# NIVEL DE MADUREZ DE LA AUTOMATIZACIÓN DE LAS PRUEBAS DEL SOFTWARE


Juan Camilo Botero Acevedo

FACULTAD DE INGENIERÍAS

Ingeniería de Sistemas

Director
Prof. Edgar Serna M.

**INSTITUTO TECNOLÓGICO METROPOLITANO**


Febrero 2016

# RESUMEN

Históricamente, la estructura, clasificación y terminología de los atributos y las métricas aplicables a la gestión de la calidad del software se han derivado o extraído de los modelos ISO 9126-3 y el posterior ISO 25000:2005, conocido como SQuaRE. Con base en estos modelos, Consortium for IT Software Quality (CISQ) definió cinco características estructurales deseables y necesarias en un sistema para proporcionar valor de negocio: confiabilidad, eficiencia, seguridad, mantenibilidad y tamaño adecuado. La gestión de la calidad del software cuantifica en qué medida un sistema cumple cada una de estas dimensiones. Además, una medida agregada de la calidad se puede calcular a través de un esquema de puntuación cualitativo o cuantitativo, o una mezcla de ambos, que luego se ponderan para ver como refleja esas prioridades. Pero de nada sirve tener una estructura para conocer esta información si generalmente los resultados son adversos. Es decir, la calidad del software es baja. Con el objetivo de aportar en la búsqueda de una solución, desde hace tiempo los investigadores y la industria se han propuesto automatizar el proceso de prueba del software. Este proyecto se orienta a realizar una búsqueda sistemática para identificar, analizar y verificar el nivel de madurez que se ha alcanzado en este sentido. Conociendo lo que se ha propuesto hasta el momento, las buenas prácticas, los casos de éxito y de fracaso, y con la información recolectada y el conocimiento generado, se conformará la base para estructurar un proyecto con el objetivo de hacer un aporte a la automatización de las pruebas, pero integrando conceptos como la Inteligencia Artificial, las redes Neuronales y las Realidad Virtual.

*Palabras clave*: Ingeniería del software, calidad, fiabilidad, automatización.

# TABLA DE CONTENIDO

1. INTRODUCCIÓN ..... 4
2. MARCO TEÓRICO ..... 7
3. METODOLOGÍA ..... 10
4. RESULTADOS Y DISCUSIÓN ..... 11
5. CONCLUSIONES ..... 21
6. ANEXOS ..... 24

REFERENCIAS ..... 25

APÉNDICE (Artículo científico) ..... 28

# 1. INTRODUCCIO N

La sociedad de este siglo es software-dependiente. Los sistemas software siguen teniendo 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. 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, con el objetivo de brindarle información a las organizaciones que les permita evaluar y mejorar sus procesos relacionados. En un sentido amplio la prueba se aplica para cubrir todas las actividades relacionadas con la calidad de 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 podrán reducir los esfuerzos y el tiempo de producción.

En un mundo mágico utópico existirían viejos sabios que encarnarían la experiencia de años de estudio y de práctica, a los que las personas acudirían en busca de un objeto mágico para desaparecer los problemas. En el mundo real esta magia no existe pero sí la tecnología, y como Arthur C. Clarke dijo alguna vez: cualquier tecnología suficientemente avanzada es indistinguible de la magia. Esto encaja bien con la labor de los probadores, porque uno de los problemas que enfrentan es los cambios en los requisitos del producto. Esto introduce nuevos procedimientos y por tanto nuevos errores en lo que antes solía trabajar adecuadamente, por lo que son responsables de encontrarlos. Para lograrlo tienen que volver a ejecutar las pruebas, lo que hace que el rendimiento sea bajo y el costo del proceso se incremente. De esta manera enfrentan el otro problema: escapar del círculo vicioso de volver a ejecutar pruebas en cada proceso de cambio y ahorrar el tiempo que dedican a crear otras. En estas circunstancias vendría bien poder consultar al mago (vendedor de herramientas para automatización) y recibir un objeto mágico (herramienta de prueba) que haga desaparecer el problema.

Así bastaría con estructurar el plan de pruebas una vez y dejar que la herramienta haga el milagro a partir de ahí. Pero, ¿por qué no se puede lograr esto? Simplemente porque la tecnología, es decir, el objeto mágico que lo logra, todavía no está lo suficientemente madura. Además, en manos incultas ese objeto no funciona y sin la asistencia del mago los probadores estarían en mayores problemas que al principio, y esa dependencia también cuesta tiempo y dinero.

Debido a que en el mundo actual 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 con fallas los sistemas, por eso la automatización de las pruebas de software surgió como una alternativa para probar en menos tiempo un ámbito más amplio de funcionalidades del mismo. En términos generales consiste en utilizar software para ejecutar o apoyar las actividades de prueba, tales como, la gestión de pruebas, los casos de prueba, la ejecución de pruebas y la evaluación de resultados. Sin embargo, la industria la concibe como una actividad para incrementar la productividad del equipo a la vez que la calidad de los productos.

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 tanto soportar su introducción de forma más disciplinada y programada. Entre todas esta propuesta 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 entender y comprender cuáles son las mejores prácticas de la automatización y cómo introducirlas en contexto. En todo caso el software debe ser probado para garantizar que funciona como debería hacerlo en un entorno de trabajo, y las pruebas tienen que ser eficaces en la búsqueda y hallazgo de cualquier defecto en el producto. Pero también deben ser eficientes en economizar tiempo y dinero. Sin embargo, la instrucción de la automatización puede reducir significativamente el esfuerzo requerido para aplicarlas o aumentar dramáticamente el tiempo necesario para hacerlo. 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.

Cuando el proceso de la automatización alcance un nivel de madurez adecuado será posible aplicar las pruebas solamente con el toque de un botón y 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, e incluso el más pequeño de los cambios durante el mantenimiento será plenamente probado con un mínimo esfuerzo. Pero, aunque pueda sonar sorprendente, probar es una habilidad, y para cualquier sistema existe un número astronómico de posibles casos de prueba y solamente se tiene tiempo para correr un pequeño número de ellos. Sin embargo, se espera que este pequeño número 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 experimentación y la experiencia han demostrado por décadas que la selección al azar no es un método eficaz, por lo que se requiere un enfoque más reflexivo para lograrlo.

En este trabajo se propone un modelo para determinar la madurez actual del proceso de la automatización de las pruebas del software, lo cual es importante para que las empresas puedan determinar en qué nivel se encuentra y colaborar para incrementarlo y mejorarlo para su beneficio. Por todo esto es que se necesita conocer esa madurez, porque de nada le sirve a una organización esta tecnología sino tiene los argumentos suficientes para implementarla y obtener todos sus beneficios. El objetivo de esta investigación es encontrarlo y entregarle a los probadores y a las empresas la información que les ayude a seleccionar el momento adecuado para introducirlo en su contexto de desarrollo. Este artículo se estructura de la siguiente forma: en la primera parte se hace un recorrido por las pruebas del software y su automatización, en la segunda se propone un modelo de madurez para el proceso de automatización de las pruebas, en la tercera se describen los trabajos relacionados, en la cuarta se presenta la metodología aplicada en la investigación y en la quinta se presentan y analizan los resultados.

