



## TALLER DE TESTING Y CALIDAD DE SOFTWARE

*U2. Desarrollo y ejecución de casos de prueba.  
Semana 4*





## **ESCUELA DE CONSTRUCCIÓN E INGENIERIA**

**Director:** Marcelo Lucero

### **ELABORACIÓN**

**Experto disciplinar:** Aída Villamar Gallardo.

**Diseño instruccional:** Carla Silva Alvarado.

**Editor instruccional:** David Villagrán Ruz.

### **VALIDACIÓN**

**Experto disciplinar:** Andrés del Alcázar

**Jefa de Diseño Instruccional:** Alejandra San Juan Reyes.

### **EQUIPO DE DESARROLLO**

AIEP

**AÑO**

2021

---



## Tabla de contenidos

<b>Aprendizaje esperado .....</b>	<b>4</b>
<b>Introducción.....</b>	<b>4</b>
<b>1. Identificación del concepto de prueba de sistema, considerando su utilidad en el proceso de desarrollo de software. ....</b>	<b>5</b>
<b>1.1. Concepto de pruebas de sistema .....</b>	<b>5</b>
1.1.1. Visión sistémica de la prueba.....	6
<b>2. Caracterización de tipos de pruebas de sistema en el proceso de desarrollo de software, según requerimientos de la industria. ....</b>	<b>11</b>
<b>2.1. Tipos de pruebas.....</b>	<b>11</b>
2.1.1. Pruebas estáticas.....	11
2.1.2. Pruebas dinámicas .....	14
2.1.3. Pruebas manuales .....	16
2.1.4. Pruebas automáticas .....	17
2.1.5. Pruebas funcionales y no funcionales .....	19
2.1.6. Pruebas de escritorio .....	23
<b>3. Herramientas disponibles en el mercado para la realización de pruebas de software en proyectos.....</b>	<b>26</b>
<b>3.1. Herramientas para realización de pruebas de sistema existentes en el mercado. ....</b>	<b>26</b>
3.1.1. Herramientas open source comerciales y personalizadas .....	26
<b>3.2. Función de las pruebas de sistema .....</b>	<b>29</b>
<b>4. Determina la función de pruebas de sistema en el proceso de desarrollo de software, considerando crecimiento de uso de software, según estándares del mercado.....</b>	<b>30</b>
<b>4.1. Crecimiento de uso de un software .....</b>	<b>30</b>
<b>Conclusiones.....</b>	<b>32</b>
<b>Referencias bibliográficas .....</b>	<b>33</b>

---



## Aprendizaje esperado

Determinan función de pruebas de sistema en el proceso de desarrollo de software, considerando tipos de pruebas y herramientas asociadas, según estándares del mercado.

## Introducción

- ¿Qué son las pruebas funcionales manuales?
- ¿Qué defectos encontramos generalmente al realizar pruebas manuales?
- ¿Cuándo y por qué conviene automatizar pruebas de software?

Hasta ahora hemos estudiado los conceptos, de manera general, asociados al testeo, junto con el impacto que tienen en nuestros desarrollos.

Si analizamos las preguntas anteriores, podemos vislumbrar que hay más de un tipo de pruebas, pero, además, encontramos técnicas y herramientas asociadas. Todo lo anterior, veremos que está directamente relacionado a la etapa en que nos encontremos, a las características de nuestro proyecto y al objetivo de las pruebas que vamos a aplicar.

---

# 1. Identificación del concepto de prueba de sistema, considerando su utilidad en el proceso de desarrollo de software.

## 1.1. Concepto de pruebas de sistema

El concepto de 'pruebas de sistema' está asociado al estudio del producto completo, ya sea que se haya desarrollado o comprado. En ambos casos, podemos verlo como un sistema que debe ser probado, por ejemplo, como una forma de decidir con respecto a su aceptación, estudiar defectos globales, o bien, para analizar aspectos específicos de su comportamiento, como son la seguridad o el rendimiento.

Generalmente, la persona que prueba el sistema no tiene acceso al código fuente, por lo que comúnmente las pruebas son de caja negra.



**Figura 1.** Prueba de caja negra

**Fuente:** Nebulargroup.com (2020)

Revisaremos el proceso de prueba de sistema, considerando la relación que existe con el proceso de toma de requerimientos y el proceso restante de desarrollo de software.

---

### 1.1.1. Visión sistémica de la prueba

De acuerdo a Peña (2006),

Cuando se deben realizar pruebas, debe mantenerse un enfoque sistémico, es decir integral, que está detrás de todo desarrollo de software. Al hablar de enfoque sistémico se indica que:

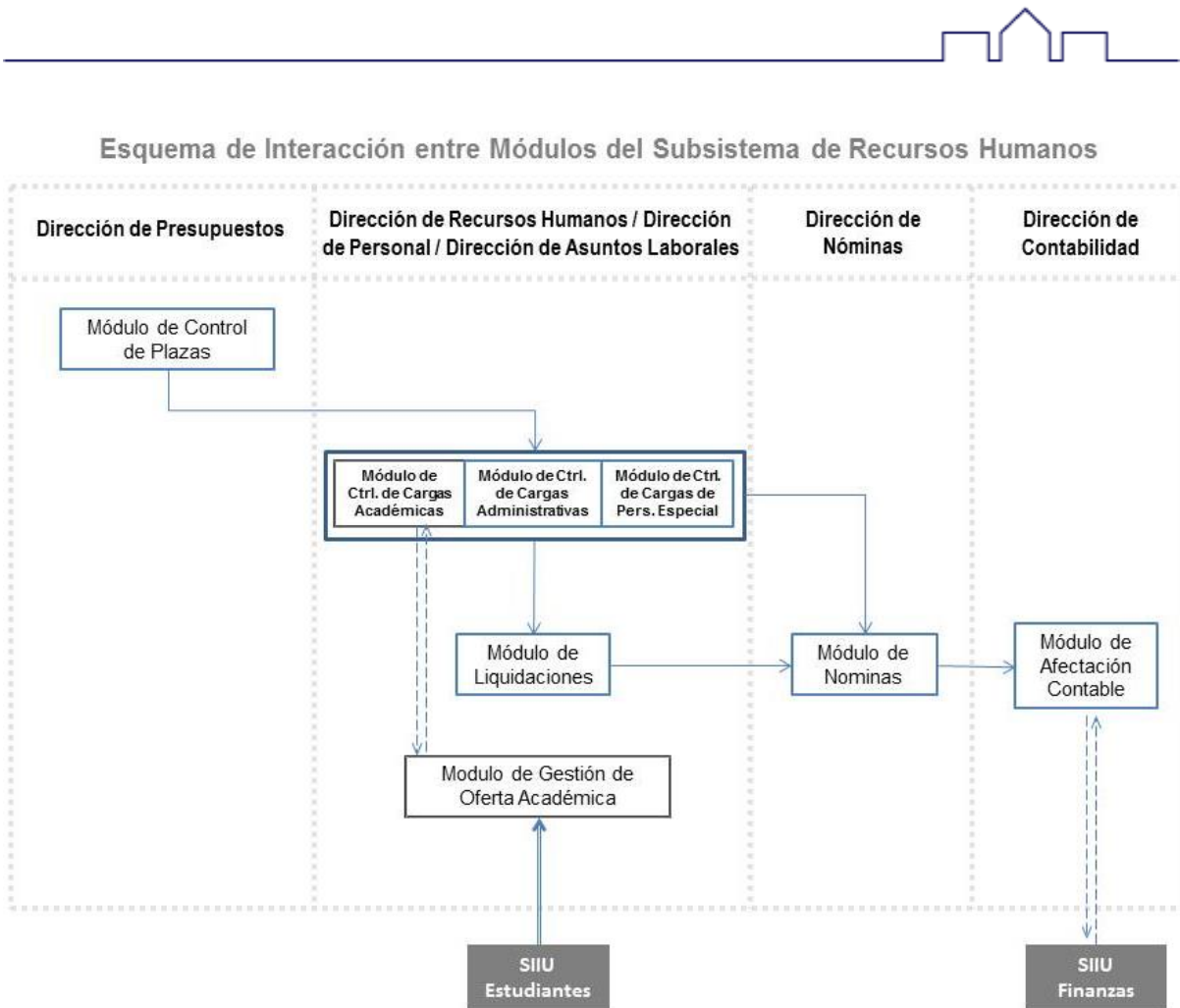
- a) Todo sistema tiene una serie de objetivos que le dan sentido. Esos objetivos están asociados con indicadores de éxito que permiten determinar si los objetivos se cumplen y en qué medida.
- b) Todo sistema tiene una serie de elementos que lo forman y la interacción de tales elementos se orienta a satisfacer los objetivos.
- c) Todo sistema tiene una frontera que lo separa de un medioambiente. Los elementos de ese medioambiente influyen sobre el sistema, proporcionándoles una serie de entradas y obteniendo del mismo un conjunto de salidas.



**Figura 2.** Limite de los sistemas

**Fuente:** Sites.google.com. (s. f.)

- d) Ningún sistema existe en aislamiento; siempre interaccionan con otros sistemas constituyendo un sistema mayor. (p. 1)



**Figura 3.** Ejemplo esquema de interacción módulos

**Fuente:** Universidad Veracruzana (s. f.)

De acuerdo a Sumano *et al.* (2010),

Al aplicar esos conceptos a la prueba de software, se obtiene una serie de principios que servirán de base para la prueba:

**a.** Debe asegurarse de conocer con precisión los objetivos del software a probar, así como sus indicadores de éxito. Estos elementos se localizan en los documentos obtenidos en la etapa de recolección de requerimientos, así como en las especificaciones del software. Esta información será



indispensable para preparar el plan de pruebas y será la base para iniciar el desarrollo de los casos de prueba.

**b.** Deben determinarse las entradas y salidas del sistema a probar. Este aspecto es necesario en la preparación de los casos de prueba y también en el establecimiento de procedimientos de prueba, orientados especialmente a los casos de prueba que muestran el cumplimiento de los objetivos.

**c.** Considerar el sistema mayor donde opera el software a probar. Generalmente es un ambiente organizacional, formado por elementos de hardware, de software y personas (usuarios). Todos estos elementos influyen mucho sobre el sistema y ayudan especialmente en la preparación de casos de prueba de situaciones no deseadas, relacionadas con datos inadecuados, ausencia de elementos necesarios y ocurrencia de excepciones.

Para un sistema, el proceso de prueba se compone de **dos etapas**, las que pudiesen estar bastante alejadas en el tiempo una de otra, ellas son:

- La primera etapa que consiste en la **preparación de las pruebas**, muy relacionada con la obtención de los requerimientos, en general, se realiza en las primeras etapas del proyecto.
- La segunda etapa corresponde a **la aplicación de las pruebas**, en la que se necesita que el sistema se encuentre “completo o al menos una integración (como se designa a un producto parcial) aún no liberado, para poder ejecutar las pruebas, por lo mismo se realiza en etapas avanzadas del proyecto” (Sumano *et al.*, 2010).





La situación exacta de estas etapas va a depender del modelo de ciclo de vida que se esté utilizando.

La primera etapa **preparación de pruebas** considera por lo menos tres actividades, que incluyen la preparación de:

- Un plan de pruebas
- Una lista de verificaciones de los requerimientos
- Casos de prueba

Para realizar las actividades dos y tres, se necesita disponer del documento de requerimientos.

La segunda etapa **aplicación de pruebas**, de acuerdo a Sumano *et al.* (2010), necesita el plan de pruebas y una versión del sistema que sea ejecutable o una integración, sobre la que se realizarán los casos de prueba preparados anteriormente, se estudiarán los resultados y se identificarán posibles defectos.

