ISSN 1870-4069

Investigación sobre técnicas y estrategias de prueba del software: un estudio de mapeo sistemático sobre la última década

Naomi García, Julio C. Díaz, Raúl A. Aguilar

Universidad Autónoma de Yucatán,
Facultad de Matemáticas, Mérida,
México

A17016302@alumnos.uady.mx, {julio.diaz, avera}@correo.uady.mx

Resumen. Pruebas de Software, es un área de conocimiento de la Ingeniería de Software que aborda aspectos vinculados con la mejora de la calidad, en el contexto del proceso. El presente estudio tiene como objetivo ofrecer una visión general de la investigación desarrollada en la última década, con base en las peguntas de investigación formuladas, con la intención de identificar áreas de oportunidad para continuar con estudios primarios o secundarios en temáticas específicas. Se realizó un Estudio de Mapeo Sistemático mediante el cual se seleccionaron 123 estudios primarios, los cuales fueron analizados para identificar la frecuencia y el tipo de investigaciones realizadas en la última década; el estudio plantea el análisis de aspectos como la innovación en cuanto a técnicas de prueba, la frecuencia en las que los estudios mencionan las estrategias y niveles de prueba, así como los mecanismos de evaluación utilizados en el contexto de la investigación empírica. Los hallazgos indican una amplia variedad de factores de calidad analizados, de manera que las técnicas y estrategias varían dependiendo del tipo de software, el área al que se aplica, entre otros factores. El estudio realizado permitió concluir que la investigación en el ámbito de pruebas del software es aún vigente en el contexto de la Ingeniería de Software, por lo que resulta pertinente continuar la investigación en temáticas específicas mediante estudios tanto primarios, como secundarios.

Palabras clave: Calidad del software, pruebas del software, validación, verificación.

Research on Software Testing Techniques and Strategies: A Systematic Mapping Study over the Last Decade

Abstract. Software Testing is an area of knowledge of Software Engineering that addresses aspects related to quality improvement, in the context of the process. The objective of this study is to offer an overview of the research developed in the last decade, based on the research questions formulated, with the intention of identifying areas of opportunity to continue with primary or

secondary studies on specific topics. A Systematic Mapping Study was carried out through which 123 primary studies were selected, which were analyzed to identify the frequency and type of research carried out in the last decade; The study proposes the analysis of aspects such as innovation in terms of testing techniques, the frequency with which studies mention testing strategies and levels, as well as the evaluation mechanisms used in the context of empirical research. The findings indicate a wide variety of quality factors analyzed, so that the techniques and strategies vary depending on the type of software, the area to which it is applied, among other factors. The study carried out allowed us to conclude that research in the field of software testing is still valid in the context of Software Engineering, so it is pertinent to continue research on specific topics through both primary and secondary studies.

Keywords: Software quality, software testing, validation, verification.

1. Introducción

La Ingeniería de Software (IS) dispone desde hace casi un par de décadas, de un cuerpo de conocimientos reconocido por académicos y profesionistas vinculados con la disciplina; la parte medular de la misma integra diez áreas de conocimiento relacionadas con los procesos de desarrollo y de gestión del software, las cuales fueron documentadas en 2004 [1] y actualizadas en 2014 [2]. Una de las áreas de desarrollo directamente vinculadas con el proceso de mejora a la calidad del software, es la de Pruebas de Software; en este sentido, encontrar un proceso de desarrollo software que satisfaga las necesidades reales del cliente y con ello generar software de mejor calidad, ha sido uno de los objetivos de la IS desde el momento de su concepción como disciplina, para ello, los cuestionamientos: (1) ¿Cómo saber si el producto construido funciona correctamente? y (2) ¿Cómo saber si el producto corresponde a lo que el cliente necesitaba? Han tratado de ser atendidos con la investigación vinculada a las técnicas y/o estrategias para verificación y validación del software [3].

De acuerdo con [4] el proceso pruebas tiene como objetivo directo el revelar defectos en el software, y de manera indirecta medir el grado de calidad que posee, con respecto a un conjunto de atributos seleccionados. Una primera postura sobre las pruebas del software establece que estas representan un proceso de verificación dinámica de los comportamientos esperados del software, en función de un conjunto de casos seleccionados [2], y con ello, la no adecuación de dichos comportamientos generaría fallos en el sistema.

Una segunda postura un tanto más flexible considera que hay muchas formas de evaluar, o en su caso, probar un sistema sin requerir ejecutarlo [5]; por ejemplo, las revisiones son un tipo de técnica de prueba que se puede usar para verificar la calidad de un artefacto de software, como puede ser un documento de especificación de requisitos o un listado de código; dicha técnica permite identificar problemas o faltas en dichos artefactos. Ambas posturas, tanto la del análisis dinámico, como la del estático, representan maneras de dar respuesta al primer cuestionamiento antes citado.

Por otro lado, el segundo cuestionamiento que involucra al cliente, es de carácter más subjetivo, y se vincula con el proceso de validación del software, dicho proceso es

definido como aquel que se encarga de la evaluación del software durante o al final del ciclo de desarrollo, para determinar si satisface los requisitos especificados y acordados con el cliente [6].

Durante mucho tiempo se afirmaba que realizar pruebas al software era una actividad que debía ocurrir al final del proceso de desarrollo, no obstante, esta perspectiva ha cambiado en las últimas décadas, en virtud de que se ha demostrado que el realizar pruebas en etapas tempranas del desarrollo evita encontrar defectos en el software al momento de su entrega, permitiendo así, desarrollar un software de mejor calidad.

En relación con el enfoque abordado por la estrategia de prueba, existen dos maneras: (1) Conociendo el funcionamiento interno del software, se pueden diseñar y ejecutar pruebas para asegurar que las operaciones internas se realicen de acuerdo con las especificaciones y que todos los componentes internos se hayan ejercitado adecuadamente; y (2) Conociendo la función especificada para la que se ha diseñado el software, se pueden diseñar y ejecutar pruebas que demuestren que cada función es completamente operativa, al mismo tiempo que se identifican fallos en cada función.

En el primer enfoque las pruebas requieren una vista interna del software y son conocidas como pruebas de caja blanca; mientras que el segundo, las pruebas tienen una visión externa y son conocidas como pruebas de caja negra [7].

Así mismo, en cuanto al nivel de la prueba, es posible distinguir cuatro niveles en función del objeto de prueba: (1) pruebas unitarias, (2) pruebas de integración, (3) pruebas del sistema y algún tipo de (4) pruebas de aceptación [8]. Es importante mencionar que tanto las pruebas de caja negra, como las de caja blanca, pueden clasificarse con base en lo que evalúan, dividiéndose así en pruebas funcionales y no funcionales.

Por un lado, tenemos las pruebas funcionales que han sido mencionado anteriormente (unitarias, integración, sistema y aceptación) y, por otro, tenemos las pruebas no funcionales que evalúan cierto tipo de comportamiento en el software como la compatibilidad, seguridad, estrés, usabilidad, rendimiento, entre otros.