# 2. MARCO TEO RICO

Actualmente existen diversos modelos y estándares de calidad de software que se pueden clasificar según el nivel de intervención: organizacional, proceso, producto software y datos [51]. Los primeros tienen dos niveles de trabajo: entidad y proceso, cuyo objetivo es la creación de una estructura organizativa apropiada para fomentar el trabajo de la calidad de todos los interesados y de los departamentos de la organización, y su adaptación en las diferentes actividades de desarrollo y mantenimiento. Para realizar esta gestión, ISO define la norma ISO-9000, y particularmente para el software la ISO-9001 y la guía especifica ISO-90003. En cuanto al nivel de proceso, la calidad se refiere a la satisfacción del personal involucrado, de la administración, y del usuario final. Lo que se evalúa a través de Revisiones Técnicas Formales. Entre los principales modelos y estándares del nivel de proceso se encuentra: CMMI, TickIT, Bootstrap, Personal SW Process (PSP), Team SW Process (TSP), ISO-12207, ISO-15504 (SPICE), IEEE/EIA 12207, e ISO-20000, entre otros. A nivel de producto, la calidad implica la utilización de metodologías o procedimientos estándares para el análisis, el diseño, el desarrollo, y la prueba del software. En este nivel se encuentran el modelo de calidad de Boehm [52], el de McCall [53], las Metodologías SQAE y GQM, Furps, Dromey, ISO-9126-1, ISO-25000 (SQUARE), e IEEE Std 1061-1998, entre otros.

Todos estos aportes proponen y estructuran metodologías, métodos y procedimientos para gestionar la calidad en el desarrollo de software, pero la mayoría se queda en eso, en proponer. Hacen falta más estudios que difundan resultados de su aplicación, y que les sirvan a los ingenieros como soporte para definir los procedimientos de gestión de la calidad en el software.

Por otro lado, existen trabajos que no se centran un nivel específico de intervención, como los anteriores. Se pueden mencionar los de Villalba [54], quien define un proceso sistemático de desarrollo de modelos de calidad orientados a un dominio y teniendo como base la experiencia y el conocimiento de expertos. El valor agregado de esta propuesta es que tiene en cuenta la importancia relativa o el peso de las características de calidad incluidas en él y las relaciones entre dichas características. Vega [55] define

una metodología para el aseguramiento de la calidad en la adquisición de software, incluyendo las dimensiones de proceso y producto. Esta metodología especifica las etapas y tareas a realizar por parte del cliente y del proveedor, incorporando en el proceso las buenas prácticas recomendadas por modelos y estándares como CMMI-ACQ, ISO-9126, PRINCE2, entre otros.

En Estayno et al. [56] se define un *framework* para la evaluación de la calidad de productos software, cuyo objetivo es integrar la información de la gestión de calidad del producto para posibilitar las evidencias y el monitoreo de los esfuerzos del equipo de desarrollo en pro de lograrlo. Como valor agregado a este trabajo Demchum et al. [57] proponen una metodología para la medición de atributos de calidad y para la determinación del nivel de complejidad en aplicaciones orientadas a objeto desarrolladas en Java; Sánchez et al. realizaron un estudio de los estándares de evaluación de productos software y una evaluación a la calidad; y Fernández et al [58] desarrollaron una herramienta web para medir el nivel de madurez del proceso de desarrollo de proyectos específicos, siguiendo las pautas del modelo Competisoft.

Por su parte, Pinto desarrolla una herramienta orientada a la evaluación de calidad de software web, denominada QUCO2, cuyo objetivo de implementación es automatizar la evaluación de calidad de aplicaciones web desde el punto de vista del usuario. Su desarrollo se enmarca en un proyecto de investigación en el que se estudian diversos modelos de calidad, estándares y normas como como la ISO-14598 y la ISO-9126, que en la actualidad hacen parte del estándar ISO-25000 que define la forma como se debe evaluar la calidad de los productos software y el modelo de calidad que se debe seguir.

Comercialmente existen varias herramientas reconocidas que permiten evaluar la calidad del software desde varios aspectos:

- Control y gestión de versiones: Subversion, ClearCase, SourceSafe, Git, CVS.
- Generación de documentación técnica: Docbook.
- Análisis estático de código: CodePro Analytix, JustCode, ReSharper.
- Apoyo en pruebas unitarias y de estrés: JUnit, Jmeter, NUnit, Parasoft dotTEST.

Aunque la aplicación de algunos de estos trabajos ofrece resultados prometedores en cuanto a la automatización de las pruebas desde la concepción de la calidad del

software, el asunto de la comercialización no permite conocer sus estructuras, Las propuestas para gestionar la calidad de estos productos debe ser más abierta, porque de esta manera se podrían realizar más aportes para mejorarlas, en busca de una automatización total de las pruebas del software.

Polo y Piatinni [59] realizan la automatización del proceso de pruebas unitarias para código java mediante la construcción de una herramienta denominada TESTOOJ. En ella se combinan las pruebas funcionales y las estructurales para obtener un proceso de pruebas riguroso y completo. Laukkanen [60] presenta un *framework* para automatizar las pruebas a gran escala conducidas por datos y por palabras claves. Esta propuesta la utiliza Rantanen [61] para realizar un estudio acerca del desarrollo de software. La conclusión de este estudio es que el *framework* presenta limitaciones que impiden la aplicación de todos los casos de prueba, en especial los de las pruebas de aceptación. Sin embargo, esta limitación se puede trabajar mediante la planificación detallada de los casos. Esmite et al. [62] presentan una metodología y un conjunto de herramientas de código abierto (Selenium, Eclipse y extensiones de Mozilla Firefox como Firebug, XPath Checker y XPather) para la automatización de las pruebas funcionales de productos con interfaz web. La metodología definida consta de las siguientes etapas: definición de las pruebas automatizadas, definición de los procedimientos de prueba, generación de suites y *scripts*, ejecución de las pruebas automatizadas del ciclo funcional, investigación y modificación de herramientas, configuración del entorno, validación de las pruebas automatizadas y organización de las pruebas automatizadas. Por su parte, Figueiredo y Ferreira [63] presentan una metodología basada en las experiencias y los resultados de la implementación de la automatización de pruebas utilizando código abierto, y herramientas integradas en los proyectos de software que aplica la metodología SCRUM.

Todas estas propuestas hacen aportes significativos a la automatización de las pruebas del software, pero ninguna se puede considerar completa y terminada debido a que sólo cubren algunas pruebas; no consideran al mismo tiempo las funcionales y las estructurales con sus diferentes tipos; requieren algún componente manual en el proceso; o no consideran la gestión de la calidad como objetivo para validar en los casos de prueba.

# 3. METODOLOGIA

Con el fin de encontrar el nivel de madurez actual de la automatización de las pruebas del software se realizó una revisión sistemática de la literatura siguiendo los procedimientos descritos por Serna M. [16]. La pregunta de investigación central 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 examinaron todos los modelos de madurez de la automatización de las pruebas publicados y disponibles en las bases de datos entre 1990 y 2014; se consultaron los trabajos que presentan análisis comparativos a la eficiencia y eficacia de los modelos y el nivel de aceptación y proyección, y los aportes que hicieran observación a la madurez de la automatización, los modelos o los procesos de prueba del software. Los trabajos seleccionados se limitaron solamente 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 el proceso de pruebas del software, no hiciera referencia a una matriz de comparación o no analizara el nivel de madurez del proceso de automatización. 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, ISI web of Science, ScienceDirect y Wiley Interscience.

