ISSN 2007-9737

Una revisión a la realidad de la automatización de las pruebas del software

Edgar Serna M.[1], Raquel Martínez M.[2], Paula Tamayo O.[2]

[1] Instituto Antioqueño de Investigación, Medellín, Antioquia, Colombia

[2] Institución Universitaria de Envigado, Envigado, Antioquia, Colombia

eserna@eserna.com, raquel.martinez@iue.edu.co, patamayo@correo.iue.edu.co

Resumen. Probar el software es una de las actividades más importantes en el ciclo de vida del desarrollo, pero tradicionalmente se ha llevado a cabo al final del proceso, cuando el producto está terminado y está a punto de ser liberado. La complejidad del software actual exige que la prueba se ejecute de forma paralela al desarrollo, de tal manera que los errores se encuentren a tiempo y se puedan corregir a bajo costo. La automatización de las pruebas surgió como una alternativa para agilizar su ejecución, a la vez que para mejorar la fiabilidad del producto y su calidad. Pero, aunque su estudio e investigación se inició hace casi 50 años, su progreso todavía no satisface la demanda por el mejoramiento de la calidad. En este artículo se presentan los resultados de una revisión integradora de la literatura, que se realizó con el objetivo de establecer una visión general de las experiencias reportadas sobre la automatización de las pruebas. Se encontró que todavía no hay acuerdo acerca de su definición y que las ventajas y limitaciones son casi opiniones. Además, que muchas empresas asumen la automatización como un remplazo total de las pruebas manuales, aunque su nivel de madurez todavía no la permita.

Palabras clave. Fiabilidad del software; calidad del software; casos de prueba; pruebas manuales.

A Review of Reality of Software Test Automation

Abstract. Testing the software is one of the most important activities in the development life cycle, but has traditionally been carried out at the end of the process, when the product is completed and is about to be released. The complexity of today's software requires the test to run in parallel with the development, so that problems are found early and can be corrected at low cost. The test automation emerged as an alternative to speed up its implementation, as well as to improve product reliability and quality. But, although his study and research began almost 50 years ago, progress still does not meet demand for quality improvement. In this article are presented the results of an integrative literature review, conducted with the aim of establishing an overview of the experiences reported on test automation. It was found that there is still no agreement on its definition and the advantages and limitations are almost personal opinions. In addition, many companies assume automation as a total replacement for manual testing, although their level of maturity still not allowed.

Keywords. Software reliability, software quality, test cases, manual testing.

1. Introducción

Aunque en la literatura se discute acerca de la automatización de las pruebas, su definición todavía se concentra en la creación de un mecanismo para automatizar casos de prueba, lo que significa que se utiliza un software especial para ejecutarlos a través de secuencias de comandos preparados, en lugar de un probador. En términos generales, se asume que con la automatización las actividades de pruebas manuales tienden a ser sustituidas por actividades automatizadas [1], lo que significa que no se requiere la participación humana para generar una prueba [2]. A diferencia de las pruebas manuales, en las que en todo el proceso se requiere que un probador que interactúe con cada caso de prueba

para analizarlo y reportar los resultados. Por el contrario, la prueba automatizada consiste en utilizar un software especial para controlar la ejecución de las pruebas y la comparación de los resultados reales con los resultados esperados [3]. Hoffman [4] declara que las pruebas de software se consideran automatizadas cuando un mecanismo libre prueba el funcionamiento de los casos de prueba. En [5], también describen la automatización como la tarea de crear una representación mecánicamente interpretable de un caso de prueba manual. Para Sommerville [6] son casos de prueba que se ejecutan automáticamente para acelerar el proceso de prueba, y una de las razones por las que son importantes es que garantizan la fiabilidad del software, especialmente con las actualizaciones.

Por otro lado, diversos autores e investigadores recalcan la importancia de las pruebas del software. Una de las razones es que a través de ellas es posible detectar los defectos y reducir los riesgos asociados [7]. Además, el aseguramiento de la calidad es un factor crítico para el éxito del desarrollo de sistemas, es la clave para la satisfacción del cliente y tiene un impacto directo en el costo y el desarrollo del producto [8].

Cuando las pruebas se aplicas adecuadamente pueden garantizar que el software y los procesos en el ciclo de vida se ajustan a las necesidades específicas [9]. Pero tradicionalmente se conciben como una entidad separada que funciona independientemente del área de desarrollo, aunque las personas que las llevan a cabo son responsables de la definición del proceso y del seguimiento de los detalles de su ejecución [9]. Debido a su importancia y necesidad, debe ser un procedimiento involucrado en todas las fases del ciclo de vida [10-12]. Por su parte, el proceso de automatización es crucial y aunque las pruebas parezcan rezagadas con respecto al ritmo de escritura del código y el número de líneas que cada programador desarrolla por día, la impresión es que permanece relativamente fijo [13], mientras los reprocesos se incrementan debido a la complejidad de los sistemas actuales.

El término pruebas manuales, al igual que automatizadas, significa algo más que la ejecución de casos de prueba, y la idea de que este proceso podría hacerse sin la intervención humana es una cuestión de amplio debate en la comunidad [33]. Si bien es cierto que los sistemas complejos requieren más de la automatización, mucho software trivial y con interfaces sencillas se puede probar sin estas herramientas, porque el costo es un ítem que se debe tener en cuenta. Por eso es que en todo proyecto de desarrollo se debe tener la posibilidad de elegir entre unas u otras para garantizar la fiabilidad del producto. La automatización acelera el proceso de prueba y una de las razones por las que es importante es que asegura de mejor forma la fiabilidad del software, especialmente cuando se realizan actualizaciones [6]. Otra manera de lograrlo es utilizando pruebas redundantes, lo que disminuye los costos de mantenimiento y asegura la integridad del conjunto de pruebas [14].

Se ha demostrado que una vez que se escriben los casos de prueba es necesario actualizarlos cada vez que se realizan cambios en el software. Si este proceso no se ejecuta con cuidado, la integridad de las suites de prueba puede ser cuestionable. La investigación en este campo es activa y los resultados se difunden en trabajos de todo tipo, en los que se comparan técnicas, herramientas y prácticas de prueba. Para Antonia Bertolino [15] la prueba se ha convertido en una actividad esencial en la Ingeniería de Software, y propone una hoja de ruta que puede ser establecida a través de un conjunto de cuatro objetivos finales y alcanzables. Uno de ellos es la cantidad de pruebas que se debe o puede hacer, donde el factor humano es un recurso fundamental, por lo que la capacidad, el compromiso y la motivación de los probadores pueden hacer una gran diferencia entre el éxito o el fracaso del proceso. Las personas ejecutan pruebas manuales sin necesidad de utilizar ninguna herramienta automatizada o cualquier secuencia de comandos. El probador asume el papel de un usuario final y prueba el software para identificar cualquier comportamiento inesperado del programa.

