ISSN 2007-9737

Un modelo para determinar la madurez de la automatización de las pruebas del software como área de investigación y desarrollo

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

[1] Corporación Universitaria Remington, Medellín, Antioquia, Colombia

[2] Institución Universitaria de Envigado, Envigado, Colombia

edgar.serna@uniremington.edu.co, raquel.martinez@iue.edu.co, patamayo@correo.iue.edu.co

Resumen. En este artículo se propone un modelo para determinar la madurez actual de la automatización de las pruebas como área de investigación y de desarrollo en la industria del software. Se describe el proceso y los resultados de una investigación que tiene como objetivo determinar el nivel de madurez de la automatización. Se realizó una revisión sistemática de la literatura en la que se encontraron 978 trabajos que describen modelos de madurez de las pruebas del software y/o analizan la eficacia y la eficiencia de la automatización hasta el momento. Luego de un proceso de análisis y de aplicar los criterios de inclusión/exclusión y de valoración a la calidad se extrajeron 26 trabajos. La conclusión final es que el nivel de madurez actual de la automatización de las pruebas del software es Adolescente.

Palabras clave. Modelo de madurez, calidad, ingeniería de software, ciclo de vida, industria del software.

A Model for Determining the Maturity of Automation of Software Testing as a Research and Development Area

Abstract. This article proposes a model to determine the current maturity of test automation as an area of research and development in the software industry. The process and the results of a research that aims to determine the maturity level of automation is described. A systematic review of the literature in which 978 papers were found where are described the models of maturity of software testing and/or analyze the effectiveness and efficiency of automation was performed. After a process of analysis and apply the inclusion/exclusion criteria and quality assessment 26 papers were extracted. The final conclusion is that the current maturity level of automation of software testing is teenager.

Keywords. Maturity model, quality, software engineering, life cycle, software industry.

1. Introducción

La sociedad de este siglo es software-dependiente, porque este desarrollo tecnológico tiene cada vez más un fuerte impacto en las operaciones vitales de la vida humana, tales como la medicina, la aeronáutica, la investigación espacial, las telecomunicaciones y la protección de datos, entre otras [1]. Por eso es imperativo abordar los problemas de calidad relacionados tanto con el proceso del desarrollo de software como con el producto mismo. Esta investigación se centra en el proceso y se orienta a analizar el nivel de madurez que ha alcanzado la automatización de las pruebas.

En un sentido amplio la prueba se aplica para cubrir todas las actividades relacionadas con la calidad del software, por lo que si se mejora el proceso y la aplicación de criterios de madurez se logrará un impacto positivo en la productividad de la Ingeniería de Software, y se reducirán los esfuerzos y el tiempo de producción [2].

Por otro lado, y debido a que los problemas son cada vez más complejos y a que los clientes son más exigentes sobre la calidad de los

productos, la prueba es una actividad esencial en el desarrollo de software. Es necesario probar para minimizar el riesgo de entregar sistemas con fallas y la automatización de las pruebas surgió como una alternativa para probar en menos tiempo un ámbito más amplio de sus funcionalidades. En términos generales consiste en utilizar software para ejecutar o apoyar las actividades de prueba, tales como la gestión, los casos de prueba, la ejecución y la evaluación de resultados. Sin embargo, la industria todavía la concibe como una actividad para incrementar la productividad del equipo a la vez que la calidad de los productos [3].

Otra cuestión relevante para incrementar la calidad en el desarrollo de software es utilizar modelos de madurez para apoyar el mejoramiento continuo de los procesos del ciclo de vida. Pero, aunque existen diversos modelos que cubren este ámbito de actividades, hay otros que se construyen específicamente para desarrollar la cultura de las pruebas y, por lo tanto, soportar su introducción de forma más disciplinada y programada.

Entre todas estas propuestas nadie puede discutir que la automatización de las pruebas hace parte del abanico para lograrlo. Pero a las organizaciones, que buscan introducirla en sus procesos, no les es fácil seleccionar un modelo que les ayude a comprender cuáles son las mejores prácticas de la automatización y cómo introducirlas en contexto [3]. Las organizaciones que deciden aplicar esta tecnología pueden lograr uno de dos objetivos: ahorrar tiempo y dinero en lo cotidiano de la prueba o no alcanzar este ahorro, pero sí mejorar la calidad del software más rápidamente que con las pruebas manuales [4].

Cuando la automatización alcance un adecuado nivel de madurez será posible aplicar las pruebas solamente con el toque de un botón, a la vez que seleccionar el mejor momento para hacerlo. Además, serán repetibles y utilizarán exactamente las mismas entradas en la misma secuencia de tiempo, algo que no se puede garantizar con las manuales. Pero, aunque pueda sonar sorprendente, probar es una habilidad, y para cualquier sistema existe un número astronómico de posibles casos de prueba, aunque solamente se tiene tiempo para correr un pequeño número de ellos. Sin embargo, se espera con ello se encuentre la mayoría de los defectos en el software, por lo que hay que ser hábil para seleccionar y generar los casos de prueba más importantes. En este sentido, la experiencia ha demostrado por décadas que la selección al azar no es un método eficaz, por lo que se requiere un enfoque más reflexivo para lograrlo [5].

En este trabajo se propone un modelo para determinar la madurez de la automatización de las pruebas del software, a partir de una revisión a los que se han publicado en la literatura. La propuesta se sustenta en el hecho de que los modelos analizados tienen objetivos diversos y no logran determinar el nivel alcanzado por la automatización como área de investigación y desarrollo. Esto es importante para determinar en qué nivel se encuentra en las empresas y colaborar para incrementarlo y mejorarlo para su beneficio. Se necesita conocer esa madurez, porque de nada le sirve a una organización esta tecnología si no tiene los argumentos suficientes para implementarla y obtener todos sus beneficios.

Este artículo se estructura de la siguiente forma: en la segunda parte se describe el estado del arte acerca de las pruebas y su automatización; en la tercera se analizan los trabajos relacionados con el tema tratado; en la cuarta se describe la metodología aplicada en la investigación; en la quinta se presentan los resultados; en la sexta se explica el modelo de madurez de la automatización de las pruebas propuesto, al tiempo que se aplica a los resultados para determinar el nivel de madurez de la automatización; en la parte siete se analizan los resultados de la investigación; y en la ocho se presentan las conclusiones.

2. Estado del arte

Probar es aplicar una serie de actividades con el objetivo de descubrir y/o evaluar las propiedades de cada elemento del software [6]. Estas actividades pueden incluir la planificación, la preparación, la ejecución, la presentación de informes y la gestión. Meyers [2] establece que las pruebas son el proceso de ejecución de un

programa con la intención de encontrarle errores, y Hass [7] opina que se puede considerar como una actividad de soporte, porque no tiene sentido sin los procesos de desarrollo y porque no produce nada en sí misma: si no hay nada desarrollado no hay nada que probar. Estas afirmaciones ofrecen una idea general de la definición de las pruebas de software y esencialmente conducen al objetivo general de las mismas: no se trata de encontrar todos los errores que pueda tener el sistema, sino de descubrir situaciones que podrían afectar negativamente su funcionamiento [8, 9].