**CRONOGRAMA DE ACTIVIDADES**

| **Actividad** | **Meses** | | | | |
|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** |
| 1. Estructurar el protocolo de búsqueda | X | | | | |
| 2. Analizar las fuentes y su aceptabilidad | X | X | | | |
| 3. Definir criterios para evaluar el nivel de madurez | | X | | | |
| 4. Estructurar y presentar los resultados | | X | X | X | X |

# 4. RESULTADOS Y DISCUSIO N

**Modelo de madurez del desarrollo de la automatización de las pruebas**

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 propuesta en este trabajo es determinar la madurez del proceso de automatización de las pruebas del software analizándolo 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 el 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 aún así todavía se desvía de sus responsabilidades. Las características de este nivel son:

- 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.
- 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.

### *Modelos de madurez para las pruebas del software*

La búsqueda se realizó entre enero y abril de 2015 y en total se recuperaron 978 trabajos. En una primera etapa se revisaron rápidamente los títulos y resúmenes, se descartaron los irrelevantes y/o duplicados y los que no describían completamente el modelo propuesto. 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 del proceso de 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 modelos analizados tienen en cuenta la automatización, aunque algunos presentan una orientación marcada.

**Tabla 1**. Modelos de madurez para las pruebas del software

| **Modelo** | **Año** | **Orientación** | **Niveles** | **Escuela de pruebas** [17] |
|---|---|---|---|---|
| MMAST [18] | 1994 | Automatización | 4 | Factory school |
| TAP [19-21] | 1995 | Evaluación | 5 | Test-driven school |
| TCMM [22] | 1996 | Capacidad | 4 | Quality School |
| TSM [23] | 1996 | Capacidad | 3 | Quality School |
| TMM [24, 25] | 1996 | Procesos | 5 | Context-drive school |
| TIM [26] | 1998 | Mejoramiento | 5 | Standard School |
| TOM [27] | 1998 | Mejoramiento | 3 | Standard School |
| TPI [28] | 1999 | Mejoramiento | 3 | Standard School |
| ATLM [29] | 1999 | Automatización | 5 | Factory school |
| MB-$V^2M^2$ [30] | 2002 | Verificación y Validación | 5 | Control school |
| CB-VVCM [31] | 2005 | Verificación y Validación | 5 | Control school |
| SAMM [32] | 2009 | Aseguramiento | 4 | Context-drive school |
| CMMI-DEV [33] | 2010 | Calidad | 4 | Quality School |
| TMMi [34] | 2012 | Actividades de prueba y desarrollo | 5 | Analytical school |
| MPT.BR [35] | 2012 | Mejores prácticas | 5 | Agile School |
| ISO/IEC/IEEE [36] | 2013 | Estándares | 6 | Control school |

### *Eficacia y eficiencia de la automatización*

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, y se convierte en la base para determinar el nivel de madurez actual de la misma. Esta valoración de los modelos se resume de los trabajos recolectados en los que: se describe el modelo o se analiza su aplicación en casos de la

industria [8, 37-39, 41, 43, 44]; el nivel de aceptación es la ponderación a la valoración que los autores hacen de cada uno [9, 39, 40, 42]; y la proyección demuestra si el modelo tiene una vida útil referente o si fue efímero su paso por la industria [9, 37, 39, 40, 44].

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

| Modelo | Eficacia y eficiencia | 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 |

### *Madurez del desarrollo 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.

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

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

## 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 de las que califica los modelos como deficientes o ineficaces es porque han intentado saltarse niveles para llegar a los superiores.

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 [45-47] 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.*

# 5. 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. Luego de realizar y presentar los resultados de esta investigación podemos concluir que:

- Es importante definir el propósito de iniciar un esfuerzo de automatización de 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.

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.

# ANEXOS

| Institución Universitaria | FICHA TÉCNICA TRABAJO DE GRADO PARA REGISTRO EN EL SISTEMA DE INFORMACIÓN ACADÉMICA- SIA | Código | FDE 098 |
|---|---|---|---|
| | | Versión | 02 |
| | | Fecha | 2015-02-16 |

## 1. INFORMACIÓN GENERAL

**Título**

NIVEL DE MADUREZ DE LA AUTOMATIZACIÓN DE LAS PRUEBAS DEL SOFTWARE

**Objetivo**

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 de las que califica los modelos como deficientes o ineficaces es porque han intentado saltarse niveles para llegar a los superiores.

| **Plazo:** | **Inicio** | DD | MM | AA | **Fin** | DD | MM | AA |
|---|---|---|---|---|---|---|---|---|
| | | 25 | 08 | 2015 | | 25 | 02 | 2016 |

| | | | | | |
|---|---|---|---|---|---|
| **Intensidad Horaria Semanal** | **16 Horas** | | | | |
| **Horas Práctica Social** | | | | | |
| **En funcionamiento – Negocio Incubado** | SI | | NO | X | |

## 2. PERSONAL Y EMPRESA

| | |
|---|---|
| **Empresa** | **Juan Camilo Botero Acevedo** |
| **Representante** | **Juan Camilo Botero Acevedo** |
| **Cargo** | Estudiante de Ingeniería en Sistemas |
| **Documento** | 98706528 |
| **Dirección** | Carrera 59 A # 52 a 12 Bello |
| **E-mail** | Jcabo65@gmail.com |
| **Teléfono** | 3113200062 |
| **Razón Social** | Trabajador Independiente |
| **Asesor** | Edgar Serna |
| **Jurado** | |

## 3. DESCRIPCIÓN Y ALCANCE

**Descripción**

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. Además, que facilite la construcción de

1

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.

**Alcance**

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.

## 4. RECURSOS

**Recursos**

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.

## 5. PARTICIPANTES

| Nombre | Cedula |
|---|---|
| Juan Camilo Botero Acevedo | 98706528 |
| **Observación** | |
| | |

| |
|---|
| |

## 6. SEGUIMIENTO

**Seguimiento**

| | | | | |
|---|---|---|---|---|
| | | | | |
| **Deserción** | | | | |
| **Vinculación Laboral** | | | | |
| **Práctica Profesional** | | | | |
| **Trabajo de Grado Terminado** | SI | | NO | |
| **Visita Empresarial Realizada** | SI | | NO | |

3

# REFERENCIAS


[1] ISO/IEEE. ISO/IEEE 29119 – Part I International Standard. Software and systems engineering/software testing, concepts and definitions. IEEE, USA. 2013.

[2] J. Myers. "The art of software testing". John Wiley and Sons. New York, USA. 1979.

[3] A. Hass. "A guide to advanced software testing". Artech House. Boston, USA. 2008.

[4] E. Serna M. "Functional test of software - A constant Verification process". Fondo Editorial ITM. Medellín, Colombia. 2013.

[5] E. Serna M. y F. Arango. "Effectiveness analysis of the set of test cases generated with the Requirements by Contracts technique". En R. Villa y L. Orozco (Eds.), V Congreso Colombiano de Computación (pp. 1-6). April 14-16, Cartagena, Colombia. 2010.

[6] ISTQB. "Standard glossary of terms used in software testing". International Software Testing Qualifications Board. Munich, Germany. 2012.

[7] M. Grottke. "Software Process Maturity Model study". IST-1999-55017. PETS Project. 1999.