En este trabajo se presentan los resultados de una investigación cuyo objetivo fue establecer el dinamismo con que se divulga acerca de la automatización de las pruebas. En primer lugar, la idea fue reforzar los conocimientos y la experiencia existentes mediante la identificación de los temas y conceptos investigados hasta el momento; en segundo lugar y con base en esos

temas y conceptos, establecer una visión general de la experiencia existente sobre la automatización. Para lograrlo se identificaron y revisaron los reportes de experiencias y las publicaciones relacionadas, como producto de la primera fase del desarrollo del proyecto de investigación: *Evaluación al estado del arte de la automatización de las pruebas del software*, cuyo objetivo fue establecer el nivel, el alcance y la efectividad de las propuestas para automatizar las pruebas del software.

2. Trabajos relacionados

Dustin, Rashka y Paul [16] afirman que la popularidad de la automatización de las pruebas también se debe a la aplicación de desarrollo rápido de aplicaciones (RAD) y a su popularidad. Para cumplir la mayor cantidad de actividades relacionadas con las pruebas se deben utilizar herramientas automatizadas, porque a menudo las manuales son laboriosas y propensas a errores, y simplemente no pueden competir con la calidad de la prueba automatizada, especialmente cuando hay que tener en cuenta el cronograma del proyecto. Estos autores no tienen en cuenta el costo de la automatización, debido a que su trabajo se centra en productos generalmente pequeños, pero otro punto de vista sería analizarla para productos complejos y críticos, donde la calidad y fiabilidad es el objetivo central.

El tema del trabajo de Antonia Bertolino [15] es la automatización de las pruebas del software, vista como un enfoque de validación extendido en la industria pero que sigue siendo en gran medida *ad hoc*, costoso y con una eficacia impredecible. De hecho, es un término amplio que abarca una variedad de actividades a lo largo del ciclo de vida y más allá y que tiene diferentes objetivos. Por lo tanto, cualquier investigación relacionada con la automatización de las pruebas se enfrenta a una serie de desafíos. La autora propone un plan de trabajo coherente para afrontar los más relevantes, y su punto de partida lo constituyen algunos logros importantes del pasado sustentados en los cuatro objetivos que identifica, pero que siguen siendo sueños inalcanzables.

De acuerdo con Petr Bavin [17], para automatizar las pruebas existen herramientas eficientes y efectivas, pero debido a que su valor es alto tienen que ayudar a realizar actividades de prueba más rápidas y eficaces que las manuales. Entre sus beneficios se encuentra que la información sobre los errores encontrados se puede reutilizar y que la búsqueda puede comenzar mucho antes en el ciclo de vida. Esto es importante porque en el desarrollo de productos software hay que tener en cuenta el tiempo y los aspectos de presupuesto, y el uso de estas herramientas ayuda a acelerar el proceso. El autor no hace una comparación a la efectividad de esas herramientas y se limita a describir la valoración subjetiva que algunos autores publican.

Whyte y Mulder [18] proponen que las pruebas de software son uno de los principales métodos utilizados en la validación y verificación de la producción y desarrollo de software en la industria. La mayoría las ve como un método clave para la consecución de la calidad y la fiabilidad del software y la satisfacción del cliente. Sin embargo, es un proceso costoso que representa alrededor del 50% del valor de desarrollo de sistemas. Debido a que en los últimos años esta disciplina ha estado bajo presión para cumplir con el tiempo, el costo y las limitaciones del producto, es fundamental identificar y aplicar herramientas que reduzcan el impacto negativo de la prueba y que incrementen su eficacia. Las principales conclusiones de este estudio es que no hay un enfoque individual que produzca resultados satisfactorios, pero una combinación de dos o más tipos de pruebas automatizadas podría incrementar la eficacia y mitigar los efectos de sus limitaciones.

Para Dudekula Rafi y sus compañeros [19] existe una brecha documentada entre los puntos de vista académicos y profesionales acerca de las pruebas del software. En su trabajo tratan de cerrarla mediante una investigación a ambos puntos de vista con respecto a los beneficios y las limitaciones de la automatización. Estudian la visión académica mediante una revisión sistemática de la literatura y la profesional la evalúan con una encuesta. En la academia la evidencia publicada es poco profunda y que los beneficios se sustentan a partir de fuentes como experimentos y estudios de casos, mientras que las limitaciones lo hacen a partir de reportes de experiencias. Los beneficios de la automatización

para los profesionales están relacionados con la reusabilidad, la repetitividad, la cobertura y el esfuerzo ahorrado. Las limitaciones son la alta inversión inicial en su configuración, la selección de las herramientas y la capacitación del personal.

Majikes, Pandita y Xie [20] opinan que los dispositivos médicos controlados por software evolucionan desde monolíticos a sistemas médicos modulares cyber-físicos, y que en ellos el software es un componente crítico, se añade mayor complejidad y se incrementa la probabilidad de fallos en los mismos.

En este caso, las pruebas automatizadas se han utilizado como un medio para asegurar la calidad del software, lo que es conveniente para este tipo de sistemas. En su trabajo, presentan una revisión a las técnicas de pruebas automatizadas existentes que exponen fallos en estos dispositivos.

En particular, categorizan las fallas y luego aplican el método de revisión sistemática de la literatura para comparar los estudios existentes contra su categorización. Los autores sugieren mejoras que pueden ayudar a futuros estudios de investigación acerca la automatización de las pruebas para dispositivos médicos complejos.

Para comprender mejor los retos que enfrentan los usuarios y los desarrolladores de pruebas automatizadas, Wiklund, Sundmark, Eldh y Lundqvist [21] realizaron una investigación empírica sobre un panel de discusión que soporta un *framework* de automatización de pruebas, y que tiene cientos de usuarios. Encontraron que gran parte de los problemas y solicitudes de ayuda se relacionan con ambientes TI centralizados y con el comportamiento erróneo en el uso del *framework* y sus componentes.

En cuanto a las solicitudes de ayuda, la mayoría se refiere al diseño de *scripts* de prueba y no a las áreas que parecen ser más problemáticas. A partir de estos resultados observaron una clara necesidad de simplificar el uso, la instalación y la configuración de la prueba automatizada. Encontraron que los problemas detectados sugieren que los probadores necesitan desarrollar habilidades diferentes, lo que históricamente se ha asumido como cubierto.

Por último, proponen nuevas investigaciones acerca de los beneficios de la automatización y de su implementación y uso eficiente en entornos TI.

El propósito del estudio de caso presentado por Dayu Guan [22] es entender el punto de vista de las organizaciones que utilizan pruebas automatizadas, y cómo lo relacionan en la literatura. Este autor describe y analiza los factores clave para tener en cuenta al configurar una prueba automatizada, además de los riesgos que implica. Encontró que la cognición de estas pruebas no se limita solamente a conocer su definición y los beneficios que pueden ser introducidos en la organización, sino que también tienen que centrarse en el ámbito de su aplicación y las condiciones previas. Determinó que es necesario gestionar consideraciones clave, tales como la resistencia de las personas, el proceso de trabajo y entrenamiento y algunas otras.

