Proceso de Pruebas Unitarias Bajo Entornos de Desarrollo Ágiles: Un Estudio Sistemático

TRABAJO DE GRADO

José Andrés Moncada Quintero
Diego Fernando Navarro Reyes

Asesor
Hugo Fernando Arboleda Jiménez
Ingeniero de Sistemas y Computación
Magister en Sistemas y Computación
Doctor (PhD) en Ingeniería
Docteur (PhD) en Informatique

FACULTAD DE INGENIERÍA
MAESTRÍA EN GESTIÓN INFORMÁTICA Y TELECOMUNICACIONES
SANTIAGO DE CALI
2014

Proceso de Pruebas Unitarias Bajo Entornos de Desarrollo Ágiles: Un Estudio Sistemático

José Andrés Moncada Quintero
Diego Fernando Navarro Reyes

Trabajo de grado para optar al título de
Magister en Gestión de Proyectos y Tecnología

Asesor
Hugo Fernando Arboleda Jiménez
Ingeniero de Sistemas y Computación
Magister en Sistemas y Computación
Doctor (PhD) en Ingeniería
Docteur (PhD) en Informatique

FACULTAD DE INGENIERÍA
MAESTRÍA EN GESTIÓN INFORMÁTICA Y TELECOMUNICACIONES
SANTIAGO DE CALI
2014

Nota de aceptación

_____
_____
_____
_____
_____
_____

_____
Firma del Presidente del Jurado

_____
Firma del Jurado

_____
Firma del Jurado

Santiago de Cali, Junio 16 de 2014

CONTENIDO

pág.

LISTA DE CUADROS

pág.

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS

pág.

RESUMEN

La industria de TI ha tenido un crecimiento constante durante los últimos años, esto ha creado un mercado altamente competitivo que requiere de prácticas más eficaces y eficientes que permitan mejorar el desempeño de las organizaciones. Intenciones como el aseguramiento de la calidad buscan garantizar que los productos cumplan con los objetivos para los cuales fueron desarrollados, y han dado paso a múltiples estrategias que buscan incrementar la competitividad y favorecer el crecimiento de la industria. Actualmente existe información sobre diferentes estrategias de calidad disponible en el mercado, sin embargo obliga a la organización a una revisión bibliográfica extensa que requiere de análisis y un alto costo en relación al tiempo invertido, tiempo con el cual la industria generalmente no cuenta.

Este trabajo presenta un estudio confiable y accesible que se base en el contenido de múltiples fuentes de información acerca del proceso de inclusión de pruebas unitarias en ambientes ágiles. Éste amplía el conocimiento necesario para facilitar la toma de decisiones en organizaciones interesadas en sacar provecho de las tendencias ágiles y las pruebas unitarias. El estudio se desarrolla por medio de una revisión sistemática de literatura, basándose en buenas prácticas referentes al estudio de fuentes bibliográficas confiables, recogidas en múltiples dominios de actuación.

A través del desarrollo de un estudio sistemático se consolida la información disponible sobre la implementación de pruebas unitarias en entornos ágiles de desarrollo, encontrando una fuerte tendencia de adopción en lo ágil y una serie de técnicas especializadas para que el uso de pruebas unitarias en estos ambientes se convierta en una estrategia de aseguramiento de la calidad en entornos con requerimientos incompletos o altamente cambiantes.

1. INTRODUCCIÓN

1.1 CONTEXTO DE TRABAJO

En Colombia TI como industria es relativamente reciente si se compara con otros sectores; las características propias del entorno económico así lo han determinado. Acceso a la educación y la bien conocida brecha tecnológica resultan ser factores clave en la formación de una industria que se basa en conocimiento más que en tecnología. Sin embargo, a pesar de los elementos previamente mencionados, la industria ha tenido un crecimiento constante durante los últimos 5 años. De acuerdo con el reporte de Proexport Colombia 2012, el crecimiento de la industria de TI ha estado alrededor del 230%.

Esto no solo lo hace un sector prometedor sino que muestra cómo políticas de estado han venido impactando de manera significativa. Un ejemplo de esto es el Programa de Transformación Productiva y el FiTI (Programa de transformación productiva, 2013). Según datos de IDC (Internacional Data Corporation), Colombia es el tercer país en América Latina en ventas de TI, alcanzando en 2011 US$ 6.118 millones (Proexport Colombia, 2013).

Entre 2005 y 2011 los ingresos del sector de TI en Colombia se triplicaron. Ésto producto del potencial de exportación del software producido. Es importante resaltar que el 90% de las 850 empresas productoras de software, cuentan con equipos de 25 personas o menos, lo que indica que es un sector casi completamente compuesto por Pymes (Fedesoft, 2013).

Se espera que para él año 2013 la industria de TI continúe con la tasa de crecimiento que presentó entre 2011 y 2012 de alrededor del 27% alcanzando para ese momento ventas por US$291 millones, de los cuales alrededor de US$139 millones fueron aportados por la industria de producción de software lo que representa un 47% de las ventas de la industria (El Espectador, 2013).

Éste se ha vuelto un mercado altamente competitivo. Competir en el mundo requiere prácticas eficaces y eficientes es por esto que varias de las empresas del sector buscando mejorar su competitividad se encuentran buscando alternativas para la mejor producción de software, elementos como la calidad buscan asegurar el lugar de una organización en este mercado en crecimiento. Un contraste con la necesidad de mejoramiento se presenta al observar las empresas que se encuentran en proceso de certificación de calidad, acotamiento de estándares o implementación de normativas internacionales, las cuales, según Fedesoft, no supera el 50% del total, esto último aunque resulta ser preocupante es comprensible por el tamaño de las empresas del sector (Paola Restrepo, Presidente Ejecutiva de Fedesoft, 2013)

Producto de las necesidades de implementación de técnicas que incrementen la productividad de las empresas del sector, Fedesoft y el SENA han desarrollado "El programa de Formación Especializada para Apoyar la Implementación de SCRUM", el cual busca mejorar significativamente los indicadores de costos, tiempo y calidad en las empresas del sector, proporcionar un conjunto de herramientas a los profesionales beneficiarios para que mejoren de manera significativa los procesos para la prestación de servicios, control de costos, mejora del desempeño y la satisfacción del cliente (Paola Restrepo, Presidente Ejecutiva de Fedesoft, 2013).

Es importante tener claro qué significa calidad, sin embargo no es un término que sea fácil de definir. La Calidad es un concepto que es trasparente cuando está presente, pero que es muy fácil de detectar cuando no lo está (Hamill, 2004). En la Tabla 1 se incluyen algunas de las definiciones del termino Calidad.

Tabla 1. Definiciones de Calidad

Fuente	Definición
OED, 1990	El grado de excelencia
Fruhauf, 1994	Calidad es cuando el cliente regresa, no el producto
Crosby, 2000	Cero defectos
ISO9000:2008	Es el total de características de un producto o servicio puede soportar con el objetivo de satisfacer una necesidad especifica
ISO 25000	Grado en que el producto software satisface las necesidades expresadas o implícitas, cuando es usado bajo condiciones determinadas

De acuerdo con la norma ISO9001:2008 definen el termino de calidad como: "La calidad es algo que se puede determinar comparando un conjunto de características inherentes con un conjunto de requerimientos. Si estas características inherentes cumplen con todos los requisitos, un alto o excelente nivel de calidad es alcanzado. Si estas características no cumplen con todos los requerimientos, un bajo o pobre nivel de calidad es obtenido" (Charles A. Cianfrani, 2009). Sin embargo el término de calidad también se puede aplicar cuando hablamos de software, en este sentido se entiende como la coherencia entre lo especificado y si cumple con el objetivo para el cual fue desarrollado. Esto se puede asegurar planteándonos dos preguntas: ¿Es una buena solución?, ¿Está enfocado a la solución problema? (Gillies, 2011)

Por otro lado se cuentan con las pruebas unitarias, estas son herramientas que proporcionan a los desarrolladores o evaluadores una forma rápida de buscar errores lógicos en los métodos de clases de proyectos, representando una forma de

asegurar la calidad de un producto y mejorarlo mediante la identificación de defectos y problemas oportunamente.

Durante mucho tiempo, los procesos de desarrollo de software han estado mediados por estructuras de verificación fuertes que tenían como objetivo acotar los requerimientos y variaciones de manera específica, es decir, mantener en control las funcionalidades y necesidades del cliente en un contexto futuro. Procesos tradicionales de desarrollo de software basadas en elementos como el modelo de cascada son usuales enseñanzas de las facultades de ingeniería, sin embargo las estructuras propias del mercado se encuentran rápido movimiento. Los requerimientos no solo cambian más rápido, sino que son difíciles de predecir hacia el futuro pues algunos que hasta el momento eran inexistes surgen en la medida en que se avanza el proceso, estos cambios son se difícil manejo en un modelo rígido de recolección de requerimientos y procesos de implementación no iterativos.

Los modelos tradicionales han estado presentes por muchos años y exitosos proyectos se han llevado a cabo con ellos sin embargo ante los cambios en la industria del software presentan una dificultad importante por la inclusión de nuevas funcionales en la mayoría de los casos afecta un sistema que ha sido basado en la predicción y no en la adaptabilidad. Sobre este último punto se fundamentan las conocidas como metodologías de desarrollo ágil.

Las Metodologías Ágiles o “ligeras” constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de e-projects que las metodologías convencionales (ISO-9000, CMM, etc) debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración (Gallo, 2009).

Ante el mercado cambiante y las dificultades de los métodos convencionales, surge el manifiesto Ágil basado en una serie de principios que buscan que la producción sea ligera y adaptativa centrada en entregar valor al cliente y en iteraciones de ajuste rápido, su principal objetivo es permitir que el software este vivo todo el tiempo y las necesidades se vayan incluyendo en el proceso a medida que se necesiten (agilemanifesto.org, 2001)

Existen diferentes frameworks de pruebas unitarias, uno de ellos es el Desarrollo Guiado por Pruebas (TDD), el cual representa una de las más significativas prácticas utilizadas en metodologías de desarrollo ágil tales como SCRUM y XP entre otras.

La clave de TDD se resume en “Escribir las pruebas primero y luego refactorizar”, se basa en el desarrollo de pruebas unitarias, una vez se codifiquen las pruebas y estas fallen, se refactoriza el código hasta que la prueba pase satisfactoriamente. El objetivo es alcanzar un código limpio al final del ciclo de desarrollo (Hamill, 2004).

Es un hecho que la distancia entre la academia y la industria en países como Colombia, es aún muy amplia, esto se debe a cuestiones relacionadas con la cultura frente a la investigación, la baja existencia de departamentos de investigación y desarrollo o determinadas prioridades económicas dadas por el ya mencionado tamaño de las organizaciones productoras de software. Esta situación genera en muchos casos la incapacidad de adaptarse rápidamente al mercado cambiante, alinearse con tendencias internacionales e incrementar la competitividad guiada por la innovación en los procesos, es claro que la solución evidente resultaría de la gestión de innovación y la generación de una nueva cultura de investigación dentro de las organizaciones, sin embargo, la operación misma agota el tiempo de las organizaciones y las deja inmersas en su día a día. Es necesario resaltar que la importancia de la inclusión de procesos sistemáticos de investigación de literatura, permiten que de manera eficiente y confiable, se generen herramientas para la toma de decisiones fundamentales en relación a las inversiones y la gestión de TI.

Por último, es importante aclarar lo que es un Estudio Sistemático, éste representa una manera de identificar, evaluar e interpretar información disponible sobre investigaciones de una pregunta específica, tema o fenómeno de interés (Kitchenham B. , Procedures for Performing Systematic Reviews, 2004). Generalmente un estudio sistemático es una forma de realizar una investigación secundaria sobre un tema en particular.

De acuerdo con Kitchenham existen algunas razones para realizar un estudio sistemático, entre las cuales se encuentran:

- Permite resumir la evidencia existente sobre el tema de interés o alguna tecnología, por ejemplo resumir la evidencia empírica entre los beneficios y desventajas de un método de desarrollo ágil.
- Para identificar aquellos vacíos en la investigación actual, que permitan trabajar en temas de una investigación mayor.
- Proporciona un marco con el objetivo de posicionar nuevas actividades dentro de la investigación.

El presente estudio tomará como base el método propuesto por Kitchenham para la realización de estudios sistemáticos (Kitchenham B. , 2004). La decisión de tomar como base esta metodología surge a partir de la revisión previa de diferentes artículos sobre estudios sistemáticos, en donde se analizó el método de investigación de cada uno de estos y se detectó que la mayor parte de estos toma como base la propuesta definida por Kitchenham, entre alguno de los artículos encontrados se encontraron: “Systematic literature reviews in software engineering – A systematic literature review” (Kitchenham, y otros, 2009) y “A systematic review of systematic review process research in software engineering” (Kitchenham & Brereton, 2013).

1.2 PLANTEAMIENTO DEL PROBLEMA

La industria no cuenta con un estudio confiable y accesible que evidencie las posibilidades descritas en múltiples fuentes alrededor de las pruebas unitarias en ambientes agiles de desarrollo de software, nuestra hipótesis es que la carencia de dicho estudio supone una barrera para que las organizaciones tomen decisiones acerca del proceso de implementación de las mismas, limitando las probabilidades de éxito de las iniciativas y obligándolas a una extensa revisión bibliográfica que enfrenta a la industria con tiempos y dinámicas que no debería manejar para seleccionar una técnica que tenga en cuenta argumentos valor para la organización.

1.3 OBJETIVOS

1.3.1 Objetivo General.

Desarrollar un estudio sistemático sobre la implementación de pruebas unitarias en ambientes ágiles con miras a la generación de una base de conocimiento para la toma de decisiones dentro de la industria de TI en Colombia.

1.3.2 Objetivos Específicos:

1. Adaptar un protocolo para el estudio sistemático acorde a la metodología propuesta por Kitchenham en 2004.

2. Determinar las preguntas de investigación que guíen el estudio sistemático, permitiendo que éste genere valor para las empresas de la industria de software.

3. Desarrollar el estudio sistemático a partir de las fases propuestas por el protocolo seleccionado.

4. Realizar el análisis de los resultados obtenidos tras desarrollar el estudio sistemático.

1.4 RESUMEN DEL MODELO PROPUESTO

El presente estudio sistemático se desarrolló con un protocolo adaptado a partir de la guía propuesta por Kitchenham (2004). Teniendo en cuenta ésta generamos las bases para definir la estrategia que se implementó en el presente estudio sistemático y el cual incluye las fases que descritas a continuación. En La Figura 1

se muestra el esquema general de cada una de las fases del método propuesto que guía el estudio.

Figura 1. Diagrama del Modelo

1. *Creación de las Preguntas de Investigación*
   Como primera fase se realiza la definición de las preguntas de investigación que guiaran el estudio. Estas preguntas se generan a partir de la necesidad que se logra detectar acerca de un tema específico de manera que se pueda obtener un consolidado de la información de una manera más organizada.

2. *Definición del protocolo de Investigación*
   Dentro de este proceso debe realizarse la definición del protocolo que guiará el estudio. En este punto debe definirse claramente cada una de las etapas que se desarrollaran para la selección, filtrados y extracción de los datos que darán respuesta a las preguntas de investigación planteadas.

   1. *Selección de Criterios de Elegibilidad*
      Se definen aquellos criterios que se tomaran como base para la inclusión y exclusión dentro de las búsquedas de los artículos.

   2. *Recolección y Filtrado de información*
      En este proceso se definen las bases de datos que se seleccionaran para la extracción de los artículos, así como el conjunto de palabras claves de búsqueda y la estrategia de recolección definida para cada una de estas fuentes de datos.

   3. *Selección de Herramientas*

Se definen las herramientas o procesos sistemáticos que ayuden a la consolidación de la información recolectada.

4. *Filtrado por Criterios de Exclusión*
   En este proceso se eliminan aquellos artículos que no cumplan con los filtros de inclusión y exclusión que se definieron.

5. *Filtrado por Título y Resumen*
   Se realiza una búsqueda dentro del título y el resumen de acuerdo a las palabras que guarden mayor relevancia dentro del estudio.

6. *Filtrado de Evaluación Semántica*
   Revisión manual del artículo de forma rápida a través de la definición de preguntas que permitan asegurar la relación de los temas de investigación.

7. *Filtrado de Evaluación de la Relación con las Preguntas de Investigación*
   Se realiza de manera manual una revisión de los resúmenes y resultados de los artículos que hasta el momento hayan logrado pasar los filtros previos para validar el grado de relación con cada una de las preguntas que dirigen el estudio.

8. *Evaluación de la Calidad*
   Categorización de los estudios recolectados a partir de una serie de preguntas planteadas con base en una serie de criterios definidos por el autor con el objetivo de agrupar de manera concisa la información obtenida.

9. *Extracción de Datos*
   Consolidación de la información recolectada de los artículos finales a partir de agrupación y categorías definidas por el autor.

3. *Resultados Finales*
   Con base en los artículos arrojados durante el proceso previo y la información consolidada se da respuesta a las preguntas planteadas dentro del estudio con base en las propuestas de cada uno de los artículos base.

1.5 RESUMEN DE RESULTADOS OBTENIDOS

Una vez que definimos el protocolo a seguir para la ejecución del estudio sistemático ejecutamos cada una de las fases propuestas en el modelo y registramos los resultados obtenidos en cada una de ellas. Al terminar la última fase, contamos con una base consolidada de artículos, los cuales guardaban la mayor relación con las

preguntas de investigación previamente definidas y esperábamos dieran respuestas a los interrogantes planteados. Para el proceso final tuvimos cuenta el estudio de los artículos arrojados y consolidamos la información recolectada para dar un aporte al propósito de la investigación. En la Figura 2 se pueden visualizar el número de artículos recolectados en cada uno de los filtros que incluye el protocolo.

En la parte superior izquierda se observa el proceso de *"Recolección y filtrado de información"* y después los subprocesos asociados a cada una de las bases de datos con el número de artículos resultado de cada proceso. Se puede observar el consolidado de todas las fuentes que fue llevado a la herramienta de gestión de referencias *"Mendeley",* la cual será descrita en la sección de herramientas de soporte*.* Se repite el proceso de filtrado de información mediante criterios de exclusión y se obtienen el número total de artículos validos hasta el momento. A continuación se observa el resultado de los sub procesos de filtrado por título y resumen. Los resultados de cada uno de los filtros se observan como números enteros al lado de cada uno de los procesos.

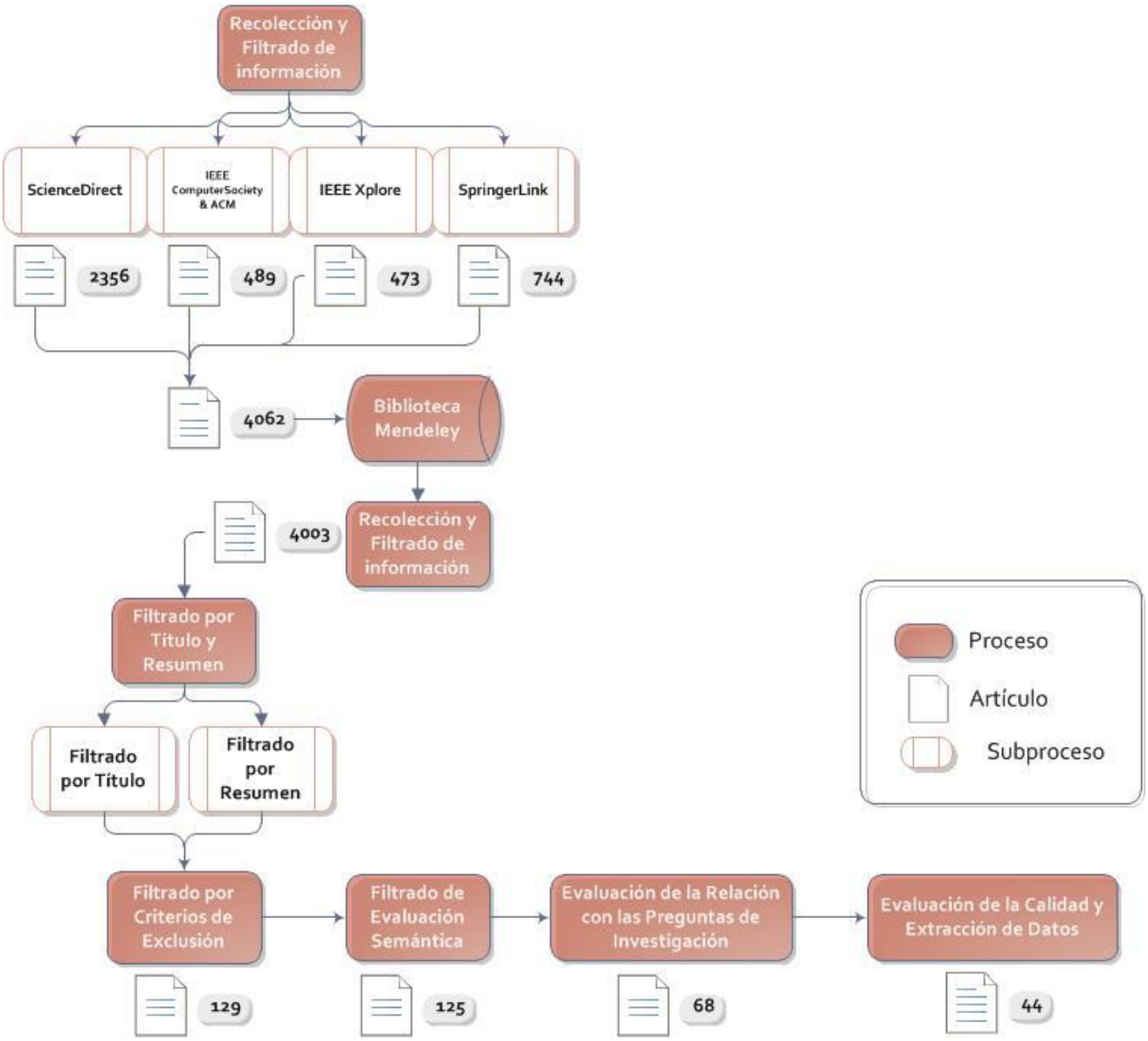

Figura 2. Diagrama del Modelo y Resultados

1.6 ORGANIZACIÓN DEL DOCUMENTO

Este documento está dividido en tres partes claramente definidas:

Parte 1: Marco Teórico. Compuesto por la definición de Estudio Sistemático, así como los conceptos que incluyen las pruebas unitarias y las metodologías de desarrollo ágil.

Parte 2: Protocolo Adaptado. Contiene la definición y descripción del protocolo del estudio, incluyendo cada una de las fases a seguir en la ejecución del estudio sistemático.

Parte 3: Resultados Obtenidos. Describe los resultados que se detectaron durante el desarrollo del estudio sistemático para cada una de las fases que se habían definido previamente en la propuesta del protocolo.

2. MARCO TEÓRICO

2.1 Estudio Sistemático

Kitchenham propone una guía para realizar un estudio sistemático que incluye un conjunto de actividades específicas para su ejecución. Se dividen en tres fases: Planeación, Ejecución y Reporte (Kitchenham, 2004).

Planeación

En esta etapa se incluyen las siguientes actividades:

1. *Identificación de necesidades del estudio:* Antes de iniciar con el estudio sistemático el investigador debe estar completamente seguro de que el estudio es realmente necesario y cual es objetivo final de la información que se logre obtener. Dentro de la guía se proponen una lista de preguntas que orientaran al investigador asegurarse de la necesidad del estudio, dentro de estas se cuenta con los puntos de vista de CRC (2001) y Greenlaugh (1997).

2. *Desarrollo de la estrategia del estudio sistemático:* En este punto se especifica cuál será el protocolo para la búsqueda de la información, se incluyen componentes individuales que se deban ejecutar:

    a. *Pregunta de Investigación:* Se considera una de las actividades más importantes que dirigen el estudio sistemático. Las preguntas que se planteen en esta etapa son las que dirigen el enfoque de la investigación y limitan el alcance del mismo.
    b. *Revisión del Protocolo:* El protocolo es uno de los elementos más críticos dentro de cualquier estudio sistemático, por esta razón debe ser revisado por expertos que permitan validar la efectividad del mismo.

Ejecución

En la etapa se incluyen las siguientes actividades:

1. *Identificación de la investigación:* El objetivo de esta etapa es encontrar tantos estudios como se considere necesario y que se encuentren en relación con las preguntas de investigación. Se debe tener claro cuál será el alcance inicial de los artículos seleccionados sin tratar de sesgar el objetivo de investigación.

2. *Selección de estudios*: Una vez se tienen seleccionados los artículos primarios que se consideran potencialmente relevantes, se procede a evaluar su relevancia con las preguntas de investigación. Se define cuáles son los

filtros para tomar las decisiones en relación a las inclusiones y exclusiones de los artículos que fueron seleccionados inicialmente.

3. *Evaluar calidad de los estudios:* Adicionalmente a los factores de exclusión e inclusión definidos anteriormente, se evalúa la calidad de los artículos que fueron seleccionados. En esta fase se pueden definir jerarquías que permitan clasificar la calidad de acuerdo con diferentes factores que se consideren necesarios.

4. *Extracción de la información:* En ésta fase se diseñaron las reglas de extracción de datos para registrar con precisión la información que puede ser obtenida de los estudios primarios. Los formularios de extracción de datos fueron definidos y empleados de acuerdo con el protocolo que fue definido previamente.

5. *Síntesis de la información:* Se realiza la síntesis de los datos que consiste en recopilar y resumir los resultados de los estudios primarios seleccionados.

Reporte

En esta fase se comunican los resultados obtenidos de la revisión sistemática. Según lo descrito generalmente este reporte tiene una restricción de tamaño, con el objetivo de garantizar que los lectores sean capaces de evaluar adecuadamente el rigor y la validez del estudio realizado.

Variaciones metodológicas

Encontramos que múltiples autores han realizado estudios a partir de la metodología de revisión sistemática de literatura, estos han realizado modificaciones en relación a los criterios de calidad, las fuentes de información utilizadas y la forma de utilización de los criterios usados para la inclusión y exclusión. En 2009 Kitchenham y su equipo realizaron: *"Systematic literature reviews in software engineering – A systematic literature review"* (Kitchenham, y otros, 2009) reportando el hallazgo de 20 estudios relevantes. En 2013 dado el crecimiento en la utilización de la metodología, realizaron: *"A systematic review of systematic review process research in software engineering"* (Kitchenham & Brereton, 2013) con el objetivo de verificar los procesos que se habían realizado y la valides actual de la metodología en el contexto de investigación actual, en este estudio identificaron 68 artículos para 63 estudios publicados entre 2005 y 2012 con las diferentes criticas frente a la metodología: a) La metodología genera un alto consumo de tiempo frente a otro tipo de técnicas de investigación, b) Las bibliotecas digitales no cuentan con las herramientas apropiadas para las búsquedas de amplio espectro c) La evaluación de calidad en diferentes tipos de estudios empíricos resulta de difícil aplicación. Este mismo estudio concluye con algunas recomendaciones, entre las que se destacan:

a) La recomendación para la eliminación de la utilización de preguntas estructuradas como cadenas de búsqueda
b) La inclusión de consejos para el uso de una norma cuasi-oro producto de una búsqueda manual limitada, esto con el objetivo de ayudar a la construcción de las cadenas de búsqueda y evaluación del proceso del mismo proceso.
c) La necesidad de herramientas de análisis textual, al considerarse útiles para las decisiones de inclusión/exclusión y la construcción de la cadena de búsqueda, sino se requiere una evaluación más rigurosa.
d) El posible beneficio de herramientas especializadas que permitan gestionar el proceso de revisión sistemática.
e) La no estandarización del proceso de evaluación de la calidad de los estudios, visto por los autores como problema importante.

2.2 Pruebas Unitarias

El concepto de prueba unitaria o de unidad no es un concepto nuevo, este se encuentra presente desde los años 70 con el lenguaje de programación *SmallTalk* donde fue introducido por Kent Beck, de esta manera es descrito en el libro *"The Art of Unit Testing"* (Osherove, 2009). También se describe a las pruebas unitarias como la mejor forma en que un desarrollador puede incrementar la calidad de su código mientras obtiene una mejor compresión de los requerimientos funcionales de una clase o un método.

En una definición clásica, una prueba unitaria es una pieza de un código, por lo general un método, que invoca otro fragmento de código y comprueba el resultado en relación a algunas hipótesis previamente creadas. Si el resultado no puede ser verificado, es decir, el supuesto realizado resulta ser equivocado, la prueba de la unidad ha fallado. Lo que quiere decir que la unidad o sistema bajo prueba, representado por un método o función no cumple con los requerimientos planeados. (Osherove, 2009).

De acuerdo con Oshere, las propiedades que determinan la validez de una prueba unitaria son:

- Debe poderse automatizar y repetir en múltiples ocasiones.
- Debe ser fácil de implementar y mantener.
- Una vez escrita debe poder reutilizarse en el futuro.
- Cualquier desarrollador del equipo debe estar en capacidad de ejecutarla.
- Debe poderse ejecutar ante una única acción.
- Debe ser de rápida utilización siempre y cuando sea posible.

Si bien las pruebas son utilizadas por todos los procesos de desarrollo de software, existen diferentes técnicas y enfoques. En las metodologías convencionales la ejecución de pruebas es considerada como una fase que se inicia al final del desarrollo del software.

Cuando se habla de pruebas, estas pueden ser categorizadas en dos tipos: pruebas de caja negra y pruebas de caja blanca, también conocidas como pruebas funcionales y estructurales. Una prueba de caja negra, se caracteriza por ser una prueba que simplemente ejecuta una funcionalidad del programa y chequea que se ejecute correctamente sin entrar en detalle sobre cómo fue escrito el programa. Por otra parte las pruebas de caja blanca revisan funcionalidades entrando en detalle con las estructuras internas del código desarrollado (Hamill, 2004).

La relación entre las pruebas unitarias contenidas en un marco de trabajo de pruebas unitarias XUnit y el código generado responde por lo generalmente a una relación uno a uno, es decir, a una clase de prueba por cada clase a probar (Hamill, 2004).

Los Desarrolladores realizan las pruebas de acuerdo con su percepción de cómo fue diseñado y desarrollado el código, sin embargo generalmente estas son pruebas de bajo nivel orientadas a revisar funciones e interfaces. Las pruebas unitarias también pueden contemplar características como desempeño o comportamientos esperados por los usuarios del sistema en este caso podrían ser usadas como pruebas de alto nivel o de aceptación.