[8] R. Swinkels. "A comparison of TMM and other Test Process Improvement Models". 12-4-1-FP Report. Frits Philips Institute, Technische Universiteit Eindhoven. Eindhoven, Netherlands. 2000.

[9] S. Kulkarni. "Test process maturity models - Yesterday, today and tomorrow". Proceedings of the 6th Annual International Software Testing Conference (pp. 1-15). Delhi, India. 2006.

[10] G. de Souza. "Modelo de maturidade em testes com foco em ambientes de testes hetorogêneos". Dissertação (mestrado) - Universidade Federal de Pernambuco. Pernambuco, Brasil. 2007.

[11] M. Sulayman. "A systematic literature review of software process improvement for small and medium web companies". Communications in Computer and Information Science. Volume 59, pp. 1-8. 2009.

[12] C. von Wangenheim, J.C. Rossa, C. Salviano y A. von Wangenheim. "Systematic literature review of software process capability/maturity models". Proceedings International Conference on Software Process Improvement and Capability Determination. May 18-20, Pisa, Italy. 2010

[13] H. Heiskanen, M. Maunumaa y M. Katara. "A Test Process Improvement Model for Automated Test Generation". En: O. Dieste, A. Jedlitschka y N. Juristo (Eds.), PROFES 2012, LNCS 7343 (pp. 17-31). Springer-Verlag. Berlin, Germany. 2012.

[14] C. García, A. Dávila y M. Pessoa. "Test process models: Systematic literature review". Communications in Computer and Information Science. Volume 477, pp 84-93. 2014.

[15] A. Furtado, S. Meira y M. Gomes. "Towards a maturity model in software testing automation". En: H. Mannaert (Eds.), The Ninth International Conference on Software Engineering Advances (pp. 282-285). Curran Associates, Inc. New York, USA. 2014.

[16] E. Serna M. (In press). Methodology for perform reliable literature reviews. Revista Investigación Económica.

[17] C. Kaner. "Schools of software testing". 2006. Marzo 2015. http://kaner.com/?p=15.

[18] M. Krause. "Maturity model for automated software testing". Medical Device & Diagnostic Industry Magazine. December, pp. 1-10. 1994.

[19] P. Crosby. "Quality is free: The art of making quality certain". McGraw-Hill. London, UK. 1979.

[20] M. Paulk, C. Mark, C. Weber, B. Curtis, y M. Chrissis. Capability Maturity Model. IEEE Software. Volume 10. Issue 4, pp. 18-27. 1993.

[21] M. Paulk, C. Weber, S. Garcia, M. Chrissis y M. Bush. “Key practices of the capability maturity model”. CMU/SEI-93-TR-25. Carnegie Mellon University. Pittsburgh, USA. 1993.

[22] S. Burgess y R. Drabick. “The I.T.B.G. Testing Capability Maturity Model”. (No more details). 1996.

[23] D. Gelperin. “A Testability Support Model”. Presented at the Fifth International Conference on Software Testing, Analysis & Review. May 13-17, Orlando, USA. 1996.

[24] I. Burnstein, T. Suwanassart y C. Carlson. “The Development of a Testing Maturity Model”. Proceedings of the Ninth International Quality Week Conference. May 21-24, San Francisco, USA. 1996.

[25] I. Burnstein, T. Suwanassart y C. Carlson. “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). 1996.

[26] T. Ericson, A. Subotic y Ursing. “TIM - A Test Improvement Model”. Software Testing Verification and Reliability. Volume 7, Issue 4, pp. 229-246. 1998.

[27] Systeme Evolutif. “Test Organization Maturity Model”. 1998. Febrero 2015. http://gerrardconsulting.com/?q=node/485.

[28] T. Koomen y M. Pol. “Test process improvement: A practical step-by-step guide to structured testing”. Addison-Wesley. New York, USA. 1999.

[29] E. Dustin, J. Rashka, J. Paul. “Automated Software Testing - Introduction, management, and performance”. Addison-Wesley. Boston, USA. 1999.

[30] J. Jacobs y J. Trienekens. “Towards a metrics based Verification and Validation maturity model”. Proceedings of the 10th International Workshop on Software Technology and Engineering Practice (pp. 1-6). October 6-8, Montréal, Canada. 2002.

[31] K. Yoon, S. Park, D. Bae. H. Chang y J. Jung. “A framework for the V&V capability assessment focused on the safety-criticality”. 13th IEEE International Workshop on Software Technology and Engineering Practice (pp.17-24). September 24-25, Budapest, Hungary. 2005.

[32] P. Chandra. “Software assurance maturity model - A guide to building security into software development, Version - 1.0”. Open Web Application Security Project (OWASP). 2009.

[33] CMMI-DEV. “CMMI for Development, Version 1.3”. CMU/SEI-2010-TR-033, Software Engineering Institute, USA. 2010.

[34] TMMI. “Test Maturity Model Integration, Release 1.0”. TMMi Foundation. 2012.

[35] A. Furtado, M. Wanderley, E. Carneiro e I. de Farias. “MPT.BR: A Brazilian maturity model for testing”. Proceedings 12th International Conference on Quality Software (pp. 220-229). August 27-29, Xi'an, China. 2012.

[36] ISO/IEC. “Software and systems engineering – Software testing, Part 2: Test processes”. ISO/IEC/IEEE 29119-2:2013. 2013.

[37] D. Mosley y B. Posey. “Just enough software test automation”. Prentice Hall Professional. Boston, USA. 2002.

[38] G. Meszaros. “xUnit test patterns: Refactoring test code”. Addison-Wesley. New York, USA. 2007.

[39] K. Bhaggan. “Test automation in practice”. Master’s Thesis Report. DSW Zorgverzekeraar, Department of ICT. Netherlands. 2009.

[40] M. Kohlegger, R. Maier y S. Thalmann. “Understanding maturity models results of a structured content analysis”. Proceedings of I-KNOW ’09 and I-SEMANTICS ’09 (pp. 51-61). September 2-4, Graz, Austria. 2009.

[41] P. Kumar. “Test process improvement – Evaluation of available models”. Maveric’s Point of View Volume 8. Maveric. Egmore, India. 2012.

[42] K. Wiklund, S. Eldh, D. Sundmark y K. Lundqvist. "Technical debt in test automation". Fifth International Conference on Software Testing, Verification and Validation (pp. 887-892). April 17-21, Montreal, Canada. 2012.

[43] R. Balaraman y H. Krishnankutty. "Need for a comprehensive test maturity model". Infosys. Bangalore, India. 2013.

[44] S. Karthikeya y S. Rao. "Adopting the right software test maturity assessment model". Cognizant 20-20 Insights. Cognizant. New Jersey, USA. 2014.

[45] N. Nasir y S. Sahibuddin. "Critical success factors for software projects: A comparative study". Scientific Research and Essays. Volume 6. Issue 10, pp. 2174-2186. 2011.

[46] The Standish Group International. "Modernization - Clearing a pathway to success". The Standish Group International Inc. Boston, USA. 2010.

[47] J. Cangussu, R. DeCarlo y A. Mathur. "A formal model of the software test process". IEEE Transactions on Software Engineering. Volume 28. Issue 8, pp. 782-796. 2002.

[48] D. Prado. "Project Management in Organizations". Editora de Desenvolvimento Gerencial. Belo Horizonte, Brasil. 2003.

