UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE INGENIERÍA DE SISTEMAS

ANÁLISIS COMPARATIVO ENTRE DOS HERRAMIENTAS DE EVALUACIÓN DE SOFTWARE BASADAS EN MODELOS.

Trabajo de titulación previo a la obtención del

Título de Ingenieros de Sistemas

AUTORES: WILIAN SANTIAGO CALDERÓN MERA

HENRY DAVID FLORES TITUAÑA

TUTOR: FRANKLIN EDMUNDO HURTADO LARREA

Quito – Ecuador

2022

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Wilian Santiago Calderón Mera con documento de identificación N° 1722089214 y Henry David Flores Tituaña con documento de identificación N° 1722738562, manifestamos que:

Somos autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 12 de septiembre del 2022

Atentamente,

Wilian Santiago Calderón Mera
1722089214

Henry David Flores Tituaña
1722738562

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Wilian Santiago Calderón Mera con documento de identificación N° 1722089214 y Henry David Flores Tituaña con documento de identificación N° 1722738562, expresamos nuestra voluntad y por medio del presente cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud que somos autores del Artículo Académico: "Análisis comparativo entre dos herramientas de evaluación de software basadas en modelos..", el cual ha sido desarrollado para optar por el título de Ingeniera de Sistemas, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 12 de septiembre del 2022

Atentamente,

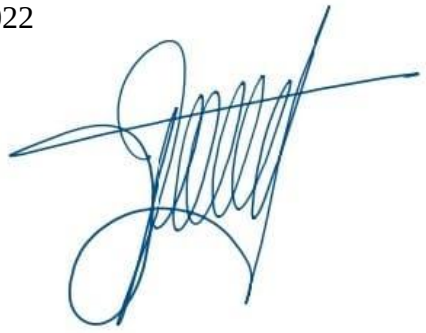
Wilian Santiago Calderón Mera
1722089214

Henry David Flores Tituaña
1722738562

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN.

Yo, Franklin Edmundo Hurtado Larrea con C.I.: 1713382016, docente de la Universidad Politécnica Salesiana, declaro que bajo mi dirección y asesoría fue desarrollado el Trabajo de Titulación: ANÁLISIS COMPARATIVO ENTRE DOS HERRAMIENTAS DE EVALUACIÓN DE SOFTWARE BASADAS EN MODELOS, realizado por Wilian Santiago Calderón Mera con documento de identificación N° 1722089214 y Henry David Flores Tituaña con documento de identificación N° 1722738562, obteniendo como resultado final el trabajo de titulación bajo la opción de Artículo Académico con lo cual cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 12 de septiembre del 2022



Ing. Franklin Edmundo Hurtado Larrea, Msc

1713382016

ANÁLISIS COMPARATIVO ENTRE DOS HERRAMIENTAS DE EVALUACIÓN DE SOFTWARE BASADAS EN MODELOS.

COMPARATIVE ANALYSIS BETWEEN TWO MODEL-BASED SOFTWARE ASSESSMENT TOOLS.

Wilian S. Calderón[1], Henry D. Flores [2], Franklin E. Hurtado[3]

Resumen

La evaluación de software es uno de los puntos críticos durante el ciclo de vida del software así como en la mejora continua de dichos productos, durante un proyecto de software el tiempo es un factor de suma importancia y reducir el tiempo en dicha fase es un gran aporte para dichos proyectos, la técnica de testeo basadas en modelos (MBT), tiene como objetivo el reducir el tiempo de pruebas funcionales y ayudar a los expertos a tener una visión más amplia de los macro y micro procesos que se llevan a cabo en el producto de software, para así tener un entendimiento más amplio sobre las características y posibles falencias en la funcionalidad del producto de software. En la presente investigación se propuso realizar una comparación de dos herramientas de dicha técnica de testeo de software y difundir sus capacidades en los dos escenarios en los cuales fueron puestas a prueba, el objetivo de esta investigación no es definir cuál herramienta es superior a la otra, más bien tiene como propósito destacar las fortalezas de dichas herramientas en las diferentes configuraciones y entornos de desarrollo.

Palabras clave: Model based testing(MBT), testeo de software, herramientas de testeo de software, análisis comparativo.

Abstract

Software evaluation is one of the critical points during the software life cycle as well as in the continuous improvement of said products, during a software project time is a very important factor and reducing the time in said phase is a great Contribution to these projects, the model-based testing technique (MBT), aims to reduce the time of functional tests and help experts to have a broader vision of the macro and micro processes that are carried out in the system. software product, in order to have a broader understanding of the characteristics and possible shortcomings in the functionality of the software product. In the present investigation it was proposed to make a comparison of two tools of said software testing technique and to disseminate their capabilities in the two scenarios in which they were tested, the objective of this investigation is not to define which tool is superior to the other, rather it is intended to highlight the strengths of such tools in different configurations and development environments.