Como se ha mencionado anteriormente, el proceso de realización de pruebas de software se realiza con el objetivo de asegurar la calidad del producto final, y es por ello que esta actividad considera un conjunto de factores de calidad [9] que deben evaluarse como la usabilidad, mantenibilidad, seguridad, portabilidad, eficiencia, capacidad de prueba, flexibilidad, fiabilidad, robusticidad, por mencionar algunas. La evaluación de dichos factores pretende verificar que el software a entregar no sólo cumpla con la parte funcional acordada, sino también, con aquellos requisitos no funcionales acordados con los clientes.

Con la intención de explorar áreas de oportunidad para continuar la investigación en el área de Pruebas del Software, los autores se plantearon el desarrollo de un estudio secundario —un mapeo sistemático de literatura— para identificar y clasificar las principales características de la investigación realizada en el área de pruebas de software en la última década; se acordó analizar estudios primarios publicados entre 2010 y 2021, con la intención de actualizar en cierta medida el estado del arte estructurado en el Swebok [1, 2], pero sobre todo, evaluar la pertinencia de continuar la investigación con un estudio primario, o incluso secundario sobre algún tópico particular —una revisión sistemática de literatura.

En la siguiente sección se describe la metodología utilizada para la realización del estudio reportado; la sección tres presenta las tareas desarrolladas para el planeamiento

del estudio; en la sección cuatro se citan las tareas realizadas para la obtención de los 123 estudios primarios finalmente seleccionados; la quinta sección da respuesta a las preguntas de investigación que conducen el estudio; finalmente, en la última sección se hace un análisis de la información recopilada y se reflexiona en torno al propósito planteado para el estudio.

2. Metodología

Con el propósito de recopilar resultados empíricos publicados a lo largo de los últimos años en el área de pruebas del software, y con ello caracterizar la investigación realizada, se optó por realizar un Mapeo Sistemático de Literatura, conocido también como Estudio de Mapeo (EM); dicho tipo de estudio secundario tiene como objetivo, explorar y proporcionar una visión global sobre un área de investigación, lo que permite identificar nichos de oportunidad para estudios primarios, o en su caso, estudios secundarios más específicos, como pudiera ser una revisión sistemática de literatura; de acuerdo con [10] en los EM es común clasificar los hallazgos obtenidos según algún esquema de clasificación predefinido.

Para el desarrollo del presente estudio se utilizó como referencia la guía propuesta en [11] en la cual se establecen las siguientes tareas:

1 Formulación de las preguntas de investigación: El objetivo principal de los EM es proporcionar una descripción general de un área de investigación e identificar la cantidad, tipo de investigación y resultados disponibles dentro de ella.

2 Búsqueda de estudios primarios: Los estudios primarios se identifican mediante el uso de cadenas de búsqueda en Bases de Datos científicas o navegando manualmente a través de actas de conferencias o publicaciones de revistas relevantes.

3 Selección de artículos relevantes: Se aplican un conjunto de criterios de inclusión y exclusión para determinar la elegibilidad de los artículos primarios que serán analizados.

4 Definición de un esquema de clasificación: Los investigadores revisan los resúmenes y buscan palabras clave y conceptos que permitan identificar un esquema de clasificación.

5 Extracción de datos y elaboración del reporte: Cuando se tiene el esquema de clasificación, los artículos relevantes se clasifican en el esquema, es decir, se lleva a cabo la extracción de datos reales y se procede con análisis de los resultados, presentando las frecuencias de las publicaciones para cada categoría, lo anterior hace posible ver qué categorías se han enfatizado en investigaciones anteriores y, por lo tanto, identificar brechas y posibilidades para investigaciones futuras.

Tabla 1. Categorías identificadas para el análisis de los estudios primarios seleccionados.

Aspecto	Categoría
PI02. Innovación.	Nuevo, Mejora, Existente.
PI03. Estrategia de prueba.	Caja Negra, Caja Blanca, Mixto.
PI03. Nivel de la prueba.	Unitaria, Integración, Sistema, Aceptación.
PI04. Mecanismo de validación.	Caso de Estudio, Experimento, Análisis Comparativo.
PI05. Contexto.	Industria, Academia.

3. Planeación del estudio

La revisión del estado del arte plasmada en el SWEBOK y el análisis de algunos materiales considerados como referencias obligadas para el área de pruebas del software [4, 5, 9], permitieron acumular información suficiente para la formulación de las preguntas de investigación.

3.1. Preguntas de investigación

Las preguntas de investigación que guiaron la realización del estudio son:

- PI01 ¿Cuál es la distribución en la última década de los estudios primarios publicados sobre pruebas del software?
- PI02 ¿Cuáles son los niveles de prueba del software abordados por los estudios primarios en la última década?
- PI03 ¿Qué tipo de pruebas de software han sido las más reportadas en la última década?
- PI04 ¿Cuáles son los factores de calidad que se han evaluado en el software durante la última década?
- PI05 ¿Cuáles son las características de las estrategias utilizadas en las pruebas de software que han sido reportadas en la última década?
- PI06 ¿Cuáles han sido los métodos de validación utilizados en la última década por los estudios primarios vinculados con el área de pruebas del software?
- PI07 ¿En qué contexto han sido desarrollados los estudios primarios vinculados con el área de pruebas del software durante la última década?

La revisión del estado del arte, el análisis de algunos materiales considerados como literatura gris para el área de pruebas del software, y la formulación de las preguntas de investigación, sirvieron de base para la realización de un análisis PICOC [12] del área bajo estudio, dicho análisis permitió identificar un conjunto aspectos clave para asistir en el análisis posterior de los estudios seleccionados.

- Población: Software.

Tabla 2. Proceso de selección de estudios primarios.

BD	Fase 1	Fase 2	Fase 3
Google Scholar	105	95	82
IEEE Xplore	44	41	41
		Total	***123***

- Intervención: Técnicas o Estrategias de prueba del Software.
- Comparación: Calidad del Software.
- Resultados: Mejoramiento del desempeño y rendimiento del software.
- Contexto: Industria o Academia.

Con el análisis PICOC y las preguntas de investigación formuladas, se identificaron posibles categorías que permitiesen al investigador clasificar la información resultante del análisis de los estudios primarios que resulten seleccionados (ver Tabla 1).

3.2. Bases de datos

De acuerdo con el objetivo de la investigación propuesta, para el Mapeo Sistemático se seleccionaron las siguientes Bases de Datos (BD):

- Google Scholar: una BD de artículos académicos (de una amplia gama de fuentes, pero principalmente artículos de revistas, actas de congresos, informes técnicos y disertaciones) sin restricciones de idioma, revistas o geográficas, lo cual permite acceder a literatura que no está disponible en otras BD. Si bien dicha base de datos no asegura la disponibilidad de los textos de los artículos, en el caso de nuestro estudio, es suficiente el acceso a los resúmenes de los mismos.
- IEEE *Xplore*: una BD de investigación académica en las áreas de Ciencias de la Computación, Ingeniería Eléctrica y Electrónica. Si bien los artículos de esta BD son de acceso restringido, los resúmenes son de libre acceso.

