

Se podría considerar como una desventaja, si se comparan los tipos de prueba ya que puede variar en:

* Tiempo.
* Costo.
* Esfuerzo.
* Recursos humanos.
* Exactitud de las pruebas.

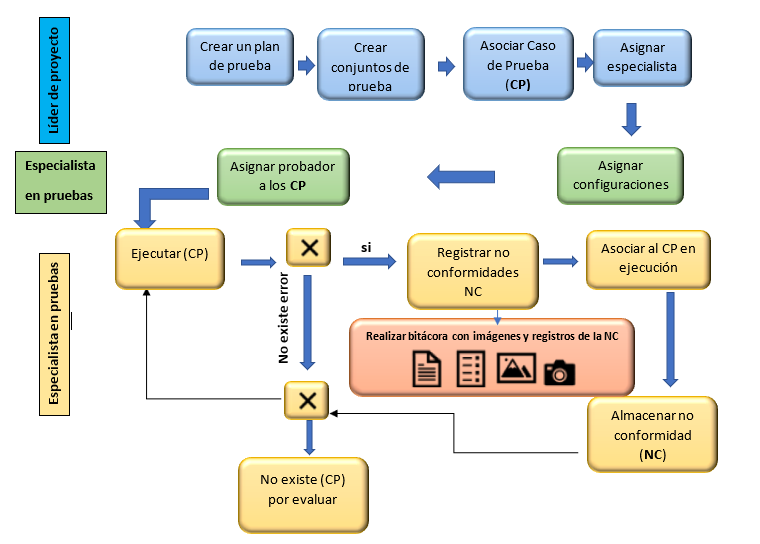
Sin embargo, la única desventaja de las pruebas de software son no hacerlas ya que el omitir este paso desencadena una nefasta serie de errores como un producto no funcional y colocarían en tela de juicio la profesionalidad de programador y su equipo de trabajo.

* + - 1. **Los casos de pruebas**

Son los documentos de los productos que se generan al momento de realizar una prueba de software, en el cual se plasma las precondiciones entradas y resultados esperados, incluyendo la implementación correcta, la identificación de errores, el chequeo de la calidad y otras informaciones consideradas importantes, en general no hay una plantilla especifica solo la bitácora, listas de chequeo, e imágenes en la cual se registran las no conformidades que se encuentran en las pruebas (ver figura 3).

**Figura 3**

*Esquema de casos de prueba*



En el diseño de las pruebas de software se deben identificar y describir casos de prueba, por lo tanto para documentar un caso de pruebas se debe comprender que este contiene un conjunto de variables o condiciones en la cual un tester determina si un sistema funciona según lo esperado; así mismo, cabe resaltar que la documentación de un caso de prueba se convierte en una actividad sin dificultad, si se tiene la información necesaria para su proceso de elaboración, teniendo en cuenta que al momento de probar un software es de mucho aporte, puesto que se transforma en una herramienta esencial en el proceso de registro, seguimiento y control.

A continuación, se nombran los elementos principales que debe contener un caso de prueba (ver tabla 1):

**Tabla 1**

*Elementos de un caso de prueba*

| Caso de prueba | |
| --- | --- |
| Identificador | Puede ser alfanumérico o numérico. |
| Nombre | Nombre del caso de prueba de manera concisa. |
| Descripción | Objetivo del caso de prueba, también describe que probara, en ciertas ocasiones se incluye el ambiente de pruebas. |
| Número de orden  Ejecución | Orden en la cual se ejecuta el caso de prueba, en la situación de que se tengan múltiples casos de prueba. |
| Requerimiento asociado | Si se plantea un caso de prueba se debe saber a qué requerimiento va asociado para mantener la trazabilidad |
| Precondición | Estado en la cual se debe encontrar el sistema antes de comenzar la prueba |
| Postcondición | El estado en que debe encontrarse el sistema luego de ejecutar la prueba |
| Resultado esperado | Objetivo que debe ser alcanzado posterior a ejecutarse la prueba. |

* + - 1. **Desarrollo guiado por pruebas**

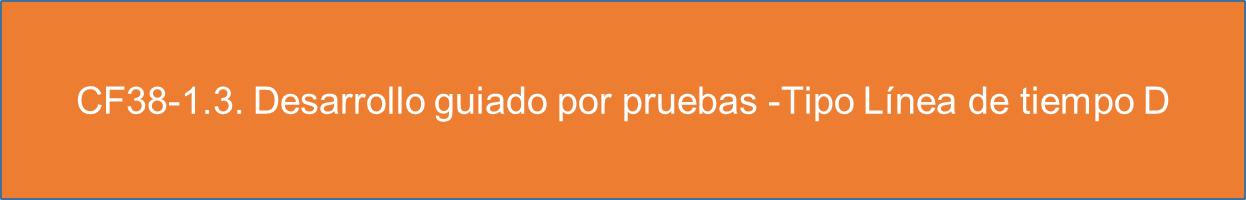
“El desarrollo de software dirigido por pruebas es una técnica de programación en la cual las pruebas que normalmente son las unitarias se escriben primero y luego se escribe el código que pase la pruebas, con esta práctica se consigue entre otras cosas: un código más robusto, más seguro, más mantenible y una mayor rapidez en el desarrollo”. (Herranz, 2021).