Keywords: Model based testing (MBT), software testing, software testing tools, comparative analysis.

[1] Estudiante de Ingeniería de Sistemas – Universidad Politécnica Salesiana, Egresado – UPS – Sede Quito-
Autor para correspondencia: wcalderonm@est.ups.edu.ec.
[2] Estudiante de Ingeniería de Sistemas – Universidad Politécnica Salesiana, Egresado – UPS – Sede Quito-
Autor para correspondencia: hflorest@est.ups.edu.ec.
[3] Magister en planificación y dirección estratégica, Ingeniero de sistemas – UPS – sede Quito.
Autor por correspondencia: fhurtado@ups.edu.ec.

1. Introducción

El Testeo de Software Basado en Modelos MBT es una técnica basada en modelos que permite automatizar casos de pruebas funcionales con la finalidad de gestionar mejor los recursos mientras se realiza el proceso de testeo de software [1]. A decir de algunos autores ésta puede ser una de las técnicas más usadas en el futuro próximo en las pruebas funcionales de software [1], pero se ha verificado que la información es insuficiente hasta el momento [2].

Las herramientas de evaluación de software basadas en modelos las cuales serán objeto de análisis comparativo son: GraphWalker y LeapWork.

El propósito de esta investigación es entregar información sistematizada a la comunidad académica y universitaria, sobre un análisis comparativo de dos herramientas de pruebas de software basadas en modelos MBT y responder a la pregunta ¿Se tiene información comparativa obtenida a través de experimentación que permita al lector recabar datos informativos de esta comparación?

El presente documento se estructurará de la siguiente manera: La sección 1 presenta la introducción. La sección 2 presenta el análisis de la literatura. La sección 3 presenta la metodología. La sesión 4 los resultados y análisis de estudio. La sección 5 las conclusiones.

2. Revisión de la literatura

En esta sección se presenta la revisión de la literatura del tema de investigación. Andrés L. Cubillos en su investigación describe un consolidado donde se presenta diferentes técnicas de testeo de software [19]. Como se puede observar en la tabla 1, se detalló una clasificación de las técnicas que facilitan la ejecución de las pruebas con su descripción, herramientas que utilizan, tipo de pruebas y descripción de la prueba.

Villalobos-Arias en su artículo describen al testeo de Software Basado en Modelos MBT, como una técnica que intenta automatizar partes del proceso de prueba de software. Los autores describen también a las herramientas de testeo de software basadas en modelos como aquellas que pueden facilitar la automatización de pruebas, evolución del testeo y que ayudan a gestionar mejor los costos y tiempo lo que puede ser difícil de lograr en los procesos tradicionales [2].

Cleva Farto & Endo en su artículo explican que las herramientas MBT son aquellas herramientas de testeo de software que automatizan las pruebas a partir de modelos que describen las posibles fallas en el software [5], a estas herramientas se pueden aplicar alguna escala de medición (métrica) para realizar una comparativa entre las mismas lo cual es el objetivo de esta investigación.

Michael, J. B., Bossuyt, B. J., & Snyder, B. B En su artículo utilizan métricas para medir las características de las herramientas de testeo de software, que posteriormente sirven de ayuda para evaluar y comparar herramientas de pruebas automatizadas de software [7].

La familia de normas ISO 25000 tiene el objetivo de servir de guía para el desarrollo de productos de software, para esta investigación se utilizará la ISO 25010 norma que sirve para evaluar la calidad de un producto de software [22].

GraphWalker que es una herramienta MBT multiplataforma para desarrollar pruebas funcionales [3] de aplicaciones web y de escritorio que automatizan procesos repetitivos y que utiliza comandos desde la consola del sistema al momento de ejecutar prueba, utiliza máquinas de estado finitas que son una abstracción de software que permite describir cómo se comporta un sistema en ese instante y cuyo funcionamiento se puede resumir en una entrada la cual modifica el estado en el que se encuentra el sistema y produce una salida [20], y máquinas de estado finitas extendidas que no solo recibe valores de entrada y de salida, sino que también evalúa los valores de entrada y los cálculos necesarios para obtener los valores de salida [20] [21].