2.3 Metodologías agiles

En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término “ágil” aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas. Tras esta reunión se creó The Agile Alliance una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida es fue el Manifiesto Ágil, un documento que resume la filosofía “ágil”. (Canós, Letelier, & Penadés, 2003)

El Manifiesto Ágil (agilemanifesto.org, 2001)

Según el Manifiesto se valora:

- Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.
- Desarrollar software que funciona más que conseguir una buena documentación. La regla a seguir es "no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante". Estos documentos deben ser cortos y centrarse en lo fundamental.
- La colaboración con el cliente más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.
- Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta. Los valores anteriores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo. Los principios son:
  I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
  II. II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
  III. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
  IV. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
  V. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
  VI. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
  VII. El software que funciona es la medida principal de progreso.
  VIII. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.

IX. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
X. La simplicidad es esencial.
XI. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

Comparación

A continuación se muestra una comparación entre las metodologías tradicionales y las metodologías agiles (Canós, Letelier, & Penadés, 2003)**.**

Metodologías Ágiles

- Basadas en heurísticas provenientes de prácticas de producción de código
- Especialmente preparados para cambios durante el proyecto
- Impuestas internamente (por el equipo)
- Proceso menos controlado y con pocos principios
- No existe contrato tradicional o de alta flexibilidad.
- El cliente es parte del equipo de desarrollo
- Grupos pequeños (menor a 10 personas) que trabajan en el mismo sitio.
- Pocos artefactos
- Pocos roles
- Menos énfasis en la arquitectura del software

Metodologías Tradicionales

- Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
- Cierta resistencia a los cambios
- Impuestas externamente
- Proceso mucho más controlado, con numerosas políticas/normas
- Existe un contrato prefijado
- El cliente interactúa con el equipo de desarrollo mediante reuniones.
- Grupos grandes y posiblemente distribuidos.
- Más artefactos
- Más roles
- La arquitectura del software es esencial y se expresa mediante modelos

Esta comparación expone las principales diferencias entre las metodologías ágiles y el modelo tradicional de desarrollo de software.

2.3.1 SCRUM

SCRUM es una metodología de desarrollo ágil inicialmente pensada para la creación de aplicaciones pero implementada en algunas áreas distintas a las relacionadas con la programación. Esta metodología se centra en el trabajo en equipo y la forma en como éste permite la construcción de un producto desde piezas pequeñas. En SCRUM se entiende el producto como algo que se crea de forma incremental apalancando cada nuevo elemento en los que han sido previamente creados. Se caracteriza por la posibilidad de dar respuesta rápida a la retroalimentación recibida y el cambio constante para construir únicamente lo necesario. (Scrum.org, s.f.)

SCRUM además es un marco para la colaboración eficaz de un equipo en proyectos complejos, cuenta con un conjunto de reglas que definen una estructura capaz de orientar a los equipos hacia la innovación en la solución, observar partes pequeñas y no un todo que podría parecer insuperable.

El concepto SCRUM tiene origen en 1986 cuando se inicial la construcción acuerda de los procesos nuevos de desarrollo de software para los productos exitosos en Japón y Estados Unidos. Este estudio se basó en industrias como Canon, Xerox y Honda (Nonaka, 1986).

El marco de trabajo además de reglas cuenta con Roles claramente definidos:

Dueño del Producto

Su función se encuentra orientada a determinar lo que hay que construir en los próximos 30 días o menos.

Equipos de Desarrollo

Se encargan de construir lo que se necesita en 30 días (o menos). Después de construido lo demuestran para que el propietario del producto determina qué construir después.

SCRUM Masters

Son los encargados de garantizar que el proceso ocurre en las mejores condiciones, Sus funcionen buscan un mejoramiento constante en las personas, el proceso y finalmente en el producto.

Ventajas

- Incremento de la velocidad respecto a procesos tradicionales de desarrollo de software.
- Independencia en los desarrolladores
- Menor gestión y procesos burocráticos
- Alta flexibilidad frente al entorno cambiante.

Desventajas

- El camino más corto resulta tentador para el equipo de desarrollo pero no siempre es el más adecuado.
- Es difícil determinar las fechas de entrega con mucho tiempo de antelación.
- Alto nivel de presión sobre el equipo, siempre en carrera.
- Requiere equipos organizados capaces de tomar decisiones sin depender de tareas concretas y especializaciones por parte de los desarrolladores.

2.4 Extremme Programming (XP)

XP es una metodología de desarrollo ágil, propuesta por Kent Beck y consignada por primera vez en el libro; Extreme Programming Explained: Embrace Change (Beck, 1999). Hasta la aparición de Scrum anteriormente mencionado, fue el más difundido método ágil para la creación de software, al igual que otras metodologías se basa en el manifiesto ágil y en la idea de brindar flexibilidad y herramientas adaptables a los equipos de creación de software. Los cambios entonces se contemplan como una situación normal y comprensible a lo largo de un proyecto de implementación de software.

Se basa en 5 valores que sirven como pilar metodológico y procedimental, el primero la simplicidad que básicamente indica que los diseños complejos deben resultar de múltiples ajustes a lo largo del ciclo de vida y no de una intención inicial. En segundo lugar encontramos la comunicación que surge de forma técnica mediante la documentación ligera solo para términos explicativos, es decir, solo se busca comunicar o documentar si algo no ha quedado claro en el código.

Para esta metodología resulta particularmente especial que el equipo pueda ser retroalimentado por el cliente, es por esto que el progreso se puede conocer en tiempo real y permite la toma de decisiones basada en situaciones cercanas en gran medida a la realidad. El cuarto pilar se fundamenta en el respeto pues se define una regla básica y es no dejar sin funcionar una prueba que ya estaba funcionando. El último pilar es la valentía, pues en muchas ocasiones se programa sin conocer todos los elementos necesarios, avanzar a ciegas entendiendo que existen elementos que no pueden ser previstos.

XP cuenta con 8 características fundamentales de acuerdo a la comunidad que apoya el proyecto en internet, se enumeran a continuación:

1. Desarrollo iterativo e incremental.
2. Pruebas unitarias constantes.
3. Equipos de programación de 2 personas.
4. Retroalimentación constante.
5. Cero errores y entregas frecuentes.
6. Refactorización del código.
7. Código compartido.
8. Simpleza en las estructuras de código.

Ventajas

- Da lugar a una programación sumamente organizada.
- Ocasiona eficiencias en el proceso de planificación y pruebas.
- Cuenta con una tasa de errores muy pequeña.
- Propicia la satisfacción del programador.
- Fomenta la comunicación entre los clientes y los desarrolladores.
- Facilita los cambios.
- Permite ahorrar mucho tiempo y dinero.
- Puede ser aplicada a cualquier lenguaje de programación.
- El cliente tiene el control sobre las prioridades.
- Se hacen pruebas continuas durante el proyecto.
- La XP es mejor utilizada en la implementación de nuevas tecnologías.

Desventajas

- Es recomendable emplearla solo en proyectos a corto plazo.
- En caso de fallar, las comisiones son muy altas.
- Requiere de un rígido ajuste a los principios de XP.
- Puede no siempre ser más fácil que el desarrollo tradicional.

2.5 Desarrollo guiado por pruebas de software (TDD)

El desarrollo guiado por pruebas se considera una práctica metodológica para el desarrollo de software que involucra 2 prácticas adicionales, la primera es iniciar con la creación de las pruebas (Test First Development) y la segunda es el alto uso de refactorización de código para realizar ajuste constante. (Crispin & Gregory, 2009)

Dentro de TDD las pruebas unitarias son ampliamente utilizadas, el objetivo inicial es verificar que las pruebas creadas en la fase inicial no sean exitosas para iniciar el proceso de desarrollo o factorización. En esta práctica es fundamental que los requisitos logren ser traducidos en pruebas que logren al final un código limpio producto de la refactorización y resultados exitosos en las pruebas. Esto último relacionado con código limpio es la intención principal de la práctica TDD. (Crispin & Gregory, 2009)

Dentro de la práctica se han definido requisitos, los cuales se enumeran a continuación:

- Se favorece el diseño al ser adaptado a las necesidades específicas.
- Se requieren interfaces claras y alta cohesión.
- Se requiere flexibilidad en el proyecto en desarrollo.
- Las pruebas deben poder ser automatizadas.
- Las pruebas deben ser unitarias y ampliamente verificadas.

Ventajas

- Mayor calidad respecto a los procesos tradicionales de desarrollo de software para entornos con requerimientos cambiantes.
- Diseño enfocado en las necesidades.
- Mayor simplicidad en el diseño.
- El diseño se va adaptando al entendimiento del problema.
- Mayor productividad.
- Menos tiempo invertido en la corrección de errores críticos.

Desventajas

- Difícil creación de Interfaz de usuario.
- Problemas con la definición de la Base de datos.
- Posibles errores no identificados o contemplados.
- Perder la visión general en detalles de iteración.
- Curva de aprendizaje elevada en la preparación del equipo.

2.6 Desarrollo basado en pruebas de aceptación (ATDD)

Esta práctica se basa en los principios de TDD, pero se basa en el uso de pruebas de aceptación automatizadas escritas previamente para las funcionalidades específicas. Los orígenes se encuentran en 2003 con la mención de Kent Beck en el libro "Test Driven Development: con el ejemplo". Posteriormente en 2004 ATDD

se convierte en una práctica aceptada a pesar de las objeciones de múltiples autores. (Beck, Test Driven Development: con el ejemplo, 2003)

Existen figuras como dueño del producto, cliente y experto en dominio, de acuerdo que se encuentre en desarrollo, se podrían especificar nuevas funcionalidades únicamente basándose en la escritura de pruebas sin contactar directamente al equipo desarrollador.

Aunque menos común el ATDD también es llamado STDD al hacer referencia a las historias de usuario.

Ventajas:

- Como el desarrollo dirigido por pruebas o TDD resulta ser sencillo probar los resultados de la unidad, las pruebas de aceptación favorecen la creación de interfaces específicas para las pruebas funcionales.

Desventajas:

- Requiere del uso de herramientas altamente específicas

3. PROTOCOLO ADAPTADO

El protocolo de investigación que seguimos para la realización del presente estudio sistemático fue desarrollado tomando como base la guía propuesta por Kitchenham (2004). En esta guía se propone una manera sistemática de identificar, evaluar e interpretar la información a investigar de una manera consolidada y ordenada, especificando las preguntas de investigación, las estrategias de búsqueda, los criterios de inclusión y exclusión, la evaluación de calidad, la extracción de datos y los métodos para sintetizar la información recolectada que dará respuesta a las preguntas de investigación.

En la Figura 3 se puede visualizar cada una de las fases propuestas dentro del protocolo. Cada una de estas etapas se describe a continuación.

Figure 3. Diagrama del Modelo

3.1 Preguntas de Investigación

Estas preguntas surgen a partir de la necesidad que detectamos en la industria acerca de tener un conocimiento claro y conciso acerca del tema de pruebas unitarias dentro de entornos ágiles de desarrollo. Actualmente muchas de las organizaciones que trabajan en desarrollo de software trabajan bajo metodologías ágiles y no toman en cuenta dentro de sus procesos la utilización de pruebas unitarias como parte fundamental de su ciclo de desarrollo.

A partir de las preguntas planteadas generamos una fuente de información que sirve para aquellas organizaciones con procesos de desarrollo de software ágil que están pensando en incluir las pruebas unitarias dentro de su ciclo de trabajo o aquellas de

corte más tradicional que inician el proceso de migración y desean obtener información relevante. Adicionalmente se consolida la información que corrobora por que las pruebas unitarias agregan valor dentro de los procesos ágiles de desarrollo y qué estilo de trabajo en las pruebas unitarias se ajusta más a estos entornos de trabajo.

El presente estudio fue dirigido a partir de las siguientes preguntas:

RQ1: ¿Cómo se logran implementar pruebas unitarias en un entorno de desarrollo ágil?

RQ2: ¿Cuáles fueron los resultados obtenidos tras la implementación de pruebas unitarias en ambientes ágiles de desarrollo de software?

RQ3: ¿Qué tipo de pruebas unitarias se deben implementar en un ambiente de desarrollo ágil?

3.2 Protocolo de Investigación

De acuerdo con el método propuesto, el siguiente paso a seguir es la definición del protocolo de investigación, en el cual se incluye todo lo referente a los filtros de inclusión y exclusión de los artículos seleccionados, y la aplicación de filtros necesarios para realizar la evaluación de calidad y obtener los artículos finales, a los cuales se les extrae la información relevante que da respuesta a cada una de las preguntas previamente planteadas.

3.2.1 Criterios de Elegibilidad

Para el levantamiento de los artículos seleccionados partimos de los criterios específicos que definen las características que deberían tener los artículos estudiados para ser tomados en cuenta en la extracción de información. Se definen dos grupos de criterios: criterios de inclusión y criterios de exclusión. A continuación se presenta cada uno, así como los criterios empleados en el presente estudio. Se sugieren utilizar los criterios definidos si se pretende realizar un estudio sistemático realizando, teniendo en cuenta las adaptaciones pertinentes de acuerdo con los objetivos particular del estudio.

3.2.1.1 Criterios de Inclusión

Los criterios de inclusión son características específicas que deben contener los artículos, fuentes de datos o información a estudiar. Para el levantamiento de los artículos que seleccionamos en este estudio, partimos de los siguientes criterios:

- ✓ *Lenguaje*
  - Los lenguajes seleccionados para las búsquedas de los artículos fueron: español e inglés.

  Definimos estos dos lenguajes debido a que son los que más fuerza poseen en cuanto a cantidad de información disponible y por las posibilidades con que contaba el equipo investigador. De igual manera se esperaba que la mayor parte de los artículos recolectados se encontraran en el idioma inglés de acuerdo con las fuentes de información o bases de datos seleccionadas.

- ✓ *Tipo de Publicación:*
  - Manuscritos originales de investigación publicados en revistas con evaluación por pares.
  - Memorias de Congresos Relevantes.

- ✓ *Diseño de Estudio:*
  - Artículos publicados en revistas indexadas.
  - Artículos presentados en eventos relevantes.

  Al igual que en el caso del criterio de tipos de publicación, esperábamos que aquellos artículos que fueran seleccionados cumplieran con un diseño estándar y un adecuado estudio en su investigación, permitiéndonos corroborar la calidad del mismo y los resultados obtenidos en su ejecución.

- ✓ *Título:*
  - Debe incluir palabras claves asociadas a las preguntas de investigación.

  Definimos una serie de palabras de acuerdo a la relevancia y relación con las preguntas de investigación objetivo que dirigieron éste estudio sistemático, con la idea filtrar aquellos artículos que pudieran ayudar a dar solución a las preguntas planteadas del estudio.

- ✓ *Tiempo:*
  - Se incluirán artículos con fecha de publicación posterior al año 2007.

  Para la selección de los artículos definimos como fecha los años posteriores al 2007. Esta decisión se toma con base al estudio realizado en el marco

teórico. Se detectó que el tema de metodologías agiles definido por primera vez en el manifiesto ágil (Kent et al, 2001) se inició en el año 2001, luego de estudios posteriores el tema tomo fuerza a mediados de 2004 en donde se iniciaron diversos estudios en temas relacionados a los procesos ágiles. Tomamos entonces la decisión de iniciar la investigación en el año 2007 para brindar un proceso de maduración de las investigaciones realizadas en años posteriores a 2004 y que estas brindaran información valida y relevante en nuestro estudio.

3.2.1.2 Criterios de Exclusión

Los criterios de exclusión son aquellas características que no son deseadas en los artículos o fuentes de información, se constituyen en límites que determinan los artículos que pertenecen y los que no pertenecen al conjunto de datos a analizar. Para el levantamiento de los artículos que se seleccionan en este estudio, partimos de los siguientes criterios:

- ✓ *Tipo de Publicación:*
  - Cualquier tipo de documento diferente a manuscritos de investigación publicados en revistas revisadas por pares o memorias de congresos.

No se toma en cuenta documentos diferentes a estudios realizados como manuscritos de investigación o memorias de congreso, esto se debe principalmente a que se espera no incluir información de poca calidad que pueda afectar los resultados del estudio.

3.2.2 Recolección y Filtrado de información

3.2.2.1 Fuentes de Información

Para la recolección de la información de los documentos y artículos relacionados a las preguntas de investigación trabajamos con las siguientes fuentes de información, las cuales son seleccionadas por ser fuentes recurrentes en los estudios sistemáticos realizados por Kitchenham y Brereton (2009) (2013), y son referentes de procesos de investigación básica y aplicada:

- ✓ ScienceDirect
- ✓ ACM
- ✓ IEEE Computer Society
- ✓ IEEE Xplore
- ✓ SpringerLink

3.2.2.2 Recolección Inicial

La Tabla 2 define el conjunto de palabras clave, de acuerdo con la relación que guardan con las preguntas de investigación. Éstas fueron generadas a partir de la revisión manual inicial durante la etapa de planeación de la investigación, este método fue basado en la exploración inicial propuesta por (Kitchenham & Brereton, 2013).

Tabla 2. Conjunto de Palabras Clave

Palabras base
Software
Development
Agile
Test
Unit
Quality

La Tabla 3 presenta las expresiones definidas para obtener el mayor número de artículos relacionados con la investigación. Las combinaciones usadas varían para las diferentes bases de datos usadas, de acuerdo con las posibilidades técnicas de búsqueda que cada base de datos provee. En las siguientes secciones se detalla la combinación de palabras clave usadas en las diferentes bases de datos.

Las búsquedas genéricas planeadas fueron:

Tabla 3. Conjunto de expresiones generales

Conjunto de expresiones generales
Software test*
Software agile test*
Software unit* test*
Software quality test*
Software develop*
Software develop* agile test*
Software develop* unit* test*
Software develop* quality test*
Software agile unit* test*
Software develop* agile unit* test*
Software develop* agile quality test*
Software develop* agile quality unit* test*

En la Tabla 3, los asteriscos representan la versión singular de la palabra en cuestión, pretendemos representar la raíz de la misma y permitir que palabras como *"developer"* o *"developing"* sean incluidas bajo la raíz *"develop"*.

3.2.2.3 ScienceDirect

Para realizar la búsqueda en la base de datos de ScienceDirect utilizamos expresiones para validar la combinación de palabras y el espacio entre ellas. Nos basamos en uso de "caracteres comodín" del siguiente tipo:

- "*" (asterisco): Representa que las palabras buscadas debe empezar por lo que precede el *, es decir, si se requiere buscar test, pero son válidas las variaciones testing, tested o testeable la expresión que usa el carácter comodín debería ser "test*" pues ésta la raíz necesaria para completar cualquiera de las otras opciones.
- "W/15": Representa que las palabras a buscar no pueden estar separadas más de 15 caracteres

Las técnicas utilizadas fueron obtenidas de la referencia de la base de datos, disponible en (Elsevier, B.V., 2014).

La Tabla 4 presenta las expresiones utilizadas para la búsqueda en la base de datos de ScienceDirect.

Tabla 4. Expresiones de Recolección en ScienceDirect

#	Grupo Preliminar Version 0.1
1	Software test*
2	Software W/15 test*
3	Software agile test*
4	Software W/15 agile W/15 test*
5	Software unit* test*
6	Software W/15 unit* W/15 test*
7	Software quality test*
8	Software W/15 quality W/15 test*
9	Software develop*
10	Software W/15 develop* W/15 test*
11	Software develop* agile test*
12	Software W/15 develop~ W/15 agile W/15 test*
13	Software develop* unit* test*
14	Software W/15 develop* W/15 unit* W/15 test*

#	Grupo Preliminar Version 0.1
15	Software develop* quality test*
16	Software W/15 develop* W/15 quality W/15 test*
17	Software agile unit* test*
18	Software W/15 agile W/15 unit* W/15 test*
19	Software develop* agile unit* test*
20	Software W/15 develop* W/15 agile W/15 unit* W/15 test*
21	Software develop* agile quality test*
22	Software W/15 develop* W/15 agile W/15 quality W/15 test*
23	Software develop* agile quality unit* test*
24	Software W/15 develop* W/15 agile W/15 quality W/15 unit* W/15 test*

3.2.2.4 IEEE Computer Society

La base de datos de IEEE Computer Society no soporta el uso de expresiones regulares por lo que en este caso se tuvo que realizar una búsqueda individual de cada una de las combinaciones generadas. En la Tabla 4 mostramos como definimos las expresiones que se implementarían más adelante para realizar la búsqueda.

Tabla 4. Expresiones de Recolección en IEEE Computer Society

ID	Expresión búsqueda
1	Software test
2	Software agile test
3	Software unit test
4	Software quality test
5	Software develop
6	Software develop agile test
7	Software develop unit test
8	Software develop quality test
9	Software agile unit test
10	Software develop agile unit test
11	Software develop agile quality test
12	Software develop agile quality unit test

3.2.2.5 ACM

IEEE Computer Society incluye a ACM como fuente para las búsquedas de los artículos. Esto quiere decir que las búsquedas realizadas dentro de ACM ya se

encuentran incluidas en las que se realizaron previamente dentro de IEEE Computer Society.

3.2.2.6 IEEE Xplore

Para la búsqueda dentro de IEEE Xplore definimos las expresiones que se incluyen dentro de la Tabla 5.

Tabla 5. Expresiones de Recolección en IEEE Xplore

ID	Expresión búsqueda
1	Software test*
2	Software agile test*
3	Software unit test*
4	Software quality test*
5	Software develop*
6	Software develop* agile* test*
7	Software develop* unit* test*
8	Software develop* quality test*
9	Software agile unit* test*
10	Software develop* agile unit* test*
11	Software develop* agile quality test*
12	Software develop* agile quality unit* test*

3.2.2.7 SpringerLink

La base de datos de SpringerLink al igual que IEEE Computer Society no soporta el uso de expresiones regulares por lo que definimos búsquedas individuales para las siguientes combinaciones incluidas en la Tabla 6.

Tabla 6. Expresiones de Recolección en SpringerLink

ID	Expresión búsqueda
1	Software test
2	Software agile test
3	Software unit test
4	Software quality test
5	Software develop
6	Software develop agile test
7	Software develop unit test
8	Software develop quality test

ID	Expresión búsqueda
9	Software agile unit test
10	Software develop agile unit test
11	Software develop agile quality test
12	Software develop agile quality unit test

3.2.3 Herramientas de Soporte

3.2.3.1 Mendeley

Mendeley es una herramienta que permite gestionar y compartir documentos de investigación, búsqueda de artículos y colaboración en línea, gestión de PDF's y referencias, ésta herramienta permite la eliminación de repetidos, la categorización por carpetas y la utilización de etiquetas en cada artículo lo que facilita el proceso operativo en la ejecución del estudio sistemático.

A partir del uso de esta herramienta se carga cada uno de los archivos .bib que se generan en las búsquedas de las bases de datos seleccionadas, y a partir de estos se realiza el filtro para eliminar aquellos artículos repetidos entre cada una de las búsquedas. De esta forma se categoriza cada artículo ordenadamente con su respectivo archivo pdf.

3.2.3.2 Parsers

Para la sistematización del proceso de recolección e implementación del primer filtro de manera optimizada, desarrollamos una serie de convertidores para migrar la información exportada por los motores de búsqueda en cada una de las bases de datos a una fuente unificada que simplificara el proceso de depuración de los archivos. Esto con el objetivo de eliminar aquellos artículos repetidos e implementar el filtro de búsqueda por resúmenes de una manera automática, con mayor facilidad y garantizando que el proceso sea sistemáticamente repetible.

Las herramientas son convertidores y extractores de información; son un grupo de clases creadas bajo el lenguaje Java que permiten obtener la información general de los artículos y el resumen de los mismos. Actualmente se encuentran en una fase inicial de desarrollo razón por la cual poseen algunas restricciones técnicas como: no cuentan con una interfaz gráfica, ni con instrucciones avanzadas para los usuarios además de requerir procedimientos específicos manuales para el correcto funcionamiento, a pesar de esto se encuentran disponibles a través del siguiente repositorio (Navarro & Moncada, 2014), su licencia es MIT, lo que en pocas palabras se refiere a que puede usarse con libertad pero sin garantía. Dado que han sido creadas pensando en una forma de extracción genérica, la utilización va más allá de los límites y el contexto pensado en esta investigación.

De acuerdo con las búsquedas realizadas en las bases de datos se identificaron tres tipos de archivos exportados por cada de una de estas:

- ScienceDirect: Archivo .bib
- IEEE Computer Society: Archivo .html
- IEEE Xplore: Archivo .csv
- SpringerLink: Archivo .csv

Para el uso de la herramienta de soporte para la gestión de referencias (Mendeley) existe una restricción y es que se requiere que todos los archivos se encuentren en formato .bib. De las búsquedas realizadas en las bases de datos solo ScienceDirect permite exportar directamente a este tipo de archivo. En la Figura 3 se observa una representación de la transformación de los archivos fuentes generados por cada base de datos hasta su conversión en el formato de comprensible por el gestor de referencias seleccionado, en este caso Mendeley.

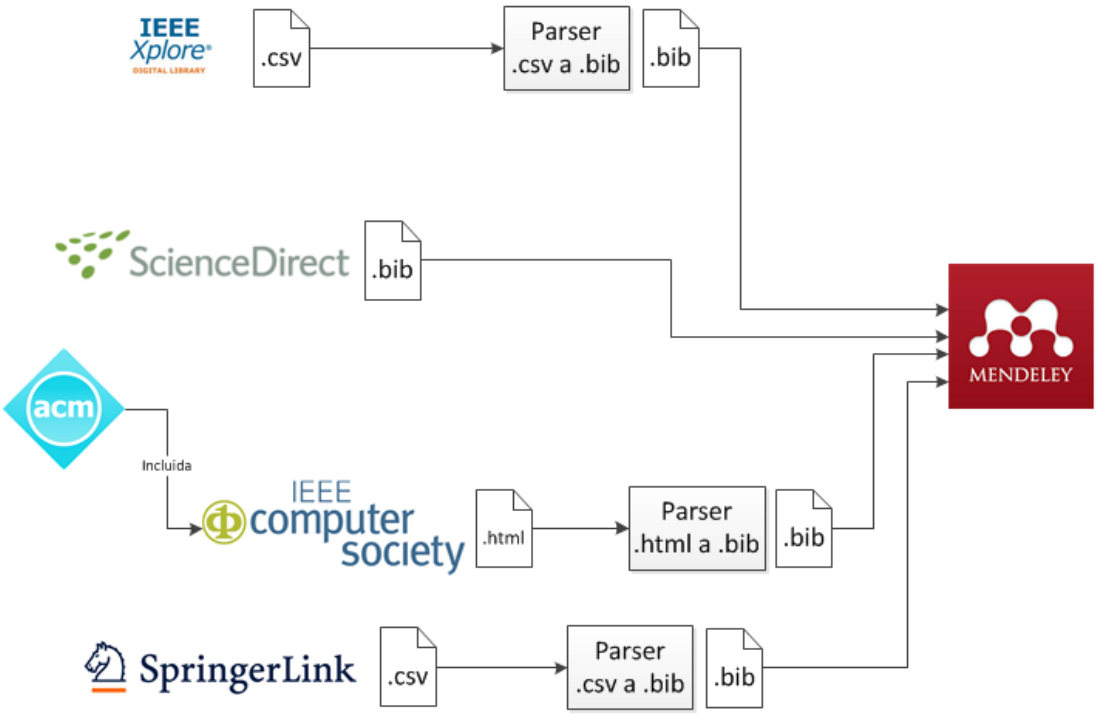

Figure 4. Diagrama de Parsers

Las herramientas para los cambios de formato de archivos de .html a .bib y de .csv a .bib se desarrollaron en el lenguaje Java como clases ejecutables que tomarán los archivos y realizarán las conversiones requeridas para generar el archivo .bib.

3.2.3.3 Buscador y selector por criterios

A partir de la necesidad de realizar un proceso automático que permita procesar la cantidad de artículos iniciales arrojados por las diferentes fuentes de datos, desarrollamos una herramienta con la capacidad de leer la biblioteca general exportada desde Mendeley y evaluar la presencia de los conjuntos de palabras requeridos. De esta manera consolidamos una estrategia de búsqueda que permite reducir el número de artículos a través de un proceso automatizado en el filtro que será descrito más adelante como *"Filtrado por Título y Resumen"*.

Esta herramienta fue creada en Java, cuenta con 2 módulos: el primero se encarga de la exploración de los resúmenes y el segundo de la exploración de títulos. Actualmente no se encuentra en una fase de usuario final, ni cuenta con una interfaz gráfica de usuario que facilite el proceso, sin embargo es de licencia MIT, por lo que su utilización está permitida sin restricciones y sin garantía expresa.

3.2.4 Filtro de Evaluación por Criterios de Exclusión

Dentro de este proceso se utiliza una herramienta de gestión de referencias, en nuestro caso usamos Mendeley como apoyo para la aplicación de los criterios de exclusión previamente definidos, estos deben filtrar aquellos artículos que estén fuera de los filtros de inclusión (fecha, tipo e idioma). Una vez se cuente con los artículos cargados directamente dentro de la herramienta, se facilita la detección y posterior eliminación de aquellos estudios que hayan pasado los filtros ya aplicados y que no cumplan con las características de inclusión. Se pueden llegar a presentar casos como lo son artículos repetidos, idiomas no contemplados, sin información sobre sus autores o que hagan parte de revistas, comentarios u otro tipo de publicación no admitida.

3.2.5 Filtrado por Título y Resumen

Una vez se tiene la primera base de artículos generados a partir del primer filtro de búsqueda, debe realizarse un nuevo filtro a partir de la información de los resúmenes de los artículos base. La idea es eliminar aquellos artículos que pasaron el filtro inicial pero que no mencionan las palabras claves dentro de sus resúmenes. De esta forma se pretende refinar aquellos artículos que realmente se encuentren relacionados a las palabras que se espera den respuesta a las preguntas de investigación.

Inicialmente se espera que se generen una cantidad significativa de artículos luego de aplicar la primera estrategia de búsqueda. Esto se debe inicialmente a que algunas de las bases de datos seleccionadas no permiten implementar una estrategia de búsqueda avanzada con los filtros sino que solo soportan estrategias de búsquedas de texto completo dentro de todo el documento.

Para lograr reducir la cantidad de artículos y depurar de una manera automática se definieron los siguientes filtros con los títulos y resúmenes de los artículos recolectados de acuerdo con las palabras de búsqueda inicialmente definidas que permita reducir la cantidad de artículos relacionados a la investigación.

El uso del asterisco (*) se refiere a la presencia total o parcial de la palabra de esta manera si la palabra encontrada puede ser test, testing, tested, testeable, etc.

Para el título se definió lo siguiente:

agil & (eval* || test*)*

Esto representa que el titulo debe estar definido como:

- Contenga la palabra ágil, *y*
- Contenga la palabra eval o test.

Para el resumen se definió lo siguiente:

software & (test || eval*) & agil* & (unit* || quality)*

De acuerdo con este filtro se espera lograr filtrar los artículos de la siguiente manera:

- Contenga la palabra software, *y*
- Contenga palabras que inicien por la palabra test o eval, *y*
- Contenga palabras que inicien por la palabra agil, *y*
- Contenga palabras que inicien por la palabra unit o contenga la palabra quality.