Sin embargo, hay que tener en cuenta que el costo de encontrar y corregir errores puede elevarse considerablemente durante el ciclo de vida. Por lo tanto, cuanto antes se descubran los errores será mejor para controlar sus efectos, moderados o graves, en etapas posteriores.

La historia de las pruebas refleja la propia evolución del desarrollo de software que, por mucho tiempo, se focalizó en grandes programas científicos y militares y en sistemas de bases de datos, producidos en plataformas mainframes o mini-computadores. Los escenarios de prueba se escribían en papel y las pruebas se orientaban a seguir las trayectorias de los flujos de control, al cálculo de algoritmos complejos y a la manipulación de datos. Un conjunto finito de procedimientos de prueba podía probar con eficacia un sistema completo, y el proceso generalmente se iniciaba hasta el final de la programación y lo ejecutaba el personal que estuviese disponible en el momento.

Con el surgimiento del computador personal se iniciaron procesos de estandarización en toda la industria y en cómo desarrollar las aplicaciones software para operacionalizarlas bajo un sistema operativo común. La introducción de los PC dio origen a una nueva era y generó un crecimiento explosivo del desarrollo de software comercial, y las aplicaciones empezaron a competir ferozmente por la supremacía y la supervivencia. Los productos líderes empezaron a rivalizar con calidad y eficiencia para satisfacer a los usuarios y las metodologías de desarrollo evolucionaron aceleradamente.

El esfuerzo de la prueba en estas nuevas metodologías requirió un enfoque diferente para probar el diseño, porque los flujos de trabajo podían ser llamados en casi cualquier orden. Esta capacidad exigía un alto número de procedimientos para soportar un sinfín de permutaciones y combinaciones.

Más tarde, la popularidad de las aplicaciones cliente-servidor introdujo una nueva complejidad en el esfuerzo de la prueba, porque el probador ya no ejercitaba una única aplicación para un sistema individual cerrado. Esta arquitectura implicaba tres componentes separados: el servidor, el cliente y la red. Por otro lado, la conectividad inter-plataformas aumentó la posibilidad de aparición de fallas, y las pruebas se tuvieron que relacionar con el rendimiento del servidor y la red, así como con el rendimiento general y la funcionalidad del sistema a través de esos componentes. Con el uso generalizado de las aplicaciones GUI, la captura de pantallas y la reproducción de escenarios se convirtieron en una forma atractiva para probar aplicaciones. Entonces, se introdujeron herramientas automatizadas para hacerlo y poco a poco se popularizaron [2].

Aunque los escenarios de prueba y los *scripts* se seguían escribiendo en alguna aplicación de procesamiento de textos, el uso de estas herramientas se incrementó. Al ampliarse la complejidad y el esfuerzo de la prueba se requirió mayor planificación, por lo que el personal encargado de aplicarla fue obligado familiarizarse más con el sistema y a cumplir con requisitos de formación más específicos, relacionados con las plataformas y redes involucradas. Actualmente, estas herramientas han madurado y ampliado su capacidad, a la vez que aparecen otras con fortalezas y nichos específicos [8]. Además, la prueba automatizada se ha convertido cada vez más en un ejercicio de programación, aunque todavía involucra las funciones tradicionales de gestión, tales como la trazabilidad de requisitos, la planificación, el diseño y el desarrollo de escenarios y de *scripts*.

La base en todo este proceso son los casos de prueba, que se describen mediante atributos que determinan su calidad. Tal vez, el más importante sea su eficacia para detectar defectos, pero también que sea reutilizable, es decir, que se pueda modificar fácilmente para probar más de una cosa, lo que reduce el número de necesario [10]. Además, debe ser económico

para realizar, analizar y depurar errores, y ser evolutivo y emplear la cantidad mínima de esfuerzo en su aplicación. A menudo estos atributos se deben equilibrar uno con otro. Hoy se acepta que la habilidad para realizar pruebas no consiste solamente en asegurar que los casos de prueba encuentren muchos defectos, sino también en que estén bien diseñados para evitar costos y tiempos excesivos [11].

Una forma de alcanzarlo es a través de la automatización, considerada actualmente por los equipos de prueba como una opción importante [8]. Aunque algunas organizaciones han fracasado rotundamente en su esfuerzo por implementarla y han tenido que recurrir nuevamente a los procesos manuales, a otras les ha permitido producir mejor software e incrementar su calidad rápidamente. Para obtener beneficios con la automatización las pruebas deben ser cuidadosamente seleccionadas y aplicadas, porque la calidad de este proceso es independiente de la calidad de la prueba, y el hecho de que se aplique automática o manualmente no afecta ni su eficacia ni su evolución.

No importa lo inteligente que se planee la automatización o lo bien que se haga, si la prueba en sí no logra nada entonces el resultado final será una evidencia de que nada se logra al hacerlo de esa forma. Habitualmente una vez implementada es mucho más económica, porque el costo de funcionamiento es una fracción de los esfuerzos necesarios para hacerlo manualmente, sin embargo, generalmente cuesta más crearla y mantenerla. De ahí la importancia de seleccionar inteligentemente el momento para automatizar, porque será más barato ponerlo en práctica a largo plazo [4]. Además, hay que pensar en el mantenimiento, porque la sola actualización de un conjunto de pruebas automatizado puede tener un alto costo.

Para que la automatización de las pruebas sea eficaz y eficiente hay que comenzar con una buena materia prima, es decir, un buen banco de pruebas, un conjunto de pruebas hábilmente diseñado por un probador con las destrezas suficientes. Posteriormente, hay que tener la habilidad para automatizarlo de tal manera que se pueda crear y mantener a un costo razonable. En resumen, es posible que las pruebas sean de buena o de mala calidad, pero en todo caso será la habilidad del probador lo que determine la calidad total de la misma. También es posible que la automatización tenga buena o mala calidad, pero será la habilidad del automatizador lo que determine lo fácil que será agregar nuevas pruebas automatizadas, a la vez que mantenerlas y obtener los beneficios [12].

Es de amplio conocimiento que la automatización de las pruebas consiste en utilizar software para realizar o soportar todo tipo de actividades de prueba, tales como la gestión, el diseño, la ejecución y el análisis de resultados [13]. A menudo, las pruebas automatizadas se consideran como la realización de pruebas sobre secuencias de comandos en lugar de tener probadores que lo realicen manualmente [6]. Sin embargo, también es cierto que muchas tareas y actividades adicionales de prueba pueden ser apoyadas por herramientas basadas en software.

En general, la actividad de la automatización supone utilizar herramientas y, de acuerdo con Hass [7], el propósito es ejecutar el mayor número posible de actividades no-creativas, repetitivas y aburridas, además de explotar la ventaja de esas herramientas para almacenar y organizar grandes cantidades de datos. En términos generales la automatización puede ayudar a resolver: 1) el trabajo repetitivo, 2) la lentitud de las pruebas manuales, y 3) la inseguridad de las pruebas manuales. Por otro lado, la introducción de la automatización debe aumentar la productividad, porque de otra manera no compensaría el costo [7].

