# Análisis de Técnicas de Testing Aplicadas en Metodologías Ágiles

*Matías Emmanuel Perez y Ma. De los Milagros Gutiérrez*
*UTN-FRSF* *UTN-FRSF CIDISI*
*perezmatiase@gmail.com* *mmgutier@frsf.utn.edu.ar*

## Resumen

*Las metodologías ágiles han revolucionado la forma en la que el software es desarrollado y las mismas también han repercutido en la manera en que es probado. Asegurar la calidad nunca ha sido fácil debido a la gran diversidad de atributos que están puesto en juego, sin embargo bajo esta nueva metodología ahora resulta aún más complejo.*

*Este trabajo pretende hacer una revisión del estado del arte de las técnicas de testing que son aplicadas en proyectos guiados bajo metodologías agiles con el fin de recopilar aquellas técnicas y analizar su aplicación.*

*Finalmente, se concluye que bajo este escenario, es indispensable la automatización de casos de prueba y el apoyo de diversas herramientas con el fin de lograr asegurar la calidad. Además, resulta igual de importante el disponer de habilidades blandas como la comunicación efectiva, la coordinación y el trabajo en equipo por parte de todos los integrantes del mismo.*

## 1. Introducción

El empleo de metodologías ágiles en el desarrollo de software ha ido incrementándose desde su aparición, en respuesta a la eficiencia que brindan en entornos que cambian rápidamente y ante la necesidad de disminuir costos y tiempo de desarrollo. Particularmente, en lo que respecta a documentación, se plantea la necesidad de eliminar la creación de toda información que sea irrelevante, lo cual en muchos casos fue interpretado como "no documentar". Esto trajo consecuencias negativas, principalmente para la persona o equipo responsable de asegurar la calidad, colocándolos en una posición difícil de sortear ya que raramente se encuentran documentos actualizados, que describen el comportamiento actual del sistema con un grado de detalle suficiente (Dingsøyr et al. 2012, pág. 1213). Es por esto que en un mundo cambiante, aquellas personas responsables de asegurar la calidad del software deben cambiar junto a él. Las técnicas que comúnmente se describen en distintas bibliografías para asegurar la calidad del software no siempre son aplicables en la práctica y es por ello que el profesional de QA debe conocer en profundidad todas ellas para poder elegir la que mejor se adecúa a la situación que le toca resolver, de manera que le permita desempeñar sus tareas en equipos de desarrollo ágil sin perder el ritmo en el camino.

Este trabajo tiene como objetivo recopilar las técnicas de testing comúnmente mencionadas en numerosas bibliografías y analizar su aplicación en el contexto de desarrollo de software ágil, con el fin de identificar aquellas que permitan elevar rápidamente el grado de confiabilidad del software bajo validación.

Este trabajo está organizado como sigue: la sección 2 describe conceptos que presentan el marco teórico del trabajo. Luego en la sección 3 se presenta la metodología de revisión bibliográfica que se utilizó y se presenta una breve descripción de los resultados encontrados. En la sección 4 se presenta el análisis de estos trabajos, sintetizando los aspectos más importantes de cada uno. Finalmente el trabajo es concluido.

## 2. Marco Teórico

### Calidad de software

Existen diferentes definiciones del término calidad de software. Una definición más ampliamente usada es la dada en la norma ISO (ISO, 2005) que define calidad como el grado en "el cual el conjunto de características inherentes del producto cumple con los requerimientos". Para la IEEE la define como "el grado en el cual un software posee una combinación de atributos deseados" (IEEE, 1998). Según Pressman, (2010.) es "la concordancia del sistema con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente". Para Albin, (2003) es "una característica directamente relacionada con la habilidad del sistema para satisfacer sus requerimientos funcionales y no funcionales tanto implícitos como explícitos".

No hay dudas que evaluar la calidad del software, implica el cumplimiento de los requerimientos del sistema.

## 3. Metodología

Para realizar un correcto abordaje sobre las técnicas de testing aplicadas en metodologías Agiles se ha realizado una búsqueda sistemática de bibliografía que permita recopilar información y analizarla posteriormente.

Se utiliza para la búsqueda bibliográfica la metodología propuesta por Medina-Lopez y col. [6] que consta de cinco etapas: (1) Identificación del campo de estudio y período a analizar, (2) Selección de las fuentes de información, (3) Realización de la búsqueda, (4) Gestión y depuración de los resultados y (5) análisis de los resultados.

### 3.1 Campo de estudio y período a analizar

La primera de las etapas a desarrollar, consiste en identificar el campo de estudio que se busca analizar. En base a ello, se ha determinado que se desea afrontar el análisis bibliográfico de técnicas de testing aplicadas en metodologías agiles y como objetivo general que se establezca el estado del arte de dicho campo.

Para alcanzar dicho objetivo, es necesario que: (i) se identifiquen y analicen las herramientas y metodologías aplicadas, (ii) se identifiquen y analicen las estrategias de documentación y (iii) se realice un análisis de las ventajas y desventajas de cada uno de los elementos identificados.