Este proceso se realiza a partir de la herramienta de soporte consignada en la sección: *"Buscador y selector por criterios"* que evalúa la existencia de los conjuntos de palabras previamente determinados y descritos en el protocolo.

3.2.6 Filtro de Evaluación por Criterios de Exclusión

Dentro de este proceso debe utilizarse una herramienta de gestión de referencias, en nuestro caso usamos Mendeley como apoyo para la aplicación de los criterios

de exclusión previamente definidos, estos deben filtrar aquellos artículos que estén fuera de los filtros de inclusión (fecha, tipo e idioma). Una vez se cuente con los artículos cargados directamente dentro de la herramienta, se facilita la detección y posterior eliminación de aquellos estudios que hayan pasado los filtros ya aplicados y que no cumplan con las características de inclusión. Se pueden llegar a presentar casos como lo son artículos repetidos, idiomas no contemplados, sin información sobre sus autores o que hagan parte de revistas, comentarios u otro tipo de publicación no admitida.

3.2.7 Filtrado de Evaluación Semántica

Una vez contamos con la base de artículos consolidados que han pasado por los filtros de título y resumen, se realiza una revisión manual del artículo de una forma rápida, lo cual implica revisar de nuevo los resúmenes y títulos de los artículos que hasta el momento se han recolectado con la intención de responder tres preguntas que se definieron para asegurar la relación con la investigación. Estas no son las preguntas de investigación definidas para el estudio, solo se trata de una estrategia para verificar el contexto de cada uno de los artículos.

1. ¿Incluye información sobre Software?
2. ¿Incluye información sobre Ambientes Ágiles?
3. ¿Incluye información sobre Pruebas o Calidad?

Este proceso es considerado fundamental debido a que permite minimizar la cantidad de artículos finales que se analizan en el siguiente proceso, logrando de esta manera reducir el tiempo en la obtención de documentos completos que resultarían rápidamente descartados.

Cada artículo debe ser analizado a partir de estas tres preguntas y definir si incluye o no (Si/No). Una vez se realice el análisis completo de los artículos solo tomaran en cuenta aquellos que respondan afirmativamente (Si) con mínimo dos de las preguntas realizadas. En caso de tener dudas en un artículo respecto a alguna de las preguntas ésta debe marcarse como Si, de esta manera se evita descartar un artículo con potencial de aporte a la investigación.

3.2.8 Filtrado de Evaluación de la Relación con las Preguntas de Investigación

Antes de proceder a realizar una evaluación completa sobre la calidad del artículo, debe ejecutarse un análisis previo para descartar aquellos artículos que hayan

pasado el filtro pero no guarden una relación directa con la solución de las preguntas de investigación.

Este proceso debe realizarse de manera manual haciendo una revisión de los resúmenes y resultados de los artículos que hasta el momento hayan logrado pasar los filtros previos. Se espera categorizar la relación con cada una de las preguntas en dos casos, si posee relación o si no. Cada artículo se analiza y se define su grado de relación con cada una de las preguntas de investigación definidas con base en el juicio de los expertos que realicen el proceso. Aquellos que no guarden relación con por lo menos dos de las preguntas serán descartados para su análisis respectivo.

La idea de realizar este proceso es asegurar que los artículos a los que se les realiza la evaluación de calidad completa guarden una relación directa a las preguntas de investigación. De esta manera se espera reducir el tiempo de lectura en artículos que no sean considerados relevantes.

3.2.9 Evaluación de la Calidad

Para asegurar la calidad de los artículos seleccionados a partir de los filtros previos que se definieron, se desarrolló una lista de chequeo a partir de unos criterios de calidad establecidos, asegurando obtener los artículos más relacionados al tema de investigación. La lista contiene 10 preguntas adaptadas de (Kitchenham, 2004), (Kitchenham, 2007), (Sfetsos, Stamelos, 2010), (Leedy, Ormrod, 2005), (Spencer, Ritchie, Lewis, Dillon, 2003), (Leedy, Ormrod, 2005). Dentro de cada uno de estos artículos se proponen diferentes opciones como preguntas para realizar un filtro de calidad adecuado para un estudio de investigación. En las propuestas de Kitchenham este proceso se incluye como una fase dentro del protocolo y se define todo un estudio incluido en una matriz con diferentes propuestas de preguntas de calidad de acuerdo al objetivo del estudio. En los siguientes artículos se incluyen diversas propuestas sobre las evaluaciones de calidad con base en una serie de preguntas que se debe plantear el investigador a la hora de asegurar la calidad de su estudio. En todas las propuestas se realiza una agrupación de acuerdo con la relación que guardan las preguntas con un criterio específico que guarda relación entre ellas.

De acuerdo con el análisis realizado en cada uno de los artículos que proponían preguntas base para la realización de un adecuado filtro de calidad en un estudio de investigación, se obtuvieron 10 preguntas que se consideraron relevantes durante el proceso y que cubren tres características principales de calidad que se necesitan para considerar el estudio del artículo: Rigor, Credibilidad y Relevancia. Estas preguntas se tomaron como base de acuerdo a la relación que se identificó dentro de los estudios base analizados. La relación de las preguntas seleccionadas

dentro de cada uno de los artículos que se evaluaron se puede apreciar en la tabla de Relación Artículos Criterios Calidad (Ver Anexo 1).

El primero de los criterios definidos, Rigor, responde a la pregunta: **¿El estudio sigue un riguroso y adecuado enfoque en la aplicación del protocolo?** Se desarrollaron tres preguntas para evaluar los fundamentos, objetivos y contexto del estudio, respondiendo a las siguientes preguntas secundarias:

1. ¿El artículo representa un estudio empírico?

✓ Las conclusiones o resultados son soportados por datos o estudios evidenciados en donde se pueda apreciar como el investigador llego a esas conclusiones. Los resultados son coherentes y lógicos. Las conclusiones son comparables a otros estudios del conocimiento o experiencia actual. Existe evidencia que corrobora o soporta los resultados encontrados.

2. ¿Los objetivos de la investigación se logran identificar con claridad?

✓ Se logra identificar cual es la meta u objetivos del estudio. Se define por qué este estudio es relevante y cuál es la importancia de su ejecución.

3. ¿Se encuentra claramente definido el contexto del estudio?

✓ Se define una descripción a fondo sobre el área o tema de estudio sobre el cual trabajará la investigación. Se explica el origen de cada uno de los documentos que se toman como base y el alcance al que se llegara para dar respuesta al objetivo de la investigación.

Se utilizaron otros cuatro criterios para evaluar la validez de la información recolectada, los métodos de análisis y la confiabilidad de los hallazgos. Estos criterios respondes a las siguientes preguntas secundarias:

4. ¿El diseño del estudio es apropiado para abordar los objetivos de la investigación?

✓ El investigador define claramente el proceso que seguirá para la realización del estudio, haciendo énfasis en los pasos definidos para llegar los resultados esperados.

5. ¿La estrategia de recolección de información es apropiada para dar respuesta a los objetivos de la investigación?

- ✓ Si el investigador justifica el diseño de la estrategia de recolección implementada en su estudio. Existe una discusión sobre la decisión de seleccionar el método de selección implementado.

6. ¿La información recolectada está enfocada a dar respuesta a los problemas de investigación?

   - ✓ Se pretende aclarar si la selección de los datos se encuentra justificada. Si es claro como los datos fueron recolectados, por ejemplo casos de industria, experimentos, artículos académicos, etc. Si el investigador ha justificado los métodos de recolección que se implementaron en su estudio, y si quedaron claros en sus proceso de investigación. En el caso de que los métodos de extracción hayan sufrido cambios en el proceso el investigador explica cómo y porque se realizaron. El investigador ha discutido el resultado de los datos y la saturación de los mismos.

7. ¿El análisis de datos es suficientemente riguroso?

   - ✓ Busca definir si se realiza una descripción profunda del análisis de los procesos. Si se usa algún tipo de agrupación para los datos, y en el caso de que se implemente si se menciona como estas categorías fueron derivadas de los datos. El investigador explica como los datos recolectados fueron seleccionados y las diversas fuentes de extracción. Si existen suficientes para validar los resultados del estudio.

Credibilidad responde a la pregunta **¿Los resultados del estudio son válidos, útiles y bien presentados?** Se desarrollaron dos preguntas para evaluar si los resultados del estudio son válidos y significativos. Estos criterios responden a las siguientes sub-preguntas:

8. ¿El involucramiento del investigador afecta los resultados?

   - ✓ Si el investigador dentro de su rol tiene una gran influencia durante la formulación de las preguntas de investigación, la recolección de datos y métodos de análisis. Como el investigador responde a cambios durante el transcurso del estudio y como considera estos cambios en el diseño del estudio.

9. ¿Existe una declaración clara de los resultados obtenidos?

   - ✓ Afirma si los resultados encontrados son explícitos. Si se realizó un adecuado análisis de la evidencia recolectada y las preguntas que guían el estudio realizado. Si el investigador se plantea la credibilidad de los datos y valida la correctitud de los mismos. Si los resultados obtenidos se

discuten en relación con las preguntas que se buscan responder durante el estudio.

Para el ultimo criterio, Relevancia, se responde a la siguiente pregunta **¿Son los hallazgos relevantes y útiles para la industria del software y la comunidad de investigación?** Una pregunta adicional fue desarrollada para evaluar la pertinencia de un estudio, respondiendo a la siguiente pregunta secundaria:

10. ¿El estudio proporciona valor para la investigación o la práctica?

    - ✓ Si el investigador evalúa la contribución del estudio al conocimiento existente. Si se plantea la existencia nuevas áreas que requieren de investigación para ampliar el conocimiento actual. Si el investigador discute si los resultados pueden ser transferidos a otras áreas o considera otros medios en el que la investigación puede ser de utilidad.

Cada uno de los criterios definidos será clasificado con un valor de “1” o “0”. El total de la suma de los grados de los criterios se utilizó como una medida de confianza para calificar la calidad de cada estudio evaluado.

De acuerdo con la escala definida, solo aquellos artículos que sumen un total de 7 puntos o más según las preguntas realizadas serán considerados como relevantes para realizar el proceso de extracción de información.

En la Figura 5 se realizar un esquema general en donde se incluye los criterios definidos y las preguntas planteadas para cada uno de ellos.

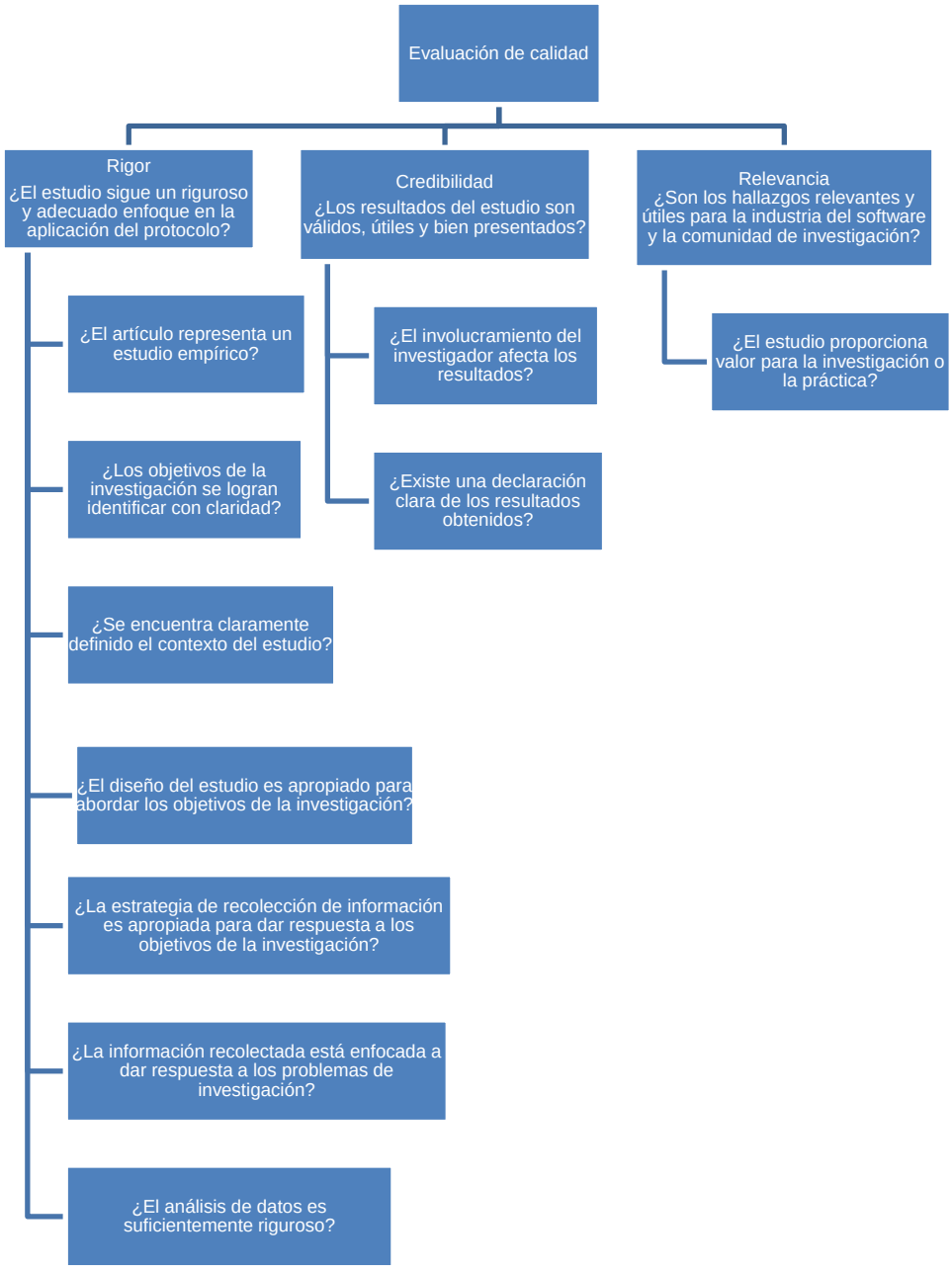

Figure 5. Criterios de Evaluación de Calidad

3.2.10 Extracción de Datos

El proceso de extracción de datos se debe realizar a partir de los artículos que pasen el proceso de calidad definido dentro de los filtros. Los datos de cada uno de los estudios recolectados debe ser almacenado en una tabla en donde se incluye la información relevante del artículo: Año, Autor, Titulo, Tipo de Estudio (Experimento,

Caso de Estudio, Mixto, etc.), Practica Ágil (Metodología descrita en el artículo), Entorno de Investigación, Prueba de Calidad (Tipos de pruebas mencionadas), Resultados (Evidencia detectada sobre pruebas de calidad en entornos ágiles y posibles conclusiones si existen). De esta forma se espera extraer la información más relevante inicialmente y detallar algunas características de cada uno de los artículos de una manera más ordena y de fácil compresión.

Si durante el proceso de Extracción de datos se presentan algunas dificultades debido a la calidad de alguno de los artículos a la hora de presentar definiciones ambiguas que no permitían clasificar adecuadamente las características del mismo, toda la información que se haya recolectado del artículo se debe someter a una discusión entre ambos autores con el objetivo de llegar a un acuerdo entre la información que se incluya como relevante

Luego de realizar la lectura detallada de cada uno de los artículos, se define una agrupación para cada una de las preguntas de investigación con factores claves que permitan dar solución a las preguntas planteadas de una manera más fácil. Este proceso se realiza a partir de la discusión de la información recolectada hasta el momento entre los actores logrando consolidar cada pregunta en una serie de categorías que ayudaran a dar respuesta al estudio de una manera más ordenada.

4. RESULTADOS OBTENIDOS

Una vez se finaliza el proceso de selección del protocolo base y el refinamiento del mismo se procede a ejecutarlo. A continuación se describen los resultados obtenidos en el proceso.

4.1 Recolección de información por base de datos

De acuerdo con lo que presentado en la sección anterior, se procede a obtener la información de cada una de las fuentes de datos seleccionadas. Se observa un alto número de artículos publicados en 2013, las fuentes fueron consultadas en el siguiente orden:

4.1.1 ScienceDirect

Como resultado del procesamiento en esta base de datos se obtuvieron 2356 artículos asociados. Una vez se eliminaron aquellos repetidos dentro del manejador de referencias, se observa un alto número de resultados producto de características propias del buscador de la base de datos, el cual trajo temas diversos que contenían las palabras clave sin que estuviera directamente relacionado con el contexto de la investigación.

Los resultados obtenidos se pueden visualizar en la Figura 6 tomando como criterio de agrupación el año de publicación de cada uno de los artículos obtenidos.

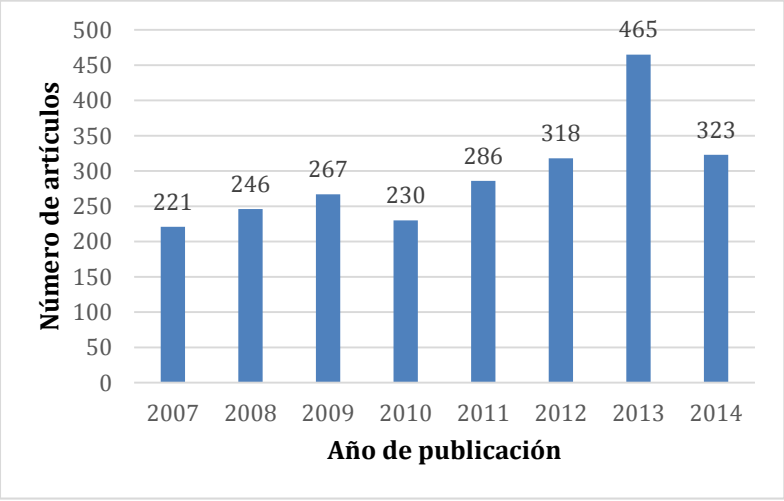

Figura 6. Distribución por Año ScienceDirect

4.1.2 IEEE ComputerSociety & ACM

Tras realizar el procedimiento descrito en el protocolo, se obtuvieron 489 artículos. En la Figura 7 se observa la cantidad de artículos distribuidos por año de publicación. Se detalle que solo 1 artículo ha sido publicado en los criterios seleccionados en 2014, atribuimos esto a la indexación de la fuente de datos o a la no publicación por parte de la fuente de datos en los primeros meses del año.

Figura 7. Distribución por Año IEEE Computer Society & ACM

4.1.3 IEEE Xplore

Después de que ejecutar el proceso seleccionado para esta base de datos, se obtuvieron 473 artículos que se ordenaron de acuerdo a su año de publicación, como se puede observar en la Figura 8.

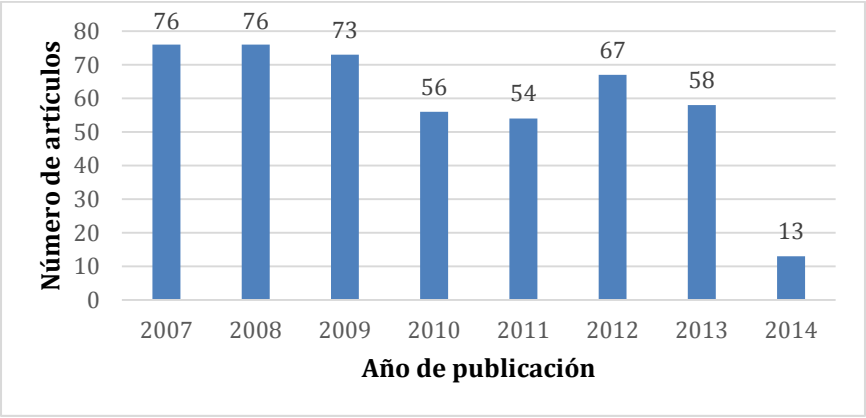

Figura 8. Distribución por Año IEEE Xplore

4.1.4 SpringerLink

Una vez que se procesó ésta base de datos, obtuvimos un total de 744 artículos que fueron referenciados de acuerdo a su año de publicación, el resultado de este proceso se puede observar en la Figura 9.

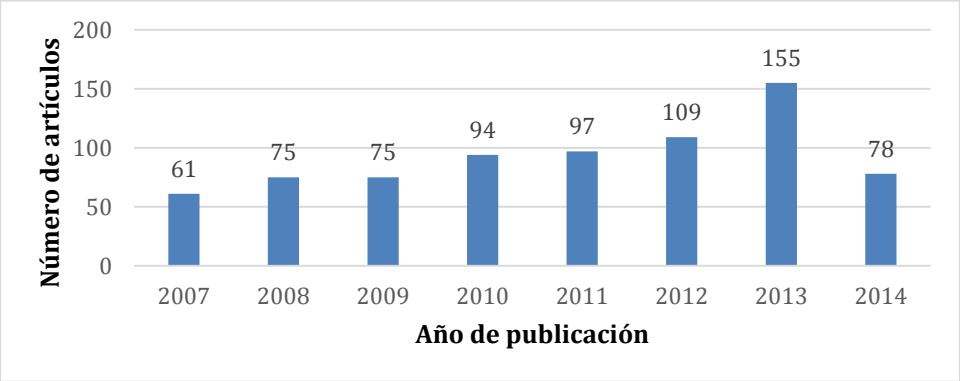

Figura 9. Distribución por Año SpringerLink

4.1.5 Resumen General

Se consolidó la información de los artículos obtenidos de cada una de las fuentes de datos. Se observa que la mayor parte de los artículos fue aportada por ScienceDirect. En la Figura 10 se visualiza los porcentajes de aporte de cada una de las fuentes de datos.

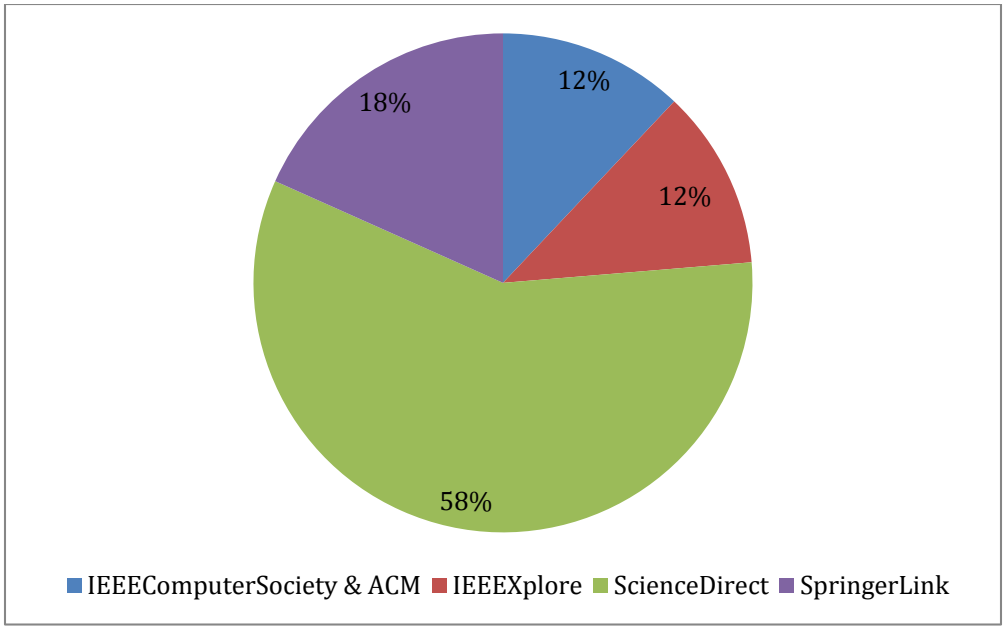

Figura 10. Distribución por Base de Datos

La información recolectada suma un total de 4062 artículos, los cuales se sometieron a una evaluación a través de los criterios de exclusión definidos. Luego de este proceso se obtuvo un total de 4003 artículos. En la Figura 11 se puede observar la distribución por año de publicación de cada uno de los artículos.

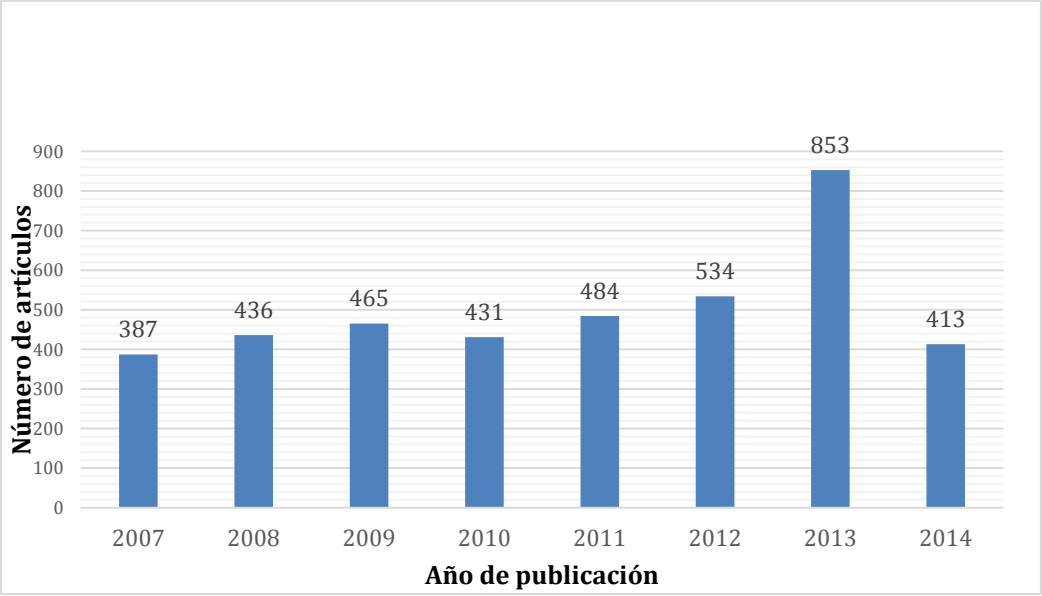

Figura 11. Distribución por Año Artículos Iniciales

4.2 Filtrado por Título y Resumen

4.2.1 Filtrado por Título

En esta fase se ejecutó el filtrado sobre los títulos de los artículos y se encontraron 19 artículos que cumplen con las expresiones definidas para esta fase en el protocolo. Los resultados segmentados por el año de publicación pueden observarse en la Figura 12. No se obtuvo ningún artículo cuyo título cumpliera con los criterios definidos y hubiese sido publicado en 2007.

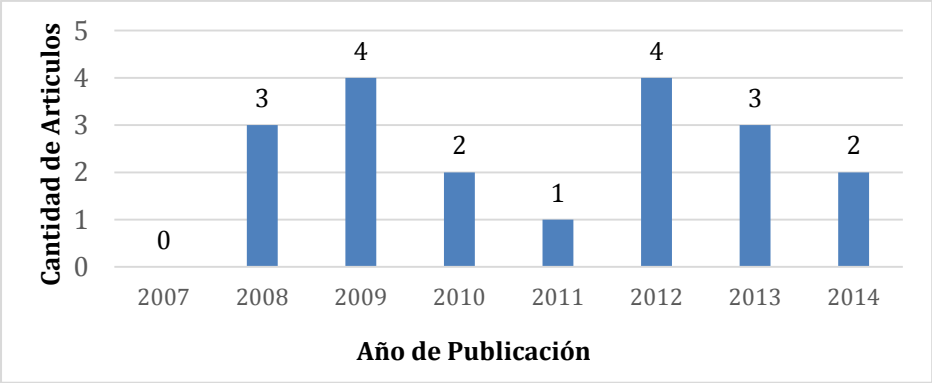

Figura 12. Filtro por Título Distribución por Año

En la Figura 13 se observa la categorización que se realizó sobre los artículos usando como criterio la distribución por publicación que lo contiene.

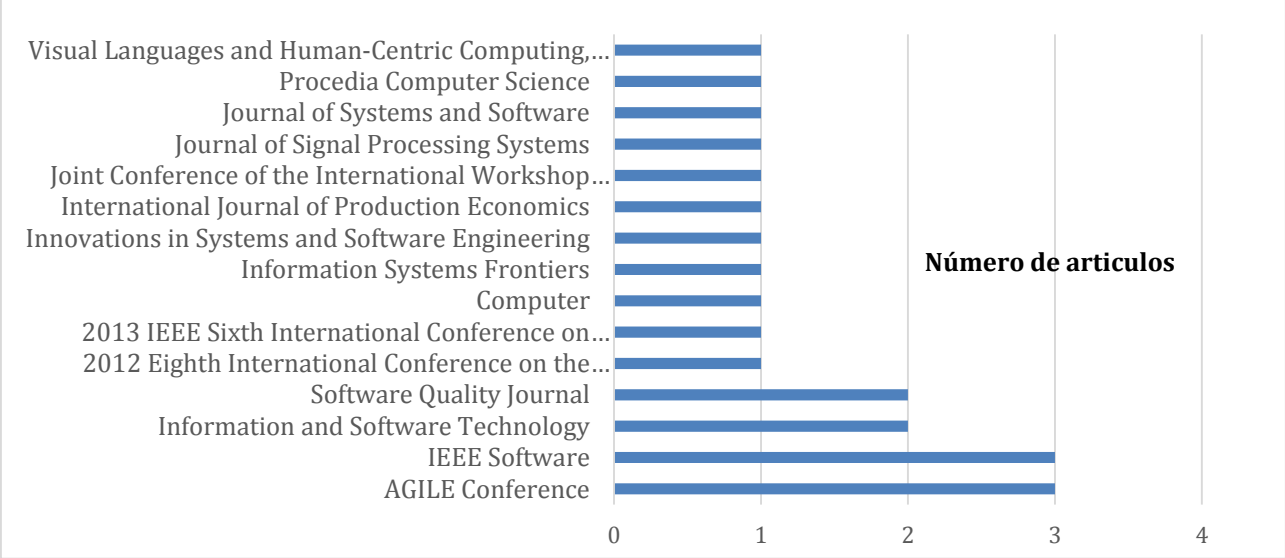

Figura 13. Filtro por Título Distribución por Fuente de Publicación

4.3 Filtrado por Resumen

Se repitió el proceso de filtrado en este caso usando el resumen de los artículos obtenidos, los resultados que se obtuvieron fueron categorizados de acuerdo a su año de publicación. En la Figura 14 se puede observar dicha categorización.