3. Trabajos relacionados

Diversos investigadores han presentado revisiones acerca de la automatización de las pruebas del software con diversos resultados, aunque no tienen en cuenta el análisis a la madurez del proceso como área de investigación y desarrollo. A continuación, se relacionan alguno de ellos. El objetivo de Michael Grottke [14] fue desarrollar un nuevo y ampliado modelo estadístico para predecir la fiabilidad de los programas software con base en la madurez de las pruebas. El modelo se implementa mediante un prototipo que les ayuda a las pequeñas

empresas de software a predecir la confiabilidad de sus desarrollos y a probarlos de manera fácil y eficiente, con una precisión superior a la disponible. Aunque no tuvo la aceptación suficiente se rescata la amplia revisión y comparación que realiza a los modelos previos. Analiza la automatización de las pruebas desde una perspectiva estadística, pero no se orienta a encontrar la madurez del proceso como tal, sino a analizar los modelos propuestos hasta el momento.

La idea de la investigación de Ron Swinkels [15] es averiguar lo que se puede aprender de otros modelos de mejoramiento de los procesos de prueba. Para materializarla presenta una comparación entre Test Maturity Model (TMM) y siete modelos de mejoramiento con el objetivo de extraer prácticas importantes para desarrollar otro modelo. El resultado es una descripción y comparación desde los objetivos, estructuras, áreas clave del proceso y procedimientos de evaluación de los modelos. Los resultados sirvieron para proponer un modelo propio, orientado básicamente al mejoramiento de los procesos de prueba. Aunque presenta una matriz de comparación el modelo base que utiliza no es el referente más importante para la industria, debido a que su objetivo no fue encontrar el nivel de madurez de la automatización, ni siquiera de los modelos existentes.

Shrini Kulkarni [16] da cuenta de la histórica de los modelos de madurez para las pruebas del software y su relevancia en el estado actual y futuro de la industria. También presenta algunas ideas iniciales y pensamientos en torno a un nuevo marco que refleje el estado actual de la prueba, en el que destaca la habilidad del probador y la importancia del pensamiento cognitivo y la inteligencia humana, lo que considera como una desviación de los modelos anteriores que ignoran el aspecto humano de las pruebas. El autor concluye que hay necesidad de modificar la mirada a los modelos y hacerlos más pertinentes a como está progresando el mundo en este aspecto.

Los modelos de madurez del futuro tienen que reconocer el aspecto humano de la prueba y resistir la tentación de elaborar diagramas de flujo para las actividades, reconociendo la importancia del pensamiento crítico. El autor hace un análisis comparativo a cuatro modelos de madurez, pero siempre orientado a los aspectos humanos de la prueba, no a conocer o analizar el nivel de madurez de la automatización en general.

Gustavo De Souza [17] afirma que las mejores prácticas en las pruebas de software contribuyen a mejorar la calidad y reducir el costo de los productos, mediante la reducción de los tiempos en las etapas de prueba y durante la implementación y el mantenimiento. Los modelos de madurez para el desarrollo de software se han utilizado a gran escala para aliviar estos problemas, pero todavía no tienen cuenta las actividades relacionadas con las pruebas, por lo que se hizo necesario desarrollar modelos de madurez.

El autor hace una comparación y valida la eficiencia de los modelos Test Improvement Model (TIM), Test Process Improvement (TPI) y TMM e intenta determinar su nivel de madurez. El objetivo fue encontrar una guía para ayudarles a las empresas a mejorar la calidad de sus productos. La matriz de comparación utilizada es amplia y los indicadores ajustados, pero el hecho de que solamente trabaje con tres modelos no le permite presentar una imagen amplia del nivel de madurez de los relacionados con las pruebas del software. Además, su objetivo es demostrar el nivel de madurez de la organización con respecto a la automatización, no de la automatización como área de investigación.

Muhammad Sulayman [18] presenta una revisión sistemática de la literatura para identificar y analizar los modelos y técnicas existentes que utilizan las PyMes Web. Después de aplicar los filtros establecidos seleccionó un total de 88 estudios, pero sorprendentemente una inspección más detallada reveló que solamente cuatro de ellos eran relevantes para el tema. El objetivo principal fue investigar modelos o técnicas específicas para el mejoramiento de los procesos software, pero no encontró ninguno específico a la medida para las PyMes Web. Aunque las métricas analizadas incluyen a los equipos de desarrollo y la satisfacción de los clientes, el aumento de la productividad, el cumplimiento de los estándares y la excelencia operativa general, no identifica modelos de automatización para las pruebas, en parte por el tamaño de las empresas analizadas y porque sus

limitaciones presupuestales no les permiten adentrarse en este campo. En resumen, esta investigación no pudo determinar el nivel de madurez de la automatización en las PyMes Web.

Para Christiane von Wangenheim et al. [19] el nivel de madurez de la evaluación y el mejoramiento del software, guiado por procesos basados en un modelo de capacidad/madurez, se ha consolidado en la práctica como un medio eficaz para mejorar los procedimientos de desarrollo en las organizaciones. Describe los resultados de una revisión sistemática de la literatura sobre los modelos que en la última década han evolucionado el desarrollado y la adaptación. Los resultados demuestran que existe gran variedad de modelos con tendencia a la especialización para dominios específicos, pero que la mayoría se concentra en el marco CMM/CMMI y la norma ISO/IEC 15504, con los inconvenientes subsecuentes.

Aunque los autores analizan el nivel de madurez su objetivo es mostrar una matriz comparativa entre los 29 modelos evaluados, por lo que no presentan una evaluación a la madurez de la automatización de las pruebas como área de investigación y desarrollo.

El estudio de Heiskanen, Maunumaa y Katara [3] muestra que la introducción de esta tecnología no siempre se realiza con éxito, en parte porque los procesos de prueba y de la organización no siempre se ajustan adecuadamente. En este trabajo se presenta un modelo de generación de pruebas automatizadas sobre el modelo TPI y traza un perfil de base para la introducción exitosa de la tecnología. Su objetivo se orienta a compaginar los procesos de las pruebas y los de organización y hace una comparación de algunos modelos de madurez, pero no profundiza más allá de seleccionar la estructura para comparar procedimientos, es decir, no presenta un acercamiento al nivel de madurez de los modelos analizados ni tiene en cuenta la automatización como área a la que se puede evaluar su madurez.

García, Dávila y Pessoa [20] afirman que la calidad de los productos software está fuertemente influenciada por la calidad del proceso que los genera, y que particularmente la prueba contribuye en mayor medida. En este contexto su estudio pretende encontrar qué modelos de procesos de prueba han sido definidos, adaptados o extendidos en la industria. Identificaron 23 modelos, muchos de ellos adaptados o extendidos desde Test Maturity Model Integration (TMMi) y TPI con diferentes arquitecturas, y desde la norma ISO/IEC/IEEE 29119 como enfoque arquitectónico alineado con otros modelos.

Debido a que las métricas e indicadores utilizados por estos investigadores se orientaron a determinar para cada modelo el grado de adopción, la arquitectura, el domino, las fuentes utilizadas, las tendencias y la información mínima necesaria para comprenderlo, los resultados no determinan el nivel de madurez de la automatización de las pruebas en la industria y la investigación.

En el trabajo de Furtado, Meira y Gomes [4] se lee que la práctica de las pruebas de software es una de las maneras de producir productos de calidad, y que la automatización puede ser vista como una solución para probar la mayor cantidad de software en un tiempo determinado. Por eso su objetivo fue proponer directrices utilizando un modelo de madurez para la automatización con el fin de ayudarlas a entrar gradualmente en esta práctica.

