

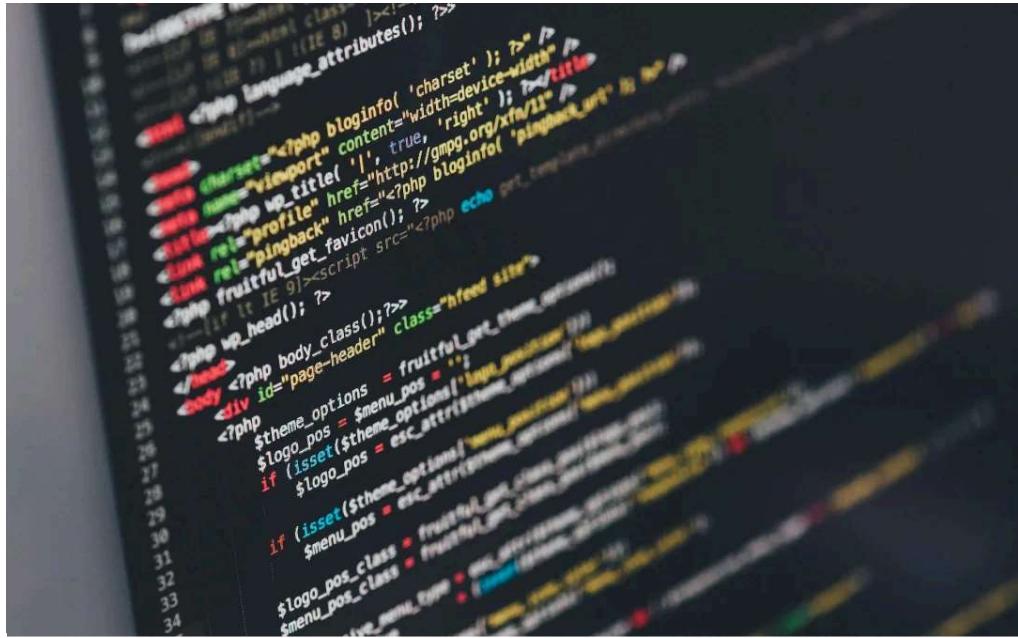


Get your 6-month No-Cost Opt-Out offer for
Unlimited Software Automation?

[Download Trial](#)

Categories: [Software Testing](#) [RPA](#) [Trends](#) [AI](#) [Videos](#)
[Courses](#) [Subscribe](#)

Particionamiento de Equivalencia en Pruebas de Software – ¡Qué es, Tipos, Proceso, Enfoques, Herramientas y Más!



puede utilizar para desbloquear los beneficios de esta técnica.

Table of Contents



Qué es la partición de clases de equivalencia en pruebas de software?



Todos los programas informáticos tienen unas condiciones de entrada particulares. En el contexto de las pruebas de software, estas condiciones de entrada describen los valores o datos que un probador debe utilizar para verificar la calidad y funcionalidad de su software. Estas entradas pueden ser algo tan simple como un clic del ratón, hasta texto y números.

Una partición equivalente en las pruebas de software explora las distintas entradas necesarias para utilizar el software y las agrupa en clases de equivalencia, es decir, conjuntos de entradas que tendrán un

ahorrar tiempo y esfuerzo en el ciclo de vida de las pruebas de software (STLC) es crucial.

Por último, cabe señalar que la prueba de equivalencia es una técnica de prueba de caja negra. En pocas palabras, significa que los probadores no necesitan conocer el código interno o el funcionamiento interno del programa. Las pruebas se basan en entradas, salidas y comportamientos externos. Como tales, estas pruebas se centran en gran medida en el comportamiento del usuario mientras utiliza el programa.

1. La partición de la equivalencia de las pruebas de software en pocas palabras

La partición de equivalencias divide los datos de entrada de las pruebas de software en dos campos: entradas válidas y no válidas. Los valores dentro de cada partición deben hacer que el software muestre el mismo comportamiento. Por ejemplo:

- Si la condición de un valor de la partición A es verdadera, también deben serlo los demás valores de la partición A.
- Del mismo modo, si las condiciones de un valor de la partición A son falsas, los demás valores de la partición A también deben ser falsos.