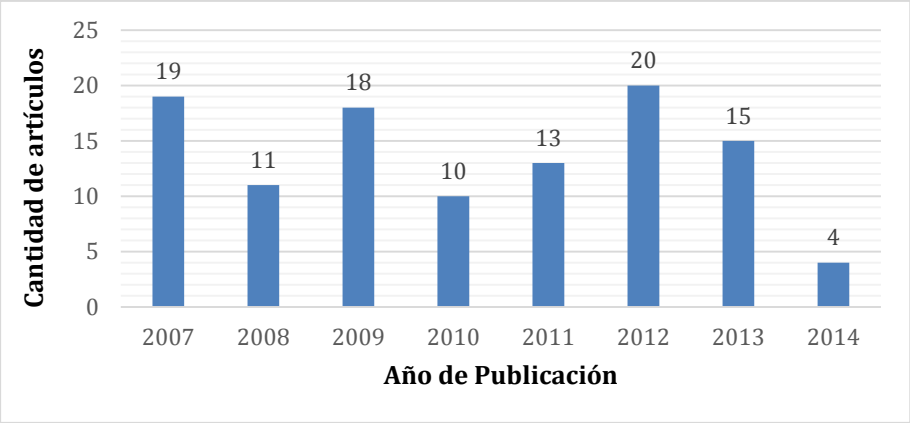

Figura 14. Filtro por Resumen Distribución por Año

En este proceso se obtuvieron 110 artículos. En la Figura 15 se muestra la segmentación realizada de acuerdo a la publicación del artículo, solo se visualizan aquellas fuentes con más de un artículo aportado.

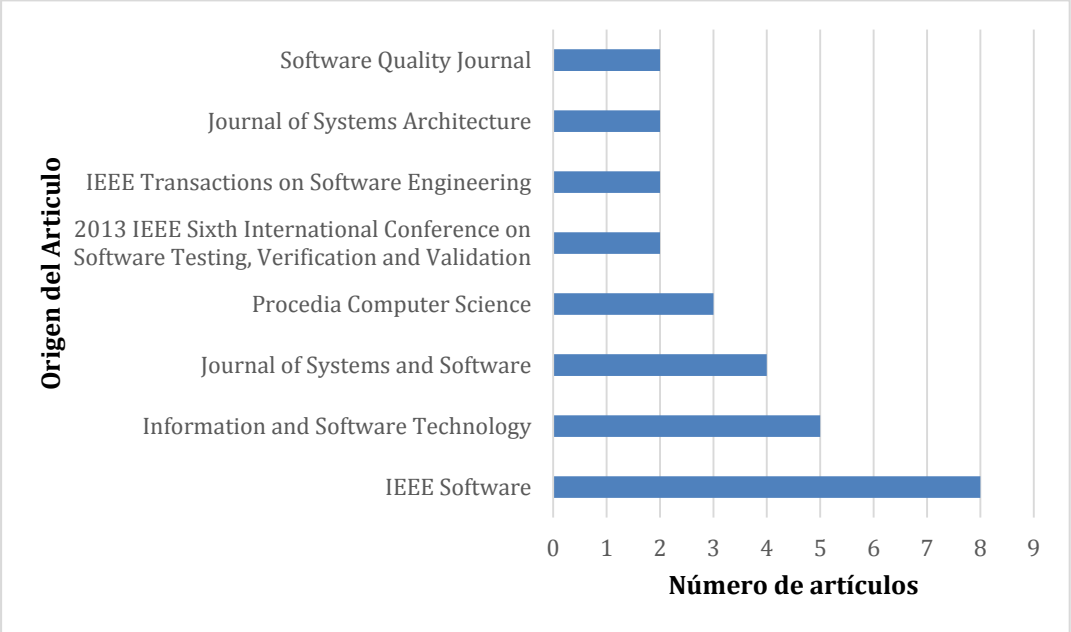

Figura 15. Filtro por Título Distribución por Fuente de Publicación

Se observa una distribución uniforme proveniente de diferentes fuentes de donde se destacan *IEEE Software, Information and Software Technology, Journal of Systems and Software, Procedia Computer Science, 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation, IEEE Transactions on Software Engineering, Journal of Systems Architecture, Software Quality Journal.* Ordenados de mayor a menor de acuerdo al número de artículos, con valores que van desde 2 hasta 8.

4.4 Filtro de Evaluación por Criterios de Exclusión

Se realizó el proceso de revisión manual de los 129 artículos provenientes del filtrado por título y resumen. Antes de iniciar el proceso de lectura se verificó que todos los artículos cumplan con los criterios de inclusión y exclusión definidos en el protocolo. Finalmente se encontró que los artículos admitidos para la siguiente fase fueron 125.

4.5 Filtrado de Evaluación Semántica

Durante la etapa de comprobación semántica, se usaron las 3 preguntas criterio definidas en el protocolo. Para esta fase la distribución de los artículos fue realizada teniendo en cuenta si hablaban o no del criterio especifico. En la Figura 16 se observa la distribución de artículos por criterio.

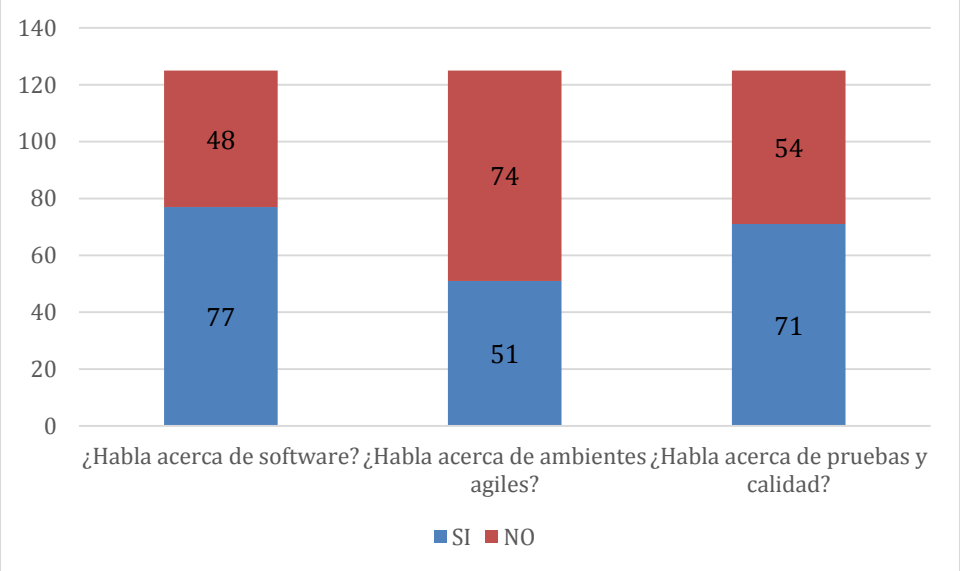

Figura 16. Evaluación Semántica Agrupación por pregunta de control

De esta etapa obtuvimos que 68 de los 125 artículos que fueron filtrados semánticamente con la investigación. En la Figura 17 se visualizan los resultados de la combinatoria posible, hemos representa cada uno de los criterios como P1, P2 y P3, como se observa a continuación:

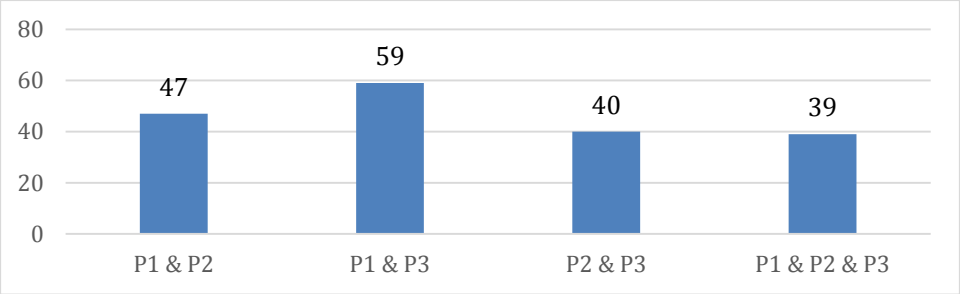

Figura 17. Filtro de Comprobación Semántica Cumplimiento de criterios básicos

Los datos que recolectamos en este paso fueron almacenados en la Tabla Filtro de Evaluación Semántica (Ver Anexo 2).

4.6 Filtro de Evaluación de la Relación con las Preguntas de Investigación

Al iniciar esta fase se contaba con 68 artículos con potencial para brindar respuestas a las preguntas de investigación. Estos artículos fueron sometidos al proceso de determinar su relación con la investigación a través de los pasos definidos en el protocolo y las preguntas que dieron inicio a la investigación (RQ 1, RQ 2, RQ 3). En la Figura 18 se puede visualizar el consolidado de los artículos para de cada una de las preguntas.

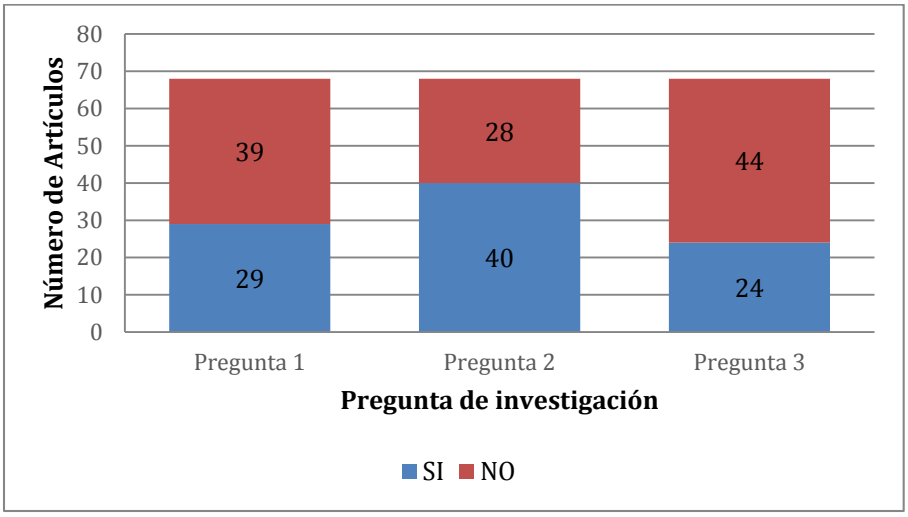

Figura 18. Filtro de Evaluación de la Relación con las Preguntas de Investigación

A partir de los resultados obtenidos se observó que de los 68 artículos, 26 responden a las pregunta 1, 38 a la pregunta 2 y 20 a la pregunta 3. Una vez se realizó este proceso, se consolidaron los artículos que responden a por lo menos dos de las preguntas de investigación. A continuación se describe el resultado obtenido:

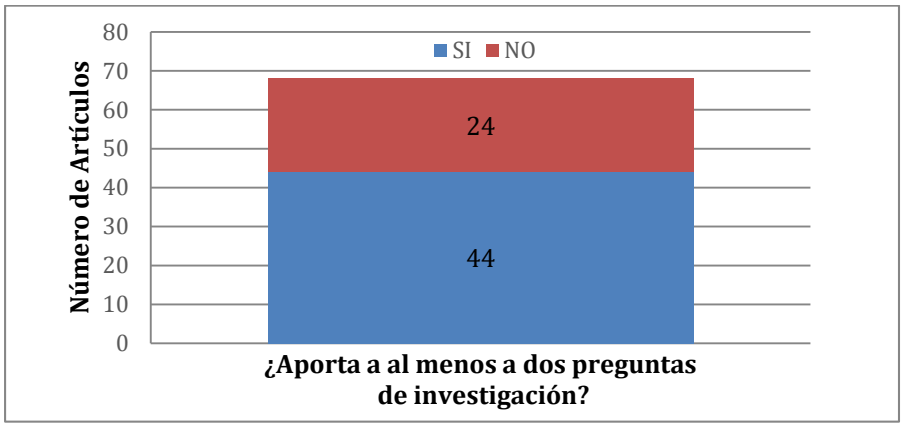

Figura 19. Consolidado Filtro de Evaluación de la Relación con las Preguntas de Investigación

Como se observa en la Figura 19 solo 44 de los 68 artículos apuntan directamente a dos de las preguntas de investigación, estos fueron categorizados usando como criterio la fuente de información de la que provienen como es visible en la Figura 19.

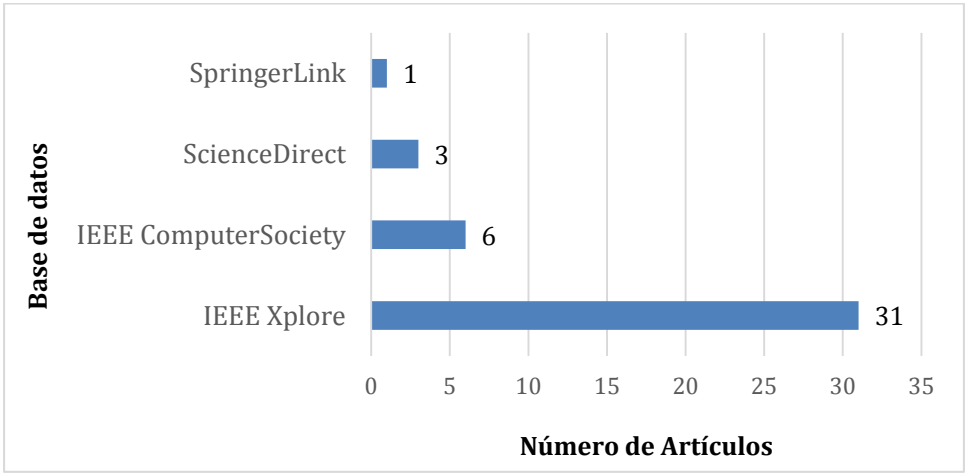

Figura 20. Artículos seleccionado por Base de Datos

En la Figura 20 se puede observar como la base de datos con mayor número de artículos aportados fue IEEE Xplore, mientras ScienceDirect que había generado el mayor número de artículos iniciales, solo obtuvo un tercer lugar en el número de aportes. La información que se obtuvo en esta fase fue registrada en la Tabla de Evaluación con las Preguntas de Investigación (Ver Anexo 3).

4.7 Evaluación de la Calidad

En esta fase se partió de los 44 artículos arrojados por el proceso anterior y se revisó cada una de las preguntas que fueron planteadas en el protocolo seleccionado como parte de la evaluación de calidad. En el anexo (Ver Anexo 4) se puede observar en detalle de cada uno de los artículos y en la Figura 21 como consolidamos los resultados obtenidos. La última columna, marcada con el rotulo "*Valor*" es nuestra primera impresión de frente a los artículos en relación al aporte posible como se describe en el protocolo.

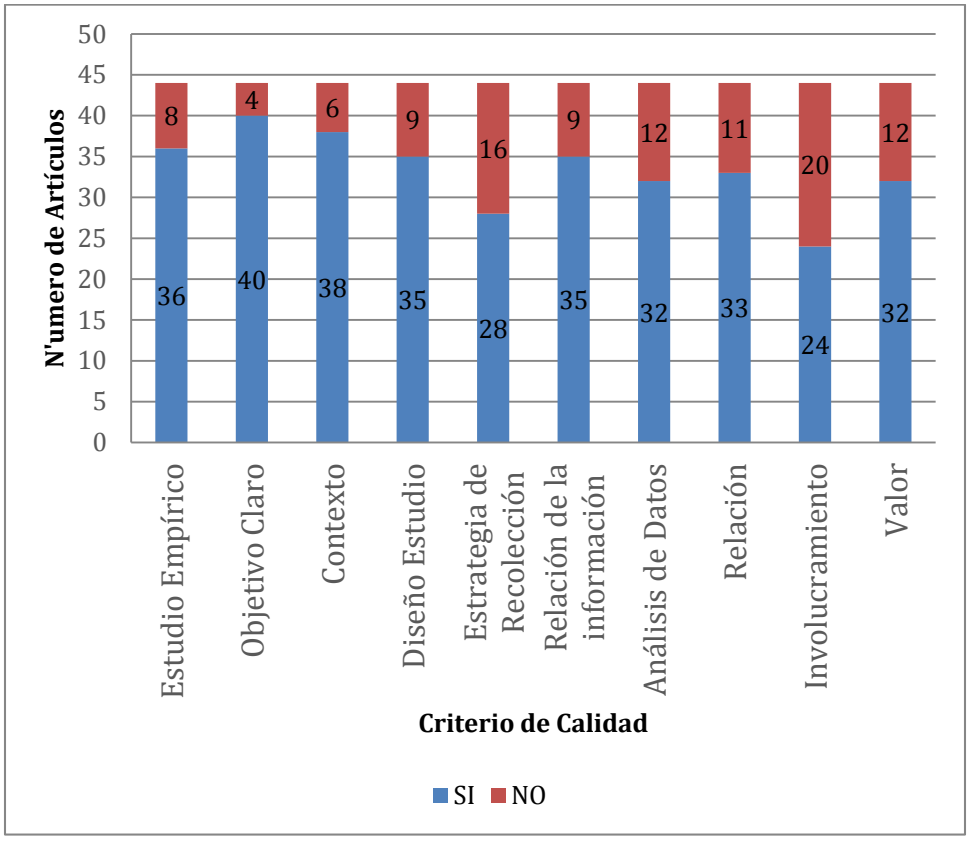

Figura 21. Evaluación de Criterios de Calidad

En La Figura 22 se describe la cantidad de artículos que presentaron la mitad de los criterios, los que presentaron 6, 7, 8, o más de 9. Se observa que 36 de los artículos cumplen más de la mitad de los criterios de calidad que fueron verificados y que la mayor agrupación se observa en los de más alto cumplimiento de calidad.

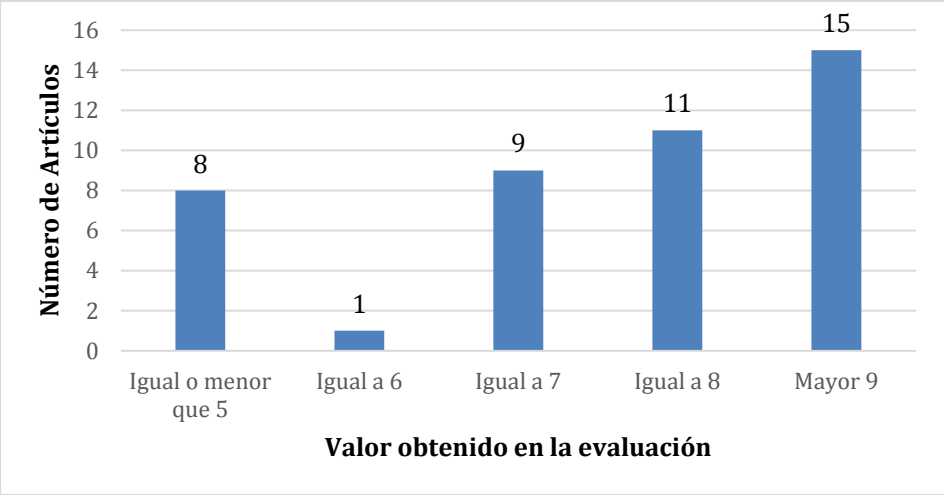

Figura 22. Relación de Artículos con los criterios de Calidad

4.8 Extracción de Datos

El proceso de extracción de datos se realizó a partir de los 44 artículos que según el filtro correspondiente, brindaban respuesta a las preguntas de investigación. La información recolecta se registró en una tabla con los criterios definidos para la etapa de Extracción de Datos del protocolo (Ver Anexo 5).

Para la síntesis de la información se almacenó la información en tablas donde se puede relacionar la relevancia de los artículos con cada una de las preguntas de investigación y los datos extraídos (Ver Anexo 6, 7 y 8). Dentro de estas tablas agrupamos por preguntas los artículos que habían pasado exitosamente los filtros de calidad.

Luego de realizar una lectura detallada de cada uno de los artículos procedimos a realizar la agrupación. Dentro de cada agrupación se definieron las siguientes categorías de acuerdo a la pregunta de investigación y lo encontrado en el grupo de artículos en revisión, tal y como fue definido en el protocolo, esto depende exclusivamente del dominio en el que se está trabajando pues debíamos determinar durante la revisión las categorías posibles:

1. Pregunta 1: ¿Cómo se logran implementar pruebas unitarias en un entorno de desarrollo ágil?

   Categorías:

   - ✓ Metodología implementada:
     Se espera identificar cual es la metodología utilizada a para trabajar el tema de pruebas unitarias en los entornos ágiles. A partir de esta información se espera dar una respuesta a que metodología es más útil dentro de los procesos agiles para abarcar las pruebas unitarias.

2. Pregunta 2: ¿Cuáles fueron los resultados obtenidos tras la implementación de pruebas unitarias en ambientes ágiles de desarrollo de software?

   Categorías:

   - ✓ Incluyen pruebas unitarias dentro de su proceso de Software:
     Se busca encontrar información que permita validar que las pruebas unitarias se utilizan dentro de los proceso de desarrollo de software y como se representan durante el ciclo de vida del software.

   - ✓ Resultados Esperados de la implementación de las pruebas:
     A partir de esta información se pretende validar cual es el valor agregado a la calidad que generan las pruebas unitarias. Qué puntos críticos atacan y cuál es la retribución o resultados de su implementación para el producto final.

3. Pregunta 3: ¿Qué tipo de pruebas unitarias se deben implementar en un ambiente de desarrollo ágil?

   Categorías:

   - ✓ Herramienta utilizada:
     Lograr identificar qué tipos de pruebas se implementan en los ciclos de desarrollo de software agiles y que tipo de herramientas utilizan para generar los casos implementados.

4.9 Resultados Finales

Luego de revisar cada uno de los filtros planteados para la ejecución del estudio sistemático se categorizaron los 44 artículos obtenidos usando criterios básicos, de acuerdo al tipo de estudio. Se obtuvo que: 5 fueron Experimentos, 28 Casos de Estudio, 5 Estudios Mixtos (Experimento/Caso de Estudio), 1 Entrevista publicada

en revista indexada y 2 Estudios Sistemáticos. En la Figura 23 puede visualizar la distribución por Tipo de Estudio de los artículos.

Figura 23. Tipos de Estudios Artículos Finales

Se realizó la distribución por año de los 44 artículos finales y se obtuvo una distribución que tiene como puntos más altos los años 2011 y 2012 y como punto más bajo el año 2010, En la Figura 24 se puede apreciar el comportamiento de la distribución.

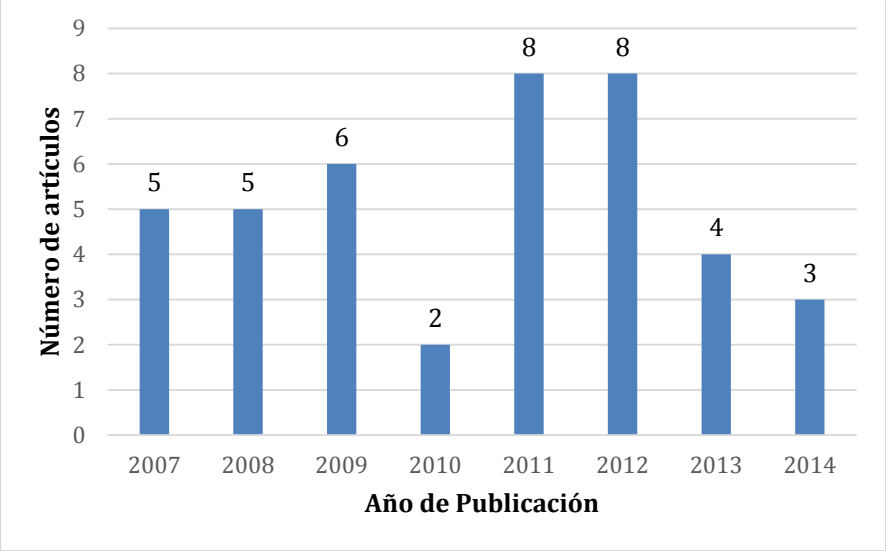

Figura 24. Distribución por Año Artículos Finales

A partir de la evaluación del tipo de metodología ágil utilizada en cada uno de los casos expuestos en los artículos, se observa que: 12 XP (Extreme Programming), 14 TDD (Test Driven Development), 13 SCRUM, 3 PP (Pair Programming) y 13 Otra (Describen metodologías ágiles pero no se menciona una específica), resultados que pueden ser apreciados en la Figura 25. Se destaca el uso de TDD como estrategia específica para pruebas ágiles.

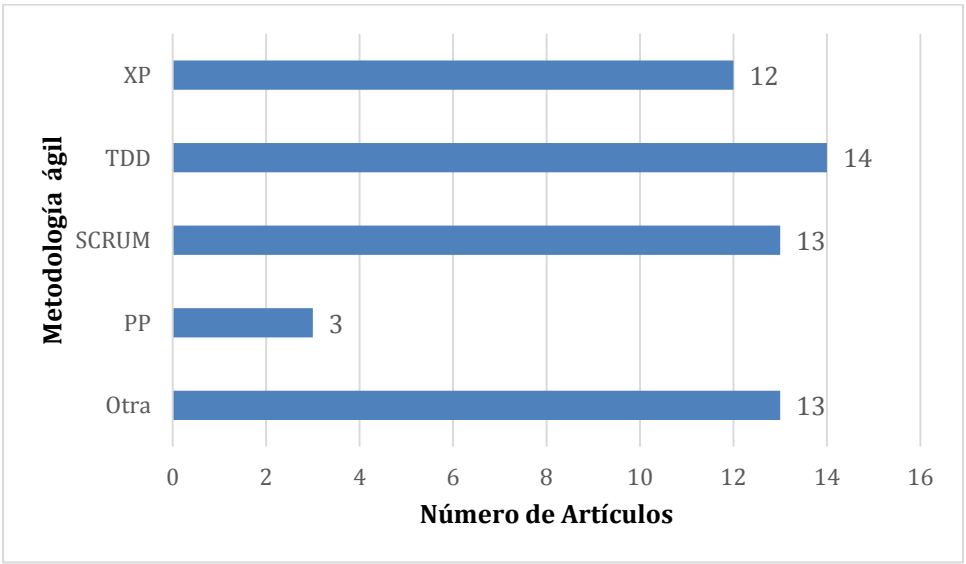

Figura 25. Metodologías Artículos Finales

En el estudio de los entornos identificamos que 12 artículos eran de Industria (Casos aplicados dentro de la industria), 23 Académicos (Estudios realizados en ambientes académicos) y 6 Mixtos (Estudio académico aplicado a casos de industria o viceversa). Esta información se puede apreciar en la Figura 26.

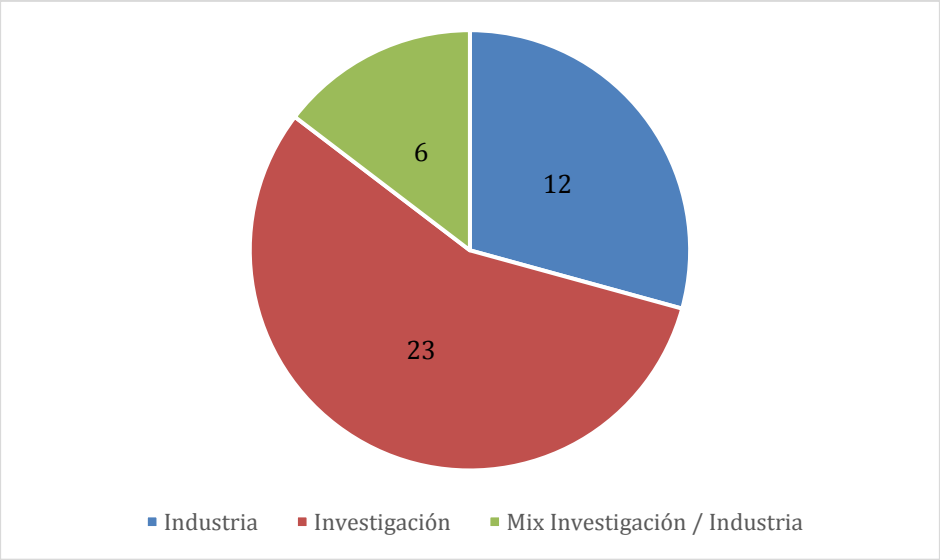

Figura 26. Distribución por Entornos Artículos Finales

Luego de realizar el análisis de la información recolectada se discutió cuáles fueron los datos recolectados con respecto al tema de pruebas unitarias en proceso de ágiles. A partir de esta discusión dimos respuesta a las preguntas de investigación (RQ1, RQ2 y RQ3) como las describimos a continuación:

4.9.1 RQ1: ¿Cómo se logran implementar pruebas unitarias en un entorno de desarrollo ágil?

Para dar respuesta a la primera pregunta del estudio, tuvimos en cuenta 24 artículos previamente seleccionados de acuerdo a la relación ésta pregunta de investigación. El análisis de estos artículos lo realizamos a través de una pregunta de categorización que busca conocer cuáles son las metodologías agiles de las reportadas incluyen pruebas unitarias y cuales son más comúnmente utilizadas (Ver Anexo 6). Esta pregunta surge tras la revisión de los artículos después de identificar que la metodología implementada influencia el procedimiento de implementación de las pruebas unitarias en los ciclos de desarrollo reportados.

Pregunta de categorización 1.1
¿Qué metodología ágil implementan en su ciclo de desarrollo de software?

Encontramos que cuando se habla del tema de pruebas unitarias y su implementación, la mayor parte de los artículos se refieren al uso de metodologías ágiles que incluyen la fase de pruebas dentro de su ciclo de vida, considerándola como una de las actividades de alta importancia dentro del proceso de desarrollo del software (Cornelissen, van Deursen, & Moonen, 2007) (Van Rompaey, 2008) (N., 2009) (Rompaey, Demeyer, & Demeyer, 2009) (Atkins & Barth, 2010) (Causevic & Sundmark, An Industrial Survey on Contemporary Aspects of Software Testing, 2010) (Fojtik, 2011) (Shelton, Li, & Ammann, 2012) (Hellmann, Chokshi, Abad, & Pratte, 2013). Muchas de estas metodologías se basan en la automatización de pruebas y poca documentación, considerando a las pruebas automatizadas como parte misma de la documentación sobre todo de los aspectos funcionales de la pieza de software (Bernardo, Sales Jr., Castor, Coelho, & Cacho, 2011).Según lo documentado un factor de gran importancia es lograr involucrar el proceso de calidad desde el inicio del ciclo de vida del software, con el objetivo que el desarrollo y la calidad sean tareas ejecutadas en paralelo (Agarwal & Garg, 2014). Las metodologías ágiles generan mejores resultados que procesos secuenciales en el contexto del estudio con ratios de incremento en diferentes métricas que van desde 4% hasta el 21% (Tarhan, 2014).