Debido al dinamismo propio del campo de estudio, el período de trabajos seleccionados está comprendido dentro del año actual y los 5 años anteriores, es decir desde el año 2015 hasta los trabajos presentados durante el corriente año. Este lapso de tiempo se considera suficiente para realizar un correcto estudio de los temas involucrados y disponer de aquellas técnicas y metodologías que se estén usando actualmente.

### 3.2 Fuentes de Información

Se eligió trabajar principalmente con libros, secciones de libros, actas de conferencias, working papers, artículos de investigación y además artículos de revistas científicas y profesionales. Esta selección de producciones se realiza con la intención de consultar referencias que hayan sido sometidas a un proceso de revisión y proporcionen calidad en la información a ser procesada.

Las fuentes de información consultadas son las siguientes: ScienceDirect[1], SciELO[2], IEEE Xplore [3]y por último, la red social académica Mendeley[4]. En cuanto al idioma, serán válidos aquellos trabajos escritos en inglés o en español.

### 3.3 Realización de Búsqueda

La metodología propone analizar los criterios de búsqueda a emplear y la manera en la cual van a ser empleados estos criterios, es decir si será en forma manual o automática. Además sugiere la realización de una prueba piloto para ayudar a depurar y mejorar la estrategia de búsqueda establecida.

Se definió inicialmente como criterio de búsqueda las palabras claves “agile testing” y se realizó una búsqueda automática en una de las fuentes de información, en particular, en la base de datos de ScienceDirect. Bajo este criterio de búsqueda se obtuvieron 25.713 resultados (fecha de consulta 07/09/2020) lo cual manifestó la necesidad de refinar el criterio para esta base de datos y se hizo de la siguiente forma: (i) los resultados tendrán un año de publicación comprendido dentro del período 2015-2020, (ii) el tipo de publicación será “Journal of Systems and Software” o “Information and Software Technology” y (iii) el tipo de articulo será: “Review articles”, “Research articles”, “Book chapters”, “Conference abstracts”, “Book reviews”, “Conference Info” o “Data articles”.

Dado que, bajo este nuevo criterio se redujo considerablemente la cantidad de resultados obtenidos para esta fuente de información, la posibilidad de analizar 492 producciones representaba un desafío considerable y no era factible en términos de tiempo según la planificación realizada. Finalmente, se observó con esta prueba piloto que era necesario refinar aún más el criterio de búsqueda y se lo hizo forzando a que las palabras claves necesariamente estén presentes en el título del trabajo.

Como resultado, el criterio de búsqueda depurado fue: Titulo contiene “agile testing” ^ año de publicación = [2015; 2020] ^ Tipo de publicación = (“Journal of Systems and Software” ∨ “Information and Software Technology”) ^ Tipo de articulo = (“Review articles” ∨ “Research articles” ∨ “Book chapters” ∨ “Conference abstracts” ∨ “Book reviews” ∨ “Conference Info” ∨ “Data articles”).

Al realizar las búsquedas en las diferentes fuentes de información se han encontrado en total 31 resultados posibles, discriminados de la siguiente manera: 2 en ScienceDirect, 3 en SciELO, 11 en IEEE Xplore y finalmente 15 en Mendeley.

Es de mencionar que los 2 trabajos encontrados en ScienceDirect estaban incluidos dentro del universo de trabajos encontrados de Mendeley. Ante esta situación, se los dejó catalogados como encontrados en la fuente de ScienceDirect y se los eliminó de la lista de Mendeley.

### 3.4 Gestión y Depuración de los Resultados

Para esta etapa, se han exportado todos los resultados obtenidos y se los ha colocado en una tabla registrando la siguiente información: fecha y base de datos consultada, tipo de publicación, nombre, palabras claves, resumen (abstract), ubicación, categoría y comentarios.

Para asegurar que cada uno de los trabajos se refiere a los conceptos que realmente se pretendía buscar, se ha analizado el título, resumen (abstract), palabras claves y las conclusiones clasificando así los trabajos en una de las

[1] https://www.sciencedirect.com/search
[2] https://search.scielo.org/
[3] https://ieeexplore.ieee.org/search/advanced
[4] https://www.mendeley.com/

siguientes categorías: (i) seleccionado, (ii) falso positivo y (iii) dudoso.

Los trabajos catalogados como falsos positivos fueron trabajos que habían cumplido con los criterios de búsqueda, sin embargo por alguna razón no correspondían al objeto real de la misma, es decir, no resultaban de interés para éste análisis. En esta categoría se colocaron 9 de 31 trabajos y al analizar las causas se pudo determinar que fue debido a la similitud de conceptos sobre los que trataba. Las palabras claves eran muy similares a trabajos seleccionados.

En relación a los trabajos catalogados como dudosos, fue necesario realizar un análisis más detallado para poder discernir si pertenecía a un trabajo seleccionado o se trataba de un falso positivo. En esta categoría fueron colocados 3 de 31 trabajos y luego de examinarlos se decidió catalogarlos como falsos positivos debido a que abordaban el tema de interés pero desde una perspectiva que no lo era.