LeapWork que a diferencia de la anterior es una herramienta MBT sin código desarrollada para Microsoft Windows, que facilita a los usuarios la automatización de pruebas con resultados más rápidos y a menor costo. Es una

herramienta intuitiva que no requiere conocimientos de programación y cuya automatización está diseñada

Tabla 1: Técnicas empleadas en el testeo de software

Clase	Técnica	Descripción	Herramientas	Tipo de Prueba	Descripción de prueba
Manuales	Manual	Las pruebas se ejecutan manualmente si utilizar ninguna herramienta de automatización	NA	Pruebas funcionales, Pruebas de desempeño, Pruebas revisión de código	
Automatizadas	Capture – Replay	En estas pruebas las secuencias o acciones son capturadas y guardadas, estas interacciones posteriormente se reproducen automáticamente	• QF-Test • Katalon Rec • SWTBot • Jubula .	Pruebas funcionales	Son el tipo de pruebas cuyo enfoque principal se basa en el cumplimiento de los requerimientos del usuario.
	Basadas en Scripts	Para realizar estas pruebas se utilizan scripts que contienen instrucciones con los cuales se pueden ejecutar varias pruebas a la vez, a su vez puede realizar una validación de código.	• Selenium, • Testpad • WebLOAD • JMeter • Codescene • visual-expert • Gerrit	Pruebas de desempeño	Son el tipo de pruebas de tipo carga y estrés, las cuales permiten determinar el rendimiento de un software (escalabilidad, tiempo de respuesta, uso de recursos) bajo una carga de trabajo.
				Pruebas revisión de Código	Son el tipo de pruebas que permiten obtener información de problemas, establecer objetivo de calidad y detectar vulnerabilidades en el código.
	Pruebas dirigidas por Datos y “KeyWords”	En una secuencia de operaciones utilizan palabras clave que simulan las acciones del usuario para realizar cualquier acción de prueba.	• TestComplete • Keyword Driven Testing • Ranorex Studio	Pruebas Funcionales	Son el tipo de pruebas cuyo enfoque principal se basa en el cumplimiento de los requerimientos del usuario.
	Pruebas Basadas en Modelos	Tiene como objetivo generar automáticamente casos de pruebas ejecutables basándose en modelos que simulan el comportamiento del usuario final.	• LeapWork, • Graphwalker • Tosca • Test Modeller • Tesct Compass	Pruebas Funcionales	Son el tipo de pruebas cuyo enfoque principal se basa en el cumplimiento de los requerimientos del usuario.

por la conexión de bloques de construcción visual, los cuales van a representar una o más acciones como: iniciar aplicación, búsqueda de elemento, clic en el elemento etc. Utilizada un controlador el cual almacena los casos, los procesos de automatización imágenes y un agente cuyo objetivo es ejecutar la automatización cuando el controlador se lo solicita [4].

3. Metodología

Durante la presente investigación se desarrolló una serie de pruebas funcionales mediante las cuales se obtuvieron resultados tanto cuantitativos como cualitativos, a partir de los cuales se determinaron las conclusiones correspondientes, las cuales serán presentadas más adelante, en esta investigación se encontrarán dos escenarios de estudio: web y aplicación de escritorio, sobre los cuales se ejecutaron las respectivas pruebas funcionales con las herramientas GraphWalker y LeapWork, las métricas obtenidas durante el proceso de experimentación fueron contrastadas con las de la otra herramienta objetivo, a su vez teniendo en cuenta su respectivo caso de estudio, a continuación se detallará el proceso de selección de las herramientas.

3.1. Selección de las herramientas

Para la selección de las herramientas, se consideraron algunos puntos que se enumeran a continuación:

- Herramientas de pruebas de software basadas en modelos.
- Herramientas que son ampliamente utilizadas en la industria.
- Una herramienta deberá tener licencia privada y la otra deberá ser de software libre.

Una vez satisfechos cada uno de estos parámetros fueron seleccionadas las herramientas GraphWalker y LeapWork, ya que fueron las que mejor cumplieron los parámetros definidos, en caso de GraphWalker es una herramienta muy utilizada en el ámbito de la investigación así como en el de la industria tal como se puede observar en [7] o [8]. A su vez LeapWork también cumple con este requisito en investigaciones como: [9] o [10] con respecto a los demás requisitos GraphWalker es una aplicación open source con un gran respaldo de su comunidad, en el caso de LeapWork es una herramienta privada desarrollada por una empresa que lleva el mismo nombre de la herramienta.