### ¿Qué vamos a obtener?

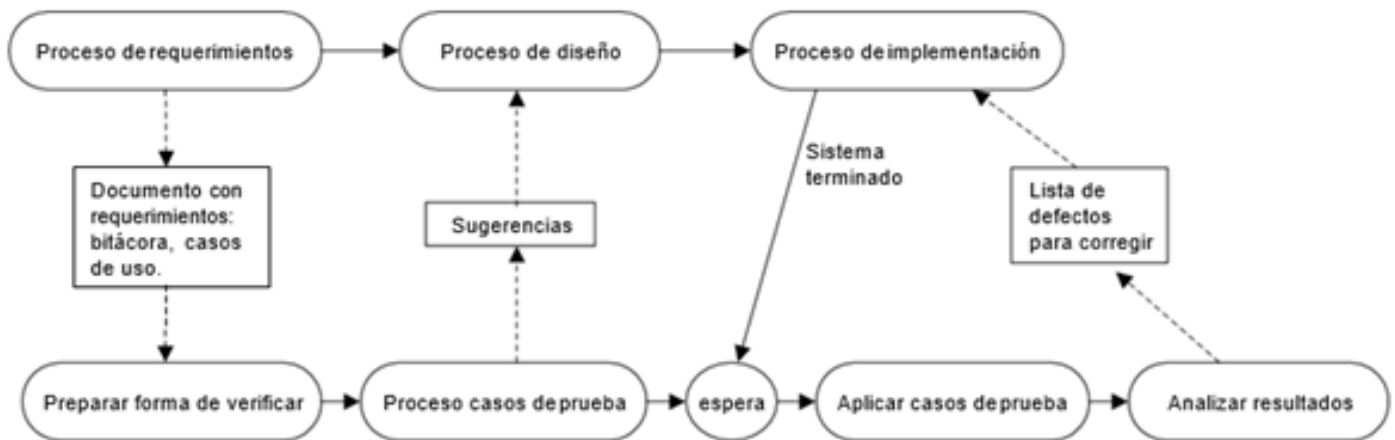
**Etapas 1:** nos entrega retroalimentación para el análisis de los requerimientos, se identifican falencias, ambigüedades y otros inconvenientes. Por otro lado, si se está comenzando con el desarrollo, entrega sugerencias valiosas para el diseño y la implementación del sistema.

**Etapas 2:** nos entrega retroalimentación para la implementación y el diseño, dejando al descubierto posibles defectos que deben ser rectificados. Además, entrega información útil para las etapas de

---

liberación del software, aceptación, estimación de confiabilidad y posterior mantenimiento.

En la Figura 7 se muestra el proceso de prueba de sistema y su relación con otros procesos.



**Figura 4.** Proceso de prueba de sistema

**Fuente:** Peña (2006, p. 3)

De acuerdo a Sumano *et al.* (2010),

La prueba de sistemas tiene varias suposiciones importantes:

- a) Cada unidad que forma el sistema ha sido probada por separado y se han eliminado sus defectos.
- b) Las interfaces humano-computadoras (de texto o gráficas) han sido probadas también por separado.
- c) Se han realizado pruebas de integración para analizar la interacción entre partes del sistema y se han eliminado los defectos identificados.



El segundo punto es importante, ya que algunas veces se confunde la prueba de sistema con la prueba de la interfaz.

La primera verifica la interacción de todas las partes, mientras que la segunda únicamente analiza los elementos de la interfaz y posiblemente el manejo de eventos asociados. Sin embargo, las herramientas que ayudan a la prueba de interfaz pueden utilizarse para iniciar las pruebas de sistema.

## **2. Caracterización de tipos de pruebas de sistema en el proceso de desarrollo de software, según requerimientos de la industria.**

### **2.1. Tipos de pruebas**

#### **2.1.1. Pruebas estáticas**

Las pruebas estáticas, o revisiones, corresponden a un **tipo de técnica** en que se realizan sin ejecutar el código de la aplicación, ya que son a nivel de especificaciones. Sí se efectúa un análisis del código, pero es de carácter estático.

Se refieren a la revisión de los documentos del proyecto, por ejemplo, las especificaciones de diseño, los requerimientos, diagramas de casos de uso y toda la documentación asociada, ya que como se mencionó no se realiza ejecución de código.

#### **a. Principios básicos**



El análisis estático es un proceso formal y estructurado, con un sistema de listas de verificación y roles definidos que tiene como finalidad detectar defectos, ya sea en los modelos o en el código fuente del software.

El foco de la revisión está en identificar defectos de manera temprana, no resolverlos. Estas revisiones permiten identificar defectos que con las pruebas dinámicas no resultaría tan sencillo.

Por medio de este proceso de revisión se obtienen datos que deben ser usados para control de la calidad del producto, monitoreo y mejoras del proceso de revisión.

De acuerdo a Sánchez (2015), los defectos típicos que se encuentran en las revisiones son.

- Referencias a una variable con un valor definido.
- Interfaces inconsistentes entre módulos.
- Variables que no se utilizan o que se declaran de forma incorrecta.
- Código inaccesible y ausencia de lógica.
- Infracciones de los estándares así como de la sintaxis de código y modelos de software. (p. 29)

Existen varios tipos de revisiones:

- **Informal:** en este tipo de revisiones, no hay documentado un procedimiento formal, ni se llevan por escrito los resultados, se detectan defectos con un menor costo y, por lo general, lo realiza el líder técnico de diseño y codificación.
- **Guiada:** en este caso encontramos que se pueden llevar a cabo de una manera informal, o bien pueden llegar a ser muy formales.



Se detectan defectos y se establece un entendimiento general, y lo realiza el autor de un documento del proyecto.