### 3.5 Descripción de los trabajos evaluados

En esta etapa se analizan los trabajos seleccionados exponiendo el tema del que tratan, realizando un análisis y comparando con los demás trabajos seleccionados y con el tema de la investigación de este artículo.

En [7] se establece que, pese a la gran cantidad de pruebas que pueden ser ejecutadas en los diferentes niveles de testing: pruebas de unidad, pruebas funcionales, pruebas de integración o pruebas de sistema, no es hasta que el cliente utiliza la aplicación, que se comprueba su adecuación a los requerimientos. Tradicionalmente esta validación puede realizarse en las metodologías agiles al final de cada sprint, sin embargo requiere una mayor participación del cliente en comparación con las metodologías tradicionales donde la participación se limitaba al principio y final del proceso del ciclo de vida de desarrollo. Se expone entonces que, pese al cambio en la metodología de desarrollo y testing, algunos clientes no están preparados para seguir este ritmo rápido y al momento de evaluar el producto lo hacen ineficazmente, en la medida de sus posibilidades.

A través de un estudio no sistemático de bibliografía detectaron que esta situación se producía debido a tres causas principalmente: (i) falta de tiempo, (ii) falta de motivación y (iii) falta de conocimientos por parte de los clientes para realizar las pruebas de aceptación (UAT). Para resolver estos problemas desarrollaron siete meta requerimientos que a través del empleo de wikis, mapas mentales y una estrecha colaboración asincrónica podían mejorar la calidad de las pruebas de aceptación que realizaba el cliente, beneficiándose de esta forma tanto el equipo de desarrollo y testing como el cliente en sí.

Es interesante lo que se plantea en este trabajo ya que deja entrever que por más recursos especializados que se tengan dentro del equipo de desarrollo y testing, existen pruebas que sólo pueden ser realizadas por el cliente y los resultados de éstas significan la aprobación o no del producto entregado poniendo en juego todos los recursos (léase como tiempo y dinero) involucrados. Por lo que el hecho de aprovechar el acercamiento de la figura del cliente que plantean las metodologías agiles, permite que a través de las reuniones y el apoyo de una biblioteca de conocimientos que en el trabajo se describen como wikis y mapas mentales, clientes y desarrolladores puedan colaborar activamente para elevar la calidad de las pruebas de aceptación y de esta manera, la calidad del producto entregado.

En [8] también se analiza cómo la información se ubica como pieza clave en el aseguramiento de la calidad y se aborda esta temática desde sus origines. Se analiza a través de cinco casos de estudio que involucran proyectos de diferentes magnitudes, cómo los mecanismos de coordinación, las actividades comprendidas dentro de la ingeniería de requerimientos y las pruebas del software pueden ser mejoradas. Sin embargo para llegar a este punto primero es necesario identificar en primer medida los factores a mejorar y para esto se muestra un método llamado REST-bench. Este método está basado en la premisa de que la información, la forma en la que se crea, utiliza y relaciona es la clave para comprender los requisitos y los pruebas necesarias.

Sin embargo una crítica que se le podría hacer al método anterior, es que las actividades de ingeniería de requerimiento en el desarrollo ágil todavía no es una práctica bien definida y entendida (Curcio et al. 2018). Debido a la naturaleza de la metodología ágil es difícil determinar el momento exacto en el que se puede dar finalizada la definición de un requerimiento y se encuentra listo para ser documentado.

En [10] se menciona que los requisitos detallados a menudo se encuentran documentados como casos de prueba en lugar de una especificación formal de requerimiento. En este trabajo se estudia a través del análisis de tres compañías, cómo los casos de pruebas pueden respaldar las actividades de requisitos principales y puntualizan en la diversidad de opciones para llevar a cabo esta práctica.

Algunas de estas opciones, como el caso de "behaviour-driven" permiten a través de una escritura estructurada, capturar los requerimientos y documentarlos como tests de aceptación como parte del proceso de elicitación, a su vez posibilitan con poco esfuerzo que las especificaciones sean ejecutables por una máquina, proporcionando también grandes ventajas en la actualización y mantenibilidad de los requerimientos y casos de prueba.

Otra de las opciones presentes en su caso de estudio fue la de "story-test driven" donde los requerimientos son documentados como historias de usuario y criterios de aceptación para lo cual test automáticos y manuales serán posteriormente creados. A pesar de estar expresados en una forma semiestructurada es posible automatizar algunos escenarios a través del uso de herramientas específicas.

Por último, estuvo presente en su observación una combinación de casos de prueba manuales y automáticos como la principal fuente de información de requisitos, ya que contenían la información más confiable y actualizada sobre los productos implementados.