Pero también afirman que la automatización puede no ser la solución para las necesidades de todas las empresas, por lo que su introducción puede complicar más de lo necesario el proceso de pruebas. Aunque esta investigación presenta una revisión de la literatura acerca de la automatización de las pruebas, su objetivo no es el de mostrar su nivel de madurez sino justificar la propuesta de otro modelo para implementarla. Es decir, no analizan la madurez de la automatización como un tema de investigación y desarrollo de la industria del software.

4. Metodología

Se realizó una revisión sistemática de la literatura para encontrar el nivel de madurez actual de la automatización de las pruebas del software, siguiendo los procedimientos descritos por Serna M. [21]. La pregunta de investigación a responder fue: ¿Cuál es el nivel de madurez actual de la automatización de las pruebas como

área de investigación y de desarrollo en la industria del software?

Para responderla se analizaron los trabajos que cumplieran con: cubrir expresamente a la automatización como área de investigación y desarrollo; haber sido publicados entre 1990 y 2014; describir modelos de madurez; presentar análisis comparativos a la eficiencia, la eficacia, el nivel de aceptación y la proyección; y realizar observaciones a la madurez de la automatización, los modelos o los procesos de prueba del software. Los trabajos seleccionados se limitaron a artículos publicados en revistas o actas de congresos y reportes técnicos.

Se excluyó cualquier publicación que no describiera explícitamente un modelo de madurez relacionado con la automatización de las pruebas, no hiciera referencia a una matriz de comparación, no analizara el nivel de madurez del proceso de automatización, o no lo concibiera desde las perspectivas de investigación y desarrollo. La calidad de los aportes se validó con los procedimientos establecidos por el medio de publicación, es decir, la revisión por pares de las revistas y congresos como criterio principal.

Se utilizaron las bases de datos IEEEXplore, la biblioteca digital ACM, Springer, Web of Science, ScienceDirect y Wiley Interscience. Las razones para seleccionar están bases de datos es que concentran la mayor cantidad de revistas relacionados con la automatización de las pruebas del software, y a que recopilan las memorias de conferencias, simposios y publicaciones de organizaciones, normas y libros. La información es actualizada diariamente y permiten realizar búsquedas por múltiples opciones. Los términos se buscaron en español e inglés.

5. Resultados

La búsqueda se realizó entre enero y abril de 2015 y en total se recuperaron 978 trabajos. En una primera etapa se revisaron títulos y resúmenes y se descartaron los irrelevantes y/o duplicados; posteriormente se analizaron los contenidos y no se tuvieron en cuenta aquellos que no describían completamente un modelo o no analizaban la madurez de la automatización como área de investigación y desarrollo.

Este proceso arrojó 16 modelos, que se describen en la Tabla 1, y 10 reportes de análisis a la eficiencia y eficacia de los modelos y a la madurez de la automatización. Con el fin de organizarlos se clasificaron por año de publicación, para determinar la derivación subsecuente y para identificar los modelos originales. Es de anotar que, de una u otra forma, todos los trabajos analizados tienen en cuenta la automatización, aunque algunos presentan una orientación más marcada que otros.

En la Tabla 2 se aprecian los resultados del análisis a las opiniones publicadas acerca de la eficiencia y eficacia de los modelos de madurez de la automatización de las pruebas del software. Esta valoración de los modelos se resume de los trabajos recolectados en los que: 1) se describe el modelo o se analiza su aplicación en casos de la industria [10, 12, 15, 39-41]; 2) el nivel de aceptación es la ponderación a la valoración que los autores hacen de cada uno [16, 41, 43, 44]; y 3) la proyección demuestra si el modelo tiene una vida útil referente o si fue efímero su paso por la industria [10, 16, 39, 41, 43].

6. Modelo de madurez propuesto

Conocer todo el potencial de la automatización de las pruebas del software será posible en la medida en que las organizaciones aprovechen sus beneficios, y esto se logra ubicando el proceso mismo dentro de un nivel de madurez. Cuanto más maduro sea el proceso de la automatización mayor será la eficiencia y la eficacia del plan de pruebas. Aunque la mayoría de investigadores y de personas en todo el mundo están familiarizados con los niveles de madurez utilizados por Capability Maturity Model (CMM), la razón de proponer un modelo con otra escala de valoración es que CMM se centra en evaluar la evolución de los procesos de una organización, y en este trabajo los autores quisieron acercarse al estado actual de la automatización como área de investigación y desarrollo. Es decir, la idea no es presentar la

Tabla 1. Modelos de madurez para la automatización de las pruebas del software

Modelo	Año	Orientación	Niveles	Escuela de pruebas [17]
MMAST [22]	1994	Automatización	4	Factory school
TAP [5, 23]	1995	Evaluación	5	Test-driven school
TCMM [24]	1996	Capacidad	4	Quality School
TSM [25]	1996	Capacidad	3	Quality School
TMM [26, 27]	1996	Procesos	5	Context-drive school
TIM [28]	1998	Mejoramiento	5	Standard School
TOM [29]	1998	Mejoramiento	3	Standard School
TPI [30]	1999	Mejoramiento	3	Standard School
ATLM [31]	1999	Automatización	5	Factory school
MB-V^2M^2 [32]	2002	Verificación y Validación	5	Control school
CB-VVCM [33]	2005	Verificación y Validación	5	Control school
SAMM [34]	2009	Aseguramiento	4	Context-drive school
CMMI-DEV [35]	2010	Calidad	4	Quality School
TMMi [36]	2012	Actividades de prueba y desarrollo	5	Analytical school
MPT.BR [37]	2012	Mejores prácticas	5	Agile School
ISO/IEC/IEEE [38]	2013	Estándares	6	Control school

Tabla 2. Eficacia y eficiencia de la automatización de las pruebas

Modelo	Eficacia y eficiencia	Nivel de aceptación	Proyección
MMAST	Baja	Bajo	Ninguna
TAP	Baja	Medio	Poca
TCMM	Baja	Bajo	Ninguna
TSM	Baja	Bajo	Ninguna
TMM	Baja	Bajo	Ninguna
TIM	Baja	Bajo	Ninguna
TOM	Baja	Bajo	Ninguna
TPI	Media	Alta	Alta
ATLM	Baja	Bajo	Poca
MB-V^2M^2	Media	Bajo	Poca
CB-VVCM	Baja	Bajo	Poca
SAMM	Media	Medio	Alta
CMMI-DEV	Alta	Alto	Alto
TMMi	Alta	Alto	Alta
MPT.BR	Alta (Brasil)	Alto (Brasil)	Alta (Brasil)
ISO/IEC/IEEE	Media	Medio	Alta

evolución sino el estado en el que se encuentra al momento de la revisión.

El objetivo de este modelo es determinar la madurez de la automatización de las pruebas del software analizándola desde una perspectiva y con niveles comparativos a la madurez de los seres humanos, es decir: 0. Infantil, 1. Adolescente, 2. Adulto y 3. Veterano.