En un contexto de pruebas, cada partición debe cubrirse al menos una vez. Lógicamente, esto significa que si una entrada de la Partición A falla, el resto de entradas también fallarán. Este proceso debería ahorrar tiempo, ya que en lugar de probar cada entrada que se encuentra en la Partición A, los probadores pueden probar sólo una y extrapolar el resultado basándose en sus puntos en común.

2. Por qué son importantes las pruebas de equivalencia de clase en las pruebas de

significa que los responsables de las pruebas deben aprovechar al máximo sus recursos. La partición de equivalencias en pruebas de software ayuda a los equipos a encontrar un equilibrio entre eficacia y fiabilidad en sus pruebas reduciendo el número de entradas.

Ventajas de la partición por equivalencia en pruebas de software

Los equipos de pruebas se decantan por una partición equivalente en las pruebas de software por diversas razones. He aquí algunas de las más convincentes.

1. Eficacia

La gran ventaja de las pruebas de partición de equivalencia reside en su eficacia. Cuando los evaluadores utilizan la partición por equivalencia, pueden reducir el número de casos de prueba necesarios sin comprometer la cobertura de la prueba. Al seleccionar un caso de

es mucho más sencilla. Probar cada entrada individualmente requiere mucha documentación y coordinación. Reducirlo a un ejemplo representativo agiliza el proceso de prueba.

Cobertura mejorada

El uso de clases de equivalencia en las pruebas también permite utilizar el tiempo de prueba de forma más eficaz. Reducir las entradas de prueba en clases significa que puede probar más a fondo cada clase. Este enfoque global sería francamente imposible si se probara cada entrada por separado. La partición de equivalencias permite a los equipos ser minuciosos y probar datos válidos e inválidos, casos límite, valores límite, etc.

3. Reutilización

El tiempo inicial que se invierte en establecer cada clase de equivalencia en las pruebas de software se amortiza más adelante si se reutilizan estas clases para futuras pruebas de entrada. Aunque no todas las particiones serán relevantes para futuras pruebas, las que lo sean le ahorrarán mucho tiempo en futuros proyectos o incluso en situaciones de **pruebas de regresión**.

Inconvenientes de la partición por equivalencia en pruebas de software

Aunque la partición por equivalencia ofrece algunas ventajas importantes, no es la solución ideal para todos los casos. Exploraremos algunas de sus limitaciones.

1. Orden de entrada

En determinadas situaciones, el orden de entrada es una parte fundamental para probar la funcionalidad de una aplicación. Eso no es algo que se pueda reducir realmente utilizando la partición por equivalencia. Los encargados de las pruebas deben ser conscientes de estas situaciones y utilizar técnicas alternativas para proporcionar una buena cobertura.

2. Dependencias de entrada complejas

El software complejo con dependencias de entrada complejas es otra área en la que se ponen de manifiesto las limitaciones de la partición de equivalencias. Por ejemplo, programas informáticos que realizan cálculos a partir de diversas entradas. En este escenario, los probadores tendrían que utilizar diversas técnicas para reducir la explosión combinatoria y aumentar la probabilidad de aislar los defectos.

Enfoques alternativos para complementar la limitaciones de las pruebas de equivalencia

dependencias intrincadas entre los valores de entrada puede requerir enfoques complementarios adicionales.

Cuando se trata de escribir casos de prueba para software complejo, utilizar una combinación de estos enfoques es una idea sólida.

1. Pruebas por pares

La prueba por pares es una técnica de prueba de software que comprueba todas las combinaciones posibles de cada par de parámetros de entrada. Este enfoque garantiza que cada par de parámetros se pruebe conjuntamente al menos una vez.

IS YOUR COMPANY IN NEED OF
ENTERPRISE LEVEL
TASK-AGNOSTIC
SOFTWARE
AUTOMATION?