3.3. Cadena de búsqueda

Con base en los resultados obtenidos con la metodología PICOC, se elaboró la siguiente cadena genérica de búsqueda para realizar el rastreo:

("software testing" AND "software development") AND "testing software"

La cadena fue configurada en función de los repositorios considerados; las cadenas de búsqueda resultantes permitieron realizar un rastreo en los repositorios de *Google Scholar* e *IEEE Xplore* para obtener información de los últimos 10 años con respecto a las pruebas de software de manera más precisa; dichas cadenas fueron:

Google Scholar: *allintitle: Software AND Testing AND (Types OR Techniques OR Strategies).*

IEEE Xplore: *("Document Title":Software Testing) AND (("Abstract":Techniques) OR ("Abstract":Types) OR ("Abstract":Strategies)).*

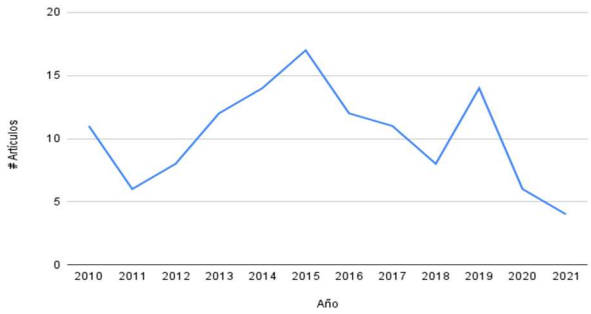

Fig. 1. Frecuencia de estudios seleccionados por año de publicación.

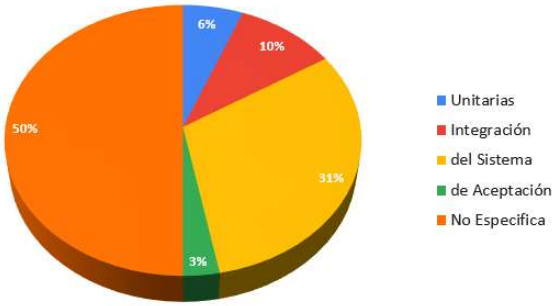

Fig. 2. Número de estudios primarios seleccionados de acuerdo con el criterio de nivel de prueba.

3.4. Criterios de inclusión y exclusión

Para filtrar la búsqueda y reducir los resultados únicamente a aquellos documentos relevantes para el objeto de investigación planteado, el cual permitiese dar respuesta a las preguntas de investigación, se definieron y utilizaron algunos criterios de inclusión y exclusión; estos criterios permitieron obtener una mejor organización de los resultados obtenidos en las búsquedas realizadas en este estudio.

Criterios de Inclusión:

- Publicaciones entre 2010 y 2021.
- Artículos publicados en revistas.
- Artículos con títulos y resúmenes en idioma inglés.

Criterios de Exclusión:

- Artículos de difusión.
- Artículos duplicados.

Tabla 3. Estudios primarios seleccionados de acuerdo al tipo de prueba reportado.

Nombre	#
Adaptativas	1
de Automatización	13
de Concurrencia	1
de Regresión	5
Escalabilidad	1
Estrés	1
Mantenibilidad	2
No Especifica	14
Rendimiento	73
Seguridad	6
Usabilidad	5

4. Ejecución del estudio

El estudio reportado fue realizado durante el mes de julio de 2021, en dicho estudio se filtraron los resultados recuperados de las búsquedas en los repositorios de *Google Scholar* e *IEEE Xplore* para excluir aquellos resultados que no estén alineados con los objetivos del mapeo sistemático de acuerdo con las tres fases definidas para el estudio:

- Fase 1: Aplicación de la cadena de búsqueda en las BD seleccionadas.
- Fase 2: Aplicación de los criterios de inclusión al conjunto de estudios obtenidos en la fase 1.
- Fase 3: Aplicación de los criterios de exclusión al conjunto de estudios obtenidos en la fase 2.

La tabla 2 presenta el número de estudios obtenidos al final de cada fase del proceso de selección.

Durante las fases anteriormente mencionadas, se descartaron un total de 26 artículos de ambas bases de datos, esto porque algunos no estaban relacionados con el tema de pruebas de software, no se apegaban a los criterios mencionados o estaban en otro idioma desde su descripción. Así pues, se obtuvo un total de 123 artículos reportados que están directamente relacionados con las pruebas de software en la última década, esto incluyendo investigaciones sobre su importancia e innovación en el área, así como experimentos y estudios de caso tanto en el ámbito industrial como en el académico. De cada estudio seleccionado se extrajo información con base en el esquema de clasificación predefinido, analizando de manera minuciosa el resumen de cada uno de los artículos seleccionados, lo anterior, en virtud del objetivo y tiempo disponible para el estudio.

5. Resultados

A continuación, se plantean los resultados obtenidos del análisis de los 123 estudios primarios seleccionados de las dos bases de datos, 82 de Google Scholar [13]-[94] y 41 de IEEE Xplore [95]-[135]. Resulta importante mencionar que se observó que buena parte de estudios analizados no proporcionaban información de manera tan detallada como para clasificarlos en una de las categorías identificadas, por lo que en algunos casos como los niveles y estrategias de prueba, se incluyó la categoría de "No Especifica".

PI01 ¿Cuál es la distribución en la última década de los estudios primarios publicados sobre pruebas del software?

La figura 1 ilustra el número de publicaciones realizadas por año en la última década; como puede observarse, en la primera mitad de la década hubo una tendencia creciente de publicaciones, no obstante —con excepción de 2019— en los últimos cinco años, la tendencia cambio a decreciente, aunque es claro que 2020 y 2021 han sido años atípicos en todas las actividades del ser humano —y la investigación no ha ido la excepción— debido a la pandemia del COVID-19.

PI02 ¿Cuáles son los niveles de prueba del software abordados por los estudios primarios en la última década?

De acuerdo con las categorías consideradas (Unitarias, Integración, del Sistema, de Aceptación), se obtuvo un total de treinta y ocho artículos que se orientaban a la investigación de pruebas del sistema, mientras que únicamente doce se orientaban a pruebas de integración, siete a unitarias y cuatro al nivel de pruebas de aceptación (ver figura 2). Resulta importante destacar la mayoría de los estudios analizados (62) fueron ubicados en la categoría de "No Especifica" dado a que en el resumen del artículo no se proporcionaba la clasificación de manera específica.

PI03 ¿Qué tipo de pruebas de software han sido las más reportadas en la última década?

Como puede observarse en la tabla 3, a pesar de que la gran mayoría de los artículos reportados fueron clasificados como pruebas de software que se orientan al rendimiento del mismo, existen otros que están clasificados dentro de la categoría de pruebas automatizadas que en su mayoría son mejoras o propuestas nuevas; por otro lado, existe un subconjunto de estudios primarios reportan otro tipo de pruebas, aunque en menor número.

PI04 ¿Cuáles son los factores de calidad que se han evaluado en el software durante la última década?

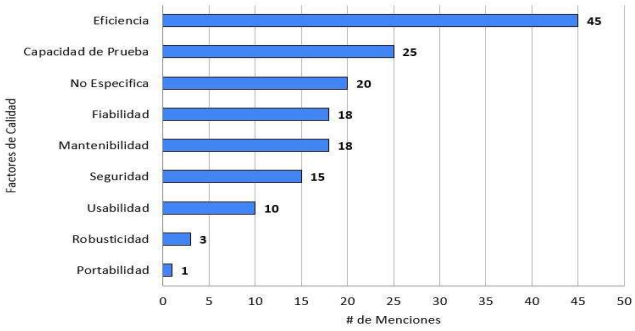

Fig. 3. Estudios primarios seleccionados de acuerdo con el factor de calidad abordado.