Nivel 0: Infantil

El proceso de automatización de las pruebas está en un nivel de madurez *Infantil* cuando necesita mucho cuidado, atención y afecto. Las características de este nivel son:

- La mayoría de las pruebas se ejecutan solamente al finalizar el desarrollo del producto, porque el plan de pruebas automatizado no está en sintonía con el ciclo de vida del desarrollo.
- La cantidad de errores detectados hace que la prueba falle y que probablemente se detenga la secuencia de comandos automatizados, porque las incoherencias entre el software bajo prueba y el marco de automatización invalida cualquier caso de prueba que se intente aplicar.
- Se generan procesos de reingeniería porque en cualquier caso hay que corregir el error o actualizar el caso de prueba y luego volver a ejecutar el plan de pruebas para encontrar el siguiente problema.
- El equipo de prueba invierte más tiempo trabajando en los casos de prueba que en su automatización.
- La mayoría de organizaciones que intentan introducir la automatización de las pruebas no tardan en retroceder al proceso manual. Lamentablemente se dan cuenta tarde de la madurez de la automatización, porque normalmente les demora entre dos y diez veces más tiempo que el proceso manual.
- Debido a que hay que cuidar, mimar y prestarle mucha atención, la automatización no se puede dejar sin vigilancia por mucho tiempo. Aquí es posible afirmar que la madurez de esta tecnología desalienta en lugar de alentar su introducción.

Nivel 1: Adolescente

El proceso de automatización se encuentra en el nivel *Adolescente* cuando es posible aplicar el plan de pruebas y el conjunto de casos de prueba y dejarlo solo y sin vigilancia durante un tiempo razonable, tal vez un par de horas o incluso una noche, pero todavía se desconfía de su responsabilidad. Las características de este nivel son:

- Un solo error en el software bajo prueba hace que falle gran cantidad de casos de prueba.
- Es importante conocer el error, pero no se necesitarán decenas de casos de prueba para demostrarlo.
- Se desperdicia mucho tiempo intentando analizar la causa de cada bloqueo a la vez que se deja de ejecutar muchas pruebas.
- Para encontrar las fallas el equipo debe solucionar los problemas y volver a correr la automatización, un ciclo que se tiene que repetir muchas veces.
- En este nivel y al igual que con los adolescentes, el proceso de automatización es sorprendentemente útil, pero se comporta de manera irresponsable y puede causar más daño que beneficio.

Nivel 2: Adulto

El proceso de pruebas automatizadas se encuentra en nivel *Adulto* cuando es digno de confianza y se puede dejar solo para que funcione sin vigilancia durante un largo tiempo, por ejemplo, durante todo un fin de semana, pero si todavía no es auto-dirigido, porque su madurez no ha llegado a ese nivel de independencia.

- Al final el proceso de prueba arroja mucha información útil.
- Aunque quizás la mayoría de casos de prueba ha fallado, los datos son diferentes y sobre todo reportan algo del software bajo prueba.
- La automatización es algo más que secuencias de comandos.

Tabla 3. Madurez del desarrollo de la automatización de las pruebas

Etapas del proceso	Nivel de madurez			
	Infantil	**Adolescente**	**Adulto**	**Veterano**
Percepción			x	
Sensibilización			x	
Decisión		X		
Implementación		X		
Apropiación	x			
Consolidación	x			
Institucionalización	x			
Externalización	x			
Nivel de madurez actual de la automatización de las pruebas		**X**		

- El equipo se concentra en encontrar y corregir los problemas reportados mientras los casos de prueba se continúan ejecutado sin intervención.
- En este nivel la automatización no requiere vigilancia extrema y aunque el proceso es responsable y se puede confiar en él todavía no es auto-dirigido, porque su madurez no ha llegado a ese nivel de independencia.

Nivel 3: Veterano

Para llegar al nivel de madurez *Veterano* la automatización de las pruebas del software tiene que haber recorrido y crecido a través de los anteriores, y en este momento es posible dejarlo para que la naturaleza siga su curso. En este nivel la automatización es totalmente gestionable, las herramientas disponibles son autónomas, los casos de prueba son repetibles y el proceso se puede dejar funcionando sin vigilancia por semanas enteras. Las características de este nivel son:

- El plan de pruebas y los casos de prueba son eficientes y eficaces y el equipo observa todo el proceso de automatización como una disciplina no como un arte.
- El proceso de la automatización utiliza la reutilización como base porque ahora es bien entendido y aplicado.
- Al final se tiene un conjunto de casos de prueba validado y maduro, y una serie de reportes que denotan la calidad y fiabilidad del producto y de la automatización de las pruebas.
- El plan de pruebas se estructura como un enfoque planificado que involucra el diseño de los casos de pruebas y el mantenimiento del plan de pruebas.
- Es posible aplicar procedimientos de gestión y de planeación estratégica a todos los aspectos de la automatización.
- La automatización de las pruebas es un proceso autónomo en cuanto a reproducción y selección de los valores de entrada y a la validación y exposición de resultados.
- La organización cuenta con un banco de pruebas automatizadas reutilizable y fácil de proyectar a cada producto bajo prueba.
- Como en el caso de los humanos, en este nivel la automatización aporta experiencia y madurez para ponerlas al servicio de los demás procesos del software.

ISSN 2007-9737

6.1. Madurez de la automatización de las pruebas del software

Con base en los resultados presentados en las Tablas 1 y 2 y a las opiniones, reflexiones y críticas que la industria y los investigadores manifiestan acerca de los modelos de madurez y de la automatización misma, en la Tabla 3 se resumen los resultados acerca de la madurez de este proceso. En la primera columna se ubican las etapas tradicionales de un proceso de pruebas de software y en las demás los niveles del modelo de madurez de la automatización de las pruebas del software propuesto en esta investigación.

7. Análisis de resultados

Es importante observar cómo en la última década se ha incrementado el interés de los investigadores y la industria por proponer modelos para automatizar el proceso de las pruebas del software. En parte motivados por apoyar el mejoramiento de la calidad y la escalabilidad de sus productos. Pero también llama la atención que, aunque se presentan diversos modelos de madurez a la industria parece que le falta algo en este tema, y es que no los aplican rigurosamente. Una de las principales cosas que se aprende con estos modelos es que cuando se avanza a lo largo de ellos es muy importante no saltarse los niveles. Ese es el punto para que sea un modelo de madurez. Como se propone en este trabajo las personas crecen a través de cada etapa y a partir de lo que ya son en cada una. No es posible dejar de ser niño para pasar automáticamente a ser adulto, aunque se intente desesperadamente, porque si se intenta saltar adolescencia está destinado al fracaso.

Al analizar la aplicación de un modelo de automatización de pruebas y constatar que la empresa se encuentra en un nivel alto, se podría argumentar que simplemente es un caso en el que pasó de la implementación manual y aprendió en los demás niveles hasta lograr el despliegue total de la automatización. Pero la realidad en los resultados de esta investigación es que pocas empresas tienen la paciencia y los recursos para atender el proceso completo a través de todos los niveles, hasta lograr una automatización madura.