3. Metodología

Una revisión de la literatura se realiza con el fin de establecer una comprensión del estado de la investigación sobre un área específica. El objetivo es identificar los temas que se han abordado y sistematizar los resultados relacionados. Existen diversas propuestas de clasificación para estas revisiones, pero el tipo que se aplicó en esta investigación es una revisión integradora [23]. De acuerdo con esta propuesta, el proceso consiste en identificar las publicaciones de relevancia, evaluarlas y seleccionarlas para su revisión y sistematización, de tal manera que se puedan identificar y estructurar los conceptos centrales comunes. Por lo tanto, una revisión integradora desarrolla una visión general de la investigación existente, que inicialmente no está relacionada y que servirá de base para futuras investigaciones. Desde este punto de vista se realizan tres actividades: 1) identificar la literatura potencialmente relevante, 2) evaluar la pertinencia y calidad de la literatura seleccionada para el análisis y la síntesis, y 3) analizar y sintetizar la literatura seleccionada.

1. Para la actividad 1 se buscaron los reportes de investigación en las bases de datos en línea. Esto incluyó búsquedas en ScienceDirect, ACM Digital Library, IEEE Explore y Web of Science. En conjunto, estos sitios cubren un amplio rango de revistas científicas, libros y conferencias relacionadas con el área de

interés. También se realizaron búsquedas directas en el motor de búsqueda, porque algunas fuentes pueden no estar indexadas. En todos los casos se realizaron búsquedas utilizando los siguientes términos clave: *Automated software testing, automated testing, test automation, test automation software* y *software testing automation*.

Esta actividad incluyó una búsqueda en la línea de tiempo desde finales de los años 90 hasta el 2015, dado que una publicación requiere entre dos y tres años para que la comunidad la analice y referencie. Se revisaron el título y la fuente (revista, libro o conferencia) con el objetivo de excluir los resultados no-relevantes para el área de interés.

No se tuvieron en cuenta editoriales, prólogos, resúmenes de artículos, entrevistas, noticias, críticas u opiniones, correspondencia, discusiones, comentarios, cartas de los lectores, resúmenes de talleres, paneles y sesiones de posters. El proceso dio como resultado 112 publicaciones.

2. En la actividad 2 se evaluó cada trabajo para asegurar que sí explica o discute el concepto de o el uso de la automatización de las pruebas del software, y que reporta hallazgos empíricos. Con base en una revisión de los resúmenes se eliminaron 42 trabajos, los demás constituyeron la base el estado de la práctica de esta investigación. Durante el estudio del texto completo se eliminaron 17 trabajos más, considerados no-relevante para la revisión, dejando 53 trabajos en total para la síntesis.

3. En la actividad 3 se hizo la síntesis de los trabajos. La colección resultante difiere con respecto a la variación temática, el tipo de estudio y la calidad científica. Se encontró que 18 reportan resultados de estudios empíricos, en el sentido que describen el método de estudio, el análisis y la validez de los resultados.

   El resto de los documentos puede ser clasificado como informes de experiencias o elaboraciones de conceptos o ideas. Para asegurar que no se pasaran por alto las conclusiones y las tendencias potencialmente relevantes en este campo de investigación, se decidió mantener también los trabajos estrictamente no-empíricos, tales como informes de experiencias ligeras.

De esta forma se presenta una visión general del estado actual de la investigación en la automatización de las pruebas, en la que se muestran los temas y conceptos que se han abordado hasta el momento y un resumen de esa experiencia. Aunque en principio es conocimiento útil, es necesario verificarlo y ampliarlo con estudios empíricos. También se identificaron temas y conceptos que poco se han investigado hasta ahora, pero que deben ser incluidos en futuras investigaciones. Esta consideración se basa en parte en lo que describen algunos de los trabajos y en las conversaciones con profesionales en el área. El análisis de la literatura seleccionada se realizó mediante comparación constante, donde los conceptos y temas centrales emergen a través de análisis repetidos de la literatura revisada. Parte de la literatura incluida en la revisión no se pudo clasificar como estudios empíricos rigurosos, sin embargo, dado que la automatización de las pruebas es una práctica en ascenso, también se incluyeron los artículos no-empíricos en los que se valoró la importancia de lo que reportan acerca de la situación actual del tema.

4. Resultados

Como la automatización de las pruebas del software gana popularidad, el cuerpo de la literatura sobre el tema ha ido creciendo de forma constante en los últimos años. Los resultados de esta investigación se estructuran de acuerdo con los tópicos identificados más destacados y relevantes. Esto no quiere decir que la lista de temas que se presenta aquí sea el conjunto final de lo más importante en automatización de las pruebas del software, simplemente refleja los tópicos más investigados y discutidos en la muestra de la literatura revisada. Se encontró que algunos autores le proporcionan a los ingenieros de pruebas una base teórica y práctica para una implementación exitosa de la automatización [24-27]; otros, con amplia experiencia profesional en la industria, los ayudan en la decisión de automatizar las pruebas, a navegar a través de una gran

cantidad de herramientas para seleccionar las que mejor se ajusten y les dan consejos sobre la construcción de procesos de pruebas sólidos y documentados [26]; también ofrecen orientación sobre planificación, diseño, desarrollo, ejecución y evaluación de las pruebas [24, 27].

Existen algunas razones para la automatización y las organizaciones que la aplican piensan en cuestiones tales como mejorar tiempos de entrega y mantener pruebas repetibles [19], ahorrar recursos [28], o gestionarlas en menos tiempo [29]. Si se implemente bien, la automatización puede ser un medio eficaz para reducir el esfuerzo en el desarrollo, porque se pueden eliminar o minimizar tareas repetitivas y reducir el riesgo de introducir errores humanos [30]. Las pruebas automatizadas pueden ser consideradas como una piedra angular en el proceso de desarrollo de software, donde su uso está aumentando rápidamente debido a la introducción de métodos como el desarrollo basado en pruebas [31] y la integración continua [32].

Las pruebas automatizadas son dependientes de una herramienta, ya sea para el flujo de prueba del entorno de configuración, la ejecución de la prueba o el análisis de resultados para su desmontaje, por esto es necesario extraer el máximo beneficios de ellas [33, 34]. En consecuencia, la complejidad y el tamaño total de la prueba comúnmente son del orden de, o mayores que, la complejidad y el tamaño del producto bajo prueba [35]. Si esta complejidad no se administra adecuadamente puede generar contratiempos costosos para la automatización e inclusive llevar a su abandono [36]. El propósito de realizar medición a la automatización es permitir que el probador desarrolle sus habilidades para llevar a cabo esta tarea o ejecutarla de manera más eficiente que si se hiciera manual, y si esto falla entonces la automatización no tiene valor [19].