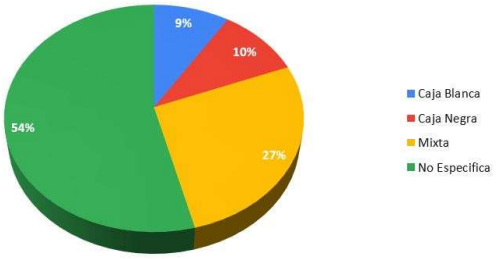

Fig. 4. Estudios primarios seleccionados de acuerdo con la estrategia de prueba seleccionada.

En relación con los factores o criterios de calidad identificados, la figura 3 ilustra la distribución por factor de calidad identificado en el estudio, por cada uno del ciento veinte y tres. Es importante mencionar que en al menos diez artículos únicamente se seleccionó un factor de calidad, mientras que en el resto era común ver más de un criterio de calidad y combinar algunos como la seguridad, capacidad de prueba, robusticidad y eficiencia. Así pues, los factores con mayor número de menciones, como puede observarse en la figura, fueron la eficiencia y la capacidad de prueba, mientras que la portabilidad y robusticidad fueron factores poco evaluados a lo largo de los artículos analizados.

Cabe destacar que el aspecto de la calidad en las pruebas denota la importancia y preocupación que existe con respecto a los aspectos de software que más se ven afectados por la falta de pruebas, investigación e innovación en el campo, así pues, como se observa en la figura, la mayoría de los estudios caen dentro de aspectos como la eficiencia, capacidad de prueba, fiabilidad y mantenibilidad; estas categorías podrían ser consideradas como primordiales para asegurar la calidad del software, lo cual confirma lo que anteriormente se pensaba con respecto a la preocupación que existe por asegurar dicha calidad en los productos en la última década.

PI05 ¿Cuáles son las características de las estrategias utilizadas en las pruebas de software que han sido reportadas en la última década?

Con el análisis de los estudios secundarios, se obtuvo que treinta y tres de los resultados aplicaban una estrategia de prueba mixta que incluía el uso de estrategias de

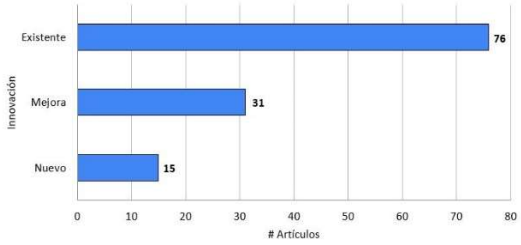

Fig. 5. Número de estudios primarios seleccionados de acuerdo con el nivel de innovación documentado.

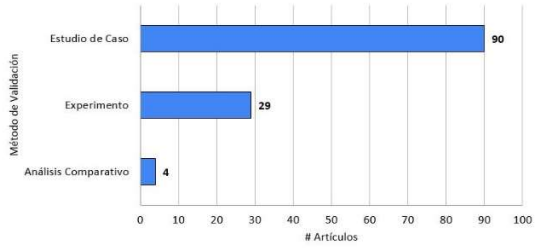

Fig. 6. Frecuencia en los Métodos de validación utilizados por los estudios primarios seleccionados.

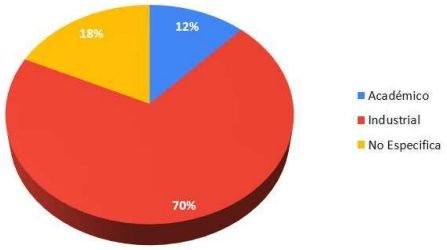

Fig. 7. Estudios primarios en función del contexto del estudio.

caja blanca y de caja negra, mientras que doce resultados reportaron aplicar estrategias de caja negra y once de caja blanca (Véase la figura 4).

Cabe destacar que sesenta y siete artículos no lograron clasificarse dado que no se mencionaba de manera explícita el tipo de estrategia de prueba seleccionado.

Por su parte, en relación con el nivel de innovación de la técnica, la figura 5 ilustra los resultados del análisis a los estudios seleccionados; se pudo contabilizar que setenta y seis de los artículos se enfocaron en técnicas o metodologías ya existentes, treinta y uno en mejoras y quince en nuevas propuestas para el área de pruebas.

Únicamente uno de los artículos no logró ser clasificado, sin embargo, se pudo observar que los pocos artículos con técnicas "nuevas" se comenzaron a volver más constantes a partir del 2015.

PI06 ¿Cuáles han sido los métodos de validación utilizados en la última década por los estudios primarios vinculados con el área de pruebas del software?

De análisis a los ciento veintitrés estudios seleccionados se pudo contabilizar los dos métodos de validación recurridos son los estudios de caso (73%) y los experimentos (29%), dos de los métodos de investigación empírica más citados en la literatura [136].

De igual forma, se hallaron cuatro artículos considerados análisis comparativos, el cual es un método de investigación que permite, como su nombre lo dice, una técnica o metodología con otra. Así pues, la figura 6 ilustra la información antes reportada.

PI07 ¿En qué contexto han sido desarrollados los estudios primarios vinculados con el área de pruebas del software durante la última década?

En relación con el contexto en el que se realizaron los estudios primarios seleccionados, se pudo observar que la mayoría se desarrolla en el ámbito industrial (86 artículos), no obstante, un alto porcentaje de éstos hacía hincapié en la investigación con respecto al ámbito de pruebas sin importar el contexto como tal (No Especificado: 22) y solo quince trabajos se reportan en el ámbito académico (ver figura 7).

Finalmente, es importante también mencionar que se obtuvieron pocos estudios de aplicación de pruebas de software en sistemas reales o experimentos aplicados al área industrial en software latente, lo cual es un indicador de que aún en la última década todavía no se profundiza en la investigación de pruebas de software en general, sobre todo para sistemas que requieren niveles altos de calidad o seguridad.

6. Conclusiones y trabajos futuros

Los Estudios de Mapeo son estudios con un alcance amplio, dado que su principal objetivo es presentar una visión global sobre un tema de interés e identificar la cantidad y tipo de investigación y resultados disponibles sobre el mismo. Con el estudio realizado en el ámbito de la Ingeniería de Software, en particular sobre el área de Pruebas de Software, se pudo observar que existe un bajo porcentaje de investigación en la que se proponen técnicas nuevas, sin embargo, la investigación se ha mantenido constante —aunque con altibajos— a lo largo de la última década.

Del análisis de los estudios primarios seleccionados se pudo identificar que las técnicas de tipo funcional son las más analizadas y que los aspectos de eficiencia y capacidad de prueba, son los criterios de calidad a los que se le ha prestado mayor atención. En cuanto a las estrategias de prueba, nos llamó la atención el uso de esquemas mixtos, es decir, una combinación de pruebas de caja blanca y de caja negra. Por otro lado, un aspecto que despertó nuestra curiosidad fue que la mayoría de los tipos de prueba fueron dirigidas al rendimiento del producto, pero también se centraban en técnicas no tan usadas que son importantes como las pruebas automatizadas.