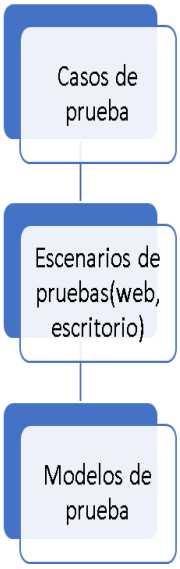
3.2. Método de evaluación

Para la evaluación de estas herramientas se seleccionaron dos aplicaciones: una web y otra de escritorio y mediante estas se crearon los escenarios de prueba los cuales fueron evaluados utilizando las herramientas de estudio antes mencionadas, en base a uno de los marcos de referencia más ampliamente reconocidos a nivel mundial para los requisitos y la evaluación de la calidad de los sistemas y el software ISO 25010, del cual se seleccionaron los siguientes atributos: funcionalidad, desempeño, usabilidad y fiabilidad [11].

3.2.1 Etapas de la metodología

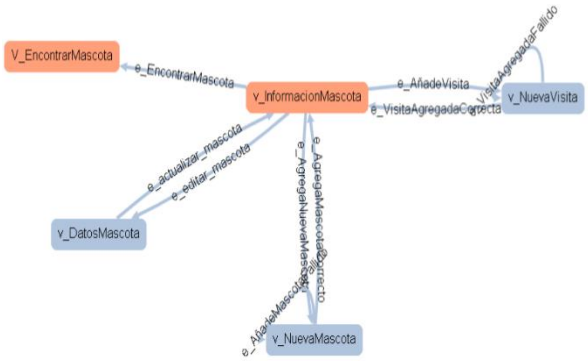
La metodología de investigación se dividió en 4 fases: revisión de la documentación de las herramientas, desarrollo de casos prueba, experimentación y finalmente el análisis de resultados. Durante la primera fase se investigó el funcionamiento de las herramientas, así como toda la literatura relevante sobre la técnica de MBT, una vez definidos los conceptos se inició la fase de desarrollo de casos de prueba en la cual se construyó 10 casos prueba por cada iteración de pruebas, los cuales contienen los pasos a seguir para desarrollar la prueba correspondiente, así como los detalles del escenario de prueba web o de escritorio según corresponda y el modelo de prueba a ser ejecutado tal como se detalla en la figura 1, se realizaron 3 iteraciones en total cambiando los casos prueba en cada uno para simular diferentes escenarios, a continuación, se detallará como se construyeron los modelos de prueba.

Figura 1: Jerarquía de pruebas.



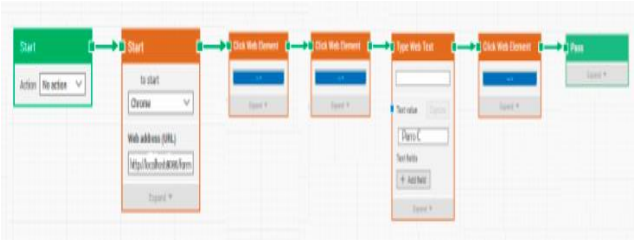
GraphWalker posee su propio editor de modelos de prueba por lo que se puede construir y determinar las condiciones de cada nodo del modelo a ser utilizado en la respectiva prueba funcional del software a ser testeado, tal como se puede observar en la Figura 2, el modelo de prueba está conformado por vértices (representados por rectángulos) y arcos (representados por flechas), los cuales representan estados del software y eventos respectivamente.

Figura 2: Modelo utilizado en GraphWalker



A su vez LeapWork presenta una interfaz gráfica que facilita la creación y configuración del modelo de prueba, al igual que GraphWalker los vértices (representados por rectángulos) y arcos (representados por flechas) representan estados y eventos, tal como se puede observar en la Figura 3.

Figura 3: Modelo utilizado en LeapWork.



Durante la fase de experimentación las dos herramientas MBT fueron ejecutadas en dos máquinas virtuales las cuales fueron levantadas en Azure con las siguientes características que se explican en la Tabla 2.

Tabla 2: Características de las máquinas virtuales levantadas en Azure.

Imagen	Disco	Tamaño	vCPU	RAM
Windows 10 pro-versión 21h2	SSD estándar 128GB	B4ms	4	8GB

Se llevaron a cabo los casos de prueba y con las herramientas de testeo sobre los diferentes escenarios de prueba en base a los modelos definidos, a su vez se realizó la recolección y medición de datos, los cuales fueron obtenidos del informe de resultados de las respectivas herramientas y en base a los atributos antes mencionados, en cada uno de dichos atributos se seleccionó 2 métricas las cuales se detallarán a continuación:

Para el atributo de funcionalidad se seleccionó como métricas de evaluación: Completitud funcional la cual es relevante debido a que permite medir el grado en el que el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario [15], como segunda métrica se decidió construir una métrica relevante al tema de estudio la cual fue definida como generación de casos de prueba, la cual tiene como objetivo medir las capacidades de crear, configurar y validar un modelo de prueba.