Los problemas que se solucionan con software son cada vez más complejos, por lo que la automatización juega un papel importante cuando se presentan cambios o mejoras y para entregar un producto con alta calidad y a tiempo [37]. En esa entrega hay tres aspectos clave: la velocidad, la calidad y la incertidumbre, que prácticamente son el mantra de la Era de la Información [38]. Esto implica que la velocidad de las nuevas funcionalidades decidirá el grado de satisfacción de los clientes, así como de mantenerlos y de atraer nuevos [25]. Además, en términos de desarrollo de software tradicionalmente la prueba se considera como el último paso antes de ir a producción, y a menudo el tiempo que se deja para hacerla no es suficiente, especialmente si se ejecuta manualmente [39]. Como resultado, la liberación del producto puede verse afectada [40]. El aumento de la complejidad de los sistemas de hoy significa que hay un número potencialmente indefinido de combinaciones de datos de entrada que dan lugar a salidas diferentes, y a menudo muchas de esas combinaciones no son cubiertas por las pruebas manuales [41].

Los beneficios de la automatización están soportados por las evidencias de los trabajos analizados, pero en comparación con los experimentos y los estudios de casos, aparecen solamente unos pocos informes de experiencias reportadas [25, 36, 42]. Los beneficios más documentados son:

- Mejora la calidad del producto, en términos de un menor número de defectos presentes en el ciclo de vida [30, 43].
- Incrementa y mejora la cobertura de la prueba y del código, y hay mayor cantidad de valores de entrada probados [30, 43-47].
- Reduce el tiempo de la prueba, es decir, ofrece la capacidad de ejecutar más pruebas en un lapso menor [25, 30, 48, 49].
- Mejora la fiabilidad, porque maneja la repetición de acuerdo con el conocimiento adquirido por los probadores [25].
- Incrementa la confianza, por ejemplo, desde la percepción de los desarrolladores y los usuarios [1, 25].
- Reutiliza casos de prueba, porque se diseñan con el mantenimiento en mente, y un alto grado de repetición trae beneficios [30, 50].
- Reduce el esfuerzo humano, lo que se puede aprovechar en otras actividades [36, 48, 50- 52].
- Disminuye costos, porque la automatización no requiere mucha intervención humana [45, 50, 53].

- Aumenta la detección de fallos, con lo que su eficacia es superior a las manuales [42, 47, 53- 56].

Por su parte, Fewster y Graham [25] afirman que uno de los beneficios más importantes de la automatización es que se puede ejecutar y repetir muchos casos de prueba, facilita el uso de herramientas de cobertura e incrementa significativamente el refinamiento iterativo de los casos a aplicar. Además, ayuda a crear mejores casos de prueba, con una calidad definida y con menos esfuerzo [36]. Otros autores vinculan los beneficios en diversos sentidos, como que al utilizar herramientas especiales se puede reducir el esfuerzo y los costos relacionados con la prueba [51], u opinan que la prueba exhaustiva es imposible por lo que alcanzar una alta cobertura sigue siendo una expectativa, y con la automatización se logra avanzar en su consecución [19]. La automatización puede reducir el costo del fracaso, al permitir una mayor cobertura para que los errores sean descubiertos antes que tengan la oportunidad de causar un daño real en el producto [38].

A diferencia de los beneficios, las limitaciones de la automatización se basan principalmente en informes de experiencias y en unos cuantos estudios empíricos [30, 57]. Algunas limitaciones reportadas son:

- La automatización no reemplaza totalmente las pruebas manuales, especialmente aquellas que requieren un amplio conocimiento del dominio [44, 52].
- Problemas para alcanzar los objetivos esperados, porque al ejecutar las pruebas en una fracción de tiempo no se pueden alcanzar beneficios duraderos o reales [28, 57].
- Dificultades en el mantenimiento, principalmente debido a los cambios en las tecnologías y la evolución de los productos software [58].
- El proceso necesita tiempo para madurar, porque crear la infraestructura y las pruebas requiere tiempo, por lo tanto, la madurez de la automatización y otras prestaciones conexas también [42].
- Crea falsas expectativas, debido a que las organizaciones no tienen claridad suficiente y su objetivo es ahorrar la mayor cantidad de tiempo y dinero que sea posible [51, 59].
- Estrategias inadecuadas, porque son difíciles de decidir, por lo tanto, no se alcanza el beneficio de la automatización [51, 58].
- Falta de personal cualificado, porque se necesita desarrollar buenas habilidades para implementar la automatización [19].

Automatizar las pruebas puede generar muchos beneficios, pero puede que no sea fácil de implementar porque hay que considerar la realización de cambios en los procesos de la estructura de la organización [60]. La iniciativa puede parecer solamente un cambio de lo manual a lo automático, sin embargo, idealmente debe ser considerada en términos de todos los factores posibles que pueden verse afectados por los cambios, tales como las personas, la tecnología y los procesos [13].

Desde la perspectiva de las personas, hay que centrarse en cómo hacer que los usuarios acepten los cambios en lugar de forzarlos a implementarlos, porque la adopción de una nueva tecnología no es una tarea fácil y es un proceso de la cognición del usuario que requiere consume mucho tiempo [13]. En este sentido, los investigadores han estudiado por largo tiempo cómo y por qué las personas adoptan nuevas tecnologías, y se han conformado varias corrientes [9]. Algunas se centran en la aceptación individual mediante la intención de uso o el uso en sí como una variable dependiente [61]. Desde esta perspectiva, hay que pensar en las limitaciones hacia el uso de la automatización, y la mayoría de organizaciones tienen su propio sistema interno que se ha vuelto cada vez más complejo después de años de acumulación [13].

Otra cuestión que se analiza en los trabajos es si la automatización puede sustituir la prueba manual. Uno de los puntos de vista propone que eso no es posible, porque las pruebas manuales detectan más nuevos defectos que las automatizadas [36]. Otro opina que entre el 60% y el 80% de los errores encontrados durante un esfuerzo de prueba automatizada ya se han logrado manualmente, y que a menos que desde el principio se creen y ejecuten nuevos casos de prueba en la herramienta, la mayoría de los errores se encuentran manualmente [62]. Estos

dos puntos de vista probablemente impliquen que la automatización puede ser utilizada como una especie de herramienta suplementaria, pero no un reemplazo completo de la prueba manual.

Mientras que estos autores visualizan a la automatización como una herramienta complementaria, otros se preocupan en cómo encajarla en un proceso en curso. Esto exige que el proceso aplicado debe ser lo suficientemente maduro, porque si es deficiente la automatización no ayudará [40]. La automatización necesita tiempo para madurar, lo mismo que para crear la infraestructura necesaria, por lo tanto, la madurez de la automatización (y otros conexos) requiere tiempo [19]. Otros autores afirman que para la automatización se necesitan personas con habilidades adecuadas, porque implementarla no significa que los probadores van a sobrar, y si no tienen las habilidades requeridas se les debe proporcionar formación [15]. Fecko y Lott [42] proponen algunos retos para la automatización de las pruebas. Uno de ellos indica que hay que enseñarles a los probadores cómo escribir y ejecutar pruebas automatizadas como parte regular de sus responsabilidades.