En cuanto a los métodos de investigación empírica utilizados, pudimos observar que los estudios de caso, así como los experimentos, son los más recurridos. Finalmente, en cuanto al contexto en el que se realizaron los estudios, en esta muestra de 123 trabajos seleccionados, el industrial fue en el que se ubicaron la mayoría de los estudios analizados.

Otro de los hallazgos interesantes, es que se identificaron algunos estudios relacionados con dominios que comúnmente no se mencionan en la gran mayoría de los estudios sobre pruebas de software, nos referimos a estudios en el ámbito de

pruebas en sistemas embebidos, pruebas de seguridad en sistemas de aviación o en microprocesadores y pruebas en aplicaciones móviles; dichos artículos llamarón la atención de los autores ya que, aunque no todos buscaban innovar en alguna técnica o estrategia, analizaban temas que no son comunes en el área de pruebas y por tanto pudieran ser nichos de oportunidad para la investigación en el área de las pruebas del software.

Resulta importante destacar también, que en la actualidad, los sistemas software comienzan a abordar problemáticas diferentes a las que fueron abordadas en el siglo pasado, las soluciones comienzan a incorporar técnicas de inteligencia artificial, abordan grandes volúmenes de datos, y requieren de mejores esquemas de seguridad; por tanto, requieren de técnicas novedosas de prueba para evaluar y asegurar la calidad de los mismos.

Finalmente, los autores consideran que en el ámbito de las pruebas de software, existen problemáticas que pueden abordarse por primera vez en un futuro mediato, por lo que resulta un área de oportunidad para continuar con un estudio primario, como es estudio tipo encuesta propuesto en [137] en torno a evaluar la madurez de las prácticas de prueba; o en su caso, con un estudio secundario de mayor especificidad, como podría ser una revisión sistemática de literatura sobre las características de las pruebas de software que incorporan técnicas de Inteligencia Artificial.

Referencias

1. Abran, A., Moore J.: SWEBOK - Guide to the software engineering body of knowledge. IEEE CS Professional Practices Committee (2004)
2. Bourque, P. Fairley, R.: Guide to the software engineering body of knowledge. IEEE Computer Society Press (2014)
3. Boehm, B.: Software engineering economics. Prentice Hall (1981)
4. Burnstein, I.: Practical software testing: A process-oriented approach. Springer (2003)
5. Hetzel, B: A complete guide to software testing, QED Information Sciences, Inc (1988)
6. IEEE Standard glossary of software engineering terminology. Standard Coordinating Committee of the Computer Society. IEEE (1990)
7. Pressman, R., Maxim, B.: Software engineering: A practitioner's approach. 8th edn. McGraw-Hill Publishing (2015)
8. Sánchez Alonso, S., Sicilia Urban, M. Á., Rodriguez García, Daniel: Ingeniería del software: Un enfoque desde la perspectiva del SWEBOK. Alfaomega (2012)
9. Laporte, C., April, A.: Software quality models. Software Quality Assurance, 1st edn (2018)
10. Genero, M. Cruz-Lemus, J. Piattini, M.: Métodos de investigación en ingeniería de software (2014)
11. Petersen, K. Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: Proceedings 12th International Conference on Evaluation and Assessment in Software Engineering (2008) doi: 10.14236/ewic/EASE2008.8
12. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Evidence Based Software Engineering (2007)
13. Adamsen, C.: Automated testing techniques for event-driven and dynamically typed software applications. Doctoral Thesis, Aarhus University (2018)
14. Akour, M., Falah, B., A., Bouriat, S., Alemerien, K.: Mobile software testing: Thoughts, strategies, challenges, and experimental study. International Journal of Advanced Computer Science and Applications, vol. 7, no. 6, pp. 12–19 (2016) doi: 10.14569/ijacsa.2016.070602

15. Alharthi, A. S.: Software Testing of Mobile Applications: Techniques and Challenges. SDIWC Organization (2014)
16. Sadia, A., Yaser, H., Shariq, H., Shunkun, Y.: Enhanced regression testing technique for agile software development and continuous integration strategies. Software Quality Journal, vol. 28, pp. 1–27 (2020)
17. Alkawaz, M. H., Silvarajoo, A.: A survey on test case prioritization and optimization techniques in software regression testing. In: IEEE 7th Conference on Systems, Process and Control (ICSPC). IEEE, pp. 59–64 (2019)
18. Alnafjan, K., Hussain, T., Ullah, H., Paracha, Z.: Comparative analysis and evaluation of software vulnerabilities testing techniques. International Journal of Computer and Information Engineering, vol. 7, no. 6, pp. 687–692 (2013)
19. Alnafjan, K., Hussain, T., Khan, G.F., Ullah, H., Alghamdi, A.S.: Using AHP to compare and evaluate software securuty testing techniques. Acta Press.
20. Alsaedi, O.: Improving student skills on software testing techniques and team coordination using a zero-fidelity collaborative simulation game. Tesis Doctoral, (2019) doi: 10.2316/P.2013.796-022
21. Alzubaidy, L., Laheeb, M., Alharid, B.: Proposed software testing using intelligent techniques. Intelligent Water Drop and Ant Colony Optimization Algorithm (2013)
22. Bagchi, T.: Statistical models for software defects and testing strategies. Association for Computing Machinery, vol. 34, no. 2 (2009) doi: 10.1145/1507 195.1507202
23. Omri, F. N.: Weighted statistical testing based on active learning and formal verification techniques for software reliability assessment. Thesis Doctoral in Karlsruher Instituts für Technologie (2018)
24. Bhargava, D., Veda, A.: A different techniques and strategies for software testing. International Journal of Engineering & research technology, vol. 2, no. 12 (2013)
25. Fleischer, C., Sauer, D. U., Barreras, J. V., Schaltz, E., Christensen, A. E.: Development of software and strategies for battery management system testing on HIL simulator. In: 11th International Conference on Ecological Vehicles and Renewable Energies, pp.1–12 (2016)
26. Chaudhary, S.: Latest software testing tools and techniques: A review. International Journal, vol. 7, no. 5 (2017)
27. Chauhan, R. K., Singh, I.: Latest research and development on software testing techniques and tools. International Journal of Current Engineering and Technology, vol. 4, no 4, pp. 2368–2372 (2014)
28. Chen, Z.: Program inspection and testing techniques for code clones and refactorings in evolving software. Tesis Doctoral, University of Nebraska at Omaha (2017)
29. Cotroneo, D., Pietrantuono, R., Russo, S.: Testing techniques selection based on ODC fault types and software metrics. Journal of Systems and Software, vol. 86, no 6, pp.1613–1637 (2013)
30. Gautam, S., Nagpal, B.: Descriptive study of software testing & testing tolos. International Journal of Innovative Research in Computer and Communication Engineering, vol. 4, no. 6 (2016)
31. Dams, G. & Rajadoral, K.: Evaluating testing strategies and processes in software development and porting software applications. International Journal of Software Engineering, vol.9, no. 2 (2016)
32. de Olivera, I., de Souza, S.: Study and definition of project attributes for selection of testing techniques for concurrent software. In: Anais Estendidos do X Congresso Brasileiro de Software: Teoria e Prática, SBC, pp. 24–30 (2019)
33. Deak, A., Stalhane, T., Sindre, G.: Challenges and strategies for motivating software testing personnel. Information and software Technology, vol. 73, pp. 1–15 (2016)
34. Deligiannis, P.: Scalable techniques for analysing and testing asynchronous software systems. Thesis of Imperial College London (2017)