En [11] también se aborda la problemática de la especificación de requisitos y pruebas en la metodología ágil, y coincide con el trabajo anterior en que la automatización de las pruebas es una práctica clave en el desarrollo ágil, sin embargo para las pruebas a nivel de sistema desarrollan un testing basado en modelo (MBT) para mejorar la eficiencia y la eficacia de los procesos de prueba, acelerando la creación de casos de prueba basándose en un procedimiento sistemático. Para ello, se utiliza una notación propuesta por los autores llamada CLARET (Central Artifact for RE and MBT). Esta notación crea especificaciones de casos de uso utilizando lenguaje natural como un artefacto central para: (i) generación de documentos de casos de uso formateados como parte de las prácticas de RE y (ii) modelos de prueba para MBT a nivel sistema.

Se puede expresar como causante de las situaciones anteriormente descriptas que las distintas actividades involucradas en el desarrollo de software tienen un ciclo de vida que está bien definido y estabilizado por la literatura y adoptado por las industrias. Sin embargo, cuando se trata de un proceso para el escenario de una metodología ágil, no sucede lo mismo.

En [12] se describe un caso de estudio donde manifiesta la complejidad de realizar una transición entre una metodología de desarrollo tradicional (en cascada) a una metodología ágil, en éste caso en particular, para Scrum.

Los autores detallan que fueron muchos los desafíos que tuvieron que afrontar, uno de ellos es la redefinición de los roles ya que figuras como "test manager" no aplicaban en la nueva metodología. Otro de los grandes desafíos fue la comunicación y colaboración entre los equipos distribuidos, situación para la que organizacionalmente exponen, no estaban preparados. Por último mencionan que la automatización de pruebas pasó a ser una necesidad y tuvieron que capacitarse para lograrla, al finalizar ese proceso, notaron mejoras importantes en todas las métricas de testing ya que anteriormente muchos test no se realizaban porque eran muy costosos en relación al tiempo que consumían y esto impactaba en la fecha de lanzamiento de un release, por lo que se decidía no hacerlo o bien hacerlo a pequeña escala donde obviamente la precisión era baja y no significativa en términos del sistema en general.

En [13] los autores analiza n esta falta de consenso en la literatura y realizan una investigación exploratoria intentando obtener información que les ayude a responder a dos interrogantes: (i) ¿Qué enfoques ágiles de implementación de pruebas están utilizando los investigadores y las industrias? y (ii) ¿Cuál es el proceso formal utilizado para insertar pruebas en equipos de desarrollo ágiles?

Como resultado, concluyen que el enfoque de test automáticos proporciona agilidad al proceso de validación de versiones que se entregarán a clientes, reduciendo así el tiempo de pruebas en cada sprint. La ejecución automática de test de regresión es indispensable para identificar defectos en funcionalidades que ya han sido probadas y han sido afectadas por la nueva funcionalidad y la posibilidad de integración continua proporciona la ejecución automática de un set de pruebas seleccionadas que serán ejecutadas en cada nueva versión del código.

En [14] se hace un análisis específico sobre los criterios para la selección de un subconjunto de pruebas adecuadas para su ejecución en la práctica de integración Continua (CI) con el fin de reducir el costo de las pruebas tanto como sea posible sin sacrificar la calidad. El criterio al que arriban los autores fue evaluado en 18 proyectos de código abierto con 261 versiones de integración continua de las comunidades Eclipse y Apache dando como resultado que la selección manual de casos de prueba a nivel de método suele ser más efectiva que la selección manual a nivel de clase. Es interesante destacar que se plantea como trabajo futuro un aprendizaje automático para mejorar el rendimiento de la selección de casos de prueba lo que permitiría mayores niveles de calidad.

En [15] también se analiza la integración continua (CI) como práctica que tiene como objetivo verificar continuamente los aspectos de calidad, pero los autores se enfocan en los siguientes requisitos no funcionales (NFR): (i) Latencia, (ii) estabilidad, (iii) escalabilidad, (iv) performance, (v) productividad, (vi) fiabilidad, (vii) eficiencia, (viii) mantenibilidad, (ix) disponibilidad.

Se puede observar que muchos de los atributos analizados en este trabajo corresponden a los que se mencionan en ISO 25010 (ver **Figura 1**)

La complejidad que exponen es que los requisitos funcionales son los insumos del desarrollo y pueden ser probados individualmente, sin embargo algunos NFR son difíciles de probar ya que a menudo son aspectos de funcionalidad y expresan aspectos de calidad.


**Figura 1. Características de Calidad según ISO/IEC 25010**

Los autores estudian en este trabajo el estado del arte en la utilización de CI para las pruebas NFR a través de una selección de 747 artículos donde identificaron 47 de interés. Como resultado, propusieron un marco de CI sintetizado para probar los NFR mencionados, mapeando también aquellas herramientas utilizadas en la industria.

Esta contribución resultó valiosa ya que los resultados del estudio muestran la factibilidad del testeo de algunos NFR en CI en la práctica, pero sin duda aún queda pendiente el estudio de algunos en particular para arribar a resultados concluyentes.

En [17] se tratan también los atributos no funcionales, pero ya no desde la perspectiva de la integración continua sino desde el punto de vista de la percepción de aquellos miembros responsables de realizar las pruebas no funcionales en sus proyectos.