Dascal y Dror [63] afirman que los avances tecnológicos, como el software, han sido una parte integral del desarrollo humano a lo largo de la historia, y que los grandes avances de las últimas décadas han hecho que muchos penetren rápidamente en la vida cotidiana. Por lo que las pruebas automatizadas se han aplicado en algunos durante años, sin embargo, todavía son un concepto nuevo para la mayoría de las organizaciones [7]. Como una nueva tecnología, la comprensión de la automatización es importante porque es un paso indispensable para su adopción, y algunas estimaciones indican que, desde la década de 1980, alrededor del 50% de las nuevas inversiones en las organizaciones ha sido en IT [61].

Aun así, para que esas tecnologías mejoren la productividad, primero deben ser comprendidas, aceptadas y utilizadas, un proceso lento en el caso de la automatización [61]. Aparte de los puntos relativos al rendimiento y la calidad del trabajo, también se hace hincapié en que para la aceptación de la automatización es necesario disfrutarla [64], porque los empleados con poco interés pueden conducir a una menor productividad y pueden entorpecer su adopción [65].

Para Antonia Bertolino [15] la automatización es una forma de mantener la calidad del análisis y las pruebas, en línea con la cada vez mayor cantidad y complejidad del software. De acuerdo con esta autora, la investigación actual en Ingeniería de Software enfatiza en la automatización de la producción de software, con una generación de herramientas de desarrollo más complejas que producen mayor cantidad de código con menos esfuerzo. Pero la otra cara de la moneda es el peligro de que los métodos para evaluar la calidad de esa cantidad de software, los métodos de prueba en particular, no puedan mantener ese ritmo de construcción, por lo que Berner, Weber y Keller [36] afirman que la investigación actual debe tener por objeto mejorar lo más posible el grado de la automatización, ya sea mediante el desarrollo de técnicas avanzadas para generar pruebas, o mediante la búsqueda de procedimientos de soporte innovadores para automatizar el proceso.

El sueño con la automatización es que pueda desarrollar por sí misma un entorno de prueba integrado, y que como una pieza de software se complete, se despliegue y se encargue automáticamente de reparar y generar o recuperar el código defectuoso, produciendo casos de prueba adecuados, ejecutarlos y finalmente emitir un informe de resultados [66]. Esta idea, aunque quimérica, ha atraído a algunos investigadores [67, 68] con el objetivo precisamente de ejecutar pruebas automáticas al mismo tiempo que el desarrollador programa. Otros trabajan en la noción de la agitación del software [69], una técnica de pruebas unitarias automáticas apoyada en la herramienta comercial Agitator, que combina diferentes análisis, tales como la ejecución simbólica y la resolución de restricciones, y que genera entradas aleatorias de datos con el apoyo del sistema Daikon [70].

Por su parte, Dustin y Garrett [27] presentan un debate constructivo sobre qué casos de prueba deben ser automatizados y sobre directrices para evaluar el rendimiento de la inversión, un tema que también tratan Mosley y Posey [26]. Además, argumentan en contra de la idea de un ciclo de vida de las pruebas del software y afirman que el resultado de la automatización depende de la

calidad de los procesos ya existentes en la organización.

Dustin, Rashka y Paul [24] introducen una metodología estructurada orientada a asegurar la implementación exitosa de las pruebas automatizadas, y promueven que debe ser una preparación deliberada y bien razonada, incluyendo estudios en profundidad de los requisitos de prueba y estableciendo expectativas realistas y una planificación estructurada.

5. Análisis de resultados

De acuerdo con muchos de los autores analizados en esta investigación, la automatización de las pruebas es una de las herramientas más eficaces para reducir los costos y el tiempo de un plan de pruebas. Desde cuando aparecieron las primeras herramientas a mediados de 1980, el proceso se implementó en diferentes tipos de pruebas y en poco tiempo se convirtió en centro de atención, tanto por sus ventajas como por los problemas que puede causar, desde el aumento de los costos de producción hasta la disminución de la fiabilidad e incluso el fracaso de los proyectos.

Sin embargo, su popularidad se ha incrementado constantemente debido esas ventajas involucradas: reduce la participación de los probadores, lo que significa una disminución en los errores humanos; disminuye el tiempo de las pruebas y por lo tanto el costo del producto final; y la comprobación automática es unas 25 veces más rápida que la manual, por lo que los errores se descubren más rápido y se atienden igualmente.

Por otra parte, la mayoría de personas se imagina al proceso de prueba como una secuencia de acciones, y de hecho se puede describir como una secuencia de interacciones intercaladas con evaluaciones. El asunto es que las interacciones se pueden predecir, pero la mayoría son mal especificadas, complejas y ambiguas. Sin embargo, los enfoques para conceptualizar una secuencia general de acciones de prueba pueden ser útiles si el objetivo principal es reducir la prueba a un conjunto de acciones rutinarias. Pero incluso en esta situación el resultado puede ser superficial y limitado.

Por otra parte, la prueba manual tiene la propiedad de adaptarse, lo que significa que puede cambiar fácilmente de acuerdo con las circunstancias de cada contexto. Por lo tanto, las personas no requieren una estricta secuencia de acciones para revelar muchos defectos y para distinguir anomalías inofensivas, lo que constituye en una gran ventaja comparada con la automatización. Es decir, la automatización es la mejor opción para un reducido espectro de pruebas. Otros afirman que un error común es creer que probar significa repetir las mismas acciones una y otra vez. Aunque la realidad es que, si en la primera ejecución del caso de prueba no se determina ningún error, los existentes solamente podrán revelarse mediante otras ejecuciones, pero en otro contexto del plan de pruebas. En este sentido, las pruebas manuales tienen una amplia variedad de casos de prueba y proporcionan una buena tasa de éxito en la detección de errores nuevos y viejos. Por lo tanto, los casos de prueba ejecutados tendrán buenos resultados solamente si son variados.

En contra de muchas opiniones generalizadas, no todas las acciones de prueba se pueden automatizar. Algunas tareas son muy fáciles para los seres humanos, pero al mismo tiempo también lo son para los computadores. Por eso es que la parte más difícil de la automatización es la interpretación de los resultados de la prueba y, además, porque el desarrollo de software es innovador, lo que significa que tiene un alto grado de incertidumbre que agrava el problema de la automatización. Asimismo, un proyecto software se desarrolla utilizando un proceso incremental que involucra diferentes tipos de componentes, incluso en las últimas fases del siclo de vida, lo que tampoco colabora con la automatización.

Por lo tanto, las pruebas automatizadas se pueden transformar fácilmente en un proceso lento, costoso e ineficaz, y esto contradice las opiniones de que una prueba automatizada es más rápida porque no necesita intervención humana. Esta expresión es incorrecta, no solamente porque el proceso puede ser más lento, sino que siempre requiere intervención humana. Porque el proceso de analizar los resultados y corregir los errores lo deben llevar a cabo personas. De la misma forma que es imposible imaginar una prueba sin contratiempos,

igualmente es imposible imaginar la automatización sin intervención humana.