[Book a Demo](#)

[Book a Demo](#)

forma de garantizar una cobertura sistemática cuando existen dependencias complejas.

3. Pruebas de transición de estado

Este tipo de prueba mide cómo el software pasa de un estado a otro en respuesta a varias combinaciones de entrada.

4. Pruebas basadas en modelos

Este enfoque requiere crear un modelo basado en la lógica interna del software y utilizar una herramienta de automatización para crear casos de prueba basados en ese modelo. Esta técnica es experta en manejar la complejidad y garantizar una cobertura adecuada.

Ejemplos de pruebas de partición de clases de equivalencia

La mejor manera de entender la partición de equivalencias es ver cómo y dónde se puede utilizar una clase de equivalencia en las pruebas de software. He aquí algunos ejemplos que le ayudarán a visualizar mejor

Supongamos que está creando una aplicación para una tienda online de material de papelería. Existe una hoja de pedido típica para fardos de papel A4. A continuación se explica cómo utilizar las clases de equivalencia para probar esta forma.

Clases de equivalencia:

Las cantidades de papel A4 están dentro de un rango específico de, por ejemplo, 1 a 100. Entonces, las tres clases son:

- 1 a 100
- Números inferiores a 1
- Números por encima de 100.

Casos de prueba:

Deben ejecutarse tres casos de prueba, con los siguientes resultados esperados

- Cualquier número entre 1 y 100 = Pedido tramitado
- Números inferiores a 1 = mensaje de error
- Números superiores a 100 = mensaje de error

2. Ejemplo nº 2 de prueba de partición de equivalencia

Una clase de equivalencia en las pruebas de software puede tratar algo más que números. En este ejemplo, exploraremos cómo puede utilizar el mismo principio para verificar un portal de carga de archivos.

Supongamos que necesita realizar pruebas para un sitio que requiere que los usuarios carguen documentos de identidad, pero sólo puede aceptar formatos específicos.

Unlock Exclusive Insights:
Subscribe Now on
CUTTING-EDGE
SOFTWARE TESTING, TCE,
& RPA

Subscribe to Newsletter

Clases de equivalencia:

- Los documentos compatibles son PDF y JPEG.
- Los documentos no compatibles son todos los demás formatos de documento
- Ningún documento

Cómo realizar una partición de equivalencia enfoque de las pruebas de software

Si desea utilizar clases de equivalencia en las pruebas, debe adoptar un enfoque estratégico. He aquí una útil guía paso a paso para aplicar la partición de equivalencias en las pruebas de software.

Paso nº 1: Identificar las variables de entrada

Cada programa responde a una serie de variables de entrada. En el caso del software complejo, estas variables pueden ser enormes. Por tanto, revisa los requisitos y especificaciones del software y señala todas las variables que influyen en su comportamiento.

Algunas de las entradas más obvias incluirán formularios de entrada de usuario. Sin embargo, debe considerar una gama más amplia de entradas para su lista. También puede tener en cuenta variables de entorno, llamadas a la API, cálculos internos, etc.

Paso 2. Determinar particiones válidas y no válidas

Observe cada variable de entrada y empiece a dividirlas en resultados válidos e inválidos. Éstas serán sus clases de equivalencia en las pruebas.

1. Particiones válidas

Las particiones válidas pueden dividirse en dos clases.

Clases de equivalencia positiva:

Valores que espera que su software gestione correctamente. Por ejemplo, para un software que registra calificaciones porcentuales, cualquier cosa entre 0 y 100 es válida.

Clases de equivalencia negativas:

Esta categoría será para los valores que están fuera de los límites de la entrada esperada, pero que su software debe manejar con un mensaje de error. Por ejemplo, la entrada es 110 para un grado de porcentaje, lo que hace que el software devuelva un mensaje de error que dice: "Todos los valores deben ser de 0 a 100".

2. Particiones no válidas

Estas clases de equivalencia incluirán entradas que desencadenarán errores o comportamientos inesperados. En nuestro ejemplo anterior, eso podría incluir intentos de introducir A+ o B o entradas similares en la calificación porcentual. Aunque estas entradas podrían ser técnicamente correctas, están fuera de las expectativas numéricas.