- **Técnica:** se pueden llevar a cabo de una manera informal o bien pueden llegar a ser muy formales, se debe realizar un informe, el objetivo es deliberar, evaluar, resolver inconvenientes, buscar acuerdos y tomar decisiones.
- **Inspección:** es un tipo de revisión más formal que considera un compendio de métricas y la realización de un informe con las conclusiones. Su finalidad es la detección de defectos que se realiza por medio de una inspección visual, por ejemplo, no cumplimiento de estándares de desarrollo.

Además, los **tipos de personas involucradas** en las revisiones, de acuerdo a Sánchez (2015), son:

- Revisor: persona involucrada en la revisión, identificando las posibles anomalías del producto o proceso bajo revisión.
- Escriba: persona que registrará en un acta cada defecto y sugerencia para la mejora durante revisión.
- Moderador: principal persona responsable de una inspección u otro proceso de revisión, que mediará entre los distintos puntos de vista. (p. 30).

Fases de las pruebas estáticas:

- Planificación
- Orientación inicial
- Preparación individual

- 
- 
- Reunión de revisión
  - Seguimiento
  - Evaluación

En las reuniones de revisión se pueden presentar tres casos posibles:

- I. La revisión es cerrada ya que se ha determinado que el producto no tiene defectos.
- II. Se debe continuar con las etapas de seguimiento y evaluación ya que se han detectado defectos menores en el producto.
- III. Es necesario realizar una corrección de los defectos y comenzar un nuevo proceso de revisión ya que se han detectado defectos graves en el producto.

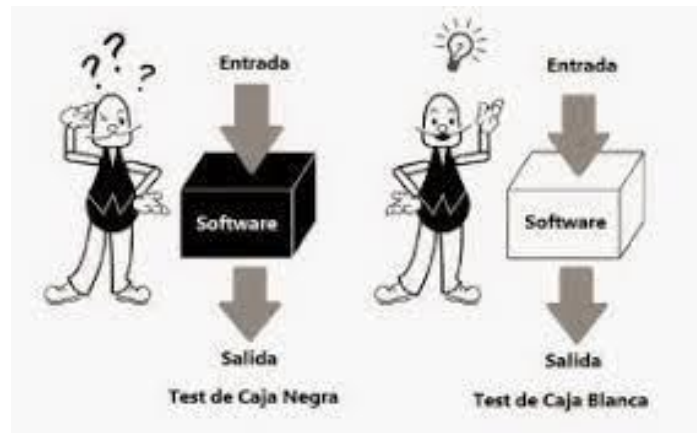
### 2.1.2. Pruebas dinámicas

Las pruebas dinámicas dicen relación al **tipo de técnica** empleada; son todas aquellas que, para su realización, necesitan la ejecución de la aplicación. Las pruebas dinámicas permiten el uso de técnicas de caja negra y caja blanca.

Debido a la naturaleza dinámica de la ejecución de pruebas, es posible medir con mayor precisión el comportamiento de la aplicación desarrollada. Por una parte, se va probando la interacción entre los distintos módulos, pero con una mirada interna gracias a las pruebas de caja blanca; por otro lado, se realizan pruebas en busca de errores de las funciones específicas, por medio de una mirada externa, realizando pruebas de caja negra.

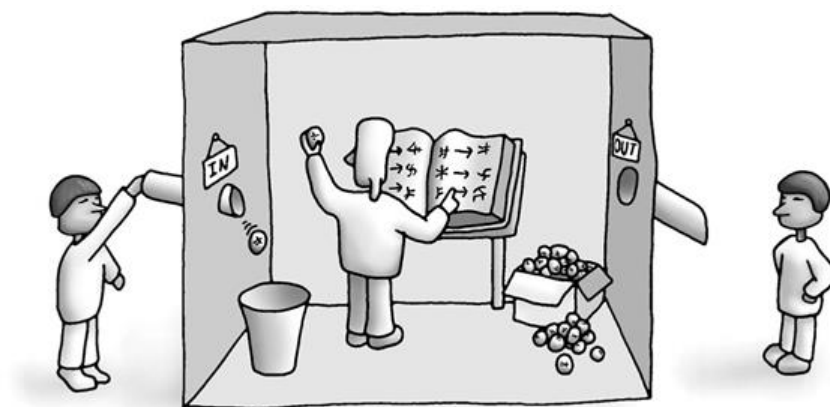
---

Por medio de ambas es que podemos definir nuestros casos de prueba, con la finalidad de tener la mayor cantidad de posibilidades en la detección de errores.



**Figura 5.** Conceptos: caja negra & caja blanca

**Fuente:** Empresas.blogthinkbig.com. (2014)



**Figura 6.** Caja blanca

**Fuente:** Panel.es. (2014)



**Figura 7.** Caja negra

**Fuente:** Commons.wikimedia.org. (2014)

Algunos ejemplos de pruebas dinámicas:

- Pruebas de aceptación
- Pruebas de regresión
- Pruebas de integración

### 2.1.3. Pruebas manuales

De acuerdo al sitio web Tecnologías-información.com (s. f.),

Testing manual o prueba manual corresponde a un tipo de prueba de software en la que uno o más testers ejecutan de forma manual los casos de prueba sin utilizar ningún tipo de herramienta de automatización.

La prueba manual es la más primitiva de todos los tipos de prueba y ayuda a encontrar errores en el sistema de software. Cualquier aplicación nueva debe probarse manualmente antes de que esta pueda automatizarse. La prueba manual requiere más esfuerzo, pero es necesaria para verificar la viabilidad de la automatización.

Dicho de una manera más simple, la prueba manual significa probar una aplicación manualmente por un humano que simula las acciones del usuario final, para garantizar que un sitio web o una aplicación funcionen correctamente, según las diversas condiciones que se escriben en los casos de prueba.





El objetivo de un probador de software es verificar el diseño, la funcionalidad y el rendimiento de la interfaz de usuario del sitio web, haciendo clic en varios elementos; pero también, es intentar corromperlo para ver si existe algún punto ciego que se pueda considerar como una vulnerabilidad.

