



**UNIVERSIDAD  
DE ANTIOQUIA**

**TÍTULO**

**REPRESENTACIÓN DE LA PRÁCTICA DE DESARROLLO  
DIRIGIDO POR PRUEBAS DE ACEPTACIÓN (ATDD) USANDO EL  
NÚCLEO DE SEMAT**

**Autor:**

**Ingrid Jakeline García Mesa**

**Universidad de Antioquia**

**Facultad de Ingeniería, Departamento de ingeniería  
de sistemas**



REPRESENTACIÓN DE LA PRÁCTICA DE DESARROLLO DIRIGIDO POR  
PRUEBAS DE ACEPTACIÓN (ATDD) USANDO EL NÚCLEO DE SEMAT

Ingrid Jakeline Garcia Mesa

Trabajo de grado  
como requisito para optar al título de: Ingeniero(a) de Sistemas

Asesor:

Juan Ricardo Cogollo  
Ingeniero de Sistemas  
Magister en Ingeniería Administrativa  
Candidato a Doctor en Ingeniería, Sistemas e Informática

Universidad de Antioquia  
Facultad de Ingeniería  
Departamento de Ingeniería de Sistemas  
Medellín, Colombia  
2019

## TABLA DE CONTENIDOS

1. INTRODUCCIÓN.....	10
2. OBJETIVOS .....	14
2.1 OBJETIVO GENERAL.....	145
2.2 OBJETIVOS ESPECÍFICOS: .....	145
3. MARCO TEÓRICO .....	17
4. DESCRIPCIÓN DE SEMAT ( <i>Software Engineering Method and Theory</i> ) .....	21
4.1 ELEMENTOS DE SEMAT .....	21
4.2 ORGANIZACIÓN DEL NÚCLEO DE SEMAT .....	24
5. ANTECEDENTES.....	27
6. DESCRIPCIÓN TEÓRICA Y VALIDACIÓN DE ATDD.....	32
6.1 DESARROLLO DIRIGIDO POR PRUEBAS DE ACEPTACIÓN (ATDD).....	32
6.2 DESCRIPCIÓN DE DESARROLLO DIRIGIDO POR PRUEBAS DE ACEPTACIÓN (ATDD) DESDE LA REVISIÓN BIBLIOGRÁFICA.....	35
6.2.1 Seleccionar y documentar historia de usuario .....	35
6.2.2 Escribir escenarios de prueba basados en criterios de aceptación .....	36
6.2.3 Implementar prueba de aceptación .....	37
6.2.4. Ejecución de prueba de aceptación .....	38
6.2.5 Refactorización .....	40

6.2.6 Superar prueba de aceptación .....	43
6.2.7 Cerrar historia de usuario.....	44
6.3 DESCRIPCIÓN ENTREVISTAS.....	45
6.3.1 Objetivo .....	45
6.3.2 Población objetivo.....	45
6.3.3 Metodología .....	46
6.3.4 Diseño de entrevistas .....	48
6.4.5 Estructura entrevista .....	48
7. RESULTADOS Y ANÁLISIS .....	54
7.1 REPRESENTACIÓN DE DESARROLLO DIRIGIDO POR PRUEBAS DE ACEPTACIÓN EN SEMAT (TEORÍA) .....	54
7.2 RESULTADOS ENTREVISTA.....	75
7.3 ANÁLISIS DE RESULTADOS .....	84
7.4 REPRESENTACIÓN DE DESARROLLO DIRIGIDO POR PRUEBAS DE ACEPTACIÓN EN SEMAT (PRÁCTICA).....	87
8. CONCLUSIONES .....	88
9. REFERENCIAS BIBLIOGRÁFICAS.....	90

## INDICE IMAGENES

Figura 1. Representación de alfas en núcleo de SEMAT.....	22
Figura 2. Representación de los espacios de actividad en el núcleo de SEMAT.....	23
Figura 3. Representación de competencias en SEMAT .....	23
Figura 4. Áreas de organización núcleo SEMAT.....	24
Figura 5. Representaciones graficas SEMAT .....	26
Figura 6. TDD asociado a desarrollo dirigido por pruebas (ATDD).....	43
Figura 7. Representación teórica desarrollo dirigido por pruebas en Semat: Seleccionar y documentar historia de usuario.....	57
Figura 8. Representación teórica desarrollo dirigido por pruebas en Semat: Escribir escenarios de prueba basados en criterios de aceptación.....	59
Figura 9. Representación teórica desarrollo dirigido por pruebas en Semat: Implementar pruebas de aceptación.....	62
Figura 10. Representación teórica desarrollo dirigido por pruebas en Semat: ejecución prueba de aceptación .....	66
Figura 11. Representación teórica desarrollo dirigido por pruebas en Semat: Refactorización.....	68
Figura 12. Representación teórica desarrollo dirigido por pruebas en Semat: Superar prueba de aceptación.....	71
Figura 13: Representación teórica desarrollo dirigido por pruebas en Semat: Cierre historia de usuario.....	74