Una corriente analizada afirma que la automatización reduce los errores humanos y por eso es la decisión perfecta para implementar una larga lista de acciones seculares que son difíciles para las personas. Esto es que alienta a la mayoría de directores de IT cuando cuantifican los costos y los beneficios de las pruebas automatizadas al compararlas con las manuales, después de implementar la nueva estrategia. Por desgracia, estos procesos son muy diferentes y cada tipo de prueba tiene diferentes conceptos y dinámicas que se estructuran para determinar diferentes tipos de errores.

Por lo tanto, no tiene sentido una comparación directa en términos de costos por número de errores encontrados. Otra expectativa que despierta la automatización es el ahorro significativo de costos laborales. La realidad es que el valor de las pruebas automatizadas tiene varios componentes: desarrollo, operación, mantenimiento, cambios en los productos y de otras tareas necesarias y originadas por la automatización. Al compararlos con el valor total de las acciones de la prueba manual, no se reduce significativamente, porque no disminuye el trabajo de los probadores hasta el punto de que no se requieran.

El costo del trabajo de los probadores depende de muchos factores, incluyendo el de las herramientas utilizadas, su destreza, la calidad del conjunto de casos de prueba y muchos otros diferentes. Obviamente, escribir una sola prueba implica menos esfuerzo que de la de un sistema grande que incluye la elección de las herramientas, la coordinación de la automatización y otras acciones. Esto puede tomar meses de trabajo duro. Los costos de la automatización también incluyen los de las nuevas tareas especiales que requiere, por lo tanto, estas pruebas son una acción costosa que incluye cargos por la documentación de los casos de prueba, el proceso, el control y análisis de resultados, las modificaciones al código, la organización del proceso de desarrollo, y todos los que necesitan la participación de personas.

Para los autores que piensan de esta forma, la automatización no es una buena herramienta para reducir costos laborales. Varios investigadores opinan que una cuestión que no se toma muy en serio es la probabilidad de dañar el proyecto de pruebas con la automatización. Por desgracia, el diseño y la implementación de una automatización equivocada pueden causar un alto número de problemas. Por eso se requiere una comprensión plena de sus posibilidades de rendimiento antes de tomar la decisión de implementarla. Cuando no se hace esto, la automatización podría transformarse en una gran cantidad de código de prueba que los probadores no entienden completamente, porque no hay estrategias claras.

En consecuencia y como resultado del análisis a los resultados de esta investigación, se puede decir que en la literatura se encuentran opiniones y resultados diversos acerca de la automatización de las pruebas. En términos generales se pueden resumir en que:

- Las pruebas automatizadas no son una simple secuencia de comandos sino una sucesión de interacciones entremezcladas con evaluaciones.
- Automatizar no significa repetir las mismas acciones una y otra vez, sino que requiere variaciones continuas.
- No todos los tipos de prueba se pueden automatizar.
- La prueba automatizada reduce los errores de acción del probador, pero no reduce los errores del diseño.
- La prueba automatizada no sustituye completamente a la prueba manual, sino que la complementa.
- La automatización disminuye el valor del proceso de pruebas, pero aumenta el costo de desarrollo de las pruebas.
- Un plan de pruebas automatizadas requiere una documentación completa, para lo cual tradicionalmente no se aparta el tiempo suficiente.
- La prueba manual encuentra más defectos que una automatizada.
- La automatización mejora el uso de recursos y brinda confianza en la prueba de atributos que manualmente son difíciles.
- La corrección de los resultados esperados tiene mayor dependencia.
- La automatización no mejora la eficacia de la prueba.

- La calidad de la automatización es independiente de la calidad de la prueba.
- Las herramientas de prueba no son muy flexibles.
- La automatización puede aumentar significativamente la productividad y la calidad de las pruebas.
- La automatización mecaniza los pasos de las pruebas manuales utilizando herramientas.
- Con la automatización se incrementa la velocidad de ejecución del plan de pruebas y es más confiable, repetible, programable, integral y reutilizable.
- La automatización soluciona todos los problemas de las pruebas manuales.

6. Conclusiones

Debido a que las tácticas de la Ingeniería de Software se mueven hacia estándares más abiertos y se centran en la reducción del tiempo de desarrollo y en la creación de capacidades reutilizables, la necesidad de pruebas eficaces y exhaustivas se vuelve hoy más importante que nunca. Los artefactos de prueba ya no pueden ser usados solamente durante el desarrollo inicial y luego desechados. Construir software incremental de forma coordinada y con capacidades de prueba altamente reutilizables les permitirá a los desarrolladores reutilizar repetidamente casos de prueba y las herramientas. Las pruebas de software automatizadas utilizan herramientas para ejecutar el software, medir las respuestas y evaluar su exactitud sin intervención humana significativa.

Aunque se ha progresado bastante y la ciencia y la academia difunden resultados continuamente, de acuerdo con Serna, Martínez y Tamayo [71], y los resultados de esta investigación, la automatización de las pruebas se encuentra todavía en una etapa de experimentación. Esta investigación revela que presenta ventajas a la vez que limitaciones, y los autores difunden resultados especialmente de experiencias no comprobadas. Vale la pena notar que uno de los principales motivos para automatizar es ahorrar tiempo, porque la prueba se continúa ejecutando como una fase al final del ciclo de vida, cuando ya no se tiene tiempo suficiente para una prueba exhaustiva. Por otro lado, parece que para los clientes es apropiado exigir requisitos en forma de pruebas de aceptación automatizadas. Pero esto necesariamente no refleja la viabilidad de la automatización, solamente es una cuestión de cómo darle prioridad y satisfacer al cliente.

Si bien la automatización de las pruebas tiene un gran potencial para mejorar la eficiencia y la calidad de los productos software, no menos importante es el mantenimiento de este tipo de pruebas y los costos asociados. Esto hace que sea importante considerar cuidadosamente y por adelantado el potencial del beneficio versus el costo. En correspondencia con que una prueba no descubra errores puede ser engañosa, automatizar todo el plan de pruebas puede crear una falsa sensación de control del proceso. Porque parece ser que automatizar aumenta el nivel de confianza entre los desarrolladores, los probadores y el cliente. La base de experiencias limitada en la automatización encontrada en esta investigación demuestra que se han alcanzado algunos logros importantes. Sin embargo, todavía hay gran necesidad de investigar en esta práctica. En este artículo se han señalado algunos temas y conceptos sobre los que viene trabajando, pero los resultados no son muy alentadores para llegar a una automatización total en corto tiempo.

La demanda por el incremento de la funcionalidad del software, de un desarrollo rápido y de la disminución de los costos de producción, sin comprometer la calidad, no deja otra opción trabajar arduamente por automatizar totalmente las pruebas del software. Para algunos autores esta estrategia puede ser sorprendentemente eficaz y altamente beneficiosa para cualquier organización, pero, aunque el costo de configuración inicial puede ser alto, el rápido retorno de la inversión proyectado lo supera, y produce beneficios clave para incrementar la productividad, la disponibilidad, la fiabilidad y el rendimiento de cualquier empresa.