### **Tipos y objetivos del testing manual**

Los siguientes son los objetivos de los testing manuales:

- Garantizar que la aplicación esté libre de errores y que funcione de conformidad con los requisitos funcionales especificados.
- Garantizar que los conjuntos de pruebas o casos estén diseñados durante la fase de prueba y que tengan una cobertura de prueba del 100%.
- Hay que asegurar que los defectos informados sean reparados por los desarrolladores, y que los probadores hayan realizado una nueva prueba en los defectos corregidos.
- Finalmente verificar la calidad del sistema y la entrega de un producto libre de errores al cliente.

#### **2.1.4. Pruebas automáticas**

Las pruebas automáticas son aquellas en las que un experto en pruebas genera casos de pruebas que, posteriormente, se automatizan por medio de scripts, con la finalidad de que se reproduzcan en las ejecuciones utilizando una herramienta para que, luego, la misma se realice automáticamente.

Estas pruebas pueden ejecutarse una y otra vez después de ser creadas. Son más rápidas que las pruebas manuales, dado que su



objetivo es detectar fallas en el software evitando que alguien tenga que ejecutar las pruebas de forma manual, comprobando que nada de lo probado con anterioridad ha dejado de funcionar como debe.

No requiere la intervención de alguna persona en cada nueva ejecución, ya que la prueba simula la interacción humana con el software.

Automatizar tiene muchas ventajas en el desarrollo y liberación de software. Dentro de los más relevantes se encuentran:

- Mayor capacidad para la ejecución de pruebas.
- Integración continua y Devops, que es una importante tendencia en la construcción de software moderno.
- Ahorro sustancial de tiempo.
- Mejora la productividad y la satisfacción laboral de los recursos.
- Pruebas de software repetibles.
- Mayor precisión para la corrección de fallas de software.

Un ejemplo de prueba automatizada sería grabar una navegación y, luego, ejecutar esa prueba de forma automática desde la herramienta que se está utilizando.

Al utilizar las pruebas automatizadas vamos a obtener un nivel de confiabilidad más alto cuando se emplean con posterioridad a alguna variación de hardware o software, en el que el sistema es ejecutado.

De acuerdo a Interware (2018), las pruebas automatizadas

Son la opción más viable cuando se necesita realizar diversos test cases de manera repetitiva y por un amplio período de tiempo. Son ideales para

---



ampliaciones de funciones, para garantizar que lo que no se ha modificado siga funcionando.

Se puede concluir que las pruebas manuales y las pruebas automatizadas son complementarias: las pruebas automatizadas se deben aplicar cuando un sistema se comporte de forma estable, o sea, que haya pasado por el proceso de pruebas manuales.

### **Objetivos de la automatización de casos**

Lo principal es buscar siempre un mayor nivel de calidad del software y evaluar si la automatización se ajusta al proyecto. Para encontrar la respuesta a esta pregunta, es recomendable analizar su viabilidad con relación a los objetivos.

Consideraremos algunos casos en los que posiblemente tenga sentido la automatización:

- Existe un problema de carácter técnico que solucionar.
- Realizar las pruebas de regresión demanda un tiempo considerable.
- Se está frente a un proyecto que posee un nivel de complejidad muy alto y es de largo plazo.

#### **2.1.5. Pruebas funcionales y no funcionales**

Las pruebas funcionales corresponden a un **tipo de prueba**, como son también las pruebas no funcionales y las pruebas estructurales.



En el caso de las **pruebas funcionales**, son aquellas que se basan en las funcionalidades del sistema que han sido definidas en la especificación de requerimientos, es decir, **se enfocan en lo que el sistema hace**.

Para la interpretación de estas pruebas se requiere contar con un nivel de experiencia alto y pueden llevar asociado documentación.

De acuerdo a Sánchez (2015),

Las características de funcionalidad las establece la ISO 25010 son idoneidad, exactitud, interoperabilidad y seguridad. En la ISO 25010:2015, indican que “la funcionalidad representa la capacidad del producto de software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones específicas” (p. 25).

La funcionalidad a su vez, siguiendo a Sánchez (2015), la dividen en las siguientes características:

**Compleitud funcional:** el grado en que las funcionalidades cubren todas las tareas y objetivos del usuario especificados.

**Corrección funcional:** capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido.

**Pertenencia funcional:** capacidad del producto de software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuarios especificados.

Las pruebas funcionales suelen estar asociadas a las técnicas de diseño de prueba de caja negra, ya que tienen en cuenta el comportamiento externo del software. (pp. 25-26).

Según lo que verifican, existen varios tipos:



- **Las pruebas funcionales**, existen varios tipos, dentro de las más importantes están las pruebas unitarias, de integración, de aceptación y de regresión (Lucerna, 2019).
- **Las pruebas exploratorias**: son pruebas en las que se obtiene un aprendizaje y conocimiento de la aplicación a probar. Ayudan a la integración de la fase de pruebas de una forma mucho más rápida. Útiles a la hora de probar aplicaciones ya desarrolladas, es decir, aquellas pruebas de software que no comienzan a la vez que el desarrollo. Para realizar las pruebas funcionales exploratorias se identificarán los distintos procesos de negocio o módulos de la aplicación y se le dará al tester libertad, poniéndose en la piel de un usuario, para probarlos. Estas pruebas exploratorias deberán ejecutarse sobre la última versión cerrada disponible de la aplicación.
- **Las pruebas unitarias** son aquellas que toman partes de código concretos para ver su funcionamiento y verificar que no tiene errores. (Lucerna, 2019).
- **Las pruebas de integración** se aplican a todos los componentes, para verificar cómo interactúan entre ellos y comprobar que cumplan con lo esperado. (Lucerna, 2019).
- **Las pruebas de aceptación** son aquellas que normalmente las realiza el usuario, lo que se busca con ellas es asegurarse que el sistema hace lo que el usuario espera, ya que es posible que un software no posea errores, que funcione bien, pero que no haga lo que debería hacer, es decir, no está haciendo lo que el usuario esperaba del sistema. (Lucerna, 2019).
- **Las pruebas de regresión** se realizan cuando se han efectuado cambios en una parte del código, habrá otras partes que no se han modificado, lo que se desea saber es que si al realizar los cambios se ha introducido algún tipo de defecto en alguna otra parte que ya funcionaba. Este tipo



de pruebas suele ser muy común que se automaticen. Se verifica que el problema se ha solucionado y que con el cambio no hemos afectado el resto del sistema. (Lucerna, 2019).