Las métricas consideradas útiles para el atributo de desempeño fueron las siguientes: tiempo de respuesta y utilización de recursos ya

que estas dos métricas son sumamente importantes para la industria del testeo de software [12], debido a que el tiempo y el consumo de recursos es crítico para cualquier producto de software.

Con respecto al atributo de usabilidad se determinó que las métricas: aprendizaje y accesibilidad son adecuadas para esta investigación debido a que el aprendizaje está totalmente ligado a mejorar la experiencia del usuario y reducir la curva de aprendizaje, con respecto a la métrica de accesibilidad es relevante mencionar que la inclusión de todo tipo de usuarios hoy en día es sumamente importante[13], ya que un software debe tener la capacidad de ser usado por los usuarios con capacidades especiales, en este caso aquellos que estén involucrados en el testeo de software basado en modelos.

Para el atributo de fiabilidad se seleccionaron las métricas: frecuencia de error y tolerancia de fallos, mismos que son sumamente importantes para la experiencia del usuario y brindarle la mayor estabilidad y fiabilidad [14]. Para concluir la fase de análisis de resultados se construyeron dos matrices de comparación a partir de la cual se realizó la discusión y se obtuvieron las respectivas conclusiones.

3.2.2 Establecimiento de las metas

Las metas de evaluación fueron establecidas a partir de otros estudios que han planteado valores con las métricas que se definen en esta investigación exceptuando la métrica planteada por los autores del presente trabajo, para el atributo de funcionalidad se determinó su fórmula y meta de evaluación en base al estudio de [15], en donde el autor plantea un marco de referencia de medición de la métrica de completitud funcional en la cual se busca medir cuantas funciones realmente están completas con respecto a las que ofrece la herramienta, la métrica construida por los autores de esta investigación surge de la necesidad de poder medir la capacidad de las herramientas para generar lo modelos de prueba, por lo cual su medición será una revisión de las características: editor de modelos de prueba, validación de modelos de prueba, posee multi formato de modelos de prueba, cada una de estas características brindara un punto hasta obtener un total de 3 lo cual será la puntuación máxima o meta en este caso. En el atributo de desempeño se tomó como meta alcanzar el tiempo de ejecución promedio determinado con la configuración Chrome/java en la investigación de [16], así como la carga sobre la memoria RAM, ya que es la misma configuración utilizada en la presente investigación, la meta de usabilidad fue tomada en base a la investigación de [17],en la cual determina características relevantes para la usabilidad de un producto de software así como la respectiva forma de medición de la métrica, finalmente para el atributo de fiabilidad específicamente para la métrica de frecuencia de error cuya meta ideal es que tienda a 0, el método de evaluación fue determinado en base al estudio de [18]. Como se puede observar en la tabla 3, se detalló los atributos y sus métricas, a su vez se muestra a detalle la formula y la descripción de sus variables, en el caso de la meta también se incluye su respectiva descripción y valor objetivo, adicionalmente los atributos que fueron medidos de manera automática están representados de color azul mientras que los de recolección manual de color amarillo.

4. Resultados y análisis

Para la evaluación de estas herramientas se construyeron dos matrices en las que se detallan los resultados de cada escenario de pruebas, como se puede observar en las tablas 4 y 5 las cuales representan los escenarios web y de escritorio respectivamente, se encuentran divididas en las 3 iteraciones con sus correspondientes herramientas y cruzan con los resultados de las métricas medidas en cada iteración y de cada herramienta, cabe mencionar que se utiliza el color verde como un distintivo para resaltar que herramienta estuvo más cerca de la meta de evaluación, mientras que el color rojo es para resaltar la falta total de características en una métrica, el atributo de funcionalidad de las herramientas resultó ser uno de los puntos más fuertes ya que en su completitud funcional tuvieron un puntaje cercano a la meta y demostraron ser herramientas adecuadas para su

objetivo principal, brindando funciones útiles y relevantes como un plugin de integración a proyectos en java en el caso de GraphWalker, también tiene la capacidad de agregarse a en APIS o web services, en el caso de Leadwork puede integrase con diversas tecnologías como:

Tabla 3: Descripción de atributos de evaluación

ATRIBUTO	Métrica	Meta	Descripción de meta	Formula	Medición
Funcionalidad (calculo manual)	Completitud funcional	1	Donde 1 es el valor óptimo de evaluación	$\frac{X}{Y}$	X =Funciones exitosas Y= Funciones totales
	Generación de casos de prueba	3	Donde 3 es el valor óptimo de evaluación	$X+Y+Z$	X=Editor de modelos de prueba Y=Validación de modelos de prueba Z=Posee multi formato de modelos de prueba
Desempeño (cálculo automático)	Tiempo de respuesta	X<=208.91s	Donde 208.91 es el tiempo óptimo en segundos de ejecución	X	X = Tiempo de ejecución.
	Utilización de recursos	X<=1586,74 Mb	Donde 1586,74 es la cantidad optima de consumo de memoria RAM	X	X = Mb de uso de memoria RAM
Usabilidad (cálculo manual)	Aprendizaje	3	Donde 3 es el valor óptimo de evaluación	$X+Y+Z$	X=Panel de ayuda Y= Soporte técnico Z=tooltips
	Accesibilidad	3	Donde 3 es el valor óptimo de evaluación	$X+Y+Z$	X = Modo daltónico Y= Soporte a usuarios con discapacidad visual Z= Multi lenguaje
Fiabilidad (cálculo manual y automático)	Frecuencia de error	0	Donde 0 es la frecuencia oprima de errores	$\frac{X}{Y}$	X = Número de errores Y = cantidad de ejecuciones
	Tolerancia a fallos	si	Donde sí, significa que, si posee tolerancia a fallos, adicionalmente se detallará el número de veces que la herramienta se recuperó de un fallo	$\sum X1+X2+X3\ldots+Xn$	Capacidad de seguir con la operación a pesar de haber fallos en la herramienta y generar una excepción detallando el error

Dynamics365, SAP, Salesforce, Office365, Oracle etc, en la generación de casos de prueba hay que destacar que LeapWorks solo acepta su propio formato de archivo de prueba, a diferencia de GraphWalker el cual si permite varios formatos de archivos como: Json o Uml. El desempeño de las herramientas se vio impactado en el escenario web debido a que el navegador tiene una carga extra en el hardware, por lo cual se puede ver una diferencia en el tiempo de ejecución y utilización de recursos entre los dos escenarios, cabe destacar que GraphWalker al realizar las pruebas más rápido que LeapWorks también tiene un uso de recursos más elevado, el atributo de usabilidad es uno de los puntos más bajos para la herramienta de GraphWalker ya que su curva de aprendizaje es muy marcada con respecto a la de LeapWorks, porque esta carece de mensajes de ayuda estilo tooltips que brinde una introducción o guía sobre las herramientas al usuario y tampoco posee un panel de ayuda, la única forma de tener una introducción o referencia para el uso de la herramienta es el foro oficial, a diferencia de LeapWorks la cual si ofrece estas características, tal como se puede observar en las matrices de comparación ninguna de las dos herramientas posee opciones de accesibilidad, por lo que no serían de ayuda para usuarios especializados en testeo de software que estén interesados en la técnica MBT con capacidades especiales. La fiabilidad a diferencia de la de usabilidad es una de los atributos más sólidos para ambas herramientas, ya que como se observa en las matrices las herramientas tienden a tener pocos fallos, así como una gran recuperación de los mismos, al poder terminar sus procesos a pesar de haber sufrido un error y comunicarle al usuario cual fue la excepción que sufrió la herramienta, es importante mencionar que GraphWalker depende de plugins externos al equipo de desarrollo de la herramienta como el plugin de Java Maven, el cual si no está en la versión correcta puede causar errores de compatibilidad, estas matrices tienen como objetivo servir al lector como una referencia de las características de estas herramientas en los diferentes escenarios propuestos, ya sea para la industria o para futuras investigaciones.

A continuación, se muestra las matrices de comparación:

Tabla 4: Matriz comparativa aplicación web

Matriz comparativa aplicación web							
Atributo	**Métrica**	**Interacción 1**		**Interacción 2**		**Interacción 3**	
		Graph Walker	Leapworks	Graph Walker	Leapworks	Graph Walker	Leapworks
Funcionalidad	**Completitud funcional**	0.6	0.8	0.6	0.8	0.6	0.8
	Generación de casos de prueba	3	2	3	2	3	2
Desempeño	**Tiempo de respuesta**	145.02	152.17	149.87	155.89	143.26	150.10
	Utilización de recursos	1488.31	1251.03	1492.02	1249.36	1505.36	1247.42
Usabilidad	**Aprendizaje**	1	3	1	3	1	3
	Accesibilidad	0	0	0	0	0	0
Fiabilidad	**Frecuencia de error**	0.2	0.1	0.1	0.2	0.1	0
	Tolerancia a fallos	2/Si	1/Si	1/Si	2/Si	1/Si	0/Si