[49] Lionbridge. "Test process assessments move into the real world - A Rethinking QA white paper". Lionbridge Technologies, Inc. Waltham, USA. 2009.

[50] I. Alsmadi. "Advanced automated software testing: Frameworks for refined practice". Information Science Reference. New York, USA. 2012.

[51] Scalone, F. (2006). Estudio comparativo de los modelos y estándares de calidad del software. Tesis de Maestría. Universidad Tecnológica Nacional. Facultad Regional Buenos Aires.

[52] Boehm, B.W. (1973). The high cost of software. Symposium on High Cost of Software. Monterey, California, pp. 27-40.

[53] McCall, J. et al (1977). Factors in Software Quality. Technical Report, RADC-TR-77-369. General Electric Company.

[54] Villalba, M. (2009). Metodología de desarrollo de modelos de calidad orientados a dominio y su aplicación al dominio de los productos finales de seguridad de tecnologías de la información. Tesis doctoral. Universidad de Alcalá. Departamento de Ciencias de la Computación.

[55] Vega, V. (2012). Metodología para el aseguramiento de la calidad en la Adquisición de Software (proceso y producto) y servicios correlacionados. (MACAD - PP). Tesis Doctoral. Universidad Politécnica de Madrid. Departamento de Lenguajes y Sistemas informáticos e Ingeniería de Software.

[56] Estayno, M. et al. (2013). QUCO2: Una herramienta para medir la calidad de aplicaciones Web. Revista Ciencia y Tecnología Vol. 13, No. 13, pp. 53-68.

[57] Demchum, D. et al. (2011). Medición de Atributos de Calidad en Aplicaciones Orientadas a Objeto. Revista La UTN en el NEA: Investigación y Desarrollo en la Regional Resistencia. Online [Agosto 2014].

[58] Fernández, L., Dapozo, G. & Greiner, C. (2011). Aplicación para la autoevaluación de Capacidad de Proceso orientada a Pymes. Primer Seminario Argentina-Brasil de Tecnologías de la Información y la Computación.

[59] Polo, M. & Piattini, M. (2006). Automatización del proceso de pruebas unitarias. Taller sobre Pruebas en Ingeniería del Software. PRIS. Sitges.

[60] Laukkanen, P. (2006). Data-Driven and Keyword-Driven Test Automation Frameworks. Tesis de maestría. Helsinki University of Technology. Department of Computer Science and Engineering Software Business and Engineering Institute.

[61] Rantanen, J. (2007). Acceptance Test-Driven Development with Keyword-Driven Test Automation Framework in an Agile Software Project. Tesis de Maestría. HELSINKI UNIVERSITY OF TECHNOLOGY. Department of Computer Science and Engineering Software Business and Engineering Institute.

[62] Esmite I. et al (2007). Automatización y Gestión de las Pruebas Funcionales usando Herramientas Open Source. XIII Congreso Argentino de Ciencias de la Computación. 1-5 de octubre, Corrientes, Argentina.

[63] Figueiredo, C. & Ferreira, L. (2012). Software test automation practices in agile development environment: An industry experience report. In 7th International Workshop Automation of Software Test (AST), pp. 57-63.

# APE NDICE (Artículo científico)

## Madurez del desarrollo de la automatización de las pruebas del software

*Maturity of development of software test automation*

Edgar Serna M., Juan C. Botero A.

**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.

**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 conclusion is that the current maturity level of automation of software testing is teenager*.

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

### INTRODUCCIÓN

La sociedad de este siglo es software-dependiente. Los sistemas software siguen teniendo 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. 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, con el objetivo de brindarle información a las organizaciones que les permita evaluar y mejorar sus procesos relacionados. En un sentido amplio la prueba se aplica para cubrir todas las actividades relacionadas con la calidad de 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 podrán reducir los esfuerzos y el tiempo de producción.

En un mundo mágico utópico existirían viejos sabios que encarnarían la experiencia de años de estudio y de práctica, a los que las personas acudirían en busca de un objeto mágico para desaparecer los problemas. En el mundo real esta magia no existe pero sí la tecnología, y como Arthur C. Clarke dijo alguna vez: cualquier tecnología suficientemente avanzada es indistinguible de la magia. Esto encaja bien con la labor de los probadores, porque uno de los

problemas que enfrentan es los cambios en los requisitos del producto. Esto introduce nuevos procedimientos y por tanto nuevos errores en lo que antes solía trabajar adecuadamente, por lo que son responsables de encontrarlos. Para lograrlo tienen que volver a ejecutar las pruebas, lo que hace que el rendimiento sea bajo y el costo del proceso se incremente. De esta manera enfrentan el otro problema: escapar del círculo vicioso de volver a ejecutar pruebas en cada proceso de cambio y ahorrar el tiempo que dedican a crear otras. En estas circunstancias vendría bien poder consultar al mago (vendedor de herramientas para automatización) y recibir un objeto mágico (herramienta de prueba) que haga desaparecer el problema. Así bastaría con estructurar el plan de pruebas una vez y dejar que la herramienta haga el milagro a partir de ahí. Pero, ¿por qué no se puede lograr esto? Simplemente porque la tecnología, es decir, el objeto mágico que lo logra, todavía no está lo suficientemente madura. Además, en manos incultas ese objeto no funciona y sin la asistencia del mago los probadores estarían en mayores problemas que al principio, y esa dependencia también cuesta tiempo y dinero.

Debido a que en el mundo actual 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 con fallas los sistemas, por eso la automatización de las pruebas de software surgió como una alternativa para probar en menos tiempo un ámbito más amplio de funcionalidades del mismo. En términos generales consiste en utilizar software para ejecutar o apoyar las actividades de prueba, tales como, la gestión de pruebas, los casos de prueba, la ejecución de pruebas y la evaluación de resultados. Sin embargo, la industria la concibe como una actividad para incrementar la productividad del equipo a la vez que la calidad de los productos.

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 tanto soportar su introducción de forma más disciplinada y programada. Entre todas esta propuesta 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 entender y comprender cuáles son las mejores prácticas de la automatización y cómo introducirlas en contexto. En todo caso el software debe ser probado para garantizar que funciona como debería hacerlo en un entorno de trabajo, y las pruebas tienen que ser eficaces en la búsqueda y hallazgo de cualquier defecto en el producto. Pero también deben ser eficientes en economizar tiempo y dinero. Sin embargo, la instrucción de la automatización puede reducir significativamente el esfuerzo requerido para aplicarlas o aumentar dramáticamente el tiempo necesario para hacerlo. 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.

Cuando el proceso de la automatización alcance un nivel de madurez adecuado será posible aplicar las pruebas solamente con el toque de un botón y 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, e incluso el más pequeño de los cambios durante el mantenimiento será plenamente probado con un mínimo esfuerzo. Pero, aunque pueda sonar sorprendente, probar es una habilidad, y para cualquier sistema existe un número astronómico de posibles casos de prueba y solamente se tiene tiempo para correr un pequeño número de ellos. Sin embargo, se espera que este pequeño número

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 experimentación y la experiencia han demostrado por décadas que la selección al azar no es un método eficaz, por lo que se requiere un enfoque más reflexivo para lograrlo.