En muchos de los casos se plantea TDD como una de las prácticas más comúnmente usadas en el desarrollo ágil dentro de la industria (Causevic & Sundmark, An Industrial Survey on Contemporary Aspects of Software Testing, 2010) (Bernardo, Sales Jr., Castor, Coelho, & Cacho, 2011) (Hellmann, Chokshi, Abad, & Pratte, 2013). La implementación de ésta requiere de un equipo apasionado y centrado en las prácticas que plantea la metodología y las herramientas necesarias para lograrlo (Sanchez & Williams, 2007). Bajo esta metodología se plantea la creación y mantenimiento de un ciclo de pruebas completo que garantice el cumplimiento de los componentes desarrollados (Cornelissen, van Deursen, & Moonen, 2007). TDD requiere de un tiempo más alto para su implementación inicial debido a la codificación de todos los casos de pruebas que se logren detectar al inicio de la iteración o ciclo ágil, sin embargo, de acuerdo a la información encontrada este proceso va disminuyendo junto con su complejidad a medida que el grupo se adapta a sus prácticas mientras genera un mayor grado de calidad en el producto (Sanchez & Williams, 2007).

Por otro lado algunos estudios plantean la metodología XP dentro de los ciclos continuos de desarrollo de software (Cornelissen, van Deursen, & Moonen, 2007) (Cannizzo & Clutton, 2008) (Sanchez & Williams, 2007) (Xu, 2009) (Fojtik, 2011). XP trabaja con 12 prácticas de la cual la de mayor relevancia es la fase de pruebas, que es considerada como un proceso esencial para el mantenimiento del software, la correctitud del código y la documentación del funcionamiento (Cornelissen, van Deursen, & Moonen, 2007). Cuando hablan de equipos que no han trabajo con esta metodología observamos algunas recomendaciones como lo son: el desarrollar las

pruebas antes de codificar, la programación por pares, las pruebas automatizadas, los diseños simples, el trabajo conjunto con el cliente y la planeación de varias entregas en pequeñas iteraciones que aseguren mejorar la calidad del producto (Xu, 2009). Durante la implementación de esta metodología es crucial el uso de pruebas unitarias automáticas para la verificación constante del proyecto, esto implica que se debe estar dispuesto a reescribir código en muchas ocasiones sí así se requiere, según lo documentado en algunos casos se debe escribir alrededor 30% de código adicional(Fojtik, 2011).

Otra de las metodologías que encontramos frecuentemente mencionada es Pair Programming (PP), en el caso de la industria se requiere de un conocimiento previo que facilite la implementación de ésta metodología. Su aprendizaje es rápido y los desarrolladores lo adoptan fácilmente. El uso de PP se vuelve más sencillo a medida que los desarrolladores tienen experiencia en diferentes proyectos (Vanhanen, 2007). También plantean la adopción de SCRUM al asegurar que permite disminuir el porcentaje de severidad de las no conformidades, es decir el impacto de errores generados en el desarrollo del software a partir de las prácticas planteadas en los conceptos ágiles (Tufail, 2012).

Por ultimo algunos autores plantean la posibilidad de combinar prácticas entre las metodologías propuestas (Cornelissen, van Deursen, & Moonen, 2007) (Cannizzo & Clutton, 2008) (Fojtik, 2011) (dos Santos, Karlsson, Cavalcante, & Correia, 2011) (Bass, 2012). Generalmente dentro de la industria se muestra la utilización de TDD como metodología de pruebas junto con XP como metodología de ciclo de desarrollo y manejo de equipos (Fojtik, 2011), comúnmente ocurre cuando ya se cuenta con una experiencia previa en la ejecución de proyectos ágiles y esto produce una adaptación más rápida (Bass, 2012), en algunos casos se evidencia la fusión de prácticas SCRUM y TDD (dos Santos, Karlsson, Cavalcante, & Correia, 2011). No obstante debe entenderse que no existe una metodología ideal para todos los tipos de proyectos y que ésta solo representa una ayuda, por esta razón puede ser modificada de acuerdo a las necesidades del tipo de proyecto y los requerimientos definidos (Fojtik, 2011).

De esta manera logramos detectar que existe una alta tendencia al uso de TDD como técnica de pruebas ágiles, encontramos también que ésta es compatible con metodologías como SCRUM y XP y además que estrategias como PP mejoran el rendimiento de los equipos tras los periodos de adaptación correspondientes, este estudio nos permite afirmar que no existe una metodología ideal para un determinado tipo de proyecto, por dominio, alcance o característica especial, por el contrario se trata no del proyecto sino del equipo de desarrollo y sus habilidades.

Para lograr la implementación de pruebas unitarias en entornos ágiles de desarrollo podríamos afirmar que debe tenerse flexibilidad y la capacidad de usar las

metodologías existentes como modelos ajustables a las necesidades del proyecto y las organizaciones involucradas.

4.9.2 RQ2: ¿Cuáles fueron los resultados obtenidos tras la implementación de pruebas unitarias en ambientes ágiles de desarrollo de software?

En el proceso de dar respuesta a ésta pregunta del estudio tuvimos en cuenta los resultados obtenidos tras el proceso de filtrado y refinamiento realizado en las diferentes fases del protocolo, para la pregunta 2 analizamos los 31 artículos que tenían potencial para dar respuesta a la misma. Este análisis fue realizado mediante 2 agrupaciones que obtuvimos usando la información existente en cada uno de los artículos. La primera agrupación busca conocer si la utilización de pruebas unitarias es una práctica común y la segunda saber cuáles son los resultados con la implementación de ésta técnica (Ver Anexo 7).

Pregunta de categorización 2.1
¿Incluyen pruebas unitarias dentro de su proceso de Software?

Durante esta exploración encontramos que 13 de los 31 artículos habilitados muestran utilización de TDD como técnica para la utilización de pruebas unitarias(A., Nagappan, & Begel, 2007) (Sanchez & Williams, 2007) (Li & Praphamontripong, 2009) (N., 2009) (Causevic & Sundmark, 2010) (Causevic & Sundmark, 2011) (Fojtik, 2011) (dos Santos, Karlsson, Cavalcante, & Correia, 2011) (Scharff, 2011) (Bhasin, 2012) (Shelton, Li, & Ammann, 2012) (Hellmann, Chokshi, Abad, & Pratte, 2013) (Qusef, 2013), los 31 artículos que encontramos muestran como la utilización de metodologías agiles y cada uno de ellos habla acerca de la inclusión de pruebas unitarias en el proceso como parte esencial del ciclo de desarrollo en ambientes agiles. Observamos un comportamiento recurrente en la creación de los casos de prueba antes de la fase de desarrollo como proceso fundamental propuesto por TDD, además el artículo (Cornelissen, van Deursen, & Moonen, 2007) menciona las pruebas unitarias como una forma de documentación funcional lo que resultaría en documentar antes de programar siendo ésta una práctica poco usual pero que facilitaría la compresión y la visualización del objetivo de la iteración.

Encontramos que la creación de las pruebas es una fuente de debate interesante, diferentes propuestas existen respecto al tema pero las que encontramos de manera más recurrente fueron: la construcción de pruebas a partir de pares desarrolladores (Vanhanen, 2007) quienes piensan de manera conjunta en posibles soluciones antes de la fase de implementación, otras posturas se inclinan por la creación automática de pruebas (Collins, 2012), (Gopularam, 2012), algunos más orientados al mejoramiento continuo sostienen posturas que combinan la generación automática con la inclusión de mutación de las pruebas en el proceso

(Gopularam, 2012) como una forma de alcanzar mayor cobertura en la verificación de los componentes en desarrollo. Observamos entonces una tendencia creciente en el refinamiento de las pruebas unitarias y un alto grado de utilización dentro de los ciclos cortos o iteraciones en los ambientes agiles de desarrollo. Una situación particular se presentó con la metodología ágil específica, pues parece no incidir en el tipo de prueba que se realiza y se favorece más la relación con el Test-Driven Development o TDD que con SCRUM (Jeon, Han, & Lee, 2011), (dos Santos, Karlsson, Cavalcante, & Correia, 2011) en donde se observaron propuestas y modificaciones para el aseguramiento de requerimientos no funcionales desde las unidades mismas de desarrollo (Jeon, Han, & Lee, 2011) o con XP ((Xu, 2009), (Fojtik, 2011), (Wood & Michaelides, 2013)) él que también encontramos ampliamente difundido por el tiempo que lleva disponible.

Pregunta de categorización 2.2
¿Resultados esperados de la implementación de pruebas?

En la búsqueda de entender la relación entre las pruebas unitarias y los procesos de aseguramiento de la calidad del software en ambientes agiles de desarrollo, encontramos que en su mayoría se encuentran asociados a la idea de dar rápida solución durante las iteraciones y no al final de las mismas (N., 2009), (Scharff, 2011)) evitando situaciones como: La aparición de micro cascadas que agoten el tiempo de pruebas, errores graves que traspasan la iteración y generación sobrecostos y reproceso (Sanchez & Williams, 2007), (Usha & Poonguzhali, 2009), (Xu, 2009), (dos Santos, Karlsson, Cavalcante, & Correia, 2011), (Agarwal & Garg, 2014).

Es indicado entonces la utilización de pruebas unitarias constantes como un proceso generador de confianza y robustez en el producto final (Vanhanen, 2007), (Cannizzo & Clutton, 2008), (Li & Praphamontripong, 2009), (Causevic & Sundmark, 2010),(Bernardo, Sales Jr., Castor, Coelho, & Cacho, 2011), (Causevic & Sundmark, 2011), (Jeon, Han, & Lee, 2011), (Jinzenji, y otros, 2013), bien sea de la iteración o del proyecto, sin embargo la creación de pruebas unitarias y los beneficios que promete no es un proceso que arroje resultados inmediatos, pues según lo que encontramos documentado los resultados no serán evidentes de manera instantánea, ya que se requieren algunas características propias de las personas para alcanzar un buen rendimiento en la práctica, por ejemplo mejorarán la calidad solo sí el proceso es asumido por todos los miembros del grupo y existe un cambio de mentalidad hacia lo que significan las pruebas (A., Nagappan, & Begel, 2007), (Bavani, 2012), también se mencionan como factores impactados: la productividad (Sanchez & Williams, 2007), (N., 2009), (Rompaey, Demeyer, & Demeyer, 2009) y la adaptabilidad al cambio (Van Rompaey, 2008), (Li & Praphamontripong, 2009) , (Rompaey, Demeyer, & Demeyer, 2009), (Atkins & Barth, 2010), (Causevic & Sundmark, 2011) con la limitación evidente de requerir equipos sólidos y bien entrenados.

Como ya habíamos mencionado, el proceso de creación de pruebas también puede ser considerado como parte de la documentación funcional del producto del día (Cornelissen, van Deursen, & Moonen, 2007), mediante el uso de herramientas de tipo XUnit (marcos de trabajo de pruebas unitarias) y la adopción de propuestas procedimentales como el control de flujos de excepción (Bernardo, Sales Jr., Castor, Coelho, & Cacho, 2011), se podría según lo documentado garantizar que bajo continua integración (Fojtik, 2011) el software continúe funcionando y en caso de fallar, los errores puedan ser fácilmente rastreados (Qusef, 2013) y cuantificados en relación a las métricas impactadas durante la iteración (Collins, 2012).

El siguiente paso en el aseguramiento de la calidad a través de la implementación de pruebas unitarias en ambientes agiles de desarrollo se trata del fortalecimiento de las pruebas en relación a 3 factores: el tiempo consumido, la valides de las pruebas y la cobertura del código fuente. Respecto al primer factor, los autores describen mejoras en los tiempos cuando se utilizan generadores de pruebas unitarias (Gopularam, 2012), (Shelton, Li, & Ammann, 2012), (Tarhan, 2014), en relación al segundo punto encontramos que se procede al despliegue automático para que las pruebas se ejecuten de forma automática en el día a día lo que permite verificar si las pruebas son válidas más allá del momento especifico de creación (Gopularam, 2012), (Wood & Michaelides, 2013), (Garijo, 2014), (Tarhan, 2014) y finalmente frente a la cobertura del código fuente, encontramos herramientas como “Cobertura”, que buscan facilitar la revisión de código no verificado, es decir, saber si cada componente fue probado o cubierto por las pruebas unitarias generadas (Shelton, Li, & Ammann, 2012). Finalmente es importante tener en cuenta que observamos que las pruebas unitarias han sido extrapoladas y llevadas al contexto de Acceptance test driven (ATTD) para que se constituyan en generadores de aceptación de usuario final y no únicamente de conformidad en el componente (Bhasin, 2012) para el equipo desarrollador.

4.9.3 RQ3: ¿Qué tipo de pruebas unitarias se deben implementar en un ambiente de desarrollo ágil?

Para dar respuesta a la tercera pregunta del estudio, tuvimos en cuenta 14 artículos previamente seleccionados por la afinidad a la misma, como parte del procedimiento que desarrollamos, se tuvo en cuenta que de acuerdo a lo encontrado en los artículos el tipo de prueba unitaria se encuentra directamente vinculado con la herramienta que se usa, la metodología de pruebas seleccionada y la intención de la prueba. Dadas estas características planteamos una agrupación a través de una pregunta de categorización para ser resuelta con cada uno de los artículos y poder dar respuesta a la pregunta general (Ver Anexo 8).

Pregunta de categorización 3.1

¿Cuáles fueron las herramientas usadas para la implementación de pruebas unitarias?

Durante esta revisión detallada encontramos que 16 de los 44 artículos presentaban afinidad con el concepto de herramienta, de igual manera detectamos variaciones en criterios como: el momento de creación, la forma de creación, el objetivo de la prueba, procedimiento de despliegue y la regularidad o frecuencia de aplicación.

A continuación realizamos una descripción de las herramientas más comúnmente referenciadas en el grupo de artículos:

JUnit: Es uno de los llamados xUnit, ampliamente usado en el universo Java, se constituye en la principal herramienta para la creación de pruebas unitarias ((Cornelissen, van Deursen, & Moonen, 2007), (Cannizzo & Clutton, 2008), (Van Rompaey, 2008), (Rompaey, Demeyer, & Demeyer, 2009), (Atkins & Barth, 2010), (Causevic & Sundmark, An Industrial Survey on Contemporary Aspects of Software Testing, 2010), (Bernardo, Sales Jr., Castor, Coelho, & Cacho, 2011), (Di Bernardo & Castor, 2011), (Bavani, 2012), (Collins, 2012), (Shelton, Li, & Ammann, 2012), (Hellmann, Chokshi, Abad, & Pratte, 2013), (Qusef, 2013)), su disponibilidad ha hecho que se popularice y la comunidad de desarrolladores continúe creando nuevos aditamentos para diferentes propósitos entre ellos la trazabilidad y manejo de rutas de excepción en el software ((Atkins & Barth, 2010), (Di Bernardo & Castor, 2011), (Qusef, 2013)).

Visual Studio: Herramienta del ambiente Microsoft cuenta con un generador de pruebas unitarias para los desarrolladores del entorno .NET ((Cannizzo & Clutton, 2008))

MuJava: Es una herramienta para la generación de pruebas unitarias con mutaciones, es una herramienta libre que toma fuerza bajo la idea de generar mayor cobertura sobre el código fuente en verificación ((Li & Praphamontripong, 2009), (Shelton, Li, & Ammann, 2012)).

Selenium IDE: Es una herramienta de automatización de pruebas para entornos WEB, de libre acceso y ampliamente usada para las pruebas unitarias sobre componentes que cuentan con una GUI ((Collins, 2012), (Hellmann, Chokshi, Abad, & Pratte, 2013)).

FitNesse: Framework para el manejo de pruebas de aceptación utilizado en etapas tempranas para la definición parcial de los documentos ((Hellmann, Chokshi, Abad, & Pratte, 2013)).

Cobertura: Cobertura es una herramienta libre de Java que calcula el porcentaje de código visitado por pruebas. Puede ser utilizado para identificar qué partes de su

programa Java se carece de cobertura de la prueba. Está basado en jcoverage. (Shelton, Li, & Ammann, 2012)

5. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo presentamos la adaptación de un protocolo de investigación, basado en la propuesta metodológica realizada por Kitchenham en 2004, y sobre la cual detectamos factores que pueden dificultar la ejecución del estudio sistemático. Estos factores se relacionaron con el tipo de estudio que se consigna en cada uno de los artículos revisados (Caso de estudio, Experimento o similar) y el tamaño de la muestra inicial, es así como detectamos que el proceso puede crecer rápidamente en su complejidad, generar una mayor posibilidad de error y hacerse difícil de llevar debido a la gran cantidad de información por recopilar y analizar. Es fundamental que se pueda generar un filtrado adecuado desde las fases iniciales, por lo cual rediseñamos el protocolo de Kitchenham creando filtros que permiten asegurar que la información seleccionada para la extracción está contextualizada en relación al estudio.

Durante el proceso de exploración, encontramos fuentes dispersas que muestran desde diferentes puntos de vista el proceso de definición y ejecución de un estudio sistemático, en su mayoría guiados por las bases metodológicas propuestas por Kitchenham, quien se evidencia como el autor más referenciado en este campo. Según lo encontrado en los reportes referenciados, el crecimiento del número de estudios sistemáticos evidencia una tendencia y constituye a la metodología como una valiosa estrategia para indagar estados del arte de una forma clara y efectiva capaz de dar respuesta a preguntas específicas. Es claro que éste resulta ser un proceso de alta complejidad pues la definición de los pasos, restricciones y características de los procesos que se ejecutan implica, en muchos casos, una suerte de ensayo y error para la posterior verificación. Situaciones como: la heterogeneidad de los motores de búsqueda de las bases de datos más relevantes y las formas en que estas entregan la información, se convierten en barreras que deben ser superadas para acortar los tiempos de investigación y mantener la intervención humana o la posibilidad de error en rangos mínimos. Dada ésta situación consideramos que la automatización y definición clara de los procesos de extracción resulta fundamental para garantizar la repetitividad y por ende la confiabilidad del estudio, a la vez que fortalece y brinda veracidad al modelo propuesto.

Como parte de nuestro estudio, luego de un proceso de consolidación y limpieza de datos de manera automática, obtuvimos un total de 129 artículos que se sometieron a diferentes filtros para garantizar la relación y el aporte a las preguntas de investigación. Logramos entonces consolidar un total de 44 estudios que cumplieron con los criterios de rigor, credibilidad y relevancia definidos en el protocolo seleccionado. Observamos la posibilidad de diferentes caminos y posibles variaciones en los resultados del estudio ante cambios en la relevancia asignada a los artículos en las diferentes fuentes de información o bases de datos seleccionadas. Los datos obtenidos son una fuente veraz de información que ha

sido consolidada alrededor de la intención, la evaluación y posterior decisión sobre la implementación de pruebas unitarias en ambientes ágiles de desarrollo de software, obtenidas a través de procedimientos repetibles y claramente documentados dentro del protocolo seleccionado para la ejecución de este estudio.

Entre las cosas más relevantes que encontramos en el estudio, fue que la información recolectada nos muestra una fuerte tendencia frente a la adopción de las llamadas metodologías ágiles. Como situación especial se logra destacar que son utilizadas en contextos cada vez más amplios y por empresas diferentes a casas de software como una técnica de ajuste a mercados cada vez más cambiantes. Observamos también una marcada tendencia en la utilización de TDD como técnica guía en el proceso de pruebas y la fusión de equipos de calidad y desarrollo desde etapas muy tempranas. En relación a este punto encontramos que se recomienda de manera recurrente la multifuncionalidad de los equipos, la eliminación de las especializaciones funcionales y el reemplazo de estas por la difusión rápida de información entre los miembros del equipo. También observamos una frecuente orientación hacia los equipos auto gestionados en actividades cotidianas, como es considerada la creación de pruebas unitarias dentro del desarrollo guiado por pruebas (TDD).

En relación al objetivo general propuesto encontramos que el proceso de pruebas ágiles difiere estructuralmente del proceso tradicional, cuando se observa desde los momentos de creación, ejecución y corrección, además de variar en objetivos, magnitudes y estrategias de aplicación. Si bien el resultado esperado es el mismo, incrementar la calidad del software producido, disminuir el reproceso y el número de incidencias en cada una de las piezas de software, el proceso de aplicación no responde a los mismos principios pues la lógica detrás de este es de seguimiento y no de solo verificación. En las fuentes de información encontramos 3 características fundamentales asociadas a la adopción de pruebas unitarias en ambientes agiles. Ubicándose en el contexto de un equipo orientado al proceso de desarrollo tradicional, la primera es que no debería intentarse ser ágil inmediatamente, se debe entender al equipo como un hibrido y tratarlo como tal, sin presionar de manera exagerada los avances y apoyando una dinámica de libertad en las practicas. No se puede olvidar que ser ágil se trata de las personas. No debe pretenderse que el equipo sea ágil y además documente o realice procesos tradicionales al mismo nivel que lo venía haciendo. La segunda cuestión que detectada es que no debe tratarse a las metodologías como una camisa de fuerza, por el contrario debe generarse un ajuste entre la metodología y la organización. No intentar cambiar las reglas generales de la organización y entender que nadie le llamará hereje por cambiar SCRUM o XP por realizar modificaciones al ciclo de desarrollo de software, por utilizar solo algunas prácticas de TDD o por proponer estrategias que favorezcan al equipo aunque no estén estrictamente acotadas a lo propuesto por algún autor. Las metodologías son modelos, cada equipo debe ajustarlo para liberar su propio potencial. La ultima característica que observamos se refiere a la división de tareas,

si se desean equipos eficientes en el aseguramiento de la calidad, no debería procederse de manera secuencial, debe pensarse en las iteraciones y construir en ciclos, asignar tareas de acuerdo a las preocupaciones actuales, definir el diseño mínimo viable por cada iteración y acudir a la refactorización cuantas veces sea necesario.

La creación de un modelo de integración de las propuestas existentes que describa en detalle las fases, tendría el potencial para la generación de un estándar de exploración de estados del arte. La posibilidad de que “no expertos” realicen este tipo de indagaciones de manera más sencilla y eficiente puede recaer en gran medida en la generación futura de herramientas de búsqueda, extracción de datos, categorización y análisis de los mismos.

BIBLIOGRAFÍA

1. Kitchenham, B. (2004). *Procedures for Performing Systematic Reviews.* Keele University.

2. Kitchenham, B. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering,* Version 2.3, Keele University and University of Durham, EBSE Technical Report.

3. Sfetsos, P., & Stamelos, I. (2010). Empirical Studies on Quality in Agile Practices: A Systematic Literature Review. Aristotle University.

4. P. D. Leedy, J. E. Ormrod. (2005). Practical Research Planning and Design, Pearson Merril Prentice Hall.

5. L. Spencer, J. Ritchie, J. Lewis, L. Dillon. (2003). Quality in qualitative evaluation: A framework for assessing research evidence. London: Government Chief Social Researcher's Office.

6. Critical Appraisal Skills Programme (CASP), Available from: www.phru.nhs.uk/Pages/PHD/CASP.htm. .

7. P. D. Leedy, J. E. Ormrod. (2005). Practical Research Planning and Design, Pearson Merril Prentice Hall.

8. *Programa de transformacion productiva*. (2013). Obtenido de http://www.ptp.com.co/portal/default.aspx

9. al., B. e. (17 de Febrero de 2001). *Agile Manifesto*. Obtenido de http://agilemanifesto.org/iso/es/principles.html

10. Barbara Kitchenham , O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, Stephen Linkman. (2013). Systematic literature reviews in software engineering – A systematic literature review. *Software Engineering Group, School of Computer Science and Mathematics, Keele University*.

11. Barbara Kitchenham, Pearl Brereton. (2013). A systematic review of systematic review process research in software engineering. *School of Computing and Mathematics, Keele University*.

12. Charles A. Cianfrani, J. J. (2009). *ISO 9001:2008 Explained.* ASQ Quality Press.

13. El espectador. (4 de Septiembre de 2013). *www.elespectador.com*. Obtenido de http://www.elespectador.com/noticias/economia/colombia-grandes-ligas-de-software-articulo-444492

14. Elisa Gallo, M. V. (2009). *European Software Institute*. Obtenido de http://www.esi.es/Berrikuntza

15. Fedesoft. (2013). Estudio de la caracterización de productos y servicios dela industria de software y servicios asociados 2012. *Estudio de la caracterización de productos y servicios dela industria de software y servicios asociados 2012*. Colombia.

16. Gillies, A. (2011). *Software quality: Theory and management (3rd edition).*

17. Greenlaugh, T. (1997). How to read a paper: Papers that summarise other papers. págs. 672-675.

18. Hamill, P. (2004). *Unit Test Frameworks: Tools for High-Quality Software Development.* O'Reilly Media.

19. Impressum. (2007). *DASMA*. Obtenido de Dr. Barbara Kitchenham Biography: http://www.dasma.org/cgi-bin/contray/contray.cgi?DATA=&search=gi&ID=000008022008&GROUP=005

20. Khan, K. S. (2001). CRD's Guidance for those Carrying Out or Commissioning.

21. Kitchenham, B. (2004). *Procedures for Performing Systematic Reviews.* Keele University.
22. Nonaka, T. &. (Enero-Febrero de 1986). The New New Product Development Gamw. *Harvard Business Review*.
23. Osherove, R. (2009). *The Art of Unit Testing.* Manning Publications.
24. Paola Restrepo, P. d. (29 de Abril de 2013). *www.elespectador.com*. Obtenido de http://www.elespectador.com/noticias/economia/articulo-418980-2012-tic-vendieron-33-billones
25. Proexport Colombia. (2013). *inviertaencolombia*. Obtenido de Proexport Colombia: http://www.inviertaencolombia.com.co/sectores/servicios/software-y-servicios-de-ti.html
26. Restrepo, P. (7 de Agosto de 2013). *www.elheraldo.co*. Obtenido de http://www.elheraldo.co/noticias/economia/crecimiento-de-industria-del-software-y-ti-seria-del-27-120154
27. A., B., Nagappan, N., & Begel, A. &. (2007). Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. IEEE Computer Society.
28. Agarwal, A., & Garg, N. K. (2014). Quality assurance for Product development using Agile. 2014 International Conference on Reliability Optimization and Information Technology (ICROIT).
29. Atkins, C., & Barth, F. &. (2010). Software Testing Using Test Sheets. 2010 Third International Conference on Software Testing, Verification, and Validation Workshops.
30. Bass, J. M. (2012). Influences on Agile Practice Tailoring in Enterprise Software Development. 2012 Agile India.
31. Bavani, R. (2012). Distributed Agile, Agile Testing, and Technical Debt. IEEE Software.
32. Bernardo, R. D., Sales Jr., R., Castor, F., Coelho, R., & Cacho, N. &. (2011). Agile Testing of Exceptional Behavior. 2011 25th Brazilian Symposium on Software Engineering.
33. Bhasin, S. (2012). Quality Assurance in Agile: A Study towards Achieving Excellence. Agile India.
34. Cannizzo, F., & Clutton, R. &. (2008). Pushing the Boundaries of Testing and Continuous Integration. IEEE.
35. Causevic, A., & Sundmark, D. &. (2010). An Industrial Survey on Contemporary Aspects of Software Testing. 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation.
36. Causevic, A., & Sundmark, D. &. (2011). Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review. 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation.
37. Collins, E. F. (2012). Software Test Automation practices in agile development environment: An industry experience report. 2012 7th International Workshop on Automation of Software Test (AST).
38. Cordeiro, L., Mar, C., Valentin, E., Cruz, F., Patrick, D., & Barreto, R. &. (2008). A Platform-Based Software Design Methodology for Embedded Control Systems: An Agile Toolkit. IEEE Computer Society.
39. Cornelissen, B., van Deursen, A., & Moonen, L. &. (2007). Visualizing Testsuites to Aid in Software Understanding. IEEE.
40. Di Bernardo, R., & Castor, F. &. (2011). Towards Agile Testing of Exceptional Behavior. 2011 Fifth Latin-American Symposium on Dependable Computing Workshops.
41. dos Santos, A. M., Karlsson, B. F., Cavalcante, A. M., & Correia, I. B. (2011). Testing in an agile product development environment: An industry experience report. 2013 14th Latin American Test Workshop - LATW.

42. Fojtik, R. (2011). Extreme Programming in development of specific software. Procedia Computer Science.
43. Frezza, S. &.-H. (2007). Testing as a Mental Discipline: Practical Methods for Affecting Developer Behavior. Education & Training, 2007. CSEET'07.
44. Garijo, ï. C. (2014). Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development. SpringerLink.
45. Gopularam, B. P. (2012). Mechanism for on demand Tag-Based software testing in virtualized environments. 2012 Fourth International Conference on Advanced Computing (ICoAC).
46. Hametner, R., & Winkler, D. &. (2012). Agile testing concepts based on keyword-driven testing for industrial automation systems. IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society.
47. Harrold, M. J. (2009). Reduce, reuse, recycle, recover: Techniques for improved regression testing. IEEE.
48. Hellmann, T. D., Chokshi, A., Abad, Z. S., & Pratte, S. &. (2013). Agile Testing: A Systematic Mapping across Three Conferences: Understanding Agile Testing in the XP/Agile Universe, Agile, and XP Conferences. 2013 Agile Conference.
49. Jeon, S., Han, M., & Lee, E. &. (2011). Quality Attribute Driven Agile Development. Software Engineering Research, Management and Applications, ACIS International Conference on.
50. Jinzenji, K., Hoshino, T., Williams, L., Takahashi, K. &., Hoshino, T., L., W., y otros. (2013). An experience report for software quality evaluation in highly iterative development methodology using traditional metrics. 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE).
51. Kakarontzas, G., & Stamelos, I. &. (2008). Product Line Variability with Elastic Components and Test-Driven Development. IEEE.
52. Li, N., & Praphamontripong, U. &. (2009). An Experimental Comparison of Four Unit Test Criteria: Mutation, Edge-Pair, All-Uses and Prime Path Coverage. Software Testing Verification and Validation Workshop, IEEE International Conference on.
53. Mugridge, R. (2008). Managing Agile Project Requirements with Storytest-Driven Development. IEEE Software.
54. N., G. R. (2009). Testing Processes in Business-Critical Chain Software Lifecycle. 2009 WRI World Congress on Software Engineering.
55. Qusef, A. (2013). Test-to-code traceability: Why and how? 2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT).
56. Rompaey, B. V., Demeyer, S. &., & Demeyer, S. (2009). Establishing Traceability Links between Unit Test Cases and Units under Test. IEEE Computer Society.
57. Sanchez, J., & Williams, L. &. (2007). On the Sustained Use of a Test-Driven Development Practice at IBM. IEEE.
58. Scharff, C. (2011). Guiding global software development projects using Scrum and Agile with quality assurance. Software Engineering Education and Training, Conference on.
59. Shelton, W., Li, N., & Ammann, P. &. (2012). Adding Criteria-Based Tests to Test Driven Development. 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation.
60. Simons, A. J. (2007). JWalk: a tool for lazy, systematic testing of java classes by design introspection and user interaction. SpringerLink.