Las pruebas automatizadas son particularmente eficaces cuando la naturaleza del trabajo es repetitivo y rutinario, tales como las pruebas unitarias y las de regresión. Otros trabajos informan que usar las pruebas automatizadas les permitió probar grandes volúmenes de código, lo que habría sido imposible de otra manera. En todo caso, y dado que no hay uniformidad acerca de las

ventajas de la automatización, se espera que en el futuro cumpla con lo prometido desde hace décadas.

Referencias

1. **Haugset, B. & Hanssen, G. (2008)**. Automated acceptance testing: A literature review and an industrial case study. *Proceedings Agile Development Conference* pp. 27–38. DOI: 10.1109 / Agile.2008.82.
2. **Huang, L. & Holcombe, M. (2008)**. Empirical investigation towards the effectiveness of test first programming. *Information and Software Technology*, Vol. 52, No. 1, pp. 182–194. DOI: 10.1016/j.infsof.2008.03.007.
3. **Safronau, V. & Turlo, V. (2011)**. Dealing with challenges of automating test execution. *Proceedings Third International Conference on Advances in System Testing and Validation Lifecycle*, pp. 14–20.
4. **Hoffman, D. (1999)**. Test automation architectures: Planning for test automation. *Proceedings 12th International Software Quality Week,* pp. 1–8.
5. **Thummalapenta, S., Sinha, S., Mukherjee, D., & Chandra, S. (2012)**. *Automating test automation.* IBM Research Report, BM Research Division.
6. **Sommerville, I. (2010)**. *Software Engineering*. Pearson.
7. **Askarunisa, A., Ramraj, N., & Priya, S. (2009).** Selecting effective coverage testing tool based on metrics. *The ICFAI University Journal of Computer Sciences*, Vol. 3, No. 3, pp. 47–70.
8. **Abdel, T. (1988)**. The economics of software quality assurance: A simulation-based case study. *MIS Quarterly*, Vol. 12, No. 3, pp. 395–411. DOI: 10.2307 / 249206.
9. **Feldman, F. (2005)**. Quality assurance: Much more than testing. *Queue – Quality Assurance* Vol. 3, No. 1, pp. 26–29.
10. **Serna, M. E. & Arango, F. (2011)**. Software testing: More than a stage in the life cycle. *Revista de Ingeniería,* Vol. 35, pp. 34–40.
11. **Serna, M. E., Serna, A. A., Sepúlveda, J., & Guevara, R. (2015)**. Software testing is more than an emergency plan. *2nd International Conference on Communication Technology*, pp. 45–54.
12. **Serna, M. E. (2013)**. *Prueba funcional del software - Un proceso de Verificación constante.* Medellín: Fondo Editorial ITM.
13. **Stobie, K. (2005)**. Too darned big to test. *Queue – Quality Assurance,* Vol. 3, No. 1, pp. 30–37.
14. **Koochakzadeh, N. & Garousi, V. (2010)**. A tester-assisted methodology for test redundancy detection. *Advances in Software Engineering*, Article 6.
15. **Bertolino, A. (2007)**. Software testing research: Achievement, challenges, dreams. *Proceedings Future of Software Engineering* pp. 85–103.
16. **Dustin, E., Rashka, J., & Paul, J. (2000)**. *Automated Software Testing*. England: Addison-Wesley.
17. **Bavin, P. (2010)**. *A proposal for a repertoire of software testing methods.* Master's Thesis. Lappeenranta University of Technology, Finland.
18. **Whyte, G. & Mulder, D. (2011)**. Mitigating the impact of software test constraints on software testing effectiveness. *The Electronic Journal Information Systems Evaluation,* Vol. 14, No. 2, pp. 254–270.
19. **Rafi, D., Moses, K., & Petersen, K. (2012)**. Benefit and limitations of automated software testing: Systematic literature review and practitioner survey. *Proceedings 7th International Workshop on Automation of Software Test,* pp. 36–42.
20. **Majikes, J., Pandita, R., & Xie, T. (2013)**. Literature review of testing techniques for medical device software. *Proceedings of the Medical Cyber Physical Systems Workshop*.
21. **Wiklund, K., Sundmark, D., Eldh, S., & Lundqvist, K. (2014)**. Impediments for automated testing - An empirical analysis of a user support discussion board. *Proceedings Seventh International Conference on Software Testing*, Verification and Validation, pp. 113– 122.
22. **Guan, D. (2014)**. Manual to automated testing. *Thesis for the degree of Master of Information*

Management. Victoria University of Wellington, Australia.

23. **Torraco, R. (2005)**. Writing integrative literature reviews: Guidelines and examples. *Human Resource Development Review,* Vol. 4, No. 3, pp. 356–367.
24. **Dustin, E., Rashka, J., & Paul, J. (1999)**. *Automated software testing: Introduction, management, and performance*, Addison Wesley.
25. **Fewster, M. & Graham, D. (1999)**. *Software test automation: Effective use of test execution tools*. Harlow: Addison Wesley.
26. **Mosley, D. & Posey, B. (2002)**. *Just enough test automation*. USA: Prentice Hall.
27. **Dustin, E. & Garrett, T. (2009)**. *Implementing automated software testing: How to save time and lower costs while raising quality*. Addison Wesley.
28. **Persson, C. & Yilmaztürk, N. (2004)**. Establishment of automated regression testing at ABB: Industrial experience report on 'avoiding the pitfalls'. *Proceedings 19th IEEE International Conference on Automated Software Engineering*, pp. 112–121.
29. **Taipale, O., Kasurinen, J., Karhu, K., & Smolander, K. (2011)**. Trade-off between automated and manual software testing. *International Journal of System Assurance Engineering and Management,* Vol. 2, No. 2, pp. 114–125.
30. **Karhu, K., Repo, T., Taipale, O., & Smolander, K. (2009)**. Empirical observations on software testing automation. *Proceedings Second International Conference on Software Testing Verification and Validation*, pp. 201– 209.
31. **Janzen, D. & Saiedian, H. (2005)**. Test-driven development concepts, taxonomy, and future direction. *Computer,* Vol. 38, No. 9, pp. 43–50.
32. **Stolberg, S. (2009)**. Enabling agile testing through continuous integration. *Proceedings Agile Conference,* pp. 369–374.
33. **Gallesdic, I. & Killiospy, G. (2013)**. Software testing as science. *Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS),* Vol. 3, No. 1, pp. 33–37.
34. **Petrova, E., Pávlov, D., & Pávlova, A. (2014)**. Automate or not to automate acceptance tests: Not is a dilemma. *Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS),* Vol. 4, No. 1, pp. 30–33.
35. **Kasurinen, J., Taipale, O., Smolander, K., Jussi, K., Ossi, T., & Kari, S. (2010)**. Software test automation in practice: Empirical observations. *Advances in Software Engineering - Special issue on software test automation*, Article 4.
36. **Berner, S., Weber, R., & Keller, R. (2005)**. Observations and lessons learned from automated testing. *Proceedings 27th International Conference on Software Engineering,* pp. 571–579.
37. **Hayes, L. (1997)**. The truth about automated test tools. *Datamation,* Vol. 43, No. 4, pp. 45.
38. **Hayes, L. (1989)**. *The automated testing handbook*. Software Testing Institute.
39. **Davis, F. (1989)**. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly,* Vol. 13, No. 3, pp. 319–340.
40. **Damm, L., Lundberg, L., & Olsson, D. (2005)**. Introducing test automation and test-driven development: An experience report. *Electronic notes in theoretical computer science,* Vol. 116, pp. 3–15.
41. **Blackburn, M., Busser, R., & Nauman, A. (2004)**. Why model-based test automation is different and what you should know to get started. *Proceedings International Conference on Practical Software Quality Techniques & Testing Techniques*, pp. 1-16.
42. **Fecko, M. & Lott, C. (2002)**. Lessons learned from automating tests for an operations support system. *Software: Practice and Experience,* Vol. 32, No. 15, pp. 1485–1506.
43. **Malekzadeh, M. & Ainon, R. (2010)**. An automatic test case generator for testing safety-critical software systems. *Proceedings 2nd International Conference on Computer and Automation Engineering*, pp. 136–167.
44. **Tan, R. & Edwards, S. (2008)**. Evaluating automated unit testing in sulu. *Proceedings First International Conference on Software*