#3. Redacción de casos de prueba eficaces

A continuación, debe diseñar casos de prueba que cubran cada

IS YOUR COMPANY IN NEED OF
ENTERPRISE LEVEL
TASK-AGNOSTIC
SOFTWARE
AUTOMATION?

[Book a Demo](#)

Book a Demo

Consejos para escribir casos de prueba sólidos

- Piense en los valores límite: Asegúrate de que compruebas los límites de tus particiones. Mínimo, máximo, inclusivo, exclusivo, etc., ya que estas áreas son firmes candidatas a los errores. Por ejemplo, si sus expectativas de entrada están entre 0 y 100, comprueba si hay valores negativos, así como números como 101.
- Considere escenarios de prueba positivos y negativos para sus casos de prueba válidos e inválidos.

- Siempre que sea posible, utilice herramientas visuales para aportar claridad y objetividad a sus casos de prueba mediante diagramas o tablas para trazar sus particiones.

#4. Programe y ejecute sus casos de prueba

Prioriza tus tareas en función de factores como:

- Qué zonas tienen más probabilidades de presentar defectos
- Qué escenarios tienen más probabilidades de provocar situaciones graves, como bloqueos o congelaciones.

A continuación, ejecute las pruebas y registre los resultados y los errores que se produzcan. Para programas complejos con muchas entradas, puede utilizar herramientas **de RPA** para imitar las acciones de los usuarios.

#5. Analizar los resultados

Agrupe los datos de las pruebas recopilados y analice los resultados.

Algunos métodos que debes utilizar son

- Examine cada caso de prueba y compare los resultados reales con los esperados.
- Encuentre cualquier discrepancia e investigue y notifique cualquier error o defecto.

#6 Consejos adicionales

Aunque estos consejos no se aplicarán en todos los casos, resultarán útiles para las pruebas de software complejas.

- Siempre que sea posible, automatice sus casos de prueba de partición de equivalencias

Partición de equivalencias y análisis de valores límite

La partición de equivalencia se basa en la suposición de que cada prueba dentro de una partición producirá el mismo resultado. Aunque eso es cierto en muchas situaciones, no siempre funciona. Por ejemplo, cualquier entrada que se haya añadido a una partición por error puede quedar sin comprobar, lo que reduciría la cobertura y podría provocar inestabilidad en el software.

La solución a este problema es la prueba del valor límite. Permite a los equipos de pruebas de software centrarse en las áreas que tienen más probabilidades de contener riesgos y probar el software sobre esa base. En resumen, propone que lo más probable es que los riesgos se produzcan en los bordes o límites de sus particiones de entrada. Por lo tanto, los probadores pueden escribir casos de prueba en los límites

Las herramientas de automatización de pruebas de software, como **ZAPTEST**, pueden ayudar a los equipos a automatizar la partición de equivalencias tanto durante la creación como durante la ejecución de las pruebas.

Exploraremos cómo ZAPTEST puede ayudarle a desbloquear los beneficios de este útil enfoque de pruebas de caja negra.

1. Selección de herramientas

Seleccionar la herramienta adecuada para el trabajo es importante. La mayoría de las **herramientas de automatización de pruebas** se especializan en pruebas web, móviles o de escritorio. ZAPTEST es capaz de realizar pruebas en diferentes plataformas y aplicaciones, lo que lo convierte en una opción sólida.

2. Escribir y ejecutar casos de prueba

ZAPTEST 1Script le permite explorar la interfaz de usuario para construir la automatización de pruebas. Además, también puede escanear maquetas de aplicaciones si se encuentra en una fase temprana de desarrollo. Utilizando la función Scan GUI, ZAPTEST escaneará todos los objetos de prueba y los añadirá a la lista de objetos.

entrada necesarios para su interfaz. A continuación, puede crear casos de prueba para cada equivalencia y ejecutarlos. Incluso puede reutilizar casos de prueba y editarlos en el editor de pasos, lo que le ahorrará mucho tiempo.