- **Las pruebas no funcionales**, son aquellas que no se basan en aspecto funcional, consideran el comportamiento externo del sistema, es decir, se enfocan en cómo funciona el sistema, lo que al igual que con las pruebas funcionales, debe estar definido en los requerimientos. Para la realización de estas pruebas se tienden a utilizar técnicas de diseño de caja negra.

Además, hay varios tipos de pruebas no funcionales que podemos ver a continuación, siguiendo a Lucerna (2019):

- **De seguridad**, el famoso hacking ético, en las que se buscan vulnerabilidades de seguridad.
- **De rendimiento**, que permiten conocer el comportamiento del software ante una carga determinada, cómo responde y cómo se recupera ante fallos.
- **De usabilidad**, que se emplean para saber cómo de usable es la aplicación, pero sin entrar en aspectos funcionales. Por ejemplo, si tiene un menú que hace que la navegación sea intuitiva, si tiene una ayuda que explica el funcionamiento de la aplicación, etc.
- **De accesibilidad**, que van más orientadas a que se visualicen bien los elementos, a que si hay sonido tenga también alguna alternativa para personas con alguna discapacidad visual o auditiva, etc.

También, la ISO 25010:2015 establece las siguientes características: fiabilidad, portabilidad, facilidad de uso, eficiencia, adecuación, compatibilidad, seguridad, mantenibilidad.



Algunas características no funcionales que se han de tener en cuenta en las pruebas, de acuerdo a Sánchez (2015):

- **Pruebas de carga:** consisten en la medición del comportamiento del sistema para aumentar la carga de este, ya sea mediante el número de peticiones que se realizan a una web al mismo tiempo, el número de usuarios que trabajan simultáneamente, etc.
- **Pruebas de rendimiento:** en estas pruebas se medirán la velocidad de procesamiento y el tiempo de respuesta del sistema.
- **Pruebas de volumen:** se mide la capacidad del sistema para procesar gran cantidad de datos, como procesar archivos con tamaños muy grandes.
- **Pruebas de esfuerzo:** se realizan pruebas donde se sobrecarga el sistema y se analiza la capacidad de recuperación.
- **Pruebas de seguridad:** se realizan diferentes pruebas de accesos no autorizados, ataque de denegación de servicio, etc.
- **Pruebas de estabilidad, eficiencia, robustez:** se realiza una medición de la respuesta del sistema a los errores de funcionamiento. (pp. 26-27).

#### 2.1.6. Pruebas de escritorio

Las pruebas de escritorio consisten en asignar valores a las variables que se han definido y se va siguiendo el flujo del programa para comprobar si el resultado es el esperado, este proceso se realiza de forma manual.

Cuando diseñamos las pruebas, debemos siempre considerar valores esperados y correctos, pero, además, valores inesperados e



incorrectos, de esa manera podremos ver cómo responde el sistema ante distintos escenarios.

Se consideran resultados previstos para datos desconocidos, a fin de que al probar cada una de sus partes podamos ir comprobando que el algoritmo sirve o requiere modificarse.

Es la etapa más importante en el desarrollo de un programa, por cuanto el realizar la prueba de escritorio nos permite saber:

1. Si el programa hace lo que debería hacer
2. En el caso de que no lo haga, permitirá detectar errores como:
  - Si algún paso o instrucción no está en el orden correcto.
  - Si falta algo.
  - Si algo esta demás.
  - Si los pasos o instrucciones que se repiten lo hacen más o menos veces de lo esperado.
  - Si las instrucciones están en un orden apropiado.
  - Otros errores que pueden presentarse. Por ejemplo:
    - Verificar cuando se ingresa la fecha de nacimiento que no sea posterior a la fecha actual.
    - Si se ingresa un número negativo pero el valor esperado es un numero positivo o entero.
    - Ingresar números cuando se espera solo texto.
    - Ingresar valores que se encuentran fuera del rango definido.
3. Elegir los datos apropiados para la prueba.





La prueba de escritorio no es más que efectuar un proceso de simulación con el algoritmo desarrollado (ver que haría el computador). Este trabajo se realiza en base a una tabla cuyos encabezados son las variables que se usan en el algoritmo y debajo de cada una de ellas se van colocando los valores que van tomando, paso a paso y siguiendo el flujo indicado por el algoritmo, hasta llegar al final.

La prueba comprende dos etapas:

- La primera: probar inicialmente que el programa funcione correctamente, para lo que se eligen algunos datos fáciles de probar.
- La segunda: si se prueba que ya funciona, se buscarán otros datos (si los hay) que hagan que falle el algoritmo, en cuyo caso se habrán de detectar otros errores. Si el algoritmo no falla, podemos concluir que el programa está terminado y revisado, por lo tanto, correcto.



### 3. Herramientas disponibles en el mercado para la realización de pruebas de software en proyectos.

#### 3.1. Herramientas para realización de pruebas de sistema existentes en el mercado.

Ya hemos visto que las pruebas automatizadas consideran el uso de herramientas que nos permiten realizarlas. En este momento, ya se sabe qué se va a automatizar, y es necesario decidir qué herramienta se utilizará, algo que puede ser bastante complejo de analizar inicialmente, ya que encontramos muchas herramientas, con diversas características, disponibles en el mercado.

¿Qué debemos considerar para la elección? El proyecto, presupuesto, conocimiento, y experiencia de quienes participan.