Testing, Verification, and Validation, pp. 62– 71.

45. **Alshraideh, M. (2012)**. A complete automation of unit testing for JavaScript programs. *Journal of Computer Science,* Vol. 4, No. 12, pp. 1012– 1019.
46. **Burnim, J. & Sen, K. (2008)**. Heuristics for scalable dynamic test generation. *Proceedings 23rd IEEE/ACM International Conference on Automated Software Engineering*, pp. 443–446.
47. **Saglietti, F. & Paint, F. (2011)**. Automated unit and integration testing for component-based software systems. *Proceedings International Workshop on Security and Dependability for Resource Constrained Embedded Systems*, pp. 1–5.
48. **Du Bousquet, L. & Zuanon, N. (1999)**. An overview of Lutess: A specification-based tool for testing synchronous software. *Proceedings 14th Conference on Automated Software Engineering,* pp. 208-215.
49. **Wissink, T. & Amaro, C. (2006)**. Successful test automation for software maintenance. *Proceedings 22nd IEEE International Conference on Software Maintenance,* pp. 265–266.
50. **Dallal, J. (2009)**. Automation of object-oriented framework application testing. *Proceedings 5th IEEE GCC Conference and Exhibition*, pp. 425-434.
51. **Pocatilu, P. (2002)**. Automated software testing process. *Economy Informatics,* Vol. 1, pp. 97–99.
52. **Leitner, A., Ciupa, I., Meyer, B., & Howard, M. (2007)**. Reconciling manual and automated testing: The autotest experience. *Proceedings 40th Hawaii International Conference on Systems Science*, pp. 261a.
53. **Shan, L. & Zhu, H. (2009)**. Generating structurally complex test cases by data mutation: A case study of testing an automated modelling tool. *Computing Journal,* Vol. 52, No. 5, pp. 571–588.
54. **Coelho, R., Cirilo, E., Kulesza, U., von Staa, A., Rashid, A., & de Lucena, C. (2007)**. Jat: A test automation framework for multi-agent systems. *Proceedings 23rd IEEE International Conference on Software Maintenance*, pp. 425–434.
55. **Hao, D., Zhang, L., Liu, M., Li, H., & Sun, J. (2009)**. Test-data generation guided by static defect detection. *Journal of Computer Science and Technology,* Vol. 24, No. 2, pp. 284–293.
56. **Swain, R., Panthi, V., Kumar, P., & Prasad, D. (2012)**. Automatic test case generation from UML state chart diagram. *International Journal of Computer Applications,* Vol. 42, No. 7, pp. 26–36.
57. **Bashir, M. & Banuri, S. (2008)**. Automated model based software test data generation system. *Proceedings 4th International Conference on Emerging Technologies*, pp. 275-279.
58. **Liu, C. (2000)**. Platform-independent and tool-neutral test descriptions for automated software testing. *Proceedings 22nd International Conference on Software Engineering,* pp. 713–715.
59. **Pettichord, B. (1999)**. Seven steps to test automation success. *Proceedings STAR West Conference*, pp. 136–167.
60. **Polo, M., Tendero, S., & Piattini, M. (2007)**. Integrating techniques and tools for testing automation. *Software Testing, Verification and Reliability,* Vol. 17, No. 1, pp. 3–39.
61. **Venkatesh, V., Morris, M., Davis, G., & Davis, F. (2003)**. User acceptance of information technology: Toward a unified view. *MIS Quarterly,* Vol. 27, No. 3, pp. 425–478.
62. **Kaner, C. (1997)**. Improving the maintainability of automated test suites. *Proceedings 10th International Software Quality Week*, pp. 1–15.
63. **Dascal, M. & Dror, I. (2005)**. The impact of cognitive technologies. *Pragmatics & Cognition,* Vol. 13, No. 3, pp. 451–457.
64. **Prusch, A., Suess, T., Paoletti, R., Olin, S., & Watts, S. (2011)**. Integrating technology to improve medication administration. *American journal of health-system pharmacy,* Vol. 68, No. 9, pp. 835–842.
65. **Attar, G. & Sweis, R. (2010)**. The relationship between information technology adoption and job satisfaction in contracting companies in Jordan. *Journal of information technology in construction,* Vol. 15, pp. 44–63.

66. Östrand, T., Weyuker, E., & Bell, R. (2005). Predicting the location and number of faults in large software systems. *IEEE Transactions Software Engineering,* Vol. 31, No. 4, pp. 340– 355.

67. Saff, D. & Ernst, M. (2004). An experimental evaluation of continuous testing during development. *Proceedings of the International symposium on Software testing and analysis,* pp. 76–85.

68. Briand, L., Labiche, Y., & Sówka, M. (2006). Automated, contract-based user testing of commercial off the shelf components. *Proceedings 28th International Conference on Software Engineering*, pp. 92–101.

69. Boshernitsan, M., Doong, R., & Savoia, A. (2006). From Daikon to Agitator: lessons and challenges in building a commercial tool for developer testing. *Proceedings International symposium on Software testing and analysis,* pp. 169–180.

70. Ernst, M., Perkins, J., Guo, P., McCamant, S., Pacheco, C., Tschantz, M., & Xiao, C. (2007). The Daikon system for dynamic detection of likely invariants. *Science of Computer Programming,* Vol. 69, No. 1-3, pp. 35–45.

71. Serna, M. E., Martínez, R., & Tamayo, P. (2015). Current maturity of development of software test automation. *Proceedings II Congreso Internacional de Ingeniería,* pp. 1– 10.

Article received on 08/08/2017; accepted on 14/08/2018.
Corresponding author is Edgar Serna M.