Cuando el TDD (Desarrollo dirigido por pruebas), se combina con el desarrollo Ágil, según Herranz (2021), tiene como consecuencia un espacio más amplio el cual cubre el desarrollo de software, hasta el diseño de software, el cual propone el proceso denominado:

**ROJO-VERDE-REFACTORY**

Según Sánchez (2017): “Para usar TDD el desarrollador de software tiene que cambiar su método de trabajo. Ahora, en vez de crear los casos de uso y después implementarlos, va a intentar convertir un caso de uso en un ejemplo. Si se realiza esto con todos los casos de uso, llegará un momento en el que los ejemplos describirán por sí solos la próxima tarea a implementar”.

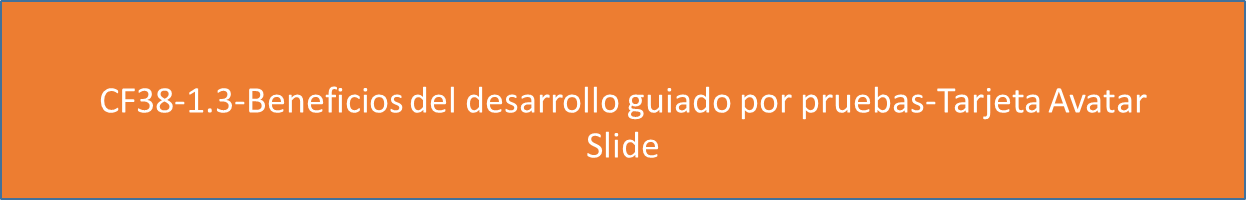
El proceso de diseño de software, combinando TDD con metodologías ágiles, sería el siguiente:



Es de comentar que este es un proceso de desarrollo relativamente nuevo, el cual está comenzando a ganar terreno y ser conocido en el ámbito del desarrollo de software en las organizaciones a nivel mundial, el cual se basa en evidencias anecdóticas y una serie de evaluaciones empíricas. Hablando del desarrollo tradicional las pruebas, se usan para validar y verificar la calidad del software y su creación se realiza después de que existe la funcionalidad.

**Nota:** en el TDD, las pruebas adicionales al tema de validación y verificación se usan con fines de especificar, donde primero se desarrolla la prueba y posteriormente la funcionalidad. Esto permite que la calidad del software aumente y la productividad del desarrollador mejore, dado que se lleva un mejor esfuerzo en depurar y se avanza rápidamente con el progreso del producto software.

A continuación, se puede observar los beneficios del desarrollo guiado por pruebas:

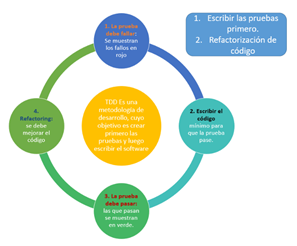


**Ciclo del desarrollo guiado por pruebas**

Ahora que ya se saben los beneficios que implica aplicar TDD, es posible ver el ciclo completo del proceso, revisando cada uno de sus pasos. En la siguiente figura 4 se puede observar el flujo de manera gráfica:

**Figura 4**

*Ciclo TDD*



Nota. https://ed.team/cursos/tdd

1. **Síntesis**

Diagrama, Escala de tiempo

Descripción generada automáticamente

1. **ACTIVIDADES DIDÁCTICAS (OPCIONALES SI SON SUGERIDAS)**

| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA | |
| --- | --- |
| Nombre de la Actividad | Conceptos básicos de Pruebas de Software 1 |
| Objetivo de la actividad | Afianzar algunos conceptos básicos de pruebas de software |
| Tipo de actividad sugerida | Arrastrar y soltar el término con la definición que corresponde |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | Anexo documento en Word llamado Actividad didáctica 1.docx  (Descargar para poder visualizar correctamente) |

| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA | |
| --- | --- |
| Nombre de la Actividad | Conceptos básicos de Pruebas de Software 2 |
| Objetivo de la actividad | Afianzar algunos conceptos básicos de pruebas de software |
| Tipo de actividad sugerida | Cuestionario de selección múltiple |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | Anexo documento en Word llamado Actividad didáctica 2.docx  (Descargar para poder visualizar correctamente) |