61. Tarhan, A. &. (2014). Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process. Information and Software Technology.
62. Tufail, R. &. (2012). A Case Study Analyzing the Impact of Software Process Adoption on Software Quality. 2012 10th International Conference on Frontiers of Information Technology.
63. Usha, K., & Poonguzhali, N. &. (2009). A quantitative approach for evaluating the effectiveness of refactoring in software development process. IEEE.
64. Van Rompaey, B. &. (2008). Exploring the composition of unit test suites. Automated Software Engineering.
65. Vanhanen, J. &. (2007). Experiences of Using Pair Programming in an Agile Project. IEEE.
66. Weiss, J., & Mandl, P. &. (2011). Functional testing of Complex Event Processing applications. 2011 IEEE 7th International Conference on Intelligent Computer Communication and Processing.
67. Williams, N. N. (2008). Realizing quality improvement through test driven development: results and experiences of four industrial teams. SpringerLink.
68. Wood, S., & Michaelides, G. &. (2013). Successful extreme programming: Fidelity to the methodology or good teamworking? Information and Software Technology.
69. Xu, B. (2009). Towards High Quality Software Development with Extreme Programming Methodology: Practices from Real Software Projects. 2009 International Conference on Management and Service Science.
70. Yehudai, B. P. (SpringerLink). GenUTest: a unit test and mock aspect generation tool. 2009.

ANEXOS

Anexo 1. Relación Artículos Criterios Calidad

Estudio	Rigor			Credibilidad					Relevancia	
	Estudio Empírico	Objetivo Claro	Contexto	Diseño Estudio	Estrategia de Recolección	Relación de la información	Análisis de Datos	Relación	Involucramiento	Valor
Procedures for Performing Systematic Reviews	X	X	X	X	X	X	X	X	X	X
Guidelines for performing Systematic Literature Reviews in Software Engineering	X	X	X	X	X	X	X	X	X	X
Empirical Studies on Quality in Agile Practices: A Systematic Literature Review. Aristotle University	X	X	X	X	X	X	X	X	X	X
Practical Research Planning and Design	X	X	-	X	-	X	X	X	-	X
Quality in qualitative evaluation: A framework for assessing research evidence	X	X	X	X	X	X	X	X	X	X
Critical Appraisal Skills Programme (CASP)	X	X	-	X	X	-	X	X	X	X

Anexo 2. Tabla Filtro de Evaluación Semántica

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
1	2007	Andrea, Jennitta	Envisioning Next-Generation Functional Testing Tools			X	NO
2	2007	Ferreira, Jennifer; Noble, James & Biddle, Robert	Agile Development Iterations and UI Design	X			NO
3	2007	Fojtik, Rostislav	Extreme Programming in development of specific software	X	X		SI
4	2007	Frezza, Stephen & Tang, Mei-Huei	Testing as a Mental Discipline: Practical Methods for Affecting Developer Behavior	X	X	X	SI
5	2007	Gamble, Rose F. & Hale, Matthew L.	Assessing individual performance in Agile undergraduate software engineering teams	X	X		SI
6	2007	Geras, A.M.; Smith, M.R. & Miller, J.	A survey of software testing practices in alberta	X	X	X	SI
7	2007	Gopularam, Bhanu Prakash & Yogeesha, C. B.	Mechanism for on demand Tag-Based software testing in virtualized environments	X		X	SI
8	2007	Guffey, Jordan D.; Wyglinski, Alexander M. & Minden, Gary J.	Agile Radio Implementation of OFDM Physical Layer for Dynamic Spectrum Access Research				NO
9	2007	Hametner, Reinhard; Winkler, Dietmar & Zoitl, Alois	Agile testing concepts based on keyword-driven testing for industrial automation systems	X	X	X	SI
10	2007	Harrold, Mary Jean	Reduce, reuse, recycle, recover: Techniques for improved regression testing	X	X	X	SI

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
11	2007	Hellmann, Theodore D.; Chokshi, Apoorve; Abad, Zahra Shakeri Hossein; Pratte, Sydney & Maurer, Frank	Agile Testing: A Systematic Mapping across Three Conferences: Understanding Agile Testing in the XP/Agile Universe, Agile, and XP Conferences	X	X	X	SI
12	2007	Hevner, Madeline Ann DominoRosann Webb CollinsAlan R	Controlled experimentation on adaptations of pair programming	X		X	SI
13	2007	Hill, James H.; Turner, Hamilton A.; Edmondson, James R. & Schmidt, Douglas C.	Unit Testing Non-functional Concerns of Component-based Distributed Systems	X		X	SI
14	2007	Liu, Sandy; Spencer, Bruce; Liang, Yong; Xu, Bo; Zhang, Libo & Brooks, Martin	Towards an Agile Infrastructure to Provision Devices, Applications, and Networks: A Service-oriented Approach				NO
15	2007	Porter, Adam; Yilmaz, Cemal; Memon, Atif M; Schmidt, Douglas C & Natarajan, Bala	Skoll: A Process and Infrastructure for Distributed Continuous Quality Assurance				NO
16	2007	Rahman, S M	Applying the TBC method in introductory programming courses				NO
17	2007	Sertic, Hrvoje; Marzic, Kresimir & Kalafatic, Zoran	A Project Retrospectives Method in Telecom Software Development				NO
18	2007	Sutherland, Jeff; Jakobsen, Carsten Ruseng & Johnson, Kent	Scrum and CMMI Level 5: The Magic Potion for Code Warriors	X			NO

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
19	2007	Wampler, Dean	Aspect-Oriented Design in Java/AspectJ and Ruby				NO
20	2008	Holzworth, Dean P.; Huth, Neil I. & DeVoil, Peter G.	Simple software processes and tests improve the reliability and usefulness of a model	X		X	SI
21	2008	Hu, Yongxiang	The Application and Research of Software Testing on Agile Software Development	X	X	X	SI
22	2008	Janzen D.S.; Saiedian, H; Janzen, David S & Saiedian, Hossein	On the Influence of Test-Driven Development on Software Design	X	X	X	SI
23	2008	Jeon, Sanghoon; Han, Myungjin; Lee, Eunseok & Lee, Keun	Quality Attribute Driven Agile Development		X	X	SI
24	2008	Jinzenji, Kumi; Hoshino, Takashi; Williams, Laurie; Takahashi, Kenji & Jinzenji K.; Hoshino, T.; Williams L.; Takahashi K	An experience report for software quality evaluation in highly iterative development methodology using traditional metrics	X	X	X	SI
25	2008	Juristo, Natalia; Moreno, Ana M & Strigel, Wolfgang	Guest Editors' Introduction: Software Testing Practices in Industry	X		X	SI
26	2008	Kakarontzas, George; Stamelos, Ioannis & Katsaros, Panagiotis	Product Line Variability with Elastic Components and Test-Driven Development	X	X	X	SI
27	2008	Karamat, T. & Jamil, A.N.	Reducing Test Cost and Improving Documentation In TDD (Test Driven Development)	X	X	X	SI

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
28	2008	Kim, Moonzoo; Kim, Yunho & Kim, Hotae	Unit Testing of Flash Memory Device Driver through a SAT-Based Model Checker				NO
29	2008	Narasimhadevara, A; Radhakrishnan, T; Leung, B & Jayakumar, R	On designing a usable interactive system to support transplant nursing.				NO
30	2008	Rizwan Jameel Qureshi, M. & Hussain, S.A.	An adaptive software development process model	X			NO
31	2008	Sison, Raymund	Investigating Pair Programming in a Software Engineering Course in an Asian Setting				NO
32	2008	Sutherland, Jeff; Jakobsen, Carsten Ruseng & Johnson, Kent	Scrum and CMMI Level 5: The Magic Potion for Code Warriors	X			NO
33	2008	Yau, Stephen S	Position Statement: Advances and Challenges of Software Engineering	X			NO
34	2009	Andrzeevski, Sergei	Planning Game for offshore XP project				NO
35	2009	Dimitropoulos, Georgios P.	Agility index of automatic production systems: Reconfigurable logic and open source as agility enablers				NO
36	2009	Knauth, Thomas; Fetzer, Christof & Felber, Pascal	Assertion-Driven Development: Assessing the Quality of Contracts Using Meta-Mutations				NO
37	2009	Lange, David ParsonsTeo SusnjakManfred	Influences on regression testing strategies in agile software development environments	X	X	X	SI
38	2009	Layman, Lucas; Williams, Laurie & Cunningham, Lynn	Motivations and measurements in an agile case study	X	X	X	SI

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
39	2009	Li, Nan; Praphamontripong, Upsorn & Offutt, Jeff	An Experimental Comparison of Four Unit Test Criteria: Mutation, Edge-Pair, All-Uses and Prime Path Coverage	X	X	X	SI
40	2009	Lindvall, Mikael; Muthig, Dirk; Dagnino, Aldo; Wallin, Christina; Stupperich, Michael; Kiefer, David; May, John & K?hk?nen, Tuomo	Agile Software Development in Large Organizations	X	X	X	SI
41	2009	Liu, Shaoying	An approach to applying SOFL for agile process and its application in developing a test support tool	X	X	X	SI
42	2009	Louridas, P	JUnit: Unit Testing and Coding in Tandem	X	X	X	SI
43	2009	Maher, Peter	Weaving Agile Software Development Techniques into a Traditional Computer Science Curriculum				NO
44	2009	Mahmud, Imran & Veneziano, Vito	Mind-mapping: An effective technique to facilitate requirements engineering in agile software development	X	X		SI
45	2009	Manhart P.; Schneider, K; Manhart, Peter & Schneider, Kurt	Breaking the Ice for Agile Development of Embedded Software: An Industry Experience Report	X	X		SI
46	2009	Martin, R.C.	The Test Bus Imperative: Architectures That Support Automated Acceptance Testing	X		X	SI

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
47	2009	Mishra, Deepti; Mishra, Alok & Ostrovska, Sofiya	Impact of physical ambiance on communication, collaboration and coordination in agile software development: An empirical evaluation	X	X		SI
48	2009	Mnkandla, E; Dwolatzky, B & Mnkandla E.; Dwolatzky, B	Defining Agile Software Quality Assurance	X	X	X	SI
49	2009	Monden, Akito; Hayashi, Takuma; Shinoda, Shoji; Shirai, Kumiko; Yoshida, Junichi; Barker, Mike & Matsumoto, Kenichi	Assessing the Cost Effectiveness of Fault Prediction in Acceptance Testing	X		X	SI
50	2009	N., Gautham Reddy	Testing Processes in Business-Critical Chain Software Lifecycle	X	X	X	SI
51	2009	Onions, Patrick & Patel, Chetankumar	Enterprise SoBA: Large-scale implementation of Acceptance Test Driven Story Cards			X	NO
52	2009	Pirenne, B. & Guillemot, E.	The data management system for the VENUS and NEPTUNE cabled observatories				NO
53	2009	Rajan, Sreeranga P.; Tkachuk, Oksana; Prasad, Mukul; Ghosh, Indradeep & Goel, Nitin	WEAVE: WEb Applications Validation Environment				NO
54	2010	Abbas, Noura; Gravell, Andrew M. & Wills, Gary B.	The Impact of Organization, Project and Governance Variables on Software Quality and Project Success			X	NO

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
55	2010	Hazzan, Orit & Leron, Uri	Disciplined and free-spirited: â€˜Time-out behaviourâ€™ at the Agile conference			X	NO
56	2010	Hill, James; Schmidt, D; Edmondson, James & Gokhale, Aniruddha	Tools for Continuously Evaluating Distributed System Qualities				NO
57	2010	Nagy, Ahmed; Njima, Mercy & Mkrtchyan, Lusine	A Bayesian Based Method for Agile Software Development Release Planning and Project Health Monitoring	X	X		SI
58	2010	Park, Shelly & Maurer, Frank	A Network Analysis of Stakeholders in Tool Visioning Process for Story Test Driven Development			X	NO
59	2010	Pautasso, Cesare	JOpera: An Agile Environment for Web Service Composition with Visual Unit Testing and Refactoring	X	X	X	SI
60	2010	Petersen, Kai & Wohlin, Claes	Software process improvement through the Lean Measurement (SPI-LEAM) method	X			NO
61	2010	Qusef, Abdallah	Test-to-code traceability: Why and how?	X		X	SI

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
62	2010	Ramos, Marcelo Paiva; Matuck, Gustavo Ravanhani; Matrigrani, Ciro Fernandes; Mirachi, Samoel; Segeti, Eliezer; Leite, Marco; Da Cunha, Adilson Marques & Dias, Luiz Alberto Vieira	Applying Interdisciplinarity and Agile Methods in the Development of a Smart Grids System	X		X	SI
63	2010	Rompaey, Bart Van; Demeyer, Serge & Van Rompaey B.; Demeyer, S	Establishing Traceability Links between Unit Test Cases and Units under Test	X	X	X	SI
64	2010	Sampaio, Am\'{e}rico SampaioAlexandre VasconcelosPedro R Falcone	Assessing agile methods: An empirical study	X	X		SI
65	2010	Sinha, Avik; Jr., Stanley M. Sutton & Paradkar, Amit	Text2Test: Automated Inspection of Natural Language Use Cases				NO
66	2010	Staron, Miroslaw; Meding, Wilhelm & S\{o}derqvist	A method for forecasting defect backlog in large streamline software development projects and its industrial evaluation				NO
67	2011	Alyahya, Sultan; Ivins, Wendy K & Gray, W.A.	Co-ordination Support for Managing Progress of Distributed Agile Projects		X		NO
68	2011	Garc\'{\i}a, Jes\'{u}s Quirce	A study on PDAs for onboard applications and technologies and methodologies				NO

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
69	2011	Sanchez, J.C.; Williams, L. & Maximilien, E.M.	On the Sustained Use of a Test-Driven Development Practice at IBM	X	X	X	SI
70	2011	Sangwan, R.S. & Laplante, P.A.	Test-Driven Development in Large Projects	X	X	X	SI
71	2011	dos Santos, Andreia M; Karlsson, Borje F; Cavalcante, Andre M; Correia, Igor B & Silva, Emanuel	Testing in an agile product development environment: An industry experience report	X	X	X	SI
72	2011	Scharff, Christelle	Guiding global software development projects using Scrum and Agile with quality assurance	X	X	X	SI
73	2011	Schooenderwoert, Nancy Van; Morsicato, Ron & Van Schooenderwoert N.; Morsicato, R	Taming the Embedded Tiger - Agile Test Techniques for Embedded Software	X	X	X	SI
74	2011	Shelton, William; Li, Nan; Ammann, Paul & Offutt, Jeff	Adding Criteria-Based Tests to Test Driven Development	X	X	X	SI
75	2011	da Silva, Jos\&\#233; Gon\&\#231;alo A Oliveira Basto; da Cunha, Paulo Rupino & da Silva J.G.A.; Rupino da Cunha, P	Reconciling the Irreconcilable? A Software Development Approach that Combines Agile with Formal	X	X	X	SI
76	2011	Simons, Anthony J H	JWalk: a tool for lazy, systematic testing of java classes by design introspection and user interaction	X		X	SI

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
77	2011	Stochel, Marek G; Wawrowski, Mariusz R & Waskiel, James J	Adaptive Agile Performance Modeling and Testing	X	X	X	SI
78	2011	Talby, David; Keren, Arie; Hazzan, Orit & Dubinsky, Yael	Agile software testing in a large-scale project	X		X	SI
79	2011	Tarhan, Ayca & Yilmaz, Seda Gunes	Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process	X	X	X	SI
80	2011	Tufail, Reham & Malik, Ali Afzal	A Case Study Analyzing the Impact of Software Process Adoption on Software Quality	X		X	SI
81	2012	Amaral, Vasco; Barroca, Bruno & Carreira, Paulo	Towards a Robust Solution in Building Automation Systems: Supporting Rapid Prototyping and Analysis				NO
82	2012	Coutinho, Caio Henrique; Marques, Johnny Cardoso; Yelisetty, Sarasuaty Megume Hayashi; Mirachi, Samoel; Ribeiro, Alexandre Lima Possebon; da Silva, Marcelo Amaral; da Cunha, Adilson Marques & Dias, Luiz Alberto Vieira	Developing a Smart Grids System as a PBL with Agile Method on Computer Engineering Courses				NO
83	2012	Dom\'{\i}nguez-Mayo, F.J.; Escalona, M.J.; Mej\'{\i}as, M.; Ross, M. & Staples, G.	Quality evaluation for Model-Driven Web Engineering methodologies	X			NO

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
84	2012	Ghanam, Yaser; Maurer, Frank & Abrahamsson, Pekka	Making the leap to a software platform strategy: Issues and challenges	X			NO
85	2012	Hayhoe, George F	ISO standards for software user documentation				NO
86	2012	Janus, Andre; Dumke, Reiner; Schmietendorf, Andreas & Jager, Jens	The 3C approach for Agile Quality Assurance			X	NO
87	2012	K\"{u}hner, Georg and Bluhm, Torsten and Heimann, Peter and Hennig, Christine and Kroiss, Hugo and Krom, Jon and Laqua, Heike and Lewerentz, Marc and Maier, Josef and Schacht, J\"{o}rg and Spring, Anett and Werner, Andreas and Zilker, Manfred	Progress on standardization and automation in software development on W7X	X			NO
88	2012	Montazemi, Ali Reza; Pittaway, Jeffrey James; Qahri Saremi, Hamed & Wei, Yongbin	Factors of stickiness in transfers of know-how between MNC units				NO
89	2012	Petrov, Plamen; Buy, Ugo & Nord, Robert L.	Enhancing the software architecture analysis and design process with inferred macro-architectural requirements				NO
90	2012	Rudorfer, Arnold; Stenzel, Tobias & Herold, Gerold	A Business Case for Feature-Oriented Requirements Engineering				NO

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
91	2012	Turnu, Ivana; Melis, Marco; Cau, Alessandra; Setzu, Alessio; Concas, Giulio & Mannaro, Katiuscia	Modeling and simulation of open source development using an agile practice	X	X	X	SI
92	2012	Usha, K.; Poonguzhali, N. & Kavitha, E.	A quantitative approach for evaluating the effectiveness of refactoring in software development process	X		X	SI
93	2012	Van Rompaey, Bart & Demeyer, Serge	Exploring the composition of unit test suites	X		X	SI
94	2012	Vanhanen, J.; Itkonen, J. & Sulonen, P.	Improving the interface between business and product development using agile practices and the cycles of control framework	X	X	X	SI
95	2012	Vanhanen, Jari & Korpi, Harri	Experiences of Using Pair Programming in an Agile Project	X	X	X	SI
96	2012	de Vries, Stephen	Software Testing for security	X		X	SI
97	2012	Wappler, Stefan; Wegener, Joachim & Baresel, Andr\'{e}	Evolutionary testing of software with function-assigned flags	X		X	SI
98	2012	Weiss, Johannes; Mandl, Peter & Schill, Alexander	Functional testing of Complex Event Processing applications	X		X	SI
99	2012	Williams, Nachiappan NagappanE. Michael MaximilienThirumalesh BhatLaurie	Realizing quality improvement through test driven development: results and experiences of four industrial teams	X	X	X	SI
100	2012	Wood, Stephen; Michaelides, George & Thomson, Chris	Successful extreme programming: Fidelity to the methodology or good teamworking?	X	X	X	SI

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
101	2012	Xu, Bin	Towards High Quality Software Development with Extreme Programming Methodology: Practices from Real Software Projects	X	X	X	SI
102	2012	Yehudai, Benny PasternakShmuel TyszberowiczAmiram	GenUTest: a unit test and mock aspect generation tool	X		X	SI
103	2013	Baca, Dejan & Petersen, Kai	Countermeasure graphs for software security risk assessment: An action research				NO
104	2013	Codabux, Zadia & Williams, Byron	Managing technical debt: An industrial case study				NO
105	2013	Korhonen, Kirsi	Evaluating the impact of an agile transformation: a longitudinal case study in a distributed context				NO
106	2013	Nanthaamornphong, Aziz; Morris, Karla; Rouson, Damian W. I. & Michelsen, Hope A.	A case study: Agile development in the community laser-induced incandescence modeling environment (CLiiME)	X			NO
107	2013	Pinheiro, Pl\'{a}cido Rogerio; Machado, Thais Cristina Sampaio & Tamanini, Isabelle	Dealing the Selection of Project Management through Hybrid Model of Verbal Decision Analysis				NO
108	2013	Turck, Anna HristoskovaBruno VolckaertFilip De	The WTE+ framework: automated construction and runtime adaptation of service mashups				NO
109	2013	Zhang, Hong; Kishore, Rajiv; Sharman, Raj & Ramesh, Ram	Agile Integration Modeling Language (AIML): A conceptual modeling grammar for agile integrative business information systems	X		X	SI
110	2013	Bavani, Raja	Distributed Agile, Agile Testing, and Technical Debt	X	X	X	SI

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
111	2013	Black, Sue; Boca, Paul P; Bowen, Jonathan P; Gorman, Jason & Hinchey, Mike	Formal Versus Agile: Survival of the Fittest		X		NO
112	2013	Ganguly, Anirban; Nilchiani, Roshanak & Farr, John V.	Evaluating agility in corporate enterprises		X		NO
113	2013	Garijo, ï¿½?lvaro CarreraCarlos A IglesiasMercedes	Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development	X	X		SI
114	2013	Korhonen, Kirsi	Evaluating the Effect of Agile Methods on Software Defect Data and Defect Reporting Practices - A Case Study				NO
115	2013	Meszaros, Gerard; Aston, Janice & Meszaros G.; Aston, J	Adding Usability Testing to an Agile Project	X	X	X	SI
116	2013	Miller, Michael SmithDongcheng DengSyed IslamJames	Enhancing Hardware Assisted Test Insertion Capabilities on Embedded Processors using an FPGA-based Agile Test Support Co-processor				NO
117	2013	Mugridge, Rick	Managing Agile Project Requirements with Storytest-Driven Development	X	X	X	SI
118	2013	Munetoh, Seiji; Yoshioka, Nobukazu & Munetoh S.; Yoshioka, N	RAILROADMAP: An Agile Security Testing Framework for Web-application Development	X		X	SI
119	2013	Puleio, Michael	How Not to Do Agile Testing	X	X	X	SI

Número	Año	Autor	Titulo	Software	Ambientes Ágiles	Pruebas/ Calidad	Viable
120	2013	Qumer, A. & Henderson-Sellers, B.	An evaluation of the degree of agility in six agile methods and its applicability for method engineering			X	NO
121	2014	Sadiq, Mohd & Hassan, Tanveer	An extended adaptive software development process model				NO
122	2014	Qumer, A. & Henderson-Sellers, B.	A framework to support the evaluation, adoption and improvement of agile methods in practice			X	NO
123	2014	Sedehi, Habib & Martano, Giovanni	Metrics to Evaluate \& Monitor Agile Based Software Development Projects - A Fuzzy Logic Approach			X	NO
124	2014	Stolberg, Sean	Enabling Agile Testing through Continuous Integration			X	NO
125	2014	Woo, Jungyub; Ivezic, Nenad & Cho, Hyunbo	Agile test framework for business-to-business interoperability			X	NO

Anexo 3. Tabla de Evaluación con las Preguntas de Investigación

#	Estudio			RQ 1	RQ 2	RQ 3	Viable
	Año	Autor	Titulo				
1	2007	Abbas, Noura	Agile Software Assurance: An Empirical Study	NO	NO	NO	NO
2	2007	Begel A.; Nagappan, N; Begel, Andrew & Nagappan, Nachiappan	Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study	NO	SI	NO	SI
3	2007	Cornelissen, Bas; van Deursen, Arie; Moonen, Leon & Zaidman, Andy	Visualizing Testsuites to Aid in Software Understanding	SI	SI	SI	SI
4	2007	Frezza, Stephen & Tang, Mei-Huei	Testing as a Mental Discipline: Practical Methods for Affecting Developer Behavior	NO	SI	NO	SI
5	2007	Hevner, Madeline Ann DominoRosann Webb CollinsAlan R	Controlled experimentation on adaptations of pair programming	NO	NO	NO	NO
6	2007	Sanchez, J.C.; Williams, L. & Maximilien, E.M.	On the Sustained Use of a Test-Driven Development Practice at IBM	SI	SI	NO	SI
7	2007	Simons, Anthony J H	JWalk: a tool for lazy, systematic testing of java classes by design introspection and user interaction	SI	NO	SI	SI
8	2007	Vanhanen, Jari & Korpi, Harri	Experiences of Using Pair Programming in an Agile Project	SI	SI	NO	SI
9	2007	de Vries, Stephen	Software Testing for security	NO	NO	NO	NO
10	2007	Zhang, Hong; Kishore, Rajiv; Sharman, Raj & Ramesh, Ram	Agile Integration Modeling Language (AIML): A conceptual modeling grammar for agile integrative business information systems	NO	NO	NO	NO
11	2008	Cannizzo, Fabrizio; Clutton, Robbie & Ramesh, Raghav	Pushing the Boundaries of Testing and Continuous Integration	SI	SI	SI	SI
12	2008	Cordeiro, Lucas; Mar, Carlos; Valentin, Eduardo; Cruz, Fabiano; Patrick, Daniel; Barreto, Raimundo & Lucena, Vicente	A Platform-Based Software Design Methodology for Embedded Control Systems: An Agile Toolkit	SI	SI	SI	SI

#	Estudio			RQ 1	RQ 2	RQ 3	Viable
	Año	Autor	Titulo				
13	2008	Kakarontzas, George; Stamelos, Ioannis & Katsaros, Panagiotis	Product Line Variability with Elastic Components and Test-Driven Development	NO	SI	NO	SI
14	2008	Mugridge, Rick	Managing Agile Project Requirements with Storytest-Driven Development	NO	SI	NO	SI
15	2008	Van Rompaey, Bart & Demeyer, Serge	Exploring the composition of unit test suites	SI	SI	SI	SI
16	2008	Williams, Nachiappan NagappanE. Michael MaximilienThirumalesh BhatLaurie	Realizing quality improvement through test driven development: results and experiences of four industrial teams	SI	SI	SI	NO
17	2009	Beckhaus, Arne; Karg, Lars M & Hanselmann, Gerrit	Applicability of Software Reliability Growth Modeling in the Quality Assurance Phase of a Large Business Software Vendor	NO	NO	NO	NO
18	2009	Harrold, Mary Jean	Reduce, reuse, recycle, recover: Techniques for improved regression testing	SI	SI	NO	SI
19	2009	Hill, James H.; Turner, Hamilton A.; Edmondson, James R. & Schmidt, Douglas C.	Unit Testing Non-functional Concerns of Component-based Distributed Systems	NO	NO	NO	NO
20	2009	Li, Nan; Praphamontripong, Upsorn & Offutt, Jeff	An Experimental Comparison of Four Unit Test Criteria: Mutation, Edge-Pair, All-Uses and Prime Path Coverage	NO	SI	SI	SI
21	2009	Liu, Shaoying	An approach to applying SOFL for agile process and its application in developing a test support tool	NO	NO	NO	NO
22	2009	N., Gautham Reddy	Testing Processes in Business-Critical Chain Software Lifecycle	SI	SI	NO	SI
23	2009	Rompaey, Bart Van; Demeyer, Serge & Van Rompaey B.; Demeyer, S	Establishing Traceability Links between Unit Test Cases and Units under Test	SI	SI	SI	SI

#	Estudio			RQ 1	RQ 2	RQ 3	Viable
	Año	Autor	Titulo				
24	2009	Usha, K.; Poonguzhali, N. & Kavitha, E.	A quantitative approach for evaluating the effectiveness of refactoring in software development process	NO	SI	NO	SI
25	2009	Wappler, Stefan; Wegener, Joachim & Baresel, Andr\'{e}	Evolutionary testing of software with function-assigned flags	NO	NO	NO	NO
26	2009	Xu, Bin	Towards High Quality Software Development with Extreme Programming Methodology: Practices from Real Software Projects	SI	SI	NO	SI
27	2009	Yehudai, Benny PasternakShmuel TyszberowiczAmiram	GenUTest: a unit test and mock aspect generation tool	SI	SI	SI	SI
28	2010	Atkins, Colin; Barth, Florian & Brenner, Daniel	Software Testing Using Test Sheets	SI	SI	SI	SI
29	2010	Causevic, Adnan; Sundmark, Daniel & Punnekkat, Sasikumar	An Industrial Survey on Contemporary Aspects of Software Testing	SI	SI	SI	SI
30	2010	Hu, Yongxiang	The Application and Research of Software Testing on Agile Software Development	NO	NO	NO	NO
31	2010	Nagy, Ahmed; Njima, Mercy & Mkrtchyan, Lusine	A Bayesian Based Method for Agile Software Development Release Planning and Project Health Monitoring	NO	NO	NO	NO
32	2011	Bernardo, Rafael Di; Sales Jr., Ricardo; Castor, Fernando; Coelho, Roberta; Cacho, Nelio & Soares, Sergio	Agile Testing of Exceptional Behavior	SI	SI	SI	SI
33	2011	Causevic, Adnan; Sundmark, Daniel & Punnekkat, Sasikumar	Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review	NO	SI	NO	SI
34	2011	Dam, Q. Binh; Gleason, Lance & McMullin, Vaughn	Using agility to build robust, real-time platforms for smart grid software services	NO	NO	NO	NO

#	Estudio			RQ 1	RQ 2	RQ 3	Viable
	Año	Autor	Titulo				
35	2011	Di Bernardo, Rafael; Castor, Fernando & Soares, S\'{e}rgio	Towards Agile Testing of Exceptional Behavior	SI	SI	SI	SI
36	2011	Fojtik, Rostislav	Extreme Programming in development of specific software	SI	SI	SI	SI
37	2011	Holzworth, Dean P.; Huth, Neil I. & DeVoil, Peter G.	Simple software processes and tests improve the reliability and usefulness of a model	NO	NO	NO	NO
38	2011	Jeon, Sanghoon; Han, Myungjin; Lee, Eunseok & Lee, Keun	Quality Attribute Driven Agile Development	NO	SI	NO	SI
39	2011	Mahmud, Imran & Veneziano, Vito	Mind-mapping: An effective technique to facilitate requirements engineering in agile software development	NO	NO	NO	NO
40	2011	dos Santos, Andreia M; Karlsson, Borje F; Cavalcante, Andre M; Correia, Igor B & Silva, Emanuel	Testing in an agile product development environment: An industry experience report	SI	SI	SI	SI
41	2011	Scharff, Christelle	Guiding global software development projects using Scrum and Agile with quality assurance	NO	SI	NO	SI
42	2011	Weiss, Johannes; Mandl, Peter & Schill, Alexander	Functional testing of Complex Event Processing applications	NO	SI	SI	SI
43	2012	Bartlett, Roscoe A; Heroux, Michael A & Willenbring, James M	Overview of the TriBITS lifecycle model: A Lean/Agile software lifecycle model for research-based computational science and engineering software	NO	NO	NO	NO
44	2012	Bass, Julian M	Influences on Agile Practice Tailoring in Enterprise Software Development	SI	NO	NO	SI
45	2012	Bavani, Raja	Distributed Agile, Agile Testing, and Technical Debt	NO	SI	SI	SI
46	2012	Bhasin, Sonali	Quality Assurance in Agile: A Study towards Achieving Excellence	SI	SI	NO	SI
47	2012	Collins, Eliane Figueiredo & de Lucena, Vicente Ferreira	Software Test Automation practices in agile development environment: An industry experience report	NO	SI	SI	SI