35. Dhiman, R.: Design and implemetartion of software testing techniques in cloud generating effective test cases for element based systems. Journal on Recent Innovation in Cloud Computing, Virtualization and Web Applications, vol. 2, no. 2 (2018)
36. Du, S. Y., Wang, J. S., Chen, Z. W., Ai, D. M.: Testing techniques of the software network interface based on the capture and analysis of the packet. Applied Mechanics and Materials. Trans Tech Publications, pp. 1786–1791 (2013)
37. Farooq, S. U., Quadri, S.: Empirical evaluation of software testing techniques–need, issues and mitigation. Software Engineering: An international Journal, vol. 3, no. 3, pp. 41–51 (2013)
38. Fefei, M.: Constraint solving techniques for software testing and analysis (2010) doi: 10.1145/1810295.1810407
39. Gómez, O. S., Cortés Verdín, M. K., Pardo, C.: Efficiency of software testing techniques: A controlled experiment replication and network meta-analysis. e-Informatica Software Engineering Journal, vol. 11, no. 1 (2017)
40. Hooda, I., Chhillar, R.: Software test process, testing types and techniques. International Journal of Computer Applications, vol. 111, no. 3 (2015) doi: 10.5120/19597-1433
41. Hussain, T., Singh, S., Motilal, S.: A comparative study of software testing techniques viz. white box testing black box testing and grey box testing, pp. 2350–1294 (2015)
42. Santos, I. Souza, S. Melo, S.: Extended Abstract - CTDSI/CTCCSI 2021 - Study and definition of project attributes for selection of testing techniques for concurrent software. In: Anais Estendidos do XVII Simpósio Brasileiro de Sistemas de Informação, pp. 103–105 (2021)
43. Isha, S.: Software testing techniques and strategieg. Department of Computer Science. SBMNE College (2014)
44. Jain, M., Gopalani, D.: Aspect oriented programming and types of software testing. In: Second International Conference on Computational Intelligence & Communication Technology (CICT). IEEE, pp. 64–69 (2016)
45. Jan, S., Shah Ullah, S. T., Johar, Z., Shah, Y., Khan, F.: An innovative approach to investigate various software testing techniques and strategies. International Journal of Scientific Research in Science, Engineering and Technology, vol. 2, no. 2, pp. 682–689 (2016)
46. Anil Job, M.: Automating and optimizing software testing using artificial intelligence techniques. International Journal of Advanced Computer Science and Applications, vol. 12, no. 5 (2021)
47. Kapur, P. K., Singh, O., Shrivastava, A., Singh, Jyotish, N. P.: A software up-gradation model with testing effort and two types of imperfect debugging. In: Proceedings of International Conference on Futuristic Trends in Computational Analysis and Knowledge Management, pp. 613–618 (2015)
48. Kapur, P. K. Singh, O., Shrivastava, A.: Optimal price and testing time of a software under warranty and two types of imperfect debugging. Journal of System Assurance Engineering and Management, vol. 5, no. 2, pp. 120–126 (2014)
49. Jovanovic, I.: Software testing methods and techniques, The IPSI BgD Transactions on Internet Research, pp. 30 (2009)
50. Kolay, P., Simha, P. V: Entry and sustainable growth strategies for firms offering independent software testing services (2010)
51. Kooli, M., Kaddachi, F., DiNatale, G., Bosio, A., Benoit, P., Torres, L.: Computing reliability: On the differences between software testing and software fault injection techniques. Microprocessors and Microsystems, vol. 50, pp. 102–112 (2017)
52. Sawant, A., Bari, P., Chawan, P.: Software testing techniques and strategies. International Journal of Engineering Research and Applications(IJERA), vol. 2, no. 2, pp. 113–117 (2016)

53. Sharma, M., Kumar, V., Kumari, M., Rd, Y.: Text data generation technique for object oriented software with comparison among black box testing and white box testing techniques (2016)
54. Lazic, L.: Application example of triz and taguchi"s robust design techniques to software testing. vol 10 (2018)
55. Lopez-Herrejon, R. E., Ferrer, J., Chicano, F., Egyed, A., Alba, E.: Comparative analysis of classical multi-objective evolutionary algorithms and seeding strategies for pairwise testing of software product lines. In: IEEE Congress on Evolutionary Computation (CEC), IEEE, pp. 387–396 (2014) doi: 10.1109/CEC.2014.6900473
56. Feifei, M.: Constraint solving techniques for software testing and analysis. In: International Conference on Software Engineering. IEEE, pp. 417–420 (2010)
57. Madhu, B., Jigalur, M., Lokesha, V.: A study on agile software testing: Emergence and techniques. African Journal of Mathematics and Computer Science Research, vol. 3, no. 11, pp. 288–289 (2010)
58. Malik, S.: Software testing: Essential phase of SDLC and a comparative study of software testing techniques. International Journal of System and Software Engineering, vol. 5, no. 2, pp. 38–45 (2017)
59. Manikandan, L. C., Selvakumar, R. K.: A Study on software process framework and testing techniques. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, pp. 94–101 (2019)
60. Melton, R.: System level integration and test leveraging software unit testing techniques. In: 29th Aerospace Testing Seminar (2015)
61. Mwambe, O.: Selection and application of software testing techniques to specific conditions of software projects. International Journal of Computer Applications, vol. 975, pp. 8887 (2013)
62. Nehra, E.: A review paper on software testing techniques and tools. ZENITH International Journal of Multidisciplinary Research, vol. 9, no. 5, pp. 129–139 (2019)
63. Oliveira Neto, F. G., Torkar, R., Machado, P. D.: An initiative to improve reproducibility and empirical evaluation of software testing techniques. In: 37th International Conference on Software Engineering - New Ideas and Emerging Results, IEEE, pp. 575–578 (2015) doi: 10.1109/ICSE.2015.197
64. Ortiz, F.: Scientific test and analysis techniques for software testing. Scientific Test and Analysis Techniques Center of Excellence (STAT COE), vol. 17 (2015)
65. Patidar, R., Sharma, A. Dave, R.: Survey on manual and automation testing strategies and tools for a software application. International Journal of Advanced Research in Computer Science and Software Engineering, vol. 7, no. 44 (2017)
66. Patil, M.: The overview of software testing: types, methods, and levels. Journal of Software Engineering Tools & Technology Trends, vol. 8, no. 1, pp. 11–14 (2021)
67. Pitchford, M.: Embedded software quality, integration, and testing techniques. Software Engineering for Embedded Systems. Newnes. pp. 269–338 (2019)
68. Poonam, P. D.: Software testing strategies and methodologies. International Journal of Advanced Research Trends in Engineering and Technology, vol. 3, no. 4 (2016)
69. Razia, A., Uma, K., Sairamesh, L. Kannan, A.: Improvement on software testing techniques and tools. Journal of Advanced Research in Dynamical and Control Systems, vol. 9, no. 2, pp. 1049–1058 (2017)
70. Reddy G. V., Chandrasekhar, A.: Tools and techniques for testing of flight critical software. Defence science journal, vol. 49, no. 4, pp. 317–322 (2013)
71. Rexhepi B., Rexhepi, A.: Software testing techniques and principles. Knowledge International Journal, vol. 28, no. 4, pp. 1383–1387 (2018)
72. Sancheti, V., Sharma, G. S.: An initiative to improve quality of software testing techniques and calculating total number of failures using bayesian method. International Journal of New Technology and Research, vol. 4, no. 6, pp. 95–97 (2018)