La mayoría que califica los modelos como deficientes o ineficaces es porque han intentado saltarse niveles para avanzar. Por otro lado, la industria del software ha tratado y ha invertido en mejorar la calidad de sus productos por años, pero ha sido una tarea difícil porque los clientes se han vuelto cada vez más exigentes y los sistemas software cada vez más complejos. El aseguramiento de la calidad se está convirtiendo en una condición permanente para la sobrevivencia de las empresas y actualmente es una realidad en la industria del software. A pesar de que algunos investigadores [48-50] afirman que la calidad del software ha mejorado en los últimos años, todavía está muy lejos del escenario ideal y de la madurez esperada. Para llenar este vacío la industria ha tratado de adaptar diferentes modelos de madurez a sus procesos del ciclo de vida, pero de acuerdo con los reportes analizados en esta investigación esos modelos describen prácticas de Verificación y Validación limitadas que no se centran directamente en la madurez del proceso de automatización de las pruebas. Esta madurez se puede definir como una forma de medir el nivel de capacidad que tiene una organización para gestionar los planes de pruebas de los proyectos [48], y el principal objetivo de conocerla es ayudarle a mejorar la construcción del software.

Aunque la prueba es un elemento esencial para lograr la satisfacción del cliente y una parte integral de todo el ciclo de vida del desarrollo de software, requiere velocidad, eficiencia y flexibilidad. Por eso es que el papel de las pruebas automatizadas es el de apoyarlo para eliminar tareas mecánicas, rutinarias y lentas. Debido a esta exigencia en velocidad, la gestión de los datos de prueba y de los diferentes entornos de plan de pruebas necesita ser muy eficiente y eficaces. Esta característica incrementa continuamente la demanda por una automatización madura que asegure que el proceso de pruebas ocurre sobre una base muy regular [49]. Además, que facilite la construcción de escenarios cada vez más predecibles, que requieran menos esfuerzo y que les permita a los equipos de desarrollo y de prueba obtener

información instantánea sobre la calidad del sistema en producción. Estas características no se logran de acuerdo con los resultados analizados en esta investigación.

La prueba automatizada es una estrategia fiable y para muchas empresas es la única opción para optimizar los estándares de calidad del software [50], pero de acuerdo con los resultados de esta investigación todavía se encuentra dentro de los tiempos de maduración que requiere cualquier aplicación compleja en el mundo actual.

Por otro lado, en la práctica los diferentes segmentos de aplicación de la automatización se encuentran en diferentes estados de madurez (ver Tabla 3), porque en cada etapa aparece un aprendizaje único que brinda la oportunidad de tener el poder para pasar a la etapa superior siguiente.

Pero un problema detectado es que actualmente en la industria la automatización de las pruebas se gestiona como otro proceso del desarrollo de software, y las herramientas disponibles para aplicarla se ofrecen de acuerdo con la demanda del mercado, no por una planeación real de las necesidades de la industria de un sistema escalable y mantenible para lograrlo. Esto genera algunos desafíos que es necesario afrontar para lograr que la automatización de las pruebas logre realmente su objetivo de alcanzar el nivel de madurez *Veterano*:

- El tiempo que se invierte para automatizar es muy extenso, porque la industria no tiene un adecuado nivel de madurez de sus procesos de desarrollo.
- El mantenimiento del plan de pruebas y del conjunto de casos de prueba es muy frecuente y no se adapta fácilmente a los escenarios cambiantes de los sistemas de hoy.
- Para las miles de líneas de código de la automatización no hay documentación y la que existe no es adecuada o está desactualizada.
- La rotación de recursos crea caos que cambia el enfoque completo del plan de pruebas, por lo que su automatización no está lista cuando se necesita.
- La industria actual invierte dinero, tiempo y esfuerzo solamente para hacer su trabajo, es decir, desarrollar, pero al automatizar las pruebas se encuentra con la frustración de no obtener rápidamente resultados.
- Las habilidades para automatizar las pruebas no son fáciles de adquirir, porque cada herramienta tiene un enfoque único y patentado para interactuar con las tecnologías de desarrollo. Por lo que la industria necesita capacitar diferentes equipos de prueba para atender las demandas de cada sistema que desarrolla, y debe utilizar diferentes herramientas debido a los diferentes requerimientos de los clientes en cuanto a lenguajes de programación.
- La industria tiene el problema de que toda la vida ha aplicado pruebas manuales y ha invertido bastante en capacitar a sus equipos de probadores. Pero resulta que ellos no saben automatizar el plan de pruebas simplemente porque no saben de programación.
- El otro desafío importante para que la automatización de las pruebas madure como área de investigación y de desarrollo es el costo de las herramientas y de la adquisición de las habilidades especiales para aplicarlas. Gran parte de la industria opta por la prueba manual porque a veces es más costosa su automatización que el mismo desarrollo del sistema.

De acuerdo con los resultados de esta investigación muchas empresas dedicadas al desarrollo de software ven y aplican la automatización de las pruebas como una especie de *bala de plata* que resolverá todos sus problemas de calidad, les ayudará a cumplir con todos los requisitos de los sistemas y les ahorrará mucho tiempo y esfuerzo. Este no es el caso, porque esta implementación implica una curva de aprendizaje progresivo hasta alcanzar la madurez adecuada. Eso significa que en realidad durante los primeros niveles las pruebas automatizadas pueden aumentar el esfuerzo y el costo de aplicación.

Como sucede con la generación del plan de pruebas, un área clave donde las expectativas de la automatización parecen estancadas en el tiempo porque actualmente se debe hacer de forma manual, aunque ya haya en el mercado herramientas que la apoyan, pero ninguna es automática. También es importante destacar que todavía no existe una herramienta que apoye todos los entornos de sistemas operativos y lenguajes de programación utilizados en una organización. Esto significa que la mayoría requiere un conjunto de diversas herramientas para automatizar sus pruebas, porque no hay tal cosa como una de *talla única*.

Por otra parte, las empresas deciden enfrascarse en la automatización porque piensan que la reducción de costos y de tiempos de entrega les permitirá cumplir con los plazos. Pero actualmente es poco probable que la automatización reduzca inmediatamente el esfuerzo de la prueba y ahorre el tiempo necesario, debido a que se requiere capacitación del personal para utilizar las herramientas y para aprender las diversas maneras eficaces de hacerlo. Además, la escritura de *scripts* de prueba aporta un nuevo nivel de complejidad al plan de pruebas, lo que requiere que los probadores y las empresas piensen en términos de fiabilidad y reutilización en el diseño, en lugar de simplemente en la eficacia de la prueba.

Otro asunto que la industria no comprende a cabalidad es que actualmente no todo el plan de pruebas se puede automatizar. Muchos sistemas contienen controles de terceros o *widgets* para mejorar su funcionalidad, lo que representa un problema porque es poco probable que una herramienta de automatización verifique su compatibilidad con todos estos controles, y puede que no sea capaz de manipular los caminos necesarios para probar la aplicación. Y pruebas como la comprobación de que un documento se imprime correctamente no se pueden automatizar, o tampoco es rentable para aquellos casos de prueba que solamente se ejecutan una vez.

Algo un poco más alejado de la realidad es que a menudo se espera que la automatización permitirá realizar pruebas al cien por ciento de la aplicación. La industria debe comprender que la prueba es una tarea potencialmente infinita, sin embargo, si se centra en las áreas clave del código puede mejorar considerablemente la fiabilidad del software.

En términos generales hasta el momento la automatización de las pruebas del software solamente permite: 1) incrementar la fiabilidad del sistema, 2) mejorar la calidad de las pruebas, 3) disminuir el esfuerzo que se dedica a las pruebas, y 4) reducir el cronograma del proyecto. Por todo esto es posible concluir que actualmente el nivel de madurez de la automatización de las pruebas del software como área de investigación y desarrollo es Adolescente.