#	Estudio			RQ 1	RQ 2	RQ 3	Viable
	Año	Autor	Titulo				
48	2012	Gopularam, Bhanu Prakash & Yogeesha, C. B.	Mechanism for on demand Tag-Based software testing in virtualized environments	NO	SI	NO	SI
49	2012	Hametner, Reinhard; Winkler, Dietmar & Zoitl, Alois	Agile testing concepts based on keyword-driven testing for industrial automation systems	NO	SI	NO	SI
50	2012	Mishra, Deepti; Mishra, Alok & Ostrovska, Sofiya	Impact of physical ambiance on communication, collaboration and coordination in agile software development: An empirical evaluation	NO	NO	NO	NO
51	2012	Shelton, William; Li, Nan; Ammann, Paul & Offutt, Jeff	Adding Criteria-Based Tests to Test Driven Development	SI	SI	SI	SI
52	2012	Stochel, Marek G; Wawrowski, Mariusz R & Waskiel, James J	Adaptive Agile Performance Modeling and Testing	NO	NO	NO	NO
53	2012	Tufail, Reham & Malik, Ali Afzal	A Case Study Analyzing the Impact of Software Process Adoption on Software Quality	SI	NO	NO	SI
54	2013	Buglione, Luigi & Abran, Alain	Improving the User Story Agile Technique Using the INVEST Criteria	NO	NO	NO	NO
55	2013	Chaipraserth, Ratchanok; Leelasantitham, Adisorn & Kiattisin, Supaporn	A test automation framework in POCT system using TDD techniques	NO	NO	NO	NO
56	2013	Gamble, Rose F. & Hale, Matthew L.	Assessing individual performance in Agile undergraduate software engineering teams	NO	NO	NO	NO
57	2013	Hellmann, Theodore D.; Chokshi, Apoorve; Abad, Zahra Shakeri Hossein; Pratte, Sydney & Maurer, Frank	Agile Testing: A Systematic Mapping across Three Conferences: Understanding Agile Testing in the XP/Agile Universe, Agile, and XP Conferences	SI	SI	SI	SI
58	2013	Jinzenji, Kumi; Hoshino, Takashi; Williams, Laurie; Takahashi, Kenji & Jinzenji K.; Hoshino,	An experience report for software quality evaluation in highly iterative development methodology using traditional metrics	SI	SI	SI	SI

#	Estudio			RQ 1	RQ 2	RQ 3	Viable
	Año	Autor	Titulo				
		T.; Williams L.; Takahashi K					
59	2013	Monden, Akito; Hayashi, Takuma; Shinoda, Shoji; Shirai, Kumiko; Yoshida, Junichi; Barker, Mike & Matsumoto, Kenichi	Assessing the Cost Effectiveness of Fault Prediction in Acceptance Testing	NO	NO	NO	NO
60	2013	Munetoh, Seiji; Yoshioka, Nobukazu & Munetoh S.; Yoshioka, N	RAILROADMAP: An Agile Security Testing Framework for Web-application Development	NO	NO	NO	NO
61	2013	Parsons, David; Susnjak, Teo & Lange, Manfred	Influences on regression testing strategies in agile software development environments	NO	NO	SI	SI
62	2013	Qusef, Abdallah	Test-to-code traceability: Why and how?	SI	SI	SI	SI
63	2013	Ramos, Marcelo Paiva; Matuck, Gustavo Ravanhani; Matrigrani, Ciro Fernandes; Mirachi, Samoel; Segeti, Eliezer; Leite, Marco; Da Cunha, Adilson Marques & Dias, Luiz Alberto Vieira	Applying Interdisciplinarity and Agile Methods in the Development of a Smart Grids System	NO	NO	NO	NO
64	2013	Wood, Stephen; Michaelides, George & Thomson, Chris	Successful extreme programming: Fidelity to the methodology or good teamworking?	SI	SI	SI	SI
65	2014	Agarwal, Anil; Garg, N K & Jain, Avirag	Quality assurance for Product development using Agile	SI	NO	NO	SI
66	2014	Day, Patrick	n-Tiered Test Automation Architecture for Agile Software Systems	NO	NO	NO	NO
67	2014	Garijo, ï¿½?lvaro CarreraCarlos A IglesiasMercedes	Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development	NO	SI	NO	SI

#	Estudio			RQ 1	RQ 2	RQ 3	Viable
	Año	Autor	Titulo				
68	2014	Tarhan, Ayca & Yilmaz, Seda Gunes	Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process	SI	SI	NO	SI

Anexo 4. Tabla de Evaluación de Calidad

Número	Estudio			Estudio Empírico	Objetivo Claro	Contexto	Diseño Estudio	Estrategia de Recolección	Relación de la información	Análisis de Datos	Relación	Involucramiento	Valor
	Año	Autor	Titulo										
1	2007	Begel A.; Nagappan, N; Begel, Andrew & Nagappan, Nachiappan	Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study	1	1	1	1	0	1	1	0	1	0
2	2007	Cornelissen, Bas; van Deursen, Arie; Moonen, Leon & Zaidman, Andy	Visualizing Testsuites to Aid in Software Understanding	1	1	1	1	1	1	1	1	1	1
3	2007	Frezza, Stephen & Tang, Mei-Huei	Testing as a Mental Discipline: Practical Methods for Affecting Developer Behavior	0	0	0	0	0	1	0	1	0	0
4	2007	Sanchez, J.C.; Williams, L. & Maximilien, E.M.	On the Sustained Use of a Test-Driven Development Practice at IBM	1	1	1	1	1	1	1	1	1	1
5	2007	Vanhanen, Jari & Korpi, Harri	Experiences of Using Pair Programming in an Agile Project	1	1	1	1	0	1	1	1	1	0
6	2008	Cannizzo, Fabrizio; Clutton, Robbie & Ramesh, Raghav	Pushing the Boundaries of Testing and Continuous Integration	1	1	1	1	1	1	1	0	0	1

Número	Estudio			Estudio Empírico	Objetivo Claro	Contexto	Diseño Estudio	Estrategia de Recolección	Relación de la información	Análisis de Datos	Relación	Involucramiento	Valor
	Año	Autor	Titulo										
7	2008	Cordeiro, Lucas; Mar, Carlos; Valentin, Eduardo; Cruz, Fabiano; Patrick, Daniel; Barreto, Raimundo & Lucena, Vicente	A Platform-Based Software Design Methodology for Embedded Control Systems: An Agile Toolkit	1	1	1	0	1	0	0	0	0	0
8	2008	Kakarontzas, George; Stamelos, Ioannis & Katsaros, Panagiotis	Product Line Variability with Elastic Components and Test-Driven Development	1	1	1	0	0	0	0	0	0	0
9	2008	Mugridge, Rick	Managing Agile Project Requirements with Storytest-Driven Development	0	0	0	0	0	1	0	1	0	0
10	2008	Van Rompaey, Bart & Demeyer, Serge	Exploring the composition of unit test suites	1	1	1	0	0	1	1	1	1	0
11	2009	Harrold, Mary Jean	Reduce, reuse, recycle, recover: Techniques for improved regression testing	1	0	0	0	0	1	0	1	0	1
12	2009	Li, Nan; Praphamontripong, Upsorn & Offutt, Jeff	An Experimental Comparison of Four Unit Test Criteria: Mutation, Edge-Pair, All-Uses and Prime Path Coverage	1	1	1	1	0	0	1	1	0	1
13	2009	N., Gautham Reddy	Testing Processes in Business-Critical Chain Software Lifecycle	1	1	1	1	0	0	1	1	1	1

Número	Estudio			Estudio Empírico	Objetivo Claro	Contexto	Diseño Estudio	Estrategia de Recolección	Relación de la información	Análisis de Datos	Relación	Involucramiento	Valor
	Año	Autor	Titulo										
14	2009	Rompaey, Bart Van; Demeyer, Serge & Van Rompaey B.; Demeyer, S	Establishing Traceability Links between Unit Test Cases and Units under Test	1	1	1	1	1	1	1	1	1	1
15	2009	Usha, K.; Poonguzhali, N. & Kavitha, E.	A quantitative approach for evaluating the effectiveness of refactoring in software development process	1	1	1	1	1	1	1	0	0	0
16	2009	Xu, Bin	Towards High Quality Software Development with Extreme Programming Methodology: Practices from Real Software Projects	1	1	1	1	1	1	0	1	0	1
17	2010	Atkins, Colin; Barth, Florian & Brenner, Daniel	Software Testing Using Test Sheets	1	1	1	1	0	1	1	1	0	1
18	2010	Causevic, Adnan; Sundmark, Daniel & Punnekkat, Sasikumar	An Industrial Survey on Contemporary Aspects of Software Testing	1	1	1	1	0	1	1	1	1	1
19	2011	Bernardo, Rafael Di; Sales Jr., Ricardo; Castor, Fernando; Coelho, Roberta; Cacho, Nelio & Soares, Sergio	Agile Testing of Exceptional Behavior	1	1	1	1	1	1	1	0	0	1

Número	Estudio			Estudio Empírico	Objetivo Claro	Contexto	Diseño Estudio	Estrategia de Recolección	Relación de la información	Análisis de Datos	Relación	Involucramiento	Valor
	Año	Autor	Titulo										
20	2011	Causevic, Adnan; Sundmark, Daniel & Punnekkat, Sasikumar	Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review	1	1	1	1	1	1	0	0	0	1
21	2011	Di Bernardo, Rafael; Castor, Fernando & Soares, S\'{e}rgio	Towards Agile Testing of Exceptional Behavior	0	1	1	1	1	1	1	0	1	0
22	2011	Fojtik, Rostislav	Extreme Programming in development of specific software	1	1	1	1	0	1	1	1	1	0
23	2011	Jeon, Sanghoon; Han, Myungjin; Lee, Eunseok & Lee, Keun	Quality Attribute Driven Agile Development	1	1	1	1	1	0	1	1	0	1
24	2011	dos Santos, Andreia M; Karlsson, Borje F; Cavalcante, Andre M; Correia, Igor B & Silva, Emanuel	Testing in an agile product development environment: An industry experience report	1	1	1	1	1	1	1	1	1	1
25	2011	Scharff, Christelle	Guiding global software development projects using Scrum and Agile with quality assurance	1	1	0	1	1	1	1	0	0	1
26	2011	Weiss, Johannes; Mandl, Peter & Schill, Alexander	Functional testing of Complex Event Processing applications	1	1	1	1	1	0	0	0	1	0
27	2012	Bass, Julian M	Influences on Agile Practice Tailoring in Enterprise	1	1	1	1	1	1	1	0	1	1

Número	Estudio			Estudio Empírico	Objetivo Claro	Contexto	Diseño Estudio	Estrategia de Recolección	Relación de la información	Análisis de Datos	Relación	Involucramiento	Valor
	Año	Autor	Titulo										
			Software Development										
28	2012	Bavani, Raja	Distributed Agile, Agile Testing, and Technical Debt	0	1	1	0	1	1	1	1	0	1
29	2012	Bhasin, Sonali	Quality Assurance in Agile: A Study towards Achieving Excellence	1	1	1	1	1	1	1	1	1	1
30	2012	Collins, Eliane Figueiredo & de Lucena, Vicente Ferreira	Software Test Automation practices in agile development environment: An industry experience report	1	1	1	1	1	1	1	1	1	1
31	2012	Gopularam, Bhanu Prakash & Yogeesha, C. B.	Mechanism for on demand Tag-Based software testing in virtualized environments	0	1	1	1	1	0	1	1	1	1
32	2012	Hametner, Reinhard; Winkler, Dietmar & Zoitl, Alois	Agile testing concepts based on keyword-driven testing for industrial automation systems	0	0	0	1	0	1	1	1	0	0
33	2012	Shelton, William; Li, Nan; Ammann, Paul & Offutt, Jeff	Adding Criteria-Based Tests to Test Driven Development	1	1	1	1	1	1	1	1	1	1
34	2012	Tufail, Reham & Malik, Ali Afzal	A Case Study Analyzing the Impact of Software Process	1	1	1	1	1	1	1	1	1	1

Número	Estudio			Estudio Empírico	Objetivo Claro	Contexto	Diseño Estudio	Estrategia de Recolección	Relación de la información	Análisis de Datos	Relación	Involucramiento	Valor
	Año	Autor	Titulo										
			Adoption on Software Quality										
35	2013	Hellmann, Theodore D.; Chokshi, Apoorve; Abad, Zahra Shakeri Hossein; Pratte, Sydney & Maurer, Frank	Agile Testing: A Systematic Mapping across Three Conferences: Understanding Agile Testing in the XP/Agile Universe, Agile, and XP Conferences	1	1	1	1	1	1	1	1	1	1
36	2013	Jinzenji, Kumi; Hoshino, Takashi; Williams, Laurie; Takahashi, Kenji & Jinzenji K.; Hoshino, T.; Williams L.; Takahashi K	An experience report for software quality evaluation in highly iterative development methodology using traditional metrics	1	1	1	1	1	1	1	1	1	1
37	2013	Qusef, Abdallah	Test-to-code traceability: Why and how?	1	1	1	1	1	1	1	1	1	1
38	2013	Wood, Stephen; Michaelides, George & Thomson, Chris	Successful extreme programming: Fidelity to the methodology or good teamworking?	1	1	0	1	1	1	0	1	1	1
39	2014	Agarwal, Anil; Garg, N K & Jain, Avirag	Quality assurance for Product development using Agile	0	1	1	1	0	1	1	1	1	1

Número	Estudio			Estudio Empírico	Objetivo Claro	Contexto	Diseño Estudio	Estrategia de Recolección	Relación de la información	Análisis de Datos	Relación	Involucramiento	Valor
	Año	Autor	Titulo										
40	2014	Garijo, ï¿½?lvaro CarreraCarlos A IglesiasMercedes	Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development	0	1	1	1	1	1	0	1	0	1
41	2014	Tarhan, Ayca & Yilmaz, Seda Gunes	Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process	1	1	1	1	1	1	1	1	1	1
42	2007	Simons, Anthony J H	JWalk: a tool for lazy, systematic testing of java classes by design introspection and user interaction	1	1	1	0	0	0	0	1	0	1
43	2008	Williams, Nachiappan NagappanE. Michael MaximilienThirumalesh BhatLaurie	Realizing quality improvement through test driven development: results and experiences of four industrial teams	1	1	1	1	1	1	1	1	0	1
44	2009	Yehudai, Benny PasternakShmuel TyszberowiczAmiram	GenUTest: a unit test and mock aspect generation tool	1	1	1	0	0	0	0	1	0	1

Anexo 5. Tabla de Resultados de Estudios Seleccionados

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
1	2007	Begel A.; Nagappan, N; Begel, Andrew & Nagappan, Nachiappan	Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study	Experimento	XP, TDD	Industria	Ninguna	* Se analiza que alrededor del 15% de los proyectos utilizan TDD, y en un 50% de los proyectos lo realizan algunas veces. * Las metodologías ágiles proveen beneficios a la hora de mejorar las comunicaciones, entregas rápidas y flexibilidad en los cambios. * Presentan falencias en proyectos de largo alcance, con demasiadas reuniones y con grupos que desconocen los procesos ágiles.
2	2007	Cornelissen, Bas; van Deursen, Arie; Moonen, Leon & Zaidman, Andy	Visualizing Testsuites to Aid in Software Understanding	Caso de Estudio	SCRUM, XP, TDD	Investigación	Pruebas Unitarias, Pruebas de Aceptación	* Utilizar abstracciones más avanzadas que definan flujos de procesos y control facilita la implementación de pruebas de aceptación. * El desarrollo de pruebas unitarias y pruebas de aceptación facilita la compresión del código desarrollado. * JUnit test es un conjunto de bibliotecas que exponen funcionalidades para realizar pruebas unitarias en lenguaje JAVA facilitando la compresión de los programas. * La implementación de pruebas

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
								unitarias requiere de abstracción del código.
3	2007	Frezza, Stephen & Tang, Mei-Huei	Testing as a Mental Discipline: Practical Methods for Affecting Developer Behavior	Caso de Estudio	TDD	Investigación	Ninguna	* El proceso de testing debe convertirse en una disciplina que evalué métodos y cuantifique el valor de cada uno de estos. * El proceso de elaboración de pruebas debe ser previo al inicio del desarrollo.

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
4	2007	Sanchez, J.C.; Williams, L. & Maximilien, E.M.	On the Sustained Use of a Test-Driven Development Practice at IBM	Mix Experimento / Caso de Estudio	TDD	Mix Investigación / Industria	Pruebas Unitarias	* TDD ofrece un mejor desempeño en términos de tiempo de desarrollo y confiabilidad del código. * El proceso de documentación debe ser detallado y especifico cuando se quiere tener un proceso de desarrollo de pruebas unitarias óptimo. * Al inicio el proceso de implementación de TDD demanda más tiempo, sin embargo después se convierte en un proceso más sencillo que se retribuye en un proceso más robusto y con mejor calidad. * El desarrollo de pruebas unitarias automatizadas permiten conocer en tiempo real el impacto en futuros cambios y el estado actual de lo ya desarrollado. * JUnit es un framework bastante elaborado que soporta correctamente el desarrollo de pruebas unitarias en java. * Crear un buen diseño y una estructura del código usando los principios de programación orientada a objetos facilita el uso de pruebas automatizadas.

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
5	2007	Vanhanen, Jari & Korpi, Harri	Experiences of Using Pair Programming in an Agile Project	Caso de Estudio	PP	Investigación	Pruebas Unitarias	* La programación por pares es soportada en TDD sin embargo incrementa la comunicación entre ambos. * Dos personas pueden desarrollar un código que es más fácil de probar y diseñando buenos casos de pruebas unitarios.
6	2008	Cannizzo, Fabrizio; Clutton, Robbie & Ramesh, Raghav	Pushing the Boundaries of Testing and Continuous Integration	Caso de Estudio	XP	Investigación	Pruebas Unitarias, Pruebas de Aceptación	* El desarrollo extensivo de pruebas unitarias y de aceptación en la primera versión del código desarrollado genera robustez y desempeño a la hora de identificar errores por procesos de cambios. De la misma forma en que reduce el tiempo para futuros cambios al detectar el impacto de las aplicaciones oportunamente sin trabajo adicional. * Contar con una suite robusta de pruebas unitarias y de aceptación completamente automatizadas generan un mejor desempeño y facilidad a la hora de detectar el alcance de los cambios.

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
7	2008	Cordeiro, Lucas; Mar, Carlos; Valentin, Eduardo; Cruz, Fabiano; Patrick, Daniel; Barreto, Raimundo & Lucena, Vicente	A Platform-Based Software Design Methodology for Embedded Control Systems: An Agile Toolkit	Caso de Estudio	XP	Investigación	Prueba Unitaria	* Las pruebas unitarias funcionan como herramienta de control para la implementación de futuros cambios, en este caso se plantea en un desarrollo de un sistema embebido
8	2008	Kakarontzas, George; Stamelos, Ioannis & Katsaros, Panagiotis	Product Line Variability with Elastic Components and Test-Driven Development	Caso de Estudio	TDD	Investigación	Ninguna	No hay información relevante sobre el tema de pruebas unitarias
9	2008	Mugridge, Rick	Managing Agile Project Requirements with Storytest-Driven Development	Caso de Estudio	TDD	Industria	Ninguna	No hay información relevante sobre el tema de pruebas unitarias
10	2008	Van Rompaey, Bart & Demeyer, Serge	Exploring the composition of unit test suites	Caso de Estudio	Otra	Investigación	Prueba Unitaria	* Modelo de propuesta para diseñar pruebas unitarias con el modelo de programación orientada a objetos a base de grafos.
11	2009	Harrold, Mary Jean	Reduce, reuse, recycle, recover: Techniques for improved	Caso de Estudio	Otra	Investigación	Prueba de Regresión	* Las pruebas de regresión son importantes, sin embargo su desarrollo es costoso, no obstante generan más efectividad y eficiencia

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
			regression testing					a la hora de implementarlos nuevos cambios.
12	2009	Li, Nan; Praphamontripong, Upsorn & Offutt, Jeff	An Experimental Comparison of Four Unit Test Criteria: Mutation, Edge-Pair, All-Uses and Prime Path Coverage	Caso de Estudio	Otra	Investigación	Ninguna	* Estudio Investigación sobre comparación de modelos para definir pruebas entre diferentes tipos.
13	2009	N., Gautham Reddy	Testing Processes in Business-Critical Chain Software Lifecycle	Caso de Estudio	TDD	Investigación	Pruebas Unitarias	* TDD propone que las pruebas unitarias deben estar terminadas antes de iniciar el proceso de desarrollo de software, para que cuando finalice el proceso de desarrollo debe aprobar correctamente las pruebas definidas con anterioridad. * El proceso de pruebas es de vital importancia durante el ciclo de vida del desarrollo de software. * Es importante definir el plan de iteración de pruebas y estimar el tiempo adecuadamente para el desarrollo de las pruebas requeridas.

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
14	2009	Rompaey, Bart Van; Demeyer, Serge & Van Rompaey B.; Demeyer, S	Establishing Traceability Links between Unit Test Cases and Units under Test	Caso de Estudio	Otra	Mix Investigación / Industria	Pruebas Unitarias	* En muchas ocasiones el desarrollador no conoce la importancia entre la relación entre pruebas unitarias y el código desarrollado. Existen herramientas que ofrecen los diferentes IDs para la implementación de pruebas unitarias pero por la falta de experiencia en muchos casos no se utilizan.
15	2009	Usha, K.; Poonguzhali, N. & Kavitha, E.	A quantitative approach for evaluating the effectiveness of refactoring in software development process	Caso de Estudio	XP	Investigación	Pruebas Unitarias	* Refactoring del código desarrollado durante todas las fases del ciclo del desarrollo de software.
16	2009	Xu, Bin	Towards High Quality Software Development with Extreme Programming Methodology: Practices from Real Software Projects	Caso de Estudio	XP, PP	Investigación	Pruebas Unitarias	* Propone que el proceso de testing a través del uso de pruebas unitarias deber ser desarrollado e implementado antes de iniciar el desarrollo del software de tal manera que cuanto este finalice solo se ejecuten las pruebas diseñadas y se verifique que se cumple con los requerimientos de los usuarios.

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
17	2010	Atkins, Colin; Barth, Florian & Brenner, Daniel	Software Testing Using Test Sheets	Experimento	Otra	Investigación	Pruebas Unitarias	* Se plantea una nueva mitología para la documentación y definición de las pruebas automatizadas a través del uso de hojas de cálculo.
18	2010	Causevic, Adnan; Sundmark, Daniel & Punnekkat, Sasikumar	An Industrial Survey on Contemporary Aspects of Software Testing	Experimento	TDD	Industria	Pruebas Unitarias	* Los casos de pruebas deben ser escritos previos al desarrollo del código. * La etapa de desarrollo solo debería iniciar cuando se finalice la etapa de diseño.
19	2011	Bernardo, Rafael Di; Sales Jr., Ricardo; Castor, Fernando; Coelho, Roberta; Cacho, Nelio & Soares, Sergio	Agile Testing of Exceptional Behavior	Caso de Estudio	Otra	Investigación	Pruebas Unitarias	* JUnit es un framework que soporta efectivament el desarrollo de pruebas unitarias y tiene una fácil adaptación con los desarrolladores
20	2011	Causevic, Adnan; Sundmark, Daniel & Punnekkat, Sasikumar	Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review	Experimento	TDD	Industria	Pruebas Unitarias	* Se describe el proceso general sobre cómo implementar TDD y sus impedimentos en la industria especificando poco sobre el tema de pruebas unitarias.

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
21	2011	Di Bernardo, Rafael; Castor, Fernando & Soares, S\'{e}rgio	Towards Agile Testing of Exceptional Behavior	Experimento	Otra	Investigación	Pruebas Unitarias	* La creación de flujos de excepciones facilita el proceso de validación mediante pruebas unitarias * El articulo propone un framework para el desarrollo agil de los flujos mencionados.
22	2011	Fojtik, Rostislav	Extreme Programming in development of specific software	Caso de Estudio	XP, TDD, PP	Mix Investigación / Industria	Pruebas Unitarias, Pruebas de Aceptación	* La importancia de las metodologías agiles recae no solo en la velocidad sino en la versatilidad que otorga al diseñador de software. * Se resalta la necesidad ce conformar equipos pequeños para la adecuada implementación de la metodología. * Brinda especial atención a los momentos de prueba y la cercanía que se debe tener con los usuarios. * Aunque no se conozcan los requerimientos completos se deben diseñar pruebas para los existentes.
23	2011	Jeon, Sanghoon; Han, Myungjin; Lee, Eunseok & Lee, Keun	Quality Attribute Driven Agile Development	Caso de Estudio	SCRUM, Otra	Industria	Pruebas Unitarias	* La utilización de backlogs que solo se basan en atributos funcionales pueden dificultar la aceptación del proyecto por el no cumplimiento de atributos de calidad. * Se realiza una propuesta de mejoramiento a SCRUM, para

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
								convertirlo en ACRUM. * Se prueba en Samsung electrónicos y las comparativas muestran mejores resultados en relación al número de defectos y horas invertidas.
24	2011	dos Santos, Andreia M; Karlsson, Borje F; Cavalcante, Andre M; Correia, Igor B & Silva, Emanuel	Testing in an agile product development environment: An industry experience report	Caso de Estudio	SCRUM, TDD	Industria	Pruebas Unitarias	* Articulo con contenido aplicado en NOKIA * Se plantea el trabajo en conjunto de 2 equipos para la generación de pruebas agiles en el contexto de Scrum y se incluye una metodología detallada de los momentos por los que pasa cada uno de los equipos. * Cada paso o interacción debe ser previamente validado en los casos de prueba * El enfoque de los equipos se basa en centrarse en la "gran imagen" y en las necesidades del cliente no en los detalles técnicos * Debe incluirse suficiente presupuesto para probar, esto incrementa el porcentaje de éxito de los proyectos agiles

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
25	2011	Scharff, Christelle	Guiding global software development projects using Scrum and Agile with quality assurance	Caso de Estudio	SCRUM	Investigación	Pruebas de aceptacion	* Se trata de un artículo Investigación, que muestra el desarrollo de un software por un grupo de estudiantes ubicados en diferentes países usando SCRUM. * Realizan una comparativa entre las herramientas usadas para el desarrollo de un proyecto de plataforma móvil. * Se realizan las siguientes recomendaciones: 1) realizar programación en pares, 2) los requerimientos deben ser capturados en el alto nivel, 3) involucramiento del cliente, 4) proceso de pruebas integrado al desarrollo no posterior, 5) priorización a través de un Pareto del tiempo. * Centrados en el proceso de auditoría de 4 categorías: 1. auditoria del backlog, 2. auditoria del diseño, 3. auditoria del proceso, 4. auditoria del código.