Tabla 5: Matriz comparativa aplicación de escritorio

Matriz comparativa aplicación de escritorio							
Atributo	**Métrica**	**Interacción 1**		**Interacción 2**		**Interacción 3**	
		Graph Walker	Leapworks	Graph Walker	Leapworks	Graph Walker	Leapworks
Funcionalidad	**Completitud funcional**	0.6	0.8	0.6	0.8	0.6	0.8
	Generación de casos de prueba	3	2	3	2	3	2
Desempeño	**Tiempo de respuesta**	136.02	147.99	138.55	149.46	131.91	145.2
	Utilización de recursos	1356.44	1165.36	1349.89	1173.89	1369.71	1170.92
Usabilidad	**Aprendizaje**	1	3	1	3	1	3
	Accesibilidad	0	0	0	0	0	0
Fiabilidad	**Frecuencia de error**	0.1	0.1	0.2	0.1	0.1	0
	Tolerancia a fallos	1 / Si	1 / Si	2 / Si	1 / Si	1 / Si	0 / Si

5. Conclusiones

Como se puede observar en cuanto al rendimiento siendo la aplicación de escritorio la que tiene mejor rendimiento respecto a la aplicación web, tanto en tiempo de ejecución como en la utilización de recursos, esto es debido a la utilización de los navegadores web que pueden consumir muchos recursos en hardware limitado, la accesibilidad es un punto muy importante a detallar ya que ambas herramientas no poseen dichas opciones, esto puede llegar a ser notable debido a la gran diversidad de usuarios especializados que existe hoy en día, también se puede destacar la capacidad de tolerancia a fallos que ambas herramientas poseen, ya que estas herramientas terminan sus procesos y detallan el error que hubo durante la ejecución, aun que como se puede observar las fallas de estas aplicaciones son muy poco comunes por lo que se puede decir que son herramientas bastante fiables.

Se pudo determinar que Leapwork mientras se llevo a cabo esta investigación tuvo soporte continuo por parte de la empresa desarrolladora, así como una notable optimización de recursos de hardware, junto con una interfaz gráfica amigable e intuitiva, además de ofrecer una gran cantidad de funciones relevantes y bien enfocadas para su objetivo principal, lo cual la convierten en una gran opción para usuarios sin mucha experiencia en esta técnica de testeo. A su vez Graphwalker tiene una comunidad detrás la cual brinda apoyo a sus integrantes, como lo es el foro oficial de la herramienta en el cual sus mismos desarrolladores intervienen en la resolución de problemas o respondiendo preguntas, otro punto a destacar son sus diferentes repositorios donde ofrecen ejemplos y tutoriales, así como el código fuente de la herramienta para que la comunidad contribuya en su mejora continua y corrección de errores, otra gran característica de esta herramienta es la facilidad con la que se puede ejecutar las pruebas ya que basta con ejecutar un par de líneas de código, cabe destacar que esta herramienta cuenta con una interfaz hibrida, el diseño de los modelos de prueba se construyen en una interfaz gráfica, mientras que la ejecución de las mismas se realiza mediante el cuadro de diálogo del sistema, lo cual puede llegar a causar confusión o problemas a usuarios con poca experiencia, uno de los problemas con los cuenta esta herramienta es la compatibilidad con pluggins externos ya que debido a estos la configuración inicial puede llegar a ser muy compleja, a diferencia de Leapworks cuya instalación es muy fácil e intuitiva.

MBT al ser una técnica de testeo emergente se podría decir que sus herramientas aún no han madurado lo suficiente como herramientas de testeo, pero están ganando un gran número de seguidores y empresas que han comenzado a utilizarlas, debido a que el tiempo de la fase de testeo funcional en los proyectos de desarrollo de software se reduce en gran medida gracias a esta técnica, pero al no ser esta de gran difusión el acceso a la información con respecto a la misma por el momento es limitado.

De igual manera, aunque el objetivo principal de este artículo no es dar un juicio de valor acerca de cuál herramienta de testeo basadas en modelos es mejor, se puede concluir que la herramienta LeapWork tiene más capacidades en ciertos atributos, principalmente en la optimización de sus recursos, así como en el soporte que brinda la empresa lo cual es crucial para entender mejor las capacidades de la herramienta, además de ser una gran opción para usuarios sin mucha experiencia en MBT y sin grandes conocimientos en programación e integración de servicios.