8. Conclusiones

Una perspectiva que se puede aplicar en esta investigación es que en lo referente a la madurez de la automatización de las pruebas del software el vaso no está vacío, pero tampoco está lleno, solamente esta medio vacío. Esta apreciación se sustenta en los resultados analizados y a que se percibe una conciencia cada vez mayor, de quienes tienen experiencia en este campo, de que muchos esfuerzos en la automatización de las pruebas no cumplen las expectativas. Para llegar a esta conclusión la presente investigación encontró que, aunque existe gran cantidad de esfuerzo orientado al desarrollo y el mantenimiento de la automatización, es muy importante realizar un análisis costo-beneficio a cualquier intento por implementarla.

Esto significa analizar los resultados y las experiencias en diferentes empresas y métodos, porque los éxitos reportados en la literatura han sido en su mayoría en áreas en las que tenía sentido automatizar algunas pruebas en lugar de todo el plan. Además, al equipo de pruebas lo asesoraban especialistas y se les permitió el tiempo para hacerlo bien. Esto no es un denominador común a toda la industria y para todos los sistemas actualmente.

Aunque la automatización puede añadir complejidad y costos al esfuerzo de un equipo de pruebas, también puede proporcionar cierta ayuda valiosa si se cuenta con las personas adecuadas, con el entorno adecuado y si se aplica cuando tiene sentido hacerlo [11]. Luego

de realizar y presentar los resultados de esta investigación podemos concluir:

- Es importante definir el propósito de iniciar un esfuerzo de automatización de las pruebas porque, aunque existen varias categorías de herramientas para hacerlo cada una tiene su propio propósito.
- Identificar qué se desea automatizar y en qué fase del ciclo de vida implementarlo es el primer paso para desarrollar una estrategia de automatización. Solamente desear que todo sea probado más rápido no es una estrategia práctica, hay que ser específico.
- Desarrollar una estrategia de automatización es más importante que decidir qué se va a automatizar, cómo se va a hacer, cómo se mantendrán los *scripts* y cuáles serán los costos y los beneficios que se espera. Al igual que con todos los esfuerzos de prueba se debe construir una estrategia o plan de pruebas para implementarla.
- Muchas herramientas de prueba son sofisticadas y utilizan lenguajes existentes o código propietario, por lo que el esfuerzo de la automatización se convierte en una rutina manual que no es diferente del trabajo de un programador, codificando en un determinado lenguaje para escribir programas para automatizar un proceso de prueba. Hay que tratar a todo el proceso de la automatización como si fuera un esfuerzo de Ingeniería de Software, es decir, definir lo que se automatizará (requisitos); diseñar la automatización; escribir, probar e implementar los *scripts*; hacerle mantenimiento y definir su vida útil.
- Para alcanzar sus beneficios hay que observar al esfuerzo de la automatización como una inversión que requiere tiempo y recursos. Esto implica que por lo general las primeras versiones del sistema no ofrecen la recompensa esperada, y que el beneficio viene luego de correr las pruebas automatizadas en cada lanzamiento posterior. De ahí la importancia de poder ubicar rápidamente la automatización de las pruebas del software en alguno de los niveles de madurez propuestos en esta investigación.
- Debido a que la automatización realmente es otro esfuerzo de desarrollo de software, es importante que quienes realizan el trabajo posean las habilidades y destrezas necesarias. Un buen probador no significa necesariamente un buen automatizador. Los buenos probadores todavía serán necesarios para identificar y escribir casos de prueba, pero se necesita al automatizador para que tome esos casos de prueba y escriba código para su automatización. Esto no quiere decir que los probadores no puedan aprender a ser automatizadores, es sólo que esos dos roles son diferentes y las habilidades necesarias también son diferentes.

Los modelos analizados en este trabajo han sufrido cambios en el tiempo, proporcionándole a la industria nuevas versiones, en periodos cada vez más cortos. Esto dificulta su apropiación y experimentación, a la vez que la pérdida de respaldo por parte de investigadores y practicantes.

Al evaluar el proceso de automatización de las pruebas de software a partir de las etapas del modelo propuesto, y debido a que las etapas de apropiación, consolidación, institucionalización y externalización se encuentran en nivel infantil, mientras que las etapas de decisión e implementación en nivel adolecente y las de percepción y sensibilidad en nivel adulto, los autores concluyen que la madurez de la automatización, como área de investigación y desarrollo, se encuentra en un nivel Adolecente.

De acuerdo con los análisis a los resultados publicados por los investigadores y la industria, y a la aplicación del modelo de madurez propuesto, la automatización de las pruebas del software se encuentra actualmente en desarrollo y en proceso de maduración, por eso no es de esperar que en corto tiempo cumpla las promesas que viene haciendo desde hace más de dos décadas.

El trabajo para alcanzarlas debe continuar y cada vez se deberán sumar más investigadores y empresas para lograrlo. El futuro es prometedor, pero intentar proyectar lo que logrará más adelante es una lotería, y ese objetivo está por fuera de esta investigación. Lo que sí se puede

afirmar, con base en los resultados encontrados y descritos aquí, es que actualmente la madurez de la automatización de las pruebas del software está en un amplio proceso de aprendizaje y de experimentación, y que como sucede con los humanos adolescentes: se puede confiar en ella, pero no dejarla sola mucho tiempo.