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
26	2011	Weiss, Johannes; Mandl, Peter & Schill, Alexander	Functional testing of Complex Event Processing applications	Mix Experimento / Caso de Estudio	TDD, CI	Investigación	Pruebas de aceptación, pruebas funcionales	* El artículo se basa en procesamiento de eventos complejos y como la utilización de TDD favorece la calidad. * Destacan la importancia de la automatización de las pruebas de aceptación a través de frameworks como Junit * Las pruebas funcionales resultan vitales para la aceptación pero deben estar enmarcadas en el proyecto no de manera posterior * No existe relación directa entre el articulo y la investigación a pesar de que realizan pruebas unitarias no se encuentra centrado en las mismas
27	2012	Bass, Julian M	Influences on Agile Practice Tailoring in Enterprise Software Development	Caso de Estudio	SCRUM, XP	Industria	Ninguna	* Habla de los fundamentos de la agilidad en términos de creación de software y como incrementa la calidad al generar equipos autosuficientes y motivados * A pesar de basarse en metodologías agiles existentes, toma en cuenta la posibilidad de realizar combinaciones para ajustarse a las políticas de las organizaciones. * Muestra cifras acerca de la posibilidad de implantación de

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
								prácticas agiles en grandes organizaciones.
28	2012	Bavani, Raja	Distributed Agile, Agile Testing, and Technical Debt	Entrevista	XP	Mix Investigación / Industria	Pruebas Unitarias, Pruebas de Aceptación	* La utilización de metodologías agiles ha demostrado resultados en equipos geográficamente separados. * Diferentes técnicas de pruebas se han desarrollado TDD, TBD entre otras. * La relación entre agilidad y pruebas es directa y definitiva. * La inclusión de expertos en pruebas en cada equipo resulta ser una técnica eficiente. * Se requiere poder automatizar las pruebas *Las iteraciones traen consigo debates sobre el que hacer, arquitectura, unit test y otras cosas. Es necesario entender el momento y poder priorizar lo mínimo requerido. * ¿Existen diferentes puntos de vista frente a la conformación de los equipos, separado pruebas o no?
29	2012	Bhasin, Sonali	Quality Assurance in Agile: A Study towards	Caso de Estudio	XP	Industria	Pruebas de aceptacion, pruebas funcionales	* Desplazamiento de métodos pesados de pruebas por entregables a pruebas del día * Comunicación y evaluación por pares como alternativa a largos

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
			Achieving Excellence					procesos de creación de pruebas. * ¿El reto de las organizaciones se encuentra en la flexibilización de QA, para que la necesitamos? * Se reporta el uso del modelo en espiral dentro de las metodologías agiles como una forma de garantizar la calidad
30	2012	Collins, Eliane Figueiredo & de Lucena, Vicente Ferreira	Software Test Automation practices in agile development environment: An industry experience report	Caso de Estudio	SCRUM	Industria	Pruebas Unitarias, Pruebas de Aceptación	* Descripción de los ítems clave para la formulación y prueba de software agile * Para la correcta construcción de un equipo agil es fundamental que este aprenda de los proyectos pasados * La automatización es fundamental y puede ser simple. * Revisal Marco de probes en el articular

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
31	2012	Gopularam, Bhanu Prakash & Yogeesha, C. B.	Mechanism for on demand Tag-Based software testing in virtualized environments	Mix Experimento / Caso de Estudio	SCRUM	Investigación	Pruebas Unitarias, Pruebas de Aceptación	* La utilización de frameworks de apoyo brindan alternativas a la creación de pruebas adicionales y existen opciones libres * Al igual que Junit existen herramientas como Jenkings. * El proceso de Scrum es altamente compatible con el modelo de pruebas continuas y automáticas. * El uso de ambientes virtual izados permite un mejor desempeño de las pruebas al ubicarlas en contextos acoplados.
32	2012	Hametner, Reinhard; Winkler, Dietmar & Zoitl, Alois	Agile testing concepts based on keyword-driven testing for industrial automation systems	Mix Experimento / Caso de Estudio	Otra, SCRUM	Industria	ninguna	* Automatización de pruebas de sistemas industriales asociados al hardware

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
33	2012	Shelton, William; Li, Nan; Ammann, Paul & Offutt, Jeff	Adding Criteria-Based Tests to Test Driven Development	Mix Experimento / Caso de Estudio	SCRUM	Investigación	Pruebas Unitarias, Pruebas de Aceptación	* TDD es una poderosa forma de trabajo, sin embargo la construcción de las pruebas puede resultar en consumo de tiempo. * El uso de mutaciones en las pruebas resulta ser una herramienta de aseguramiento de la calidad * La construcción automática de las pruebas unitarias y la mutación de las mismas con software como MuJava incrementa el rendimiento y habilidades del equipo. * Se deben construir criterios de prueba por iteración, no solo de la "gran foto"
34	2012	Tufail, Reham & Malik, Ali Afzal	A Case Study Analyzing the Impact of Software Process Adoption on Software Quality	Caso de Estudio	SCRUM	Industria	Pruebas Unitarias, Pruebas de Aceptación	* Durante el proceso de migración a metodologías agiles es posible que se incrementen los errores como se muestra en el estudio pero en el largo plazo existe alta posibilidad de una disminución de hasta el 50% en errores graves
35	2013	Hellmann, Theodore D.; Chokshi, Apoorve; Abad, Zahra Shakeri Hossein; Pratte, Sydney & Maurer, Frank	Agile Testing: A Systematic Mapping across Three Conferences: Understanding	Estudio Sistematico	XP	Investigación	Pruebas Unitarias	* Articulo de alta importancia para la construcción del resumen técnico del estudio, centrado en XP pero fundamental para la firma de presentación. Describe claramente el proceso de identificación d las

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
			Agile Testing in the XP/Agile Universe, Agile, and XP Conferences					categorías en relación a las pruebas en el mundo Agil
36	2013	Jinzenji, Kumi; Hoshino, Takashi; Williams, Laurie; Takahashi, Kenji & Jinzenji K.; Hoshino, T.; Williams L.; Takahashi K	An experience report for software quality evaluation in highly iterative development methodology using traditional metrics	Caso de Estudio	SCRUM	Mix Investigación / Industria	Pruebas Unitarias, Pruebas de Aceptación	* No existe métricas claras para la medición de la calidad en procesos de desarrollo altamente iterativos, lo que resulta en la no adopción por parte de las organizaciones. * Articulo importante acerca de las métricas de adopción de la agilidad * Las métricas de las pruebas unitarias se deben basar en el número de componentes creados
37	2013	Qusef, Abdallah	Test-to-code traceability: Why and how?	Caso de Estudio	SCRUM	Investigación	Pruebas Unitarias, Pruebas de Aceptación	* La trazabilidad no es una tarea estándar en la creación de software basada en metodologías agiles * El uso de refactorización y las pruebas unitarias es fundamental en términos de calidad para las metodologías agiles. * La refactorización golpea a las pruebas unitarias en diferentes circunstancias debe manejarse con cuidado y de forma planificada. * La sincronización del equipo debe ir desde la reunión de srum hasta la

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
								determinación de los nombres y estándares
38	2013	Wood, Stephen; Michaelides, George & Thomson, Chris	Successful extreme programming: Fidelity to the methodology or good teamworking?	Caso de Estudio	XP	Investigación	Ninguna	* Debe existir flexibilidad en la aplicación de las metodologías. * La adopción gradual cuida el ambiente de comunicación entre los equipos de desarrollo * Las medidas tomadas alrededor de la agilidad no deben ir en contra de la comunicación pues esta es fundamental para el éxito de los proyectos. * El uso de pares en la construcción de los componentes favorece la calidad en ambientes agiles

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
39	2014	Agarwal, Anil; Garg, N K & Jain, Avirag	Quality assurance for Product development using Agile	Caso de Estudio	AGILE	Industria	Ninguna	* Los equipos de desarrollo y pruebas deben moverse en paralelo para asegurar la calidad * Debe favorecerse la construcción de equipos con múltiples roles e interdisciplinarios * Construir un cambien de retroalimentación constante * Equipos en constante entrenamiento. * Alejarse los objetivos ambiguos esperar un poco a que estos se aclaren. * Genera criterios de pruebas, niveles de entrada y salida.
40	2014	Garijo, ï¿½?lvaro CarreraCarlos A IglesiasMercedes	Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development	Caso de Estudio	AGILE, SCRUM, BDD, TDD	Investigación	Pruebas Unitarias, Pruebas de Aceptación	* Propuesta metodológica para la creación de un framework de trabajo alrededor de las pruebas de software agil usando como base Scrum modificado para el contexto de los agentes inteligentes. El framework es llamado BEAST y usa BDD

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
41	2014	Tarhan, Ayca & Yilmaz, Seda Gunes	Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process	Estudio Sistematico	AGILE	Mix Academico / Profesional	Pruebas Unitarias, Pruebas de Aceptación	* La cualificación y cuantificación de los avances obtenidos permite el mejoramiento de los grupos de producción de software * Los equipos híbridos de producción deben escoger la metodología basado en el conocimiento y necesidad especifica del cliente. * Agil presenta mejores resultado que secuencial en el contexto del estudio con ratios que van desde 4% hasta el 21%
42	2007	Simons, Anthony J H	JWalk: a tool for lazy, systematic testing of java classes by design introspection and user interaction	Experimento	Otra	Investigación	Pruebas unitarias	Jwalk es una herramienta para la creación de pruebas sistemáticas de software, es una alternativa aunque menos usada que el JUnit, promueve la creación de un mayor número de pruebas o casos, de forma automatizada
43	2008	Williams, Nachiappan NagappanE. Michael MaximilienThirumalesh BhatLaurie	Realizing quality improvement through test driven development: results and experiences of four industrial teams	Caso de Estudio	TDD	Industria	Pruebas Unitarias, Pruebas de Aceptación	TDD cuenta con gran posibilidad de adaptación a diferentes dominios, la utilización de métricas como el KLOC favorece el monitoreo y control sobre los equipos de desarrollo ágil. En etapas tempranas el tiempo de desarrollo puede incrementarse por la adopción de la metodología TDD, entre el 15 % y el 35 %

Número	Estudio			Tipo de Estudio	Práctica Ágil	Entrono de Investigación	Prueba de Calidad	Resultados
	Año	Autor	Titulo					
44	2009	Yehudai, Benny PasternakShmuel TyszberowiczAmiram	GenUTest: a unit test and mock aspect generation tool	Caso de Estudio	XP	Investigación	Pruebas Unitarias	* GenUTest es un prototipo de herramienta que genera pruebas unitarias basadas en una técnica propuesta para la definición de las mismas.

Anexo 6. Tabla de Resultados Pregunta 1

Estudio	¿Cómo se logran implementar pruebas unitarias en un entorno de desarrollo ágil?
Visualizing Testsuites to Aid in Software Understanding	XP y los métodos de desarrollo ágil definen las pruebas como un proceso de alta importancia en su ciclo de vida. Este proceso no se considera solo algo esencial para el mantenimiento del software y la correctitud del código, sino que también representan la documentación del funcionamiento. TDD por su parte implica la creación y mantenimiento de un ciclo de pruebas completo que garantiza la correctitud de los componentes
On the Sustained Use of a Test-Driven Development Practice at IBM	La implementación de TDD requiere de un equipo apasionado y centrado para las prácticas que plantea la metodología y las herramientas necesarias para lograrlo. TDD requiere un tiempo más alto para su implementación inicial, pero luego se convierte en algo más fácil de manejar. Se requiere de tiempo extra para realizar la refactorización de aquellas pruebas que puedan cambiar en el tiempo. TDD puede agregar mayor calidad al desarrollo del producto.
Experiences of Using Pair Programming in an Agile Project	La aplicación de PP en la industria requiere de un conocimiento previo que facilite la implementación de la metodología. Su aprendizaje es rápido y los desarrolladores lo adoptan fácilmente. El uso de PP se vuelve más sencillo a medida que los desarrolladores tienen experiencia de su uso en diferentes proyectos. Ayuda al espíritu del equipo debido al buen desempeño de los proyectos. Se asegura que la combinación de PP con prácticas como TDD ayudaría a evadir posibles errores que no se detecten inicialmente.
Pushing the Boundaries of Testing and Continuous Integration	Las metodologías ágiles y XP tienen conceptos en común con TDD y los ciclos continuos de desarrollo de software.
Exploring the composition of unit test suites	Dentro de los ciclos de desarrollo ágil el ciclo de pruebas contribuye considerablemente al código generado, en términos de verificación y documentación.

Estudio	¿Cómo se logran implementar pruebas unitarias en un entorno de desarrollo ágil?
Testing Processes in Business-Critical Chain Software Lifecycle	Se define el ciclo de pruebas como el de más importancia y se acuerda que el plan de pruebas debe finalizarse antes de empezar el proceso de desarrollo. Se trabajará con iteraciones con principios bases de TDD pero enfocándose más en la administración de las pruebas.
Establishing Traceability Links between Unit Test Cases and Units under Test	Las fases de desarrollo y pruebas son dos actividades con alta importancia durante los ciclos de desarrollo ágiles, sin embargo la mayor parte de las veces los desarrolladores no conocen la relación directa entre el código y los artefactos de pruebas.
Towards High Quality Software Development with Extreme Programming Methodology: Practices from Real Software Projects	XP trabaja totalmente con 12 prácticas de la cual la de mayor relevancia es la fase de pruebas. Se detectan algunas recomendaciones de implementar XP para proyectos que no han utilizado esta metodología, como lo son desarrollar las pruebas antes de codificar, programación por pares, pruebas automatizadas, diseños simples, trabajar junto al cliente y pequeñas iteraciones aseguran mejorar la calidad del producto.
Software Testing Using Test Sheets	El ciclo de pruebas es una de las actividades más importantes en el desarrollo de software especialmente en proyectos que se enfocan en el desarrollo ágil.
An Industrial Survey on Contemporary Aspects of Software Testing	TDD representa una de las prácticas más comunes usadas en el desarrollo ágil dentro de la industria.
Agile Testing of Exceptional Behavior	La experiencia demuestra que gran parte de los desarrollos modernos de aplicaciones trabajan bajo metodologías ágiles. Estas metodologías se basan en la automatización de pruebas y poca documentación, considerando a las pruebas automatizadas como parte de la documentación.
Towards Agile Testing of Exceptional Behavior	Otra Framework de control del flujo de excepciones como forma de evitar bugs

Estudio	¿Cómo se logran implementar pruebas unitarias en un entorno de desarrollo ágil?
Extreme Programming in development of specific software	Tras encontrarse con requerimientos poco claros y altamente cambiantes se toma la decisión de utilizar como base de desarrollo la técnica XP, Los proyectos que llevará mucho tiempo o que tienen dificultades para obtener retroalimentación (por ejemplo, desde el punto de vista tecnológico) no son adecuados para esta metodología. Las fases básicas de desarrollo son: 1. Planificación y Gestión 2. Diseñar 3. Codificación 4. Pruebas.
	Es fundamental para la metodología la implementación de pruebas unitarias automáticas para la verificación constante del proyecto, se debe estar dispuesto a reescribir código en muchas ocasiones si es necesario. Pueden crear (automáticas) las pruebas antes o después del desarrollo usando la herramienta de pruebas unitarias de MS Visual Studio. Aunque estas son automáticas implican escribir posteriormente alrededor del 30% más de código
	Debe entenderse que la metodología es una ayuda, pero puede ser modificada a gusto y seleccionada según el tipo de proyecto y los requerimientos del mismo.
	Es común la utilización de TDD junto con XP como metodología de desarrollo. Versatilidad al diseño Equipos pequeños 2 a 10 miembros
Testing in an agile product development environment: An industry experience report	A pesar de que existen múltiples metodologías como SCRUM, XP, Kanbam, Lean, Crystal y varias más, lo más destacado y común entre ellas es: Planificación continua, Involucrar al cliente en todas las fases del proyecto, iterativo y proceso gradual, Definición clara de los roles, Los costos de iteraciones y alcance bien definido, Disciplina en el flujo de trabajo.
	(Rocha et al).la garantía de calidad debe llevarse a cabo en todas las fases del proceso de desarrollo de modo que los defectos se eliminan lo más cerca posible a la fase en la que se introducen, para evitar el aumento de los costes si se deja durante los pasos posteriores.

Estudio	¿Cómo se logran implementar pruebas unitarias en un entorno de desarrollo ágil?
	En una primera aproximación los equipos definen las pruebas después de la reunión diaria, pero el equipo de generadores de pruebas era demasiado pequeño (2personas) y no tuvo tiempo para final el sprint con todas las pruebas Bajo la metodología ágil, el equipo de pruebas n es un equipo independiente, la responsabilidad de la calidad no depende de "calidad" depende del equipo. Es común la aparición de micro cascadas, en donde errores de estimación o dificultades de implementación llevan al límite los tiempos de prueba. Esto no solo afecta la prueba sino la carrera completa. Un segundo enfoque consistía en dividir el sprint en desarrollo y pruebas posteriores, esto brindaba una ilusión de progreso pero podría ocultar problemas estructurales graves que serían descubiertos por fuera de fase y afectarían al sprint al tener que ser solucionados inmediatamente. Nuevas características podían haber sido construidas sobre esa base defectuosa. Finalmente se realizó un enfoque más simple sin subdivisiones en el equipo, todos hicieron pruebas. bases fundamentales: Mire el panorama, Colaborar con el Cliente, Construir la Fundación para las prácticas básicas ágiles, Proporcionar y obtener, retroalimentación, Automatizar pruebas de regresión, Adoptar un sistema de pruebas Ágil mentalidad, Utilizar el enfoque de equipo entero. Programadores escriben pruebas de unidad y las de integración de manera conjunta. "apoyar al equipo" Las pruebas deben incluirse en las tareas de estimación.
	Se requiere de equipos auto organizados, con diversos perfiles y altas capacidades para lograr un desempeño ágil.

Estudio	¿Cómo se logran implementar pruebas unitarias en un entorno de desarrollo ágil?
Influences on Agile Practice Tailoring in Enterprise Software Development	No es necesario ajustarse a una metodología a la perfección, las variaciones y combinaciones son comunes, la utilización de SCRUM o XP en compañía de una estrat5egia basada en TDD y el acompañamiento por PP es recurrente cuando se ha trabajado un poco más en proyectos agiles. No es necesario que cada "iteración" entregue software, en ocasiones hay iteraciones necesarias para otra y no necesarias en términos del negocio. La iteración favorece el avance y motivación del equipo. El enfoque es la clave.
	La combinación de metodologías posibilidad la inclusión de la agilidad en las grandes organizaciones. Los factores claves para la selección de las prácticas agiles son: duración de los ciclos, orientación a pruebas, actividades diarias, funcionalidad esperada, estándares de codificación, integración continua y la pertenencia del código.
	Las prácticas típicas de XP como TDD y PP no fueron tomadas al detalle (refactorización, metáfora, diseño simple) por la dificultad de adaptación a las organizaciones.
	El tamaño de los proyectos influye directamente en la posibilidad de hacer integración continua.
	En relación a los estándares se prefiere la legibilidad del código que la alta abstracción.
Quality Assurance in Agile: A Study towards Achieving Excellence	El reto fundamental en la aplicación de prácticas agiles en las organizaciones es lograr un balance entre la flexibilidad, la transparencia y la colaboración necesaria.
	Se requieren cambios fundamentales para la posibilidad de asegurar la calidad en ambientes agiles de desarrollo, el modelo de cascada que solo se centraba en probar al final de un plan, funcionaba porque había un plan, ahora no hay plan hay necesidades y un software que debe solucionarlas.
	Entre más tiempo dure en el proceso un error más costoso será arreglarlo debe tratarse de solucionar lo más rápido posible.
	Mantener al equipo motivado por los triunfos tempranos es fundamental para garantizar la alta productividad y los bajos costos de la agilidad

Estudio	¿Cómo se logran implementar pruebas unitarias en un entorno de desarrollo ágil?
Adding Criteria-Based Tests to Test Driven Development	La utilización de TDD se encuentra entre dicho por algunos autores, dada la intención propia de las pruebas y la alta probabilidad de que estas solo cubran el llamado "camino feliz" en donde todo funciona, este articulo propone la utilización de pruebas mutadas adicionales para verificar el cumplimiento del software.
	La mutación ha demostrado no afectar el ritmo de refactorización del TDD y ser más efectiva que otras técnicas.
	La utilización de la mutación aumenta la calidad y la cobertura de la pruebas
A Case Study Analyzing the Impact of Software Process Adoption on Software Quality	La adopción de SRUM disminuye el porcentaje de severidad alta (esto es sensacional y críticos) errores. En otras palabras, se reduce el impacto de un fallo de software. En segundo lugar, al pasar de la etapa de un proceso formal para la etapa ágil aumentar el ratio P / F. Además de lo que implica una mejor calidad, sino que también implica una reducción en la cantidad de esfuerzo invertido en las pruebas del software. Esto se debe al hecho de que tienen que ser fijo y, por lo tanto, menos recursos que tenga que ser dedicado a las pruebas de regresión menos errores.(passed-to- failed ratio (P/F)) Desde un punto de vista empresarial, la adopción de las metodologías agiles debe realizar secuencialmente, dejar que los equipos se sientan cómodos y trasladar practicas poco a poco, esto disminuye los errores que pueden ser introducidos al proceso como parte del proceso nuevo de desarrollo
Agile Testing: A Systematic Mapping across Three Conferences: Understanding Agile Testing in the XP/Agile Universe, Agile, and XP Conferences	La mayoría de los practicantes de ambientes agiles usan TDD para su entorno de pruebas, las automatizadas y las de aceptación son las 2 siguientes categorías y las unitarias las que continúan en la lista. La adopción de pruebas agiles es el tema más escrito en las conferencias y la calidad la mayor fuente de preocupación.
An experience report for software quality evaluation in highly iterative development	No existen métricas claras frente a la vinculación de pruebas en ambientes agiles, el artículo propone una distribución de las mismas que pueden ser usados en la adopción de los mismos.

Estudio	¿Cómo se logran implementar pruebas unitarias en un entorno de desarrollo ágil?
methodology using traditional metrics	
Test-to-code traceability: Why and how?	La posibilidad de realizar seguimiento a los proyectos agiles ha sido poco explorada pero en la medida en que la adopción crece, y los proyectos son más grandes es fundamental que se inicie el proceso, elementos como la refactorización lo dificulta.
Successful extreme programming: Fidelity to the methodology or good teamworking?	Debe existir flexibilidad en la aplicación de las metodologías ya que la adopción gradual cuida el ambiente de comunicación entre los equipos de desarrollo, es importante que las medidas tomadas alrededor de la agilidad no vayan en contra de la comunicación pues esta es fundamental para el éxito de los proyectos. El uso de la práctica de pares en la construcción de los componentes favorece la calidad en ambientes agiles
Quality assurance for Product development using Agile	Involucrar QA desde el principio es fundamental cuando se comprende que Las pruebas son componentes clave del desarrollo ágil, se debe fomentar la realización de múltiples funciones dentro de los equipos y permitir que desarrollo y prueba deben moverse en paralelo.
Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process	* La cualificación y cuantificación de los avances obtenidos permite el mejoramiento de los grupos de producción de software
	* Los equipos híbridos de producción deben escoger la metodología basado en el conocimiento y necesidad especifica del cliente.
	* Ágil presenta mejores resultado que secuencial en el contexto del estudio con ratios que van desde 4% hasta el 21%

Anexo 7. Tabla de Resultados Pregunta 2

Estudio	¿Incluyen pruebas unitarias dentro de su proceso de Software?	¿Resultados esperados de la implementación de pruebas?
Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study	Se incluye el uso de pruebas unitarias de acuerdo a la metodología planteada que brinda TDD	Mejoran la calidad del producto desarrollado cuando el proceso es claro y su forma de implementarlo es entendida por cada uno de los miembros del grupo.
Visualizing Testsuites to Aid in Software Understanding	Las pruebas son parte esencial del ciclo de desarrollo del software, en el caso de metodologías ágiles las pruebas unitarias son usadas como documentación, por lo que pueden ser de gran ayuda para la compresión del código.	Se espera que generen una mayor calidad del producto y permita documentar su funcionamiento a través de las pruebas implementadas.
On the Sustained Use of a Test-Driven Development Practice at IBM	Se implementan las pruebas unitarias como fase del ciclo propuesto dentro de TDD	La productividad incrementa y disminuye el grado de complejidad de futuros cambios del software.
Experiences of Using Pair Programming in an Agile Project	El diseño y desarrollo de las pruebas unitarias se realiza a través de pares de desarrolladores, juntos piensan sobre soluciones antes de iniciar a escribir el código.	Contribuyen al factor de encontrar pocos defectos dentro del sistema.
Pushing the Boundaries of Testing and Continuous Integration	Construir un ciclo extenso de pruebas unitarias y de aceptación que genere robustez y desempeño en el software.	Contar con un suite completo de pruebas unitarias aseguran un sistema con robustez, buen desempeño y adaptable a cambios.

Estudio	¿Incluyen pruebas unitarias dentro de su proceso de Software?	¿Resultados esperados de la implementación de pruebas?
Exploring the composition of unit test suites	Las pruebas unitarias representan un valor agregado que permite al software evolucionar en el tiempo.	Adaptabilidad al cambio.
An Experimental Comparison of Four Unit Test Criteria: Mutation, Edge-Pair, All-Uses and Prime Path Coverage	Se utilizan las pruebas unitarias como parte del ciclo de TDD	Generan eficiencia, desempeño y adaptabilidad al producto final.
Testing Processes in Business-Critical Chain Software Lifecycle	De acuerdo a lo propuesta de TDD las pruebas unitarias deben ser diseñadas e implementadas antes de iniciar el ciclo de desarrollo	Al inicio puede que el esfuerzo se vea reflejado como tiempo adicional por parte del equipo, pero se espera que este sea menor que corregir posibles errores que surjan si no se detectan a tiempo.
Establishing Traceability Links between Unit Test Cases and Units under Test	Se considera una fase dentro del ciclo de desarrollo de software ágil.	Genera mayor calidad y fácil adaptabilidad al cambio.
A quantitative approach for evaluating the effectiveness of refactoring in software development process	Se plantea como una de las fases durante el proceso de refactorización	Permiten identificar falencias que no se puedan percibir en un proceso simple de pruebas.
Towards High Quality Software Development with Extreme Programming Methodology: Practices from Real Software Projects	Fase del ciclo de XP	Ayudan a verificar si lo desarrollado es lo esperado con el cliente. Genera una alta calidad al software desarrollado. Alta productividad y efectividad.
Software Testing Using Test Sheets	El ciclo de pruebas es una de las actividades más importantes en el desarrollo de software especialmente en proyectos que se enfocan en el desarrollo ágil.	Generan un alto grado de calidad y fácil adaptación al cambio.

Estudio	¿Incluyen pruebas unitarias dentro de su proceso de Software?	¿Resultados esperados de la implementación de pruebas?
An Industrial Survey on Contemporary Aspects of Software Testing	Los casos de pruebas deben ser escritos previos al desarrollo del código y se consideran una fase del ciclo de TDD.	Al emplear pruebas unitarias asegura un alto grado de calidad en el proceso de desarrollo y despeño del producto.
Agile Testing of Exceptional Behavior	El proceso de pruebas forma parte del ciclo de desarrollo de software	Genera mayor calidad, a medida que las pruebas cubran más casos de error el grado de calidad será mayor.
Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review	TDD forma una parte esencial de XP, en esta se define el ciclo de pruebas como una de las fases más importantes del desarrollo de software.	Brinda un mayor grado de calidad en los alcances y una adaptabilidad mayor a futuros cambios.
Towards Agile Testing of Exceptional Behavior	Incluyen pruebas unitarias enriquecidas con el manejo de excepciones en el software.	Esperan lograr el aprovechamiento de la herramienta JUnit y el mejoramiento de la calidad de software a través del uso de los flujos de excepción controlados
Extreme Programming in development of specific software	Incluyen pruebas unitarias asociadas a XP y TDD. Realizan integración continua	Esperan garantizar el funcionamiento del software tras las modificaciones constantes.
Quality Attribute Driven Agile Development	Incluyen pruebas unitarias a través de una modificación de SCRUM Propuesta llamada ACRUM, esta busca estar orientada a la satisfacción de atributos de calidad del software en combinación con los funcionales	Busca promover de calidad no funcional al software creado a través de la metodología.

Estudio	¿Incluyen pruebas unitarias dentro de su proceso de Software?	¿Resultados esperados de la implementación de pruebas?
Testing in an agile product development environment: An industry experience report	Incluyen pruebas unitarias en el contexto de TDD y SCRUM	El objetivo es la rápida eliminación de no conformidades evitando que se propaguen errores a posteriores etapas donde pueda resultar más costoso.
Guiding global software development projects using Scrum and Agile with quality assurance	Incluyen pruebas estilo TDD, temprano en el proceso de creación.	Poder concluir exitosamente los Sprints en el tiempo planeado, no eliminar el tiempo de pruebas y tomar correctivos antes de construir sobre los errores.
Distributed Agile, Agile Testing, and Technical Debt	Si incluyen pruebas de estilo TDD	Se busca promover en el equipo la mentalidad de probar.
Quality Assurance in Agile: A Study towards Achieving Excellence	Incluyen en la metodología ATDD como parte de la retroalimentación del cliente	El objetivo es construir pruebas más eficientes capaces de generar no solo conformidad sino aceptación final, proponen algunas hipótesis de los factores influyentes en Calidad de Software
Software Test Automation practices in agile development environment: An industry experience report	Incluyen pruebas unitarias automáticas, autogeneradas y auto desplegadas bajo la metodología SCRUM	El objetivo es reportar eficientemente el número de defectos, simplificar el proceso manual d pruebas y generar métricas de la iteración de forma constante.
Mechanism for on demand Tag-Based software testing in virtualized environments	Implementan pruebas unitarias automatizadas a través de etiquetas predeterminadas	Facilitar el proceso de generación y despliegue de pruebas automáticas.

Estudio	¿Incluyen pruebas unitarias dentro de su proceso de Software?	¿Resultados esperados de la implementación de pruebas?
Adding Criteria-Based Tests to Test Driven Development	Si incluyen pruebas de estilo TDD modificado con mutadores de pruebas	Aumentar el porcentaje de cobertura de las pruebas y disminuir el riesgo de defectos no descubiertos. Favorecer el proceso de automatización.
Agile Testing: A Systematic Mapping across Three Conferences: Understanding Agile Testing in the XP/Agile Universe, Agile, and XP Conferences	Usan pruebas unitarias en combinación con TDD	No aplica
An experience report for software quality evaluation in highly iterative development methodology using traditional metrics	Utilización pruebas unitarias en ambientes altamente iterativos como estrategia para el aseguramiento de la calidad	El objetivo es la rápida detección de errores en cada iteración y la verificación rápida de requisitos funcionales.
Test-to-code traceability: Why and how?	Usan pruebas unitarias en combinación con TDD y un alto índice de refactorización	Las usan como la base del seguimiento y generación de rastros sobre los errores o los componentes.
Successful extreme programming: Fidelity to the methodology or good teamworking?	Utilizan pruebas unitarias en ambientes XP	generar criterios de aceptación de componentes
Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development	Utilización pruebas unitarias	Verificar la estructura de componentes y agentes del sistema
Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process	Utilizan pruebas unitarias en ambientes Agiles	Realizar verificación de componentes antes de la fase de integración del día.

Anexo 8. Tabla de Resultados Pregunta 3

Estudio	¿Cuáles fueron las herramientas usadas para la implementación de pruebas unitarias?
Visualizing Testsuites to Aid in Software Understanding	JUnit ofrece la posibilidad de crear pruebas unitarias y ayuda a generar la compresión del programa. Incluye escenarios que permite evaluar la efectividad del programa.
Pushing the Boundaries of Testing and Continuous Integration	Se implementa JUnit como herramienta para el desarrollo de las pruebas unitarias en Java.
Exploring the composition of unit test suites	Se utiliza JUnit, para generar los casos de pruebas en donde cada caso de prueba es una clase especifica.
An Experimental Comparison of Four Unit Test Criteria: Mutation, Edge-Pair, All-Uses and Prime Path Coverage	Se define pruebas unitarias a través de un modelo planteado basado en el uso de grafos y mutación de pruebas.
Establishing Traceability Links between Unit Test Cases and Units under Test	Se implementa el uso de pruebas unitarias a través de JUnit con el ID de Eclipse
Software Testing Using Test Sheets	Se basan en el uso de JUnit como herramienta para la generación pruebas unitarias utilizando hojas de texto como medio para la documentación de las pruebas
An Industrial Survey on Contemporary Aspects of Software Testing	Se utiliza el plugin de JUnit como herramienta generadora de los casos de pruebas en las clases definidas antes de iniciar el desarrollo del código.

Estudio	¿Cuáles fueron las herramientas usadas para la implementación de pruebas unitarias?
Agile Testing of Exceptional Behavior	Se desarrolla una extensión del framework de Junit, a partir del cual esperan controlar y diseñar pruebas para aquellas excepciones que inicialmente son detectadas con el objetivo de abarcar más casos de pruebas y detectar errores no controlados.
Towards Agile Testing of Exceptional Behavior	Junit framework que facilita la creación y despliegue de pruebas unitarias con un plugin desarrollado para el manejo de excepciones
Extreme Programming in development of specific software	Visual Studio cuenta con una herramienta de generación de pruebas unitarias
Distributed Agile, Agile Testing, and Technical Debt	Junit como herramienta base de desarrollo de pruebas unitarias
Software Test Automation practices in agile development environment: An industry experience report	Xunit frameworks, Cobertura que brinda criterios por línea de código en la etapa de pruebas. Y Selenium para las pruebas basadas en ambientes WEB
Adding Criteria-Based Tests to Test Driven Development	Mujava framework para la automatización de pruebas con mutación en JAVA, Junit como generador inicial de pruebas y Cobertura para medir la cantidad de pruebas realizadas sobre cada parte del código
Agile Testing: A Systematic Mapping across Three Conferences: Understanding Agile Testing in the XP/Agile Universe, Agile, and XP Conferences	Selenium para la realización de pruebas web, Junit para la generación y aplicación de pruebas unitarias, FitNesse para la creación de pruebas de aceptación agiles.
Test-to-code traceability: Why and how?	Junit como herramienta base de desarrollo de pruebas unitarias y el sistema de Log de java para la trazabilidad