##### 3.1.1. Herramientas open source comerciales y personalizadas

De acuerdo a Rodríguez (s. f.),

Las herramientas que tenemos a nuestro alcance varían en sus limitaciones y posibilidades. Por lo que, para seleccionar la herramienta correcta para automatizar una prueba, hay que tener claridad de los requisitos que deben cumplirse para continuar el análisis de costo-beneficio de su uso.

**Selenium:** Selenium es una herramienta de código abierto, ampliamente aceptada en todo el mundo para probar aplicaciones web en diferentes navegadores y plataformas.



**Figura 8.** Selenium

**Fuente:** Eistria.b-cdn.net. (2016)

**Appium:** Appium es otra herramienta open source basada en Selenium que se puede emplear para automatizar las pruebas, principalmente en dispositivos móviles para iOS y Android.



**Figura 9.** Appium

**Fuente:** Medium.com (2019)

**Cucumber:** Cucumber es parte del enfoque BDD (Behavior Driven Development) y su principal ventaja es la facilidad de uso, ya que es muy intuitiva. También proporciona una amplia variedad de características y similar a las anteriores, es una herramienta open source.



**Figura 10.** Cucumber

**Fuente:** Testingadvice.com. (s. f.)

---

**Ghost Inspector:** lo más interesante de Ghost Inspector es que nos permite automatizar sin saber codificar, lo que la convierte en una herramienta ideal para principiantes.

Ciertamente, esta herramienta solo permite 100 ejecuciones gratuitas por mes.



**Figura 11.** Ghost Inspector

**Fuente:** Federico-toledo.com. (s. f.)

De acuerdo a Khatri (2021),

**Jmeter:** JMeter, de Apache, es una herramienta de prueba de código abierto que se utiliza para probar el rendimiento de sitios web y aplicaciones web dinámicas. Una persona con muy pocos conocimientos técnicos también puede interpretar los resultados proporcionados por JMeter.



**Figura 12.** JMeter

**Fuente:** Commons.wikimedia.org. (2016)

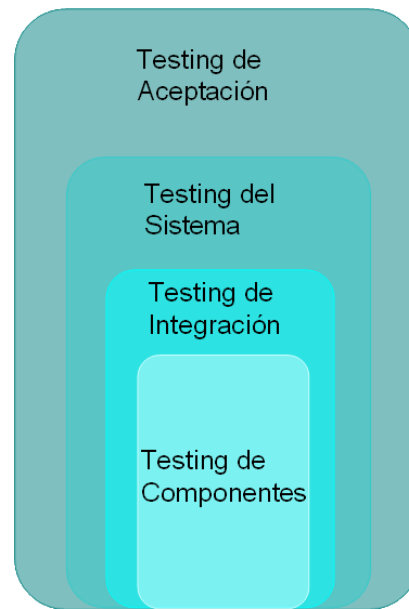


### 3.2. Función de las pruebas de sistema

Habiendo ya probado todos los componentes del sistema y la integración de ellos, se debe pasar a un nuevo nivel de pruebas en el que vamos a comprobar si el sistema cumple con los requisitos que se han especificado.

Si consideramos lo que indica la International Software Testing Qualifications Board (Comité Internacional de Certificaciones de Pruebas de Software) ISTQB, vemos que se pueden considerar a aquellas pruebas que se basan en riesgos, especificaciones de requerimientos, procesos de negocio, casos de uso, o bien alguna otra descripción escrita. Pueden ser modelos que indiquen el comportamiento del sistema, cómo interactúa con el sistema operativo y con los recursos del sistema.

Con estas pruebas vamos a analizar, por medio de la aplicación de técnicas de caja negra, los requisitos funcionales y los requisitos no funcionales, junto con las características de calidad.



**Figura 13.** Niveles de pruebas

**Fuente:** Josepablosarco.wordpress.com. (2012)

## 4. Determina la función de pruebas de sistema en el proceso de desarrollo de software, considerando crecimiento de uso de software, según estándares del mercado.

### 4.1. Crecimiento de uso de un software

De acuerdo a Guía Chile Energía (s. f.),

Según la principal firma de analistas de la industria de tecnologías de la información (TI), IDC, el sector a nivel mundial tuvo un crecimiento positivo al cierre de 2020, de 5.5% en dólares constantes y seguirá creciendo para 2021, con una estimación promedio anual de 7.7%



Una de las industrias que no hibernaron en esta pandemia fue la de la tecnología de la información, convirtiéndose en un eje estratégico de empresas, Gobiernos y personas comunes para seguir trabajando y tratando de llevar una vida normal en plena pandemia.

Para Chile, según el mismo estudio, el crecimiento de la industria fue de un 6.3% y se espera para el 2021 un crecimiento en torno al 5.5%. Bajo esta realidad es que MITI, Asociación Gremial Mejor Industria TI, explica que este crecimiento en parte se debe a la gran migración que tuvieron que realizar las empresas a la digitalización de manera acelerada.

Para el próximo año se ve una fuerte tendencia en la misma utilización de la nube, pero para tomar decisiones de negocios inteligentes. Por otro lado, el uso del Ecommerce seguirá proyectándose fuertemente para el 2021 “Debido a la pandemia el comercio se ha volcado al ecommerce.

Aún existen muchas empresas que dependen de su operación física en mayor o menor medida, como, por ejemplo, los aeropuertos, las clínicas y los establecimientos educacionales que si bien pueden ser complementados online siempre requieren de un componente social.

En este escenario, la eficiencia de procesos tiene que mejorar drásticamente. Las operaciones en general funcionan en silos de manera poco eficiente. Es fundamental que todo lo que ocurra en el día a día, lo que sea necesario para que la maquinaria cotidiana funcione, esté optimizado, digitalizado y sea constantemente analizado para poder competir con el negocio del futuro.