73. Santos, I.: Study and definition of project attributes for selection of testing techniques for concurrent software. Tesis Doctoral, Universidade de São Paulo (2019)
74. Sathyavathy, V.: Software testing techniques for embedded systems and applications. International Journal of Scientific Research in Computer Science Applications and Management Studies, vol. 7, no. 3 (2018)
75. Schneidewind, N.: Software testing and reliability strategies. Journal of Aerospace Computing, Information, and Communication, vol. 7, no. 9, pp. 294–307 (2010)
76. Selvaprita, P. B.: Different software testing strategies and techniques. International Journal of Science and Modern Engineering, vol. 2, no. 1, pp. 2319–6386 (2013)
77. Sharma, S., Magow, R., Kathpal, S., Jatain, A.: Software testing techniques and experimental research drawn from inferences. International Journal of Advanced Engineering and Global Technology, vol. 2, no. 1 (2014)
78. Shuaibu, I., Machina, M., Muazzamu, I.: Investigation onto the software testing techniques and tools: An evaluation and comparative analysis. International Journal of Computer Applications, vol. 177, no. 23, pp. 8887 (2019)
79. Sneha, K., Malle, G. M.: Research on software testing techniques and software automation testing tools. In: International Conference on Energy, Communication, Data Analytics and Soft Computing. IEEE, pp. 77–81 (2017)
80. Subashini, B., Sundaravadivazhagan, B., Ashik, M.: Amalgamation and characterization for test Suite enhancement in software testing using data mining techniques. Materials Today: Proceedings (2021)
81. Subhiyakto E. R., Utomo, D. W.: Software testing techniques and strategies use in novice software teams. SISFO, vol 5, no. 5 (2016)
82. Saha T., Palit, R.: Practices of software testing techniques and tools in Bangladesh software industry. In: IEEE Asia-Pacific Conference on Computer Science and Data Engineering, pp. 1–10 (2019)
83. Taley M., Pathak, B.: Comprehensive study of software testing techniques and strategies: a review international journal of engineering and technical research, vol. 9, no. 8 (2020)
84. Thakare, S., Chavan, S., Chawan, P. M.: Software testing strategies and techniques. International Journal of Emerging Technology and Advanced Engineering, vol. 2, pp. 980–986 (2012)
85. Umar, M. A.: A study of software testing: categories, levels, techniques, and types. TechRxiv. Preprint (2020) doi: 10.36227/techrxiv.12578714.v1
86. Umar, M. A.: Comprehensive study of software testing: Categories, levels, techniques, and types. International Journal of Advance Research, Ideas and Innovations in Technology, vol. 5, no. 6, pp. 32–40 (2019)
87. Viksne, K.: Desktop software testing techniques and tools. Thesis of Institute of Information Technology (2011)
88. Saavnet, K. V., Jasuna, V.: The pragmatic review on code coverage and analysis techniques in software testing. International Journal of Computing and Corporate Research, vol. 5, no. 3 (2015)
89. Vos, T., Marin, B., Escalona, M. J., Marchetto, A.: A methodological framework for evaluating software testing techniques and tools. In: 2th international conference on quality software, IEEE, pp. 230–239 (2012)
90. Vos, T., Marín, B., Panach, I., Baars, A., Ayala, C., Franch, X.: Evaluating software testing techniques and tolos. In: Proceedings of Jornadas de Ingeniería del Software y Bases de Datos, pp. 531–536 (2011)
91. Xu, L.: The research of software testing techniques of software major and applied curriculum standards. Liaoning Higher Vocational Technical Institute Journal, vol. 9 (2011)
92. Yu, N. T.: Utilization and deployment of software testing tools and techniques. International Journal of Trend in Research and Development, vol. 6, no. 1 (2019)

93. Zhang, H. C.: Research on new techniques and development trend of software testing. Advanced materials research, Trans Tech Publications Ltd, pp. 1298–1301 (2013)
94. Żukowicz, M.: Software testing: rationale for teaching and creating test strategies using the conclusions drawn from the "No free lunch" Thory. General and Professional Education, vol. 3, pp. 61–70 (2015)
95. Arcuri, A.: A theoretical and empirical analysis of the role of test sequence length in software testing for structural coverage. IEEE Transactions on Software Engineering, vol. 38, no. 3, pp. 497–519 (2012)
96. Krieg, A., Preschern, C., Grinschgl, J., Steger, C., Kreiner, C., Weiss, R., Bock, H., Haid, J.: Power and fault emulation for software verification and system stability testing in safety critical environments. IEEE Transactions on Industrial Informatics, vol. 9, no. 2, pp. 1199–1206 (2013)
97. Zachariah, B.: Analysis of software testing strategies through attained failure size. IEEE Transactions on Reliability, vol. 61, no. 2, pp. 569–579 (2012)
98. Henard, C., Papadakis, M., Perrouin, G., Klein, J., Heymans, P., Le Traon, Y.: Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test configurations for software product lines. IEEE Transactions on SE, vol. 40, no. 7, pp. 650–670 (2014)
99. Hayden, C. M., Smith, E. K., Hardisty, E. A., Hicks, M., Foster, J. S.: Evaluating dynamic software update safety using systematic testing. IEEE Transactions on Software Engineering, vol. 38, no. 6, pp. 1340–1354 (2012)
100. Martinez, D., Celeita, D., Clavijo, D., Ramos, G.: Hardware and Software Integration as a Realist SCADA Environment to Test Protective Relaying Control. IEEE Transactions on Industry Applications, vol. 54, no. 2, pp. 1208–1217 (2018)
101. Cotroneo, D., Pietrantuono R., Russo, S.: RELAI Testing: A technique to assess and improve software reliability. IEEE Transactions on Software Engineering, vol. 42, no. 5, pp. 452–475 (2016)
102. Sabena, D., Reorda M. S., Sterpone, L.: On the automatic generation of optimized software-based self-test programs for VLIW processors. IEEE Trans. on Very Large Scale Integration Systems, vol. 22, no. 4, pp. 813–823 (2014)
103. Xu, D., Xu, W., Kent, M., Thomas, L., Wang, L.: An automated test generation technique for software quality assurance. IEEE Transactions on Reliability, vol. 64, no. 1, pp. 247–268 (2015)
104. Barr, E. T., Harman, M., McMinn, P., Shahbaz, M., Yoo, S.: The oracle problem in software testing: A survey. IEEE Transactions on Software Engineering, vol. 41, no. 5, pp. 507–525 (2015)
105. Tang, E., Zhang, X., Muller, N. T., Chen, Z., Li, X.: Software numerical instability detection and diagnosis by combining stochastic and infinite-precision testing. IEEE Transactions on Software Engineering, vol. 43, no. 10, pp. 975–994 (2017)
106. Uzuncaova, E., Khurshid, S., Batory, D.: Incremental test generation for software product lines. IEEE Transactions on Software Engineering, vol. 36, no. 3, pp. 309–322 (2010)
107. Bianchi, F. A., Margara, A., Pezzè, M.: A survey of recent trends in testing concurrent software systems. IEEE Transactions on Software Engineering, vol. 44, no. 8, pp. 747–783 (2018)
108. Theodorou, G., Kranitis, N., Paschalis, A., Gizopoulos, D.: Software-Based self-test for small caches in microprocessors. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 33, no. 12, pp. 1991–2004 (2014)
109. Itkonen, J., Mäntylä, V., Lassenius, C.: The role of the tester's knowledge in exploratory software testing. IEEE Transactions on Software Engineering, vol. 39, no. 5, pp. 707–724 (2013)