Referencias

Artículos de revistas:

[1] Joshi.V. (8 de Febreo del 2021). Pruebas basadas en modelos: la nueva era de la automatización de software. [Online] Available https://cynoteck.com/es/blog-post/modelbased-testing-the-new-era-of-software-automation/

[2] Villalobos-Arias, Leonardo, Christian Quesada-López, Alexandra Martinez, and Marcelo Jenkins. "Model-based testing areas, tools and challenges: A tertiary study." *CLEI Electronic Journal* 22, no. 1 (2019).

[3] Sivanandan, Sandeep. "Agile development cycle: Approach to design an effective Model

Based Testing with Behaviour driven automation framework." In *20th Annual International Conference on Advanced Computing and Communications (ADCOM)*, pp. 22-25. IEEE, 2014.

[4] LeapWork (2018). Visión general de la arquitectura. [Online] Available https://normas-apa.org/referencias/citar-pagina-web/

[5] de Cleva Farto, Guilherme, and Andre Takeshi Endo. "Evaluating the model-based testing approach in the context of mobile applications." *Electronic notes in Theoretical computer science* 314 (2015): 3-21.

[6] Michael, James Bret, Bernard J. Bossuyt, and Byron B. Snyder. "Metrics for measuring the effectiveness of software-testing tools." In *13th International Symposium on Software Reliability Engineering, 2002. Proceedings.*, pp. 117-128. IEEE, 2002.

[7] Zafar, Muhammad Nouman, Wasif Afzal, Eduard Enoiu, Athanasios Stratis, Aitor Arrieta, and Goiuria Sagardui. "Model-based testing in practice: An industrial case study using GraphWalker." In *14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference)*, pp. 1-11. 2021.

[8] Arrieta, Aitor, and Goiuria Sagardui. "Model-Based Testing in Practice: An Industrial Case Study using GraphWalker." (2021).

[9] Koivisto, Hannu, and Mikko Salmenperä. "TESTING OF VALMET DNA MACHINE MONITORING APPLICATION." (2021).

[10] Martinez, Diana Estefania Hernandez, and Agnieszka Roginska. "Augmented reality tempo control tool for conducting students." In *149th Audio Engineering Society Convention 2020, AES 2020*. 2020.

[11] Krouska, Akrivi, Christos Troussas, and Maria Virvou. "A literature review of Social Networking-based Learning Systems using a novel ISO-based framework." *Intelligent Decision Technologies* 13, no. 1 (2019): 23-39.

[12] Gao, Kaiye. "Simulated software testing process and its optimization considering heterogeneous debuggers and release time." *IEEE Access* 9 (2021): 38649-38659.

[13] Miranda, Michele, and Rafael Prikladnicki. "Towards a model for Managing Diversity and Inclusion in Software Development Teams." In *Proceedings of the 34th Brazilian Symposium on Software Engineering*, pp. 325-331. 2020.

[14] Li, Qiuying, and Hoang Pham. "A generalized software reliability growth model with consideration of the uncertainty of operating environments." *IEEE Access* 7 (2019): 84253-84267.

[15] Debnath, Narayan, Carlos Salgado, Mario Peralta, Daniel Riesco, Luis Roqué, Germán Montejano, and Mouna Mazzi. "A Software Testing Strategy Based on a Software Product Quality Model." In *International Conference Europe Middle East & North Africa Information Systems and Technologies to Support Learning*, pp. 248-259. Springer, Cham, 2019.

[16] Kuutila, Miikka. "Benchmarking configurations for web-testing-Selenium versus Watir." PhD diss., Faculty of Information Technology and Electrical Engineering, University of Oulu, 2016.

[17] Sánchez, Bertha Alice Naranjo, María José Tinoco Arichavala, and Daniel Enrique Vega Bravo. "Análisis de la usabilidad del sistema web de terapias cognitivas sanamentics." *Boletín Redipe* 9, no. 5 (2020): 175-187.

[18] Jatain, Aman, and Yukti Mehta. "Metrics and models for Software Reliability: A systematic review." In *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, pp. 210-214. IEEE, 2014.

[19] Cubillos Rodríguez, Andrés Leonardo. "Pruebas de software basadas en modelos aplicadas en la generación automatizada de casos de prueba sobre interfaces gráficas de usuario." *Ingeniería de Sistemas* (2012).

[20] Xirgo, Lluis Ribas, "Estructura básica de un computador" *El procesador como generalización de las máquinas algorítmicas*, pp. 8-17. 2012.

[21] Walkinshaw, Neil, Ramsay Taylor, and John Derrick. "Inferring extended finite state machine models from software executions." Empirical Software Engineering 21.3 (2016): 3-13.

[22] Iso25000. ISO/IEC 25010. [Online] Available https://iso25000.com/index.php/normas-iso-25000/iso-25010.