Los autores realizaron un estudio de investigación empírico donde a través de 20 entrevistas a profesionales de TI de una gran empresa multinacional identificaron siete factores principales que influyen en las pruebas no funcionales, estos factores son: (i) experiencia, (ii) cultura, (iii) conciencia, (iv) prioridad, (v) costo, (vi) presión de tiempo y (vii) problemas técnicos. Para superar estos desafíos, en el trabajo se expresa que fue necesario adoptar cuatro prácticas las cuales fueron: (i) discutir los aspectos no funcionales durante el inicio del proyecto, la planificación del sprint y el desarrollo de historias de usuario, (ii) disponer un equipo multidisciplinario (desarrolladores, testers, arquitectos de software y product owners) que revise los requerimientos no funcionales y las necesidades de testing, (iii) comunicación frontal y honesta entre los miembros del equipo y cliente y (iv) trabajar con una mentalidad de calidad, evangelizando al equipo con respecto a la importancia de los requisitos no funcionales y pruebas.

En [18] se realiza un estudio exploratorio para determinar cómo impacta el análisis del grado de cobertura ante un código que fue afectado por una refactorización. Los autores plantean que cuando se refactoriza una parte del sistema, un conjunto de pruebas pueden no ser lo suficientemente robusto para revelar fallas.

Según los proyectos que estuvieron bajo análisis, se concluyó que un conjunto de pruebas que no llame al método refactorizado y/o a las secciones de código que lo relacionan aumentan la posibilidad de que la falla introducida no sea detectada. Sin embargo, existen mayores posibilidades de detectar una falla de refactorización cuando el grado de cobertura es alto.

Sin lugar a dudas los trabajos anteriores han expuestos algunas problemáticas críticas a la hora de asegurar la calidad en un contexto tan dinámico, pero no se debe perder de vista que son situaciones que la misma realidad impulsa. Un claro ejemplo de esta dinámica es el mercado de software para dispositivos móviles.

En [19] se analiza esta situación mencionando algunas características particulares de este mercado, donde las demandas específicas y las características de los diferentes dispositivos deberían ser consideradas en el desarrollo de las aplicaciones, donde es importante que el producto posea calidad desde el comienzo. Para ello, la automatización de pruebas de software en forma temprana, las pruebas funcionales e incluso de carga incluyendo a la integración continua como parte de las pruebas de regresión libera a los testers para concentrarse en casos de uso más complejos o bien ejecutar aquellas pruebas que debido a las características intrínsecas no son posibles de ser automatizadas.

Finalmente, se concluye expresando que realmente es un desafío mantener alineadas todas las especificaciones cuando se sigue una metodología ágil, donde en un corto período de tiempo el equipo debe entregar al cliente un producto funcional, pero sin lugar a dudas la automatización de pruebas es una de las vías para lograrlo.

En [20] se investiga el esfuerzo de la estimación del proceso de testing para aplicaciones móviles a través de una revisión sistemática de la literatura y una encuesta. El objetivo es identificar a través de la literatura, cómo la estimación del esfuerzo de prueba para aplicaciones móviles es distinta de cualquier otro software. Además se plantea identificar los problemas de adaptación de los métodos tradicionales de estimación a este contexto en particular.

Luego de un extenso trabajo, se concluye que el análisis de punto de función / punto de prueba es una técnica de estimación de prueba tradicional altamente adaptable al dominio móvil, ya que sortea muchos de los desafíos. Sin embargo, los requisitos inciertos, la falta de soporte de las herramientas para la estimación de pruebas, la complejidad de las pruebas y la comunicación con el cliente siguen estando presentes como puntos a superar.

En lo que respecta a la suite de pruebas, la adopción de enfoques ágiles ha impulsado esta práctica lo que en algunos casos genera una extensa variedad de test.

En [21] se analizan trabajos que explotan esta situación, ya que cada test engloba un conocimiento / regla de negocio y es posible mejorar la suite de pruebas generando automáticamente nuevas pruebas.

El análisis tomó en consideración 70 trabajos de los cuales sólo 4 fueron seleccionados como semillas para un estudio de snowballing, al finalizarlo luego de 6 iteraciones la cantidad de trabajos con los que se trabajó fue de 49.

A partir del estudio de la literatura seleccionada, se describieron muchas técnicas utilizadas en la práctica y que generan nuevos casos de pruebas. El espectro que se cubrió abarco desde ejecuciones simbólicas hasta búsqueda random y modificación de casos de prueba en tiempo de ejecución.

Finalmente, se concluye que una comparación experimental sólida y sistemática entre artículos de “test de amplificación” y técnicas tradicionales de generación de casos de pruebas proporcionaría un hito dentro del campo.

En lo que respecta a la seguridad del software, las técnicas de testing aplicadas en metodologías ágiles deben

lograr evaluar la calidad y detectar fallas, incluso cuando en el Agile manifiesto [22] y sus principios [23] se enfatice en entregar software continuamente en diversas iteraciones aceptando cambios en los requisitos en cualquier momento. Este tema desde la perspectiva de la seguridad es complejo ya que esta forma de trabajar plantea nuevos desafíos e involucra a diferentes unidades de negocio.