110. Yin, J., Cai, K.: On the asymptotic behavior of adaptive testing strategy for software reliability assessment. IEEE Transactions on Software Engineering, vol. 40, no. 4, pp. 396–412 (2014)
111. Hu, J. Cai, K. Y., Chen, T. Y.: Adaptive and random partition software testing. IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 44, no. 12, pp. 1649– 1664 (2014)
112. Fiondella, L., Gokhale, S. S.: Optimal allocation of testing effort considering software architecture. IEEE Transactions on Reliability, vol. 61, no. 2, pp. 580–589 (2012)
113. Skitsas, M. A., Nicopoulos, C. A., Michael, M. K.: DaemonGuard: Enabling O/S-Orchestrated Fine-Grained Software-Based Selective-Testing in Multi-/Many-Core Microprocessors. IEEE Transactions on Computers, vol. 65, no. 5, pp. 1453–1466 (2016)
114. Böhme, M., Paul, S.: A probabilistic analysis of the efficiency of automated software testing. IEEE Transactions on Software Engineering, vol. 42, no. 4, pp. 345–360 (2016)
115. Jamro, M.: POU-Oriented unit testing of IEC 61131-3 control software. IEEE Transactions on Industrial Informatics, vol. 11, no. 5, pp. 1119–1129 (2015)
116. Khatibsyarbini, M., Isa, M. A., Jawawi, D. N., Hamed, H. N., Mohamed Suffian, M. D.: Test case prioritization using firefly algorithm for software testing. IEEE Access, vol. 7, pp. 132360–132373 (2019)
117. Kim, M., Kim Y., Kim, H.: A comparative study of software model checkers as unit testing tools: An industrial case study. IEEE Transactions on Software Engineering, vol. 37, no. 2, pp. 146–160 (2011)
118. Khan, O., Kundu, S.: Hardware/Software codesign architecture for online testing in chip multiprocessors. IEEE Transactions on Dependable and Secure Computing, vol. 8, no. 5, pp. 714–727 (2011)
119. Bernardi, P., Cantoro, R., De Luca, S., Sanchez, E., Sansonetti, A., Squillero, G.: Software-Based Self-Test Techniques for Dual-Issue embedded processors. IEEE Transactions on Emerging Topics in Computing, vol. 8, no. 2, pp. 464–477 (2020)
120. Georgiou, P., Kavousianos, X., Cantoro, R., Reorda, M. S.: Fault-Independent Test-Generation for Software-Based Self-Testing. IEEE Transactions on Device and Materials Reliability, vol. 19, no. 2, pp. 341–349 (2019)
121. Li, Q., Li, H., Lu, M.: Incorporating S-shaped testing-effort functions into NHPP software reliability model with imperfect debugging. Journal of Systems Engineering and Electronics, vol. 26, no. 1, pp. 190–207 (2015)
122. Luo, Q., Moran, K., Zhang, L., Poshyvanyk, D.: How do static and dynamic test case prioritization techniques perform on modern software systems? An extensive study on github projects. IEEE Transactions on Software Engineering, vol. 45, no. 1, pp. 1054–1080 (2019)
123. Duffey, R. B, Fiondella, L.: Software, hardware, and procedure reliability by testing and verification: Evidence of learning trends. IEEE Transactions on Human-Machine Systems, vol. 44, no. 3, pp. 395–405 (2014)
124. Baker, R., Habli, I.: An empirical evaluation of mutation testing for improving the test quality of safety-critical software. IEEE Transactions on Software Engineering, vol. 39, no. 6, pp. 787–805 (2013)
125. Rosero, R., Gómez, O., Rodríguez, G.: Regression testing of database applications under an incremental software development setting. IEEE Access, vol. 5, pp. 18419–18428 (2017)
126. Matias, P., Barbetta, P. A., Trivedi, K. S., Filho, P. J.: Accelerated degradation tests applied to software aging experiments. IEEE Transactions on Reliability, vol. 59, no. 1, pp. 102–114 (2010)
127. Jain, R. P., Poston, R. S., Simon, J. C.: An empirical investigation of client managers' responsibilities in managing offshore outsourcing of software-testing projects. IEEE Transactions on Engineering Management, vol. 58, no. 4, pp. 743–757 (2011)

128. Shah, S., Sundmark, D., Lindström, B. Andler, S. F.: Robustness testing of embedded software systems: An industrial interview study. IEEE Access, vol. 4, pp. 1859–1871 (2016)
129. Yu, T., Huang, Z., Wang, C.: ConTesa: Directed test suite augmentation for concurrent software. IEEE Transactions on Software Engineering, vol. 46, no. 4, pp. 405–419 (2020)
130. Fragal, V. H., Simao, A., Mousavi, M. R., Turker, U. C.: Extending HSI Test Generation Method for Software Product Lines. The Computer Journal, vol. 62, no. 1, pp. 109–129 (2019)
131. Renard, Y., Lotte, F., Gibert, G., Congedo, M., Maby, E., Delannoy, V., Bertrand, O., Lécuyer, A.: OpenViBE: An open-source software platform to design, test and use brain-computer interfaces in real and virtual environments. Presence, vol. 19, no. 1, pp. 35–53 (2010)
132. Zhang, Y., Chakrabarty, K., Peng, Z., Rezine, A., Li, H., Eles, P., Jiang, J.: Software-Based Self-Testing using bounded model checking for out-of-order superscalar processors. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Syst, vol. 39, no. 3, pp. 714–727 (2020)
133. Jiang, Z. M., Hassan, A. E.: A survey on load testing of large-scale software systems. IEEE Transactions on Software Engineering, vol. 41, no. 11, pp. 1091–1118 (2015)
134. Zhou, Z. Q., Xiang, S., Chen, T. Y.: Metamorphic testing for software quality assessment: A study of search engines. IEEE Transactions on Software Engineering, vol. 42, no. 3, pp. 264–284 (2016)
135. Wang, Z., Tang, K., Yao, X.: Multi-Objective approaches to optimal testing resource allocation in modular software systems. IEEE Transactions on Reliability, vol. 59, no. 3, pp. 563–575 (2010)
136. Malhotra, R.: Empirical research in software engineering: Concepts, analysis, and applications. CRC Press (2015)
137. Garousi V., Zhi, J.: A survey of software testing practices in Canada. Journal of Systems and Software, vol. 86, pp. 1354–1376 (2013)