3. Informes y gestión de casos de prueba

ZAPTEST permite ejecutar casos de prueba en paralelo, lo que ahorra un tiempo considerable. Esto puede ayudarle a ejecutar un gran número de particiones de equivalencia diferentes a la vez o a ejecutar determinados grupos de pruebas.

Los resultados son fáciles de recopilar gracias a los informes detallados de fallos/pasados, capturas de pantalla, registros de ejecución y métricas de rendimiento relacionadas con cada caso de prueba.

4. Mantenimiento de casos de prueba

También puede realizar un seguimiento y mantenimiento sencillos de sus casos de prueba gracias a las funciones de control de versiones de calidad. Además, los usuarios de ZAPTEST pueden clonar y reutilizar pruebas para alcanzar un nuevo nivel de eficacia.

ZAPTEST ofrece muchas más funcionalidades aparte de la automatización de casos de prueba. Con un conjunto de herramientas RPA, ZAPTEST ofrece funcionalidad 2 en 1, tendiendo un puente entre DevOps y BizOps en un futuro marcado por la hiperautomatización, donde todo lo que se pueda automatizar se automatizará.

Reflexiones finales

dividir los datos de las pruebas en trozos manejables y del tamaño de un bocado, cada uno de los cuales puede probarse a fondo.

[Download post as PDF](#)

AI



[Copilotos e IA generativa en RPA / Pruebas de software](#)

[Ingeniería inmediata en automatización de software](#)

[Impacto de la IA en la RPA](#)

[RPA frente a IA](#)

[Automatización inteligente de procesos frente a RPA](#)

[La visión por ordenador es el futuro de la automatización de las pruebas de software - Una historia del pasado, el presente y el futuro](#)

Automatización robótica de procesos



[RPA en cuentas por pagar](#)

[RPA en seguros](#)

[RPA en RRHH](#)

[RPA en finanzas y banca](#)

[Tamaño y tendencias del mercado de RPA](#)

[RPA en la fabricación](#)

[RPA en sanidad](#)

[Las 10 principales ventajas de la RPA](#)

[Las 31 mejores herramientas de RPA](#)

[6 tipos de RPA](#)

Guías



[ZAPTEST para Agile DevOps](#)

[RPA frente a automatización de pruebas](#)

[Gestión de datos de prueba \(TDM\) en las pruebas de software: definición, historia, herramientas, procesos y más.](#)

[Creación de un Centro de Excelencia de Pruebas \(TCoE\) - Los pormenores de la creación de una organización ágil](#)

[Una guía completa para la automatización de pruebas de software](#)

[Guía completa sobre la automatización de procesos robóticos \(RPA\)](#)

[Hiperautogestión - Una guía completa](#)

Las mejores herramientas para probar software



[10 mejores herramientas para pruebas de regresión](#)

[10 mejores herramientas para pruebas de rendimiento](#)

[30 mejores herramientas para pruebas de software](#)

Sin categorizar



Tipos de pruebas de software



[Pruebas ETL](#)

Prueba negativa

Pruebas con monos

Pruebas incrementales

Pruebas de Remojo en Pruebas de Software: ¡Qué es, Tipos, Procesos, Enfoques, Herramientas y Más!

Pruebas de estrés en pruebas de software: Qué es, Tipos, Procesos, Enfoques, Herramientas & ¡Más!

Pruebas de compatibilidad: qué son, tipos, proceso, características, herramientas y mucho más.

Pruebas alfa: qué son, tipos, proceso, vs. pruebas beta, herramientas y mucho más.

Pruebas Beta - Qué son, Tipos, Procesos, Enfoques, Herramientas, vs. Pruebas Alfa & ¡Más!

Pruebas de aplicaciones móviles: qué son, tipos, procesos, enfoques, herramientas y mucho más.

Pruebas de Caja Blanca: ¡Qué es, Cómo funciona, Retos, Métricas, Herramientas y Más!