En este trabajo se propone un modelo para determinar la madurez actual del proceso de la automatización de las pruebas del software, lo cual es importante para que las empresas puedan determinar en qué nivel se encuentra y colaborar para incrementarlo y mejorarlo para su beneficio. Por todo esto es que se necesita conocer esa madurez, porque de nada le sirve a una organización esta tecnología sino tiene los argumentos suficientes para implementarla y obtener todos sus beneficios. El objetivo de esta investigación es encontrarlo y entregarle a los probadores y a las empresas la información que les ayude a seleccionar el momento adecuado para introducirlo en su contexto de desarrollo. Este artículo se estructura de la siguiente forma: en la primera parte se hace un recorrido por las pruebas del software y su automatización, en la segunda se propone un modelo de madurez para el proceso de automatización de las pruebas, en la tercera se describen los trabajos relacionados, en la cuarta se presenta la metodología aplicada en la investigación y en la quinta se presentan y analizan los resultados.

**PRUEBAS Y AUTOMATIZACIÓN**

De acuerdo con ISO/IEEE [1] probar es aplicar una serie de actividades con el objetivo de descubrir y/o evaluar las propiedades de cada elemento del software. Estas actividades pueden incluir la planificación, la preparación, la ejecución, la presentación de informes y la gestión. Meyers [2] establece que las pruebas del software son el proceso de ejecución de un programa con la intención de encontrarle errores, y Hass [3] 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 más bien descubrir situaciones que podrían afectar negativamente su funcionamiento [4, 5]. 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, que se producían 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. Aunque generalmente 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 y relacionados con las plataformas y redes involucradas. Actualmente estas herramientas han madurado y ampliado su capacidad y siguen apareciendo otras con fortalezas y nichos específicos. Además, la prueba automatizada del software 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 escenario y de *scripts*.

La base en todo ese proceso de las pruebas 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. Además, debe ser económico para realizar, analizar y depurar errores y debe 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 prueba no consiste solamente en asegurar que los casos de prueba encuentren muchos defectos, sino también en asegurar que estén bien diseñados para evitar costos y tiempos excesivos.

Una forma de alcanzarlo es a través de la automatización, considerada actualmente por los equipos de prueba como una opción importante. 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 la prueba 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 más barato será ponerlo en práctica a largo plazo. 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 puedan 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 prueba. 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, cómo mantenerlas y en última instancia los beneficios que se puedan lograr.

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 [6]. 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 [1]. Sin embargo, 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 [3] 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 problemas causados por: 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 técnicas de automatización debe aumentar la productividad del equipo, porque de otra manera no se compensaría el costo de hacerlo [3].

**Modelo de madurez del desarrollo de la automatización de las pruebas**

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 propuesta en este trabajo es determinar la madurez del proceso de automatización de las pruebas del software analizándolo 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 el 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 aún así todavía se desvía de sus responsabilidades. Las características de este nivel son:

- 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.
- 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.

**TRABAJOS RELACIONADOS**

Diversos investigadores han presentado revisiones acerca de la automatización de las pruebas del software con diversos resultados. El estudio de Michael Grottke [7] es un esfuerzo de investigación y desarrollo de un consorcio de tres industrias y un cuerpo de académicos. El objetivo 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 le 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 se realiza a los modelos previos. Aunque analiza la automatización de las pruebas desde una perspectiva estadística, este trabajo 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 [8] es averiguar lo que se puede aprender de otros modelos de mejoramiento de los procesos de prueba. Para materializarla presenta una comparación entre TMM y otros siete modelos de mejoramiento con el objetivo de extraer prácticas importantes para desarrollar otro modelo. El resultado es una descripción y comparación de los modelos desde sus objetivos, estructuras, áreas clave del proceso y procedimientos de evaluación. Los resultados fueron insumos que 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 [9] 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 de software. El autor concluye que hay necesidad de modificar la mirada a los modelos de procesos de prueba 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, 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 [10] 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 para esto. El autor hace una comparación y valida la eficiencia de los modelos TIM, TPI y TMM e intenta determinar su nivel de madurez. Su objetivo fue encontrar una guía para ayudarle a las grandes empresas de desarrollo a mejorar la calidad de sus productos. La matriz de comparación utilizada es amplia y los indicadores ajustados, pero el hecho de que sólo trabaje con tres modelos no le permite presentar una imagen amplia del nivel de madurez de los modelos relacionados con la madurez de las pruebas de 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 [11] observa que la mejora de procesos del software se perfila como uno de los mayores retos para las empresas y 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 del 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 de las pruebas en las PyMes Web.

Para Christiane von Wangenheim et al [12] el nivel de madurez de la evaluación y el mejoramiento de los procesos 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. En este trabajo se presenta 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 torno al marco CMM/CMMI y la norma ISO/IEC 15504. Aunque los autores analizan el nivel de madurez su objetivo es mostrar una matriz comparativa entre los 29 modelos evaluados, por lo que no presenta una evaluación a la madurez de la automatización de las pruebas del software como área de investigación y desarrollo.

Heiskanen, Maunumaa y Katara [13] presentan un estudio en el que muestran que la generación de pruebas automatizadas está ganando popularidad en la industria del software, debido a los beneficios en el ahorro de tiempo y su capacidad para lograr mayor cobertura. Sin embargo, la introducción de esta tecnología no siempre se realiza con éxito, en parte porque los procesos

de prueba y 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.

Cecilia García, Abraham Dávila y Marcelo Pessoa [14] 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 a lograrla. 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 TMMi y TPI con diferentes arquitecturas y desde la norma ISO/IEC/IEEE 29119 como enfoque arquitectónico alineado con otros modelos.

Estos modelos se caracterizan por el dominio o el contexto en el que se definen, una referencia a la fuente de datos donde se describen, el modelo de la fuente o fuentes en las que se basan y una identificación secuencial. 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 del software en la industria y la investigación.

En el trabajo de Ana Furtado, Silvio Meira y Marcos Gomes [15] se lee que la práctica de las pruebas de software es una de las maneras de producir productos de calidad, y que la automatización de estas pruebas puede ser vista como una solución para probar la mayor cantidad de software en un tiempo determinado. Este trabajo presenta una descripción del desarrollo de un modelo de madurez para la automatización de las pruebas de software, y presenta la estructura y el plan de validación del mismo. Los investigadores concluyen que las organizaciones se han dado cuenta que pueden utilizar la automatización para lograr mayores niveles de calidad, pero los modelos de madurez que utilizan les dan poca o ninguna orientación sobre cómo implementarla.

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 la investigación y el desarrollo de la industria del software.

**METODOLOGÍA**

Con el fin de encontrar el nivel de madurez actual de la automatización de las pruebas del software se realizó una revisión sistemática de la literatura siguiendo los procedimientos descritos por Serna M. [16]. La pregunta de investigación central 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 examinaron todos los modelos de madurez de la automatización de las pruebas publicados y disponibles en las bases de datos entre 1990 y 2014; se consultaron los trabajos que presentan análisis comparativos a la eficiencia y eficacia de los modelos y el nivel de aceptación y proyección, y los aportes que hicieran observación a la madurez de la automatización, los modelos o los procesos de prueba del software. Los trabajos seleccionados se limitaron solamente 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 el proceso de pruebas del software, no hiciera referencia a una matriz de comparación o no analizara el nivel de madurez del proceso de automatización. 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, ISI web of Science, ScienceDirect y Wiley Interscience.