Referencias

1. **Serna, M.E. (2012)**. Social control for science and technology. *Tenth Latin American and Caribbean Conference for Engineering and Technology*, pp. 1–8.
2. **Myers, J. (1979).** *The art of software testing*. John Wiley & Sons, Inc., New York.
3. **Heiskanen, H., Maunumaa, M., & Katara, M. (2012)**. A Test Process Improvement Model for Automated Test Generation. *PROFES 2012, LNCS 7343*, pp. 17–31. DOI: 10.1007/978-3-642-31063-8_3.
4. **Furtado, A., Meira, S., & Gomes, M. (2014)**. Towards a maturity model in software testing automation. *The Ninth International Conference on Software Engineering Advances*, pp. 282–285. New York, USA.
5. **Paulk, M., Weber, C., Garcia, S., Chrissis, M., & Bush, M. (1993).** *Key practices of the capability maturity model*. Technical report CMU/SEI-93-TR-25, Carnegie Mellon University.
6. **ISO/IEEE (2013).** *Part I International Standard, Software and systems engineering/software testing, concepts and definitions.* ISO/IEEE 29119, USA.
7. **Hass, A. (2008).** *A guide to advanced software testing*. Boston: Artech House.
8. **Serna, M.E. (2013).** *Functional test of software - A Constant Verification process*. Medellín: Fondo Editorial ITM.
9. **Serna, M.E. & Arango, F. (2010).** Effectiveness analysis of the set of test cases generated with the Requirements by Contracts technique. *V Congreso Colombiano de Computación*, Cartagena, Colombia. pp. 1–6.
10. **Karthikeya, S. & Rao, S. (2014).** Adopting the right software test maturity assessment model. *Cognizant,* Vol. 20, No. 20, Insights, New Jersey, USA.
11. **Anya, P. & Smith, G. (2014).** Qualitative research methods in Software Engineering. *Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software (RACCIS)*, Vol. 4, No. 2, pp. 1–18.
12. **Kumar, P. (2012).** *Test process improvement – Evaluation of available models*. *Maveric's Point of View*, pp. 8.
13. **ISTQB (2012).** *Standard glossary of terms used in software testing.* International Software Testing Qualifications Board. ISTQB: Munich.
14. **Grottke, M. (1999).** *Software Process Maturity Model study*. IST-1999-55017, PETS Project.
15. **Swinkels, R. (2000).** *A comparison of TMM and other Test Process Improvement Models*. 12-4-1-FP Report. Frits Philips Institute, Technische Universiteit Eindhoven. Eindhoven, Netherlands.
16. **Kulkarni, S. (2006).** Test process maturity models - Yesterday, today and tomorrow. *Proceedings 6th Annual International Software Testing Conference,* pp. 1–15, Delhi, India.
17. **De Souza, G. (2007).** *Modelo de maturidade em testes com foco em ambientes de testes hetorogêneos*. Dissertação (mestrado), Universidade Federal de Pernambuco. Pernambuco, Brasil.
18. **Sulayman, M. (2009).** A systematic literature review of software process improvement for small and medium web companies. *Communications in Computer and Information Science*, Vol. 59, pp.1–8. DOI: 10.1007/978-3-642-10619-4_1.
19. **Von Wangenheim, C., Rossa, J., Salviano, C., & Von Wangenheim, A. (2010)**. Systematic literature review of software process capability/maturity models. *Proceedings International Conference on Software Process Improvement and Capability Determination*.
20. **García, C., Dávila, A., & Pessoa, M. (2014)**. Test process models: Systematic literature review. *Communications in Computer and Information Science*, Vol. 477, pp. 84–93. DOI: 10.1007/978-3-319-13036-1_8.
21. **Serna, M.E. (2015).** Methodology for perform reliable literature reviews. *Revista Investigación Económica*.
22. **Krause, M. (1994).** Maturity model for automated software testing. *Medical Device & Diagnostic Industry Magazine*, pp. 1–10.
23. **Paulk, M., Mark, C., Weber, C., Curtis, M., & Chrissis, M. (1993).** Capability Maturity Model. *IEEE Software*, Vol. 10, No. 4, pp.18–27.
24. **Burgess, S. & Drabick, R. (1996).** *The I.T.B.G. Testing Capability Maturity Model*.
25. **Gelperin, D. (1996).** A Testability Support Model. *Proceedings Fifth International Conference on*

Software Testing, Analysis & Review, Orlando, USA.

26. **Burnstein, I., Suwanassart, T., & Carlson, C. (1996).** The Development of a Testing Maturity Model. *Proceedings Ninth International Quality Week Conference,* San Francisco, USA.
27. **Burnstein, I., Suwanassart, T., & Carlson, C. (1996)**. Developing a Testing Maturity *Model*. CROSSTALK, Software Technology Support Center, Hill Air Force Base, Utah. Part I, pp. 21–24. Part II, pp. 19–26.
28. **Ericson, T., Subotic, A., & Ursing, A. (1998).** TIM - A Test Improvement Model. *Software Testing Verification and Reliability*, Vol. 7, No. 4, pp. 229–246.
29. **Systeme Evolutif (1998).** *Test Organization Maturity Model.*
30. **Koomen, T. & Pol, M. (1999).** *Test process improvement: A practical step-by-step guide to structured testing.* New York: Addison-Wesley.
31. **Dustin, E., Rashka, J., & Paul, J. (1999).** *Automated Software Testing - Introduction, management, and performance*. Boston: Addison-Wesley.
32. **Jacobs, J. & Trienekens, J. (2002).** Towards a metrics based Verification and Validation maturity model. *Proceedings 10th International Workshop on Software Technology and Engineering Practice*, Montréal, Canada. pp. 1–6. DOI: 10.1109/STEP.2002.1267622.
33. **Yoon, K., Park, S., Bae, D., Chang, H., & Jung, J. (2005).** A framework for the V&V capability assessment focused on the safety-criticality. *Proceedings 13th IEEE International Workshop on Software Technology and Engineering Practice,* Budapest, Hungary. pp. 17–24. DOI: 10.1109/STEP.2005.5.
34. **Chandra, P. (2009).** *Software assurance maturity model - A guide to building security into software development.* Version 1.0. Open Web Application Security Project (OWASP).
35. **CMMI-DEV (2010).** *CMMI for Development*. Version 1.3. CMU/SEI-2010-TR-033, Software Engineering Institute.
36. **TMMI (2012).** *Test Maturity Model Integration.* Release 1.0*. TMMi Foundation.*
37. **Furtado, A., Wanderley, M., Carneiro, E., & De Farias, I. (2012).** MPT.BR: A Brazilian maturity model for testing. *Proceedings 12th International Conference on Quality Software*, pp. 220–229. DOI: 10.1109/QSIC.2012.53.
38. **ISO/IEC (2013).** *Software and systems engineering Software testing, Part 2: Test processes*. ISO/IEC/IEEE 29119-2:2013*.*
39. **Mosley, D. & Posey, B. (2002).** *Just enough software test automation*. Boston: Prentice Hall.
40. **Meszaros, G. (2007).** *xUnit test patterns: Refactoring test code*. New York: Addison-Wesley.
41. **Bhaggan, K. (2009).** *Test automation in practice*. Master's Thesis Report, DSW Zorgverzekeraar, Department of ICT, Netherlands.
42. **Balaraman, R. & Krishnankutty, H. (2013)**. *Need for a comprehensive test maturity model*. Infosys, Bangalore, India.
43. **Kohlegger, M., Maier, R., & Thalmann, S. (2009).** Understanding maturity models results of a structured content analysis. *Proceedings I-KNOW '09 and I-SEMANTICS '09*, pp. 51–61.
44. **Wiklund, K., Eldh, S., Sundmark, D., & Lundqvist, K. (2012).** Technical debt in test automation. *Proceedings Fifth International Conference on Software Testing, Verification and Validation,* pp. 887–892. DOI: 10.1109/ICST. 2012.192.
45. **Prado, D. (2003).** *Project Management in Organizations*. Belo Horizonte: Editora de Desenvolvimento Gerencial.
46. **Lionbridge (2009).** *Test process assessments move into the real world - A Rethinking QA white paper*. Lionbridge Technologies, Waltham, USA.
47. **Alsmadi, I. (2012).** *Advanced automated software testing: Frameworks for refined practice*. *Information Science Reference*. New York, USA.
48. **Nasir, N. & Sahibuddin, S. (2011).** Critical success factors for software projects: A comparative study. *Scientific Research and Essays*, Vol. 6, No. 10, pp. 2174–2186.
49. **SGI (2010).** *Modernization Clearing a pathway to success*. The Standish Group International Inc., Boston, USA.
50. **Cangussu, J., DeCarlo, R., & Mathur, A. (2002).** A formal model of the software test process. *IEEE Transactions on Software Engineering*, Vol. 28, No. 8, pp. 782–796. DOI: 10.1109/TSE.2002. 1027800.

Artículo recibido el 10/11/2015; aceptado el 22/02/2017.
Autor de correspondencia es Edgar Serna Montoya.