En [24] se aborda esta problemática a través de un caso de estudio, donde enfatiza en la necesidad de la independencia del equipo que analiza la seguridad de la aplicación pero que también cumpla con el aseguramiento de la seguridad del software bajo testing. Por estas razones, se identificaron 4 categorías donde se programan las actividades de pruebas de seguridad, ellas son: (i) testeo de la seguridad luego del primer sprint, (ii) testeo de la seguridad en cada sprint posterior, (iii) testeo de la seguridad con la ejecución de tareas específicas cada cierto periodo de tiempo y (iv) testeo de la seguridad en cada reléase.

En el caso de estudio se explican todos los obstáculos que debieron sortear y cómo la incorporación de software específico de seguridad facilitó la automatización de determinadas pruebas. Además se concluyó que un testing temprano trae aparejado resultados tempranos lo que permite ir ajustando el rumbo durante el proceso de desarrollo.

Para finalizar con la descripción de los resultados obtenidos, hemos observado que las metodologías agiles intentan satisfacer las necesidades cambiantes del mercado. En este contexto es necesario que las herramientas que utilice el equipo estén a la altura de la circunstancias

En [25] se realiza una investigación junto con entrevistas cualitativas semiestructuradas en 18 organizaciones, para determinar cuáles herramientas se utilizan para le entrega de software, cuáles son las razones por las que se omiten ciertas partes de la cadena y qué implicaciones tienen en la rapidez con la que el software llega a los clientes.

Como resultado, se observa que existe una gran variedad de herramientas (ver **Figura 2**) en los diferentes pasos de desarrollo y testing pero que la entrega rápida de nuevas funcionalidades está relacionada con una utilización eficiente de las herramientas a lo largo de toda la cadena.

Por último, se puede ver que es tan compleja la actividad de software que en algunos casos aún no está disponible suficiente cantidad de trabajos que orienten sobre alguna temática puntual.

En [26] se analizaron 5432 trabajos de los cuales sólo fueron seleccionados 128 para realizar una revisión sistemática sobre la ingeniería de software para sistemas ubicuos. En base a su análisis concluyen que son necesarios más trabajos de investigación relacionados al concepto de contexto, soporte en tiempo de ejecución, interacción del usuario, soporte adecuado a desarrolladores y testers, seguridad y privacidad. Además se menciona que la fase de testing necesita recibir más atención especialmente en los aspectos de las simulaciones con las cuales se prueba este tipo de sistemas.

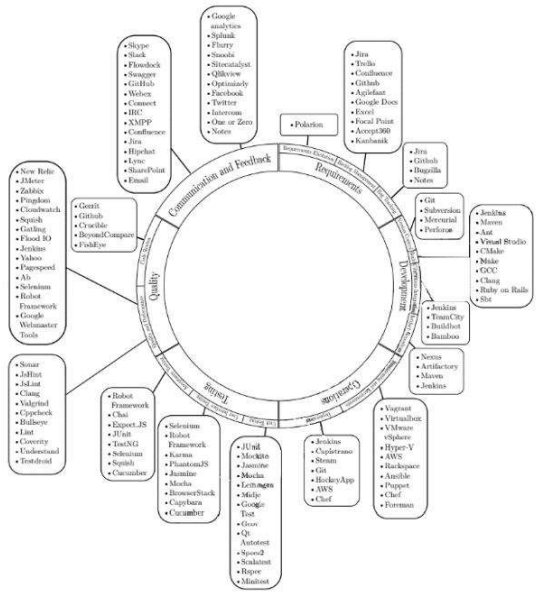


**Figura 2: Resumen de herramientas utilizadas en cada etapa de desarrollo. Extraído de [25]**

## 4. Análisis

En base a los trabajos seleccionados, se aprecia que en todos ellos la automatización de casos de pruebas está presente.

En algunos trabajos se la muestra como medio para validar que nuevas fallas no hayan sido introducidas y se profundiza sobre las herramientas que permiten realizar esta integración continua del código. Uno de los trabajos puntualmente, profundiza esta cuestión tomándolo como punto de partida y analiza diferentes enfoques que puede seguir la persona responsable de asegurar la calidad para elegir aquellos test que validarán el código bajo esta dinámica.

En otros trabajos en cambio, se estudian mejoras en los mecanismos coordinadores debido a que la comunicación efectiva es clave en las metodologías agiles. Estas mejoras en algunas ocasiones son manifestadas en el acuerdo mutuo de la escritura de documentación en bajo cierto nivel de estructuración y que posibilita que nuevos procesos automáticos sean disparados con el fin de asegurar la calidad.

Algo a destacar es que un solo trabajo de los seleccionados desarrolló el atributo de la seguridad, lo que da a suponer que no es una práctica muy estudiada en líneas generales. Caso contrario fue el análisis de los requerimientos no funcionales, donde varios trabajos han puesto su foco de investigación proporcionando grandes avances al incorporar herramientas ejecutables en la práctica.