1. **MATERIAL COMPLEMENTARIO:**

| **Tema** | **Referencia APA del Material** | **Tipo de material**  **(Video, capítulo de libro, artículo, otro)** | **Enlace del Recurso o**  **Archivo del documento**  **o material** |
| --- | --- | --- | --- |
| Tipos de pruebas y sus características | IBM. (2021). *Pruebas Funcionales.* <https://www.ibm.com/docs/es/rtw/9.1.0?topic=SSBLQQ_9.1.0/com.ibm.rational.test.ft.doc/topics/Getting_Started_With_Ivory.html> | Artículo | <https://www.ibm.com/docs/es/rtw/9.1.0?topic=SSBLQQ_9.1.0/com.ibm.rational.test.ft.doc/topics/Getting_Started_With_Ivory.html> |
| IBM. (2021). *Pruebas de rendimiento.* <https://www.ibm.com/docs/es/rtw/9.0.0?topic=phases-performance-testing> | Artículo | <https://www.ibm.com/docs/es/rtw/9.0.0?topic=phases-performance-testing> |

1. **GLOSARIO:**

| TÉRMINO | SIGNIFICADO |
| --- | --- |
| Bottom-up | Pruebas ascendente**.** |
| QA (Quality Assurance) | Especialistas en pruebas de software verifican que un software no contenga fallos**.** |
| REFACTORY | Acción de limpiar o reconstruir el código de una aplicación. |
| Sprint | Ciclo de desarrollo en metodología de desarrollo scrum. |
| TDD | Desarrollo dirigido por pruebas. |
| Top-down | Pruebas descendentes. |

1. **REFERENCIAS BIBLIOGRÁFICAS:**

Herranz, J.I. (2021). *TDD como metodología de diseño de software*. <https://www.paradigmadigital.com/dev/tdd-como-metodologia-de-diseno-de-software/>

IBM (2021). *Pruebas de rendimiento*. <https://www.ibm.com/docs/es/rtw/9.0.0?topic=phases-performance-testing>

IBM (2021). *Pruebas Funcionales*. <https://www.ibm.com/docs/es/rtw/9.1.0?topic=SSBLQQ_9.1.0/com.ibm.rational.test.ft.doc/topics/Getting_Started_With_Ivory.html>

Mera Paz, J. A. (2016). *Análisis del proceso de pruebas de calidad de software*. Ingeniería solidaria, *12* (20).

Organización ISO. (2020). *Ingeniería de sistemas y software - Requisitos de calidad y evaluación de sistemas y software (SQuaRE) - Guía de SQuaRE.* <https://www.iso.org/standard/64764.html>

Sánchez, A.F (2017). *Agile Testing*. *Estado del arte. Su aplicación en empresas TIC de Extremadura*. Universidad De Extremadura.

Sanz, L. F. (2005). *Un sondeo sobre la práctica actual de pruebas de software en España*. REICIS. Revista Española de Innovación, Calidad e Ingeniería del Software,1(2), 43-54.

1. **CONTROL DEL DOCUMENTO**

|  | **Nombre** | **Cargo** | **Dependencia** | **Fecha** |
| --- | --- | --- | --- | --- |
| **Autor (es)** | Mario Fernando Meneses Calvache | Experto Temático | Regional Cauca, Centro de teleinformática y producción industrial | Diciembre 2021 |
| Carlos Hernán Muñoz | Experto Temático | Regional Cauca, Centro de teleinformática y producción industrial | Diciembre 2021 |
| Zvi Daniel Grosman | Diseñador Instruccional | Regional Tolima, Centro Agropecuario “La Granja” | Diciembre 2021 |
| Ana Catalina Córdoba Sus | Revisora Metodológica y Pedagógica | Regional Distrito Capital – Centro para la Industria de la Comunicación Gráfica. | Diciembre 2021 |
| Rafael Neftalí Lizcano Reyes | Asesor pedagógico | Regional Santander - Centro Industrial del Diseño y la Manufactura. | Diciembre 2021 |
|  | Jhon Jairo Rodríguez Pérez | Diseñador y evaluador instruccional | Regional Distrito Capital - Centro para la Industria de la Comunicación Gráfica | Enero de 2022 |

1. **CONTROL DE CAMBIOS**

**(Diligenciar únicamente si realiza ajustes a la Unidad Temática)**

|  | Nombre | Cargo | Dependencia | Fecha | Razón del Cambio |
| --- | --- | --- | --- | --- | --- |
| Autor (es) |  |  |  |  |  |