## RESULTADOS

### *Modelos de madurez para las pruebas del software*

La búsqueda se realizó entre enero y abril de 2015 y en total se recuperaron 978 trabajos. En una primera etapa se revisaron rápidamente los títulos y resúmenes, se descartaron los irrelevantes y/o duplicados y los que no describían completamente el modelo propuesto. 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 del proceso de 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 modelos analizados tienen en cuenta la automatización, aunque algunos presentan una orientación marcada.

**Tabla 1**. Modelos de madurez para las pruebas del software

| **Modelo** | **Año** | **Orientación** | **Niveles** | **Escuela de pruebas** [17] |
|---|---|---|---|---|
| MMAST [18] | 1994 | Automatización | 4 | Factory school |
| TAP [19-21] | 1995 | Evaluación | 5 | Test-driven school |
| TCMM [22] | 1996 | Capacidad | 4 | Quality School |
| TSM [23] | 1996 | Capacidad | 3 | Quality School |
| TMM [24, 25] | 1996 | Procesos | 5 | Context-drive school |
| TIM [26] | 1998 | Mejoramiento | 5 | Standard School |
| TOM [27] | 1998 | Mejoramiento | 3 | Standard School |
| TPI [28] | 1999 | Mejoramiento | 3 | Standard School |
| ATLM [29] | 1999 | Automatización | 5 | Factory school |
| MB-V²M² [30] | 2002 | Verificación y Validación | 5 | Control school |
| CB-VVCM [31] | 2005 | Verificación y Validación | 5 | Control school |
| SAMM [32] | 2009 | Aseguramiento | 4 | Context-drive school |
| CMMI-DEV [33] | 2010 | Calidad | 4 | Quality School |
| TMMi [34] | 2012 | Actividades de prueba y desarrollo | 5 | Analytical school |
| MPT.BR [35] | 2012 | Mejores prácticas | 5 | Agile School |
| ISO/IEC/IEEE [36] | 2013 | Estándares | 6 | Control school |

### *Eficacia y eficiencia de la automatización*

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, y se convierte en la base para determinar el nivel de madurez actual de la misma. Esta

valoración de los modelos se resume de los trabajos recolectados en los que: se describe el modelo o se analiza su aplicación en casos de la industria [8, 37-39, 41, 43, 44]; el nivel de aceptación es la ponderación a la valoración que los autores hacen de cada uno [9, 39, 40, 42]; y la proyección demuestra si el modelo tiene una vida útil referente o si fue efímero su paso por la industria [9, 37, 39, 40, 44].

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

| Modelo | Eficacia y eficiencia | 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-V²M² | 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 |

### *Madurez del desarrollo 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.

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

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

## 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 de las que califica los modelos como deficientes o ineficaces es porque han intentado saltarse niveles para llegar a los superiores.

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 [45-47] 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.*

**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. Luego de realizar y presentar los resultados de esta investigación podemos concluir que:

- Es importante definir el propósito de iniciar un esfuerzo de automatización de 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.

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.

## ANEXOS

| Institución Universitaria | FICHA TÉCNICA TRABAJO DE GRADO PARA REGISTRO EN EL SISTEMA DE INFORMACIÓN ACADÉMICA- SIA | Código | FDE 098 |
|---|---|---|---|
| | | Versión | 02 |
| | | Fecha | 2015-02-16 |

### 1. INFORMACIÓN GENERAL

**Título**

NIVEL DE MADUREZ DE LA AUTOMATIZACIÓN DE LAS PRUEBAS DEL SOFTWARE

**Objetivo**

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 de las que califica los modelos como deficientes o ineficaces es porque han intentado saltarse niveles para llegar a los superiores.

| **Plazo:** | **Inicio** | DD 25 | MM 08 | AA 2015 | **Fin** | DD 25 | MM 02 | AA 2016 |
|---|---|---|---|---|---|---|---|---|

| **Intensidad Horaria Semanal** | **16 Horas** | | | |
|---|---|---|---|---|
| **Horas Práctica Social** | | | | |
| **En funcionamiento – Negocio Incubado** | SI | | NO | X |

### 2. PERSONAL Y EMPRESA

| **Empresa** | **Juan Camilo Botero Acevedo** |
|---|---|
| **Representante** | **Juan Camilo Botero Acevedo** |
| **Cargo** | Estudiante de Ingeniería en Sistemas |
| **Documento** | 98706528 |
| **Dirección** | Carrera 59 A # 52 a 12 Bello |
| **E-mail** | Jcabo65@gmail.com |
| **Teléfono** | 3113200062 |
| **Razón Social** | Trabajador Independiente |
| **Asesor** | Edgar Serna |
| **Jurado** | |

### 3. DESCRIPCIÓN Y ALCANCE

**Descripción**

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. Además, que facilite la construcción de

1

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.

**Alcance**

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.

## 4. RECURSOS

**Recursos**

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.

## 5. PARTICIPANTES

| Nombre | Cedula |
|---|---|
| Juan Camilo Botero Acevedo | 98706528 |
| **Observación** | |
| | |

| |
|---|
| |

**6. SEGUIMIENTO**

**Seguimiento**

| | | | | |
|---|---|---|---|---|
| | | | | |
| **Deserción** | | | | |
| **Vinculación Laboral** | | | | |
| **Práctica Profesional** | | | | |
| **Trabajo de Grado Terminado** | SI | | NO | |
| **Visita Empresarial Realizada** | SI | | NO | |

3

**REFERENCIAS**


[1] ISO/IEEE. ISO/IEEE 29119 – Part I International Standard. Software and systems engineering/software testing, concepts and definitions. IEEE, USA. 2013.

[2] J. Myers. "The art of software testing". John Wiley and Sons. New York, USA. 1979.

[3] A. Hass. “A guide to advanced software testing”. Artech House. Boston, USA. 2008.

[4] E. Serna M. “Functional test of software - A constant Verification process”. Fondo Editorial ITM. Medellín, Colombia. 2013.

[5] E. Serna M. y F. Arango. “Effectiveness analysis of the set of test cases generated with the Requirements by Contracts technique”. En R. Villa y L. Orozco (Eds.), V Congreso Colombiano de Computación (pp. 1-6). April 14-16, Cartagena, Colombia. 2010.

[6] ISTQB. “Standard glossary of terms used in software testing”. International Software Testing Qualifications Board. Munich, Germany. 2012.

[7] M. Grottke. “Software Process Maturity Model study”. IST-1999-55017. PETS Project. 1999.

[8] R. Swinkels. “A comparison of TMM and other Test Process Improvement Models”. 12-4-1-FP Report. Frits Philips Institute, Technische Universiteit Eindhoven. Eindhoven, Netherlands. 2000.

[9] S. Kulkarni. “Test process maturity models - Yesterday, today and tomorrow”. Proceedings of the 6th Annual International Software Testing Conference (pp. 1-15). Delhi, India. 2006.

[10] G. de Souza. “Modelo de maturidade em testes com foco em ambientes de testes hetorogêneos”. Dissertação (mestrado) - Universidade Federal de Pernambuco. Pernambuco, Brasil. 2007.

[11] M. Sulayman. “A systematic literature review of software process improvement for small and medium web companies”. Communications in Computer and Information Science. Volume 59, pp. 1-8. 2009.