## 5. Conclusiones

Uno de los problemas que se detectan en el aseguramiento de la calidad de piezas de software que son entregadas al cliente, es que no se conozcan todas las interacciones o alcance que dicha pieza tenga. Las personas o equipos encargados de garantizar la calidad, deben descubrir las vulnerabilidades críticas que podrían afectar el rendimiento y el funcionamiento del software como la seguridad de los datos que el mismo administra. De esto también depende la imagen y prestigio de la empresa que utiliza el software para la ejecución de sus actividades diarias.

Con procesos de desarrollos tradicionales resultaba imposible lograr especificar los requisitos funcionales y no funcionales antes de comenzar las etapas de diseño e implementación, motivo por el cual, diseñar pruebas eficaces que aseguren realmente la calidad del software resultaba aún más inalcanzable trabajando de esta manera.

La aparición de las metodologías agiles mejoraron muchas de las falencias ampliamente estudiadas y discutidas en las metodologías tradicionales a través de un acercamiento del equipo de desarrollo y prueba con el cliente y un ciclo de desarrollo más pequeño. Sin embargo, estas nuevas metodologías impactaron enormemente en las tareas de aseguramiento de la calidad ya que los procesos tradicionales de testing no cabían dentro del plazo de una iteración y esta práctica también tuvo que ser pensada nuevamente y redefinida.

Como resultado, el testing cambió a ser realizado lo más temprano posible, testeando a medida que el código esté disponible y sea lo suficientemente estable. No obstante, esta nueva práctica descuida otros tipos de testing tales como los de performance, seguridad, usabilidad, entre otros cuando en realidad estos tipos de testing son igualmente importantes en un proyecto ágil, como también lo son en otros proyectos que utilizan otras metodologías de desarrollo.

Para intentar suplir esta cuestión, se han presentado diversas técnicas y herramientas para asegurar la calidad en estos nuevos escenarios. En este espacio, la automatización de casos de prueba toma el rol de la figura principal, siendo acompañada por técnicas de integración continua, ejecución automática de pruebas de regresión, generación de nuevos casos de prueba en forma automática a partir de existentes y además técnicas de documentación que sientan las bases para la gestión y comunicación con los demás miembros del equipo.

Actualmente es difícil encontrar trabajos que abarquen todos estos conceptos en su conjunto, dado que se continúan desarrollando trabajos en una sola dirección para un tema específico. Por tal motivo se plantea como trabajo futuro realizar un aporte en ese sentido, que permita acercar la brecha entre aquellas explicaciones teóricas desarrolladas ampliamente en libros de pruebas de software y la realidad que enfrenta el profesional en la industria del software permitiendo así mejorar el testing realizado al evaluar la calidad del software de manera integral.

## Referencias


[1] Dingsøyr, Torgeir; Nerur, Sridhar; Balijepally, VenuGopal; Moe, Nils Brede (2012): A decade of agile methodologies: Towards explaining agile software development. En: Journal of Systems and Software 85 (6), pág. 1213–1221. DOI: 10.1016/j.jss.2012.02.033

[2] ISO, "Quality management, 1ª Edición, "ISO, 2005.

[3] IEEE Standard for a Software Quality Metrics Methodology Electrical Electronic Engineering, 1998.

[4] R. Pressman, Software Engineering: A Practitioner´s Approach, 7th ed. McGraw-Hill, 2010.

[5] S. Albin, The art of software architecture: design methods and techniques. John Wiley & Sons, 2003

[6] Medina-Lopez, C., Marin-Garcia, J., & Alfalla-Luque, R. (2010). Una propuesta metodológica para la realización de búsquedas sistemáticas de bibliografía (A methodological proposal for the systematic literature review). WPOM-Working Papers on Operations Management, 1(2), 13-30. doi:https://doi.org/10.4995/wpom.v1i2.786

[7] Otaduy, I.; Diaz, O. (2017): User acceptance testing for Agile-developed web-based applications: Empowering customers through wikis and mind maps. En: Journal of Systems and Software 133, pág. 212–229. DOI: 10.1016/j.jss.2017.01.002.

[8] Unterkalmsteiner, Michael; Gorschek, Tony; Feldt, Robert; Klotins, Eriks (2015): Assessing requirements engineering and software test alignment—Five case studies. En: Journal of Systems and Software 109, pág. 62–77. DOI: 10.1016/j.jss.2015.07.018.

[9] Curcio, Karina; Navarro, Tiago; Malucelli, Andreia; Reinehr, Sheila (2018): Requirements engineering: A systematic mapping study in agile software development. En: Journal of Systems and Software 139, pág. 32–50. DOI: 10.1016/j.jss.2018.01.036.

[10] Bjarnason, Elizabeth; Unterkalmsteiner, Michael; Borg, Markus; Engström, Emelie (2016): A multi-case study of agile requirements engineering and the use of test cases as requirements. En: Information and Software Technology 77, pág. 61–79. DOI: 10.1016/j.infsof.2016.03.008.