Pruebas ad hoc: qué son, tipos, procesos, enfoques, herramientas y mucho más.

Pruebas manuales: qué son, tipos, procesos, enfoques, herramientas y mucho más.

Pruebas de caja negra: qué son, tipos, procesos, enfoques, herramientas y mucho más.

Pruebas no funcionales: ¡Qué es, Tipos, Enfoques, Herramientas y Más!

Pruebas de mutación: tipos, procesos, análisis, características, herramientas y mucho más.

Pruebas de caja gris - Profundice en qué son, tipos, procesos,

Pruebas Exploratorias - ¡Una Inmersión Profunda en Tipos, Procesos, Enfoques, Herramientas, Marcos y Más!

Pruebas de extremo a extremo - Profunda inmersión en los tipos de pruebas E2E, procesos, enfoques, herramientas y mucho más.

¡Pruebas de Backend - Profunda inmersión en lo que es, sus tipos, procesos, enfoques, herramientas y más!

¡Smoke Testing - Profundización en Tipos, Proceso, Herramientas de Software de Smoke Test y Más!

¿Qué son las pruebas API? Profundice en la automatización de pruebas de API, procesos, enfoques, herramientas, marcos y mucho más.

¿Qué es el "Sanity Testing"? Profundice en los tipos, procesos, enfoques, herramientas y mucho más.

¿Qué es la prueba de software de interfaz de usuario? Profundización en los tipos, procesos, herramientas y aplicación

¿Qué son las pruebas de integración? Profundización en los tipos, el proceso y la aplicación

¿Qué son las pruebas de rendimiento? Profundice en los tipos, las prácticas, las herramientas, los retos y ¡más!

¿Qué son las pruebas unitarias? Profundice en el proceso, los beneficios, los retos, las herramientas y mucho más.

¿Qué es la automatización de pruebas? Una guía sencilla y sin jerga

¿Qué es la prueba de regresión? Aplicación, herramientas y guía completa

¿Qué es la prueba de carga? Profundización en los tipos, prácticas, herramientas, retos y más

¿Qué es la prueba ágil? Proceso, ciclo de vida, métodos y aplicación

Alpha Testing

API Testing

Automation

Beta Testing

Black Box Testing

Compatibility Testing

Computer Vision Technology

Functional Testing

Grey Box Testing

Integration Testing

Load Test

Manual Testing

Media

Mobile App Testing

Mockup-Tests

Mutation Testing

Non-functional testing

Regression Testing

RPA

RPA In Manufacturing

RPA Tools

RPA Use Cases

Sanity Testing

Smoke Testing

Soak Testing

WebDriver

White Box Testing

Free Test Automation Tools



Performance

Web Apps

Mobile Apps

Windows

iOS Apps

QA

UI

API

Linux

Android Apps

Courses



UI Scripted

UI Script-Less

API Scripted

API Script-Less

LOAD

[Subscribe to Newsletter](#)

Founder and CEO of **ZAPTEST**, with 20 years of experience in Software Automation for Testing + RPA processes, and application development. Read Alex Zap Chernyak's full executive profile on [Forbes](#).



This post is also available in:

Български	简体中文	繁體中文	Hrvatski	Čeština
Dansk	Nederlands	English	Eesti	Français
Deutsch	हिन्दी	Magyar	Italiano	日本語
Latviešu	Lietuvių	Polski	Português	Português
Punjabi	Română	Русский	српски	Slovenčina
Slovenčina	Svenska	Tamil	Türkçe	Українська
Albanian	ქართველი	Suomi	Ελληνικά	עברית
Íslenska	Norsk bokmål			

Testing Automation



1395 Brickell Ave. Suite 800
Miami, FL. 33131 USA
Phone (800) 795-3552



First Name
Last Name
Email

By proceeding, you confirm that you have read
and agree to ZAPTEST's [Privacy Policy](#) and [Terms
of Service](#).

[Subscribe](#)

Copyright 2024 – All rights reserved.