[12] C. von Wangenheim, J.C. Rossa, C. Salviano y A. von Wangenheim. “Systematic literature review of software process capability/maturity models”. Proceedings International Conference on Software Process Improvement and Capability Determination. May 18-20, Pisa, Italy. 2010

[13] H. Heiskanen, M. Maunumaa y M. Katara. “A Test Process Improvement Model for Automated Test Generation”. En: O. Dieste, A. Jedlitschka y N. Juristo (Eds.), PROFES 2012, LNCS 7343 (pp. 17-31). Springer-Verlag. Berlin, Germany. 2012.

[14] C. García, A. Dávila y M. Pessoa. “Test process models: Systematic literature review”. Communications in Computer and Information Science. Volume 477, pp 84-93. 2014.

[15] A. Furtado, S. Meira y M. Gomes. “Towards a maturity model in software testing automation”. En: H. Mannaert (Eds.), The Ninth International Conference on Software Engineering Advances (pp. 282-285). Curran Associates, Inc. New York, USA. 2014.

[16] E. Serna M. (In press). Methodology for perform reliable literature reviews. Revista Investigación Económica.

[17] C. Kaner. “Schools of software testing”. 2006. Marzo 2015. http://kaner.com/?p=15.

[18] M. Krause. “Maturity model for automated software testing”. Medical Device & Diagnostic Industry Magazine. December, pp. 1-10. 1994.

[19] P. Crosby. “Quality is free: The art of making quality certain”. McGraw-Hill. London, UK. 1979.

[20] M. Paulk, C. Mark, C. Weber, B. Curtis, y M. Chrissis. Capability Maturity Model. IEEE Software. Volume 10. Issue 4, pp. 18-27. 1993.

[21] M. Paulk, C. Weber, S. Garcia, M. Chrissis y M. Bush. “Key practices of the capability maturity model”. CMU/SEI-93-TR-25. Carnegie Mellon University. Pittsburgh, USA. 1993.

[22] S. Burgess y R. Drabick. “The I.T.B.G. Testing Capability Maturity Model”. (No more details). 1996.

[23] D. Gelperin. “A Testability Support Model”. Presented at the Fifth International Conference on Software Testing, Analysis & Review. May 13-17, Orlando, USA. 1996.

[24] I. Burnstein, T. Suwanassart y C. Carlson. “The Development of a Testing Maturity Model”. Proceedings of the Ninth International Quality Week Conference. May 21-24, San Francisco, USA. 1996.

[25] I. Burnstein, T. Suwanassart y C. Carlson. “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). 1996.

[26] T. Ericson, A. Subotic y Ursing. “TIM - A Test Improvement Model”. Software Testing Verification and Reliability. Volume 7, Issue 4, pp. 229-246. 1998.

[27] Systeme Evolutif. "Test Organization Maturity Model". 1998. Febrero 2015. http://gerrardconsulting.com/?q=node/485.

[28] T. Koomen y M. Pol. "Test process improvement: A practical step-by-step guide to structured testing". Addison-Wesley. New York, USA. 1999.

[29] E. Dustin, J. Rashka, J. Paul. "Automated Software Testing - Introduction, management, and performance". Addison-Wesley. Boston, USA. 1999.

[30] J. Jacobs y J. Trienekens. "Towards a metrics based Verification and Validation maturity model". Proceedings of the 10th International Workshop on Software Technology and Engineering Practice (pp. 1-6). October 6-8, Montréal, Canada. 2002.

[31] K. Yoon, S. Park, D. Bae. H. Chang y J. Jung. "A framework for the V&V capability assessment focused on the safety-criticality". 13th IEEE International Workshop on Software Technology and Engineering Practice (pp.17-24). September 24-25, Budapest, Hungary. 2005.

[32] P. Chandra. "Software assurance maturity model - A guide to building security into software development, Version - 1.0". Open Web Application Security Project (OWASP). 2009.

[33] CMMI-DEV. "CMMI for Development, Version 1.3". CMU/SEI-2010-TR-033, Software Engineering Institute, USA. 2010.

[34] TMMI. "Test Maturity Model Integration, Release 1.0". TMMi Foundation. 2012.

[35] A. Furtado, M. Wanderley, E. Carneiro e I. de Farias. "MPT.BR: A Brazilian maturity model for testing". Proceedings 12th International Conference on Quality Software (pp. 220-229). August 27-29, Xi'an, China. 2012.

[36] ISO/IEC. "Software and systems engineering – Software testing, Part 2: Test processes". ISO/IEC/IEEE 29119-2:2013. 2013.

[37] D. Mosley y B. Posey. "Just enough software test automation". Prentice Hall Professional. Boston, USA. 2002.

[38] G. Meszaros. "xUnit test patterns: Refactoring test code". Addison-Wesley. New York, USA. 2007.

[39] K. Bhaggan. "Test automation in practice". Master's Thesis Report. DSW Zorgverzekeraar, Department of ICT. Netherlands. 2009.

[40] M. Kohlegger, R. Maier y S. Thalmann. "Understanding maturity models results of a structured content analysis". Proceedings of I-KNOW '09 and I-SEMANTICS '09 (pp. 51-61). September 2-4, Graz, Austria. 2009.

[41] P. Kumar. "Test process improvement – Evaluation of available models". Maveric's Point of View Volume 8. Maveric. Egmore, India. 2012.

[42] K. Wiklund, S. Eldh, D. Sundmark y K. Lundqvist. "Technical debt in test automation". Fifth International Conference on Software Testing, Verification and Validation (pp. 887-892). April 17-21, Montreal, Canada. 2012.

[43] R. Balaraman y H. Krishnankutty. "Need for a comprehensive test maturity model". Infosys. Bangalore, India. 2013.

[44] S. Karthikeya y S. Rao. "Adopting the right software test maturity assessment model". Cognizant 20-20 Insights. Cognizant. New Jersey, USA. 2014.

[45] N. Nasir y S. Sahibuddin. "Critical success factors for software projects: A comparative study". Scientific Research and Essays. Volume 6. Issue 10, pp. 2174-2186. 2011.

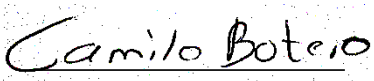
[46] The Standish Group International. "Modernization - Clearing a pathway to success". The Standish Group International Inc. Boston, USA. 2010.

[47] J. Cangussu, R. DeCarlo y A. Mathur. "A formal model of the software test process". IEEE Transactions on Software Engineering. Volume 28. Issue 8, pp. 782-796. 2002.

[48] D. Prado. “Project Management in Organizations”. Editora de Desenvolvimento Gerencial. Belo Horizonte, Brasil. 2003.

[49] Lionbridge. “Test process assessments move into the real world - A Rethinking QA white paper”. Lionbridge Technologies, Inc. Waltham, USA. 2009.

[50] I. Alsmadi. “Advanced automated software testing: Frameworks for refined practice”. Information Science Reference. New York, USA. 2012.



Juan Camilo Botero Acevedo
Estudiante

Prof. Edgar Serna M.
Asesor

FECHA ENTREGA: _____

_____
Comité trabajos de grado de la Facultad

RECHAZADO ____ ACEPTADO ___ ACEPTADO CON MODIFICACIONES ____

ACTA NO._____

FECHA ENTREGA: _____

_____
Consejo de Facultad

ACTA NO._____

FECHA ENTREGA: _____