[11] Jorge, Dalton N.; Machado, Patricia D. L.; Alves, Everton L. G.; Andrade, Wilkerson L. (2018 - 2018): Integrating Requirements Specification and Model-Based Testing in Agile Development. En: 2018 IEEE 26th International Requirements Engineering Conference (RE). 2018 IEEE 26th International Requirements Engineering Conference (RE). Banff, AB, 19/08/2018 - 23/08/2018: IEEE, pág. 336–346.

[12] Gupta, Rajeev Kumar; Manikreddy, Prabhulinga; GV, Abhinandan (2016 - 2016): Challenges in Adapting Agile Testing in a Legacy Product. En: 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE). 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE). Orange County, CA, USA, 01/08/2016 - 04/08/2016: IEEE, pág. 104–108.

[13] Menendez, Danielle Amaral; Menendez, Elisa Sousa; Sousa, Thiers Garretti Ramos; Silva, Paulo Caetano da (2015 - 2015): Experiment Report on the Implementation of Agile Testing. En: 2015 12th International Conference on Information Technology - New Generations. 2015 12th International Conference on Information Technology - New Generations

(ITNG). Las Vegas, NV, USA, 12/04/2015 - 14/04/2015: IEEE, pág. 772–773.

[14] Li, Yingling; Wang, Junjie; Yang, Yun; Wang, Qing (2020): An extensive study of class-level and method-level test case selection for continuous integration. En: Journal of Systems and Software 167, pág. 110614. DOI: 10.1016/j.jss.2020.110614.

[15] Yu, Liang; Alégroth, Emil; Chatzipetrou, Panagiota; Gorschek, Tony (2020): Utilising CI environment for efficient and effective testing of NFRs. En: Information and Software Technology 117, pág. 106199. DOI: 10.1016/j.infsof.2019.106199.

[16] ISO: ISO/IEC 25010. Disponible en línea en https://iso25000.com/index.php/normas-iso-25000/iso-25010.

[17] Camacho, Cristina Rosa; Marczak, Sabrina; Cruzes, Daniela S. (2016 - 2016): Agile Team Members Perceptions on Non-functional Testing: Influencing Factors from an Empirical Study. En: 2016 11th International Conference on Availability, Reliability and Security (ARES). 2016 11th International Conference on Availability, Reliability and Security (ARES ). Salzburg, Austria, 30/08/2016 - 01/09/2016: IEEE, pág. 582–589.

[18] Alves, Everton L.G.; Massoni, Tiago; Machado, Patrícia Duarte de Lima (2017): Test coverage of impacted code elements for detecting refactoring faults: An exploratory study. En: Journal of Systems and Software 123, pág. 223–238. DOI: 10.1016/j.jss.2016.02.00

[19] Santos, Andreia; Correia, Igor (2015 - 2015): Mobile Testing in Software Industry Using Agile: Challenges and Opportunities. En: 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST). 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST). Graz, Austria, 12/04/2015 - 16/04/2015: IEEE, pág. 1–2.

[20] Kaur, Anureet; Kaur, Kulwant (2019): Investigation on test effort estimation of mobile applications: Systematic literature review and survey. En: Information and Software Technology 110, pág. 56–77. DOI: 10.1016/j.infsof.2019.02.003.

[21] Danglot, Benjamin; Vera-Perez, Oscar; Yu, Zhongxing; Zaidman, Andy; Monperrus, Martin; Baudry, Benoit (2019): A snowballing literature study on test amplification. En: Journal of Systems and Software 157, pág. 110398. DOI: 10.1016/j.jss.2019.110398.

[22] Beck, Kent et al.: Manifesto for Agile Software Development. Disponible en línea en http://agilemanifesto.org/

[23] Beck, Kent et al.: Principles behind the Agile Manifesto. Disponible en línea en http://agilemanifesto.org/principles.html.

[24] Choliz, Jesus; Vilas, Julian; Moreira, Jose (2015 - 2015): Independent Security Testing on Agile Software Development: A Case Study in a Software Company. En: 2015 10th International Conference on Availability, Reliability and Security. 2015 10th International Conference on Availability, Reliability and Security (ARES). Toulouse, France, 23/08/2015 - 26/08/2015: IEEE, pág. 522–531.

[25] Mäkinen, Simo; Leppänen, Marko; Kilamo, Terhi; Mattila, Anna-Liisa; Laukkanen, Eero; Pagels, Max; Männistö, Tomi (2016): Improving the delivery cycle: A multiple-case study of the toolchains in Finnish software intensive enterprises. En: Information and Software Technology 80, pág. 175–194. DOI: 10.1016/j.infsof.2016.09.001.

[26] Sánchez Guinea, Alejandro; Nain, Grégory; Le Traon, Yves (2016): A systematic review on the engineering of software for ubiquitous systems. En: Journal of Systems and Software 118, pág. 251–276. DOI: 10.1016/j.jss.2016.05.024.