Por último, la seguridad informática, en la actualidad, según McAfee, el número de amenazas externas dirigidas a los servicios en la nube aumentó en un 630% durante con la mayor concentración en las plataformas de colaboración. Para su informe, la empresa dividió los intentos de inicio de



sesión y acceso sospechosos en dos categorías: uso excesivo desde una ubicación anómala y superhombre sospechoso.

Ambas han visto un patrón similar de aumento y crecimiento durante el período de tiempo analizado, "por lo que será indispensable la creación de nuevas formas de asegurar a la empresa, sus procesos y documentos".

## Conclusiones

En esta semana, estudiamos todo lo relacionado con las distintas pruebas que podemos aplicar a nuestros proyectos.

Vimos que existen tipos de pruebas, como son las funcionales y no funcionales, pero, además, que hay técnicas de prueba, como son las estáticas o dinámicas, y que en el mercado encontramos herramientas para realizar pruebas de manera automática. Lo fundamental acá es que seamos capaces de analizar qué necesitamos, cuáles son las características y, a partir de ello, elegir. Esto nos dará un respaldo muy importante a la hora de presentar el sistema al cliente, ya que tendremos como respaldo la documentación con los resultados obtenidos; en primer lugar, demostrando que el sistema hace lo que él espera, pero, además, que lo hace bien, considerando que con las pruebas exitosas, es decir, aquellas que detectaron errores, estamos minimizando las posibles ocurrencias de fallas cuando el sistema entre a explotación.



---

## Referencias bibliográficas

Commons.wikimedia.org. (2014). Blackbox. [Imagen]. Recuperado de <https://commons.wikimedia.org/wiki/File:Blackbox3D.png>

(2016). Logo JMeter. [Imagen]. Recuperado de [https://commons.wikimedia.org/wiki/File:Apache\\_JMeter.png](https://commons.wikimedia.org/wiki/File:Apache_JMeter.png)

Eistria.b-cdn.net. (2016). Logo Selenium. [Imagen]. Recuperado de <https://eistria.b-cdn.net/wp-content/uploads/2016/02/selen.png>

Empresas.blogthinkbig.com. (2014). Caja blanca y caja negra. [Imagen]. Recuperado de <https://empresas.blogthinkbig.com/qa-pruebas-para-asegurar-la-calidad-de/>

Federico-toledo.com. (s. f.). Logo Ghost Inspector. [Imagen]. Recuperado de <https://www.federico-toledo.com/chequeos-automaticos-con-ghost-inspector/ghost-inspector-logo/>

Guía Chile Energía. (s. f.). Las nuevas tendencias y proyecciones de la industria Ti para este 2021 en el país. Recuperado de <https://www.guiachileenergia.cl/las-nuevas-tendencias-y-proyecciones-de-la-industria-ti-para-este-2021-en-el-pais/>

Encora.com (2019). Debate: Las pruebas manuales de software vs pruebas automatizadas

Recuperado de <https://www.encora.com/es/blog/debate-las-pruebas-manuales-de-software-vs-pruebas-automatizadas>



Josepablosarco.wordpress.com. (2012). Niveles de pruebas. [Imagen].

Recuperado de

<https://josepablosarco.wordpress.com/2012/05/25/istqb-cap-2-testing-a-traves-del-ciclo-de-vida-del-software-ii/>

Khatri, V. (2021, 22 de abril). Las 17 mejores herramientas de prueba de software que debe conocer como evaluador. [Entrada de blog]. Recuperado de <https://geekflare.com/es/software-testing-tools/>

Lucerna, C. (2019, 12 de junio). ¿Qué es el testing de software?. [Entrada de blog]. Recuperado de <https://openwebinars.net/blog/que-es-el-testing-de-software/>

Medium.com (2019). Logo Appium. [Imagen]. Recuperado de <https://medium.com/@knoldus/introduction-to-appium-and-its-drivers-bc676cfc5799>

Nebulargroup.com. (2020). Caja negra. [Imagen]. Recuperado de <https://nebulargroup.com/knowledgebase/que-es-un-algoritmo-de-caja-negra/>

Panel.es. (2014). Caja blanca. [Imagen]. Recuperado de <https://www.panel.es/software-qa-cuales-son-los-tipos-de-pruebas-software/>

Peña, J. (2006). Capítulo 6. Pruebas de sistema. [Documento PDF]. Recuperado de <https://www.uv.mx/personal/jfernandez/files/2010/07/Pruebas-de-Sistema.pdf>



Sánchez, J. (2015). Pruebas de software. Fundamentos y técnicas. Universidad Politécnica de Madrid. Recuperado de [http://oa.upm.es/40012/1/PFC\\_JOSE\\_MANUEL\\_SANCHEZ\\_PEN\\_O\\_3.pdf](http://oa.upm.es/40012/1/PFC_JOSE_MANUEL_SANCHEZ_PEN_O_3.pdf)

Sites.google.com. (s. f.). Límite de los sistemas. [Imagen]. Recuperado de <https://sites.google.com/site/ingenieriadesistemasitt/1-2-3-entorno-o-medio-ambiente-de-los-sistemas>

Sumano, M., Fernández, J. y Cortés, M. (2010). *Ingeniería de software II. Manual de prácticas*. Universidad Veracruzana.

Rodríguez, C. (s. f.). Automatizar pruebas de software. ¿Cuándo y por qué? [Entrada de blog]. Recuperado de <https://cl.abstracta.us/blog/automatizar-pruebas-de-software/>

Tecnologias-informacion.com. (s. f.). Cómo hacer testing de software manual. Recuperado de <https://www.tecnologias-informacion.com/testingmanual.html>

Testingadvice.com. (s. f.). Logo Cucumber. [Imagen]. Recuperado de <https://testingadvice.com/cucumber-usage-command-line-parameters/>

Universidad Veracruzana. (s. f.). Esquema de interacción entre módulos del subsistema de Recursos Humanos. [Imagen]. Recuperado de <https://www.uv.mx/siiu/informacion-general/subsistema-de-recursos-humanos/>