## INDICE TABLAS

Tabla 1. Tabla representación de elementos de SEMAT .....	26
Tabla 2: Espacios de actividad, actividades y competencias en: Seleccionar y documentar historia de usuario .....	55
Tabla 3: Espacios de actividad, alfas, productos de trabajo y roles en: Seleccionar y documentar historia de usuario .....	55
Tabla 4: Espacios de actividad, actividades y competencias en: Escribir escenarios de prueba basados en criterios de aceptación.....	58
Tabla 5: Espacios de actividad, alfas, productos de trabajo y roles en: Escribir escenarios de prueba basados en criterios de aceptación.....	58
Tabla 6: Espacios de actividad, actividades y competencias en: Implementar prueba de aceptación. ....	60
Tabla 7: Espacios de actividad, alfas, productos de trabajo y roles en: Implementar prueba de aceptación. ....	61
Tabla 8: Espacios de actividad, actividades y competencias en: Ejecución prueba de aceptación. ....	63
Tabla 9: Espacios de actividad, alfas, productos de trabajo y roles en: Ejecución prueba de aceptación. ....	64
Tabla 10: Espacios de actividad, actividades y competencias en: Refactorización .....	67
Tabla 11: Espacios de actividad, alfas, productos de trabajo y roles en: Refactorización .....	67
Tabla 12: Espacios de actividad, actividades y competencias en: Superar prueba de aceptación. ....	69

Tabla 13: Espacios de actividad, alfas, productos de trabajo y roles en: Superar prueba de aceptación. ....	70
Tabla 14: Espacios de actividad, actividades y competencias en: Cierre historia de usuario ...	72
Tabla 15: Espacios de actividad, alfas, productos de trabajo y roles en: Cierre historia de usuario.....	73

## RESUMEN

Durante el proceso de desarrollo de software, cuando se sigue un enfoque tradicional, los equipos de aseguramiento de la calidad (En adelante QA, por sus siglas en inglés *Quality Assurance*) trabajan de forma independiente a los desarrolladores. Actualmente se presenta un cambio en esta dinámica, los equipos inmersos en el desarrollo de software se integran, trabajando juntos con una única responsabilidad: generar un producto que funcione y genere valor, donde el objetivo es prevenir errores en vez de detectarlos. Las herramientas que se usan bajo este enfoque se integran ayudando a mejorar la calidad del software a través de un proceso de desarrollo, pruebas, integración y despliegue automatizado, lo cual permite desarrollar software ágilmente, con mejor retorno a la inversión y de mejor calidad. Actualmente algunas empresas adoptan ATDD en su proceso de desarrollo, según experiencia propia o siguiendo parámetros señalados en la teoría. En este trabajo se busca identificar cómo realizan algunas empresas la adopción de ATDD y realizar una comparación en la teoría disponible al respecto, usando para ello representaciones mediante el núcleo de Semat (teoría y método de la ingeniería de software), el cual define un estándar llamado *Essence*, que permite modelar esfuerzos presentes en la ingeniería de software usando un lenguaje común de elementos que se definen para ello.

**Palabras claves:** Semat, ATDD, práctica, desarrollo de software, desarrollo dirigido por pruebas de aceptación, criterios de aceptación, Essence



## ABSTRACT

During the software development process, when a traditional approach is followed, the quality assurance teams (hereinafter QA, for its acronym in Quality Assurance) work independently to the developers. Currently there is a change in this dynamic, teams immersed in software development are integrated, working together with a single responsibility: to generate a product that works and generates value, where the objective is to prevent errors instead of detecting them. The tools used under this approach are integrated helping to improve the quality of the software through a process of development, testing, integration and automated deployment, which allows to develop software quickly, with better return to investment and better quality. Currently some companies adopt ATDD in their development process, according to their own experience or following parameters indicated in the theory. This paper seeks to identify how some companies adopt ATDD and make a comparison in the theory available in this regard, using representations through the core of Semat (software engineering theory and method), which defines a standard called Essence, which allows modeling efforts in software engineering using a common language of elements that are defined for it.

**Keywords:** Semat, ATDD, practice, software development, development driven by acceptance tests, acceptance criteria, Essence

## 1. INTRODUCCIÓN

El enfoque de Desarrollo Dirigido por Pruebas de Aceptación (En adelante ATDD, por sus siglas en inglés *Acceptance Test Driven Development*) permite especificar los criterios de aceptación de un producto de software que se va a construir, donde se considera su comportamiento y funcionamiento antes de iniciar el desarrollo, mientras se discute y define con ejemplos concretos de comportamiento con el cliente. Luego, los ejemplos significativos se refinan y se agregan a los *tests* del sistema para lograr entender qué es lo que hay que construir. Esto significa que, si al final del desarrollo el producto cumple con estos criterios de aceptación, entonces el cliente estará satisfecho con el resultado que se obtiene.

Actualmente algunas empresas adoptan ATDD en su proceso de desarrollo, según experiencia propia o mediante lo que se indica en la teoría. Con el objetivo de promover la adopción adecuada de ATDD, es importante identificar cómo realizan algunas empresas su adopción y realizar una comparación contra lo que señala la teoría. Éste trabajo busca identificar, mediante representaciones en el núcleo de Semat, cómo algunas empresas adoptan ATDD, con el objetivo de comparar dicha adopción contra lo descrito en la teoría.

La estrategia que se propone para lograr lo anterior consiste en usar técnicas de recolección de información, para indagar cómo se implementa ATDD por parte del personal participante del proceso de desarrollo de software en empresas de la ciudad de Medellín, así se define como instrumento de recolección de información una entrevista dirigida a desarrolladores de software que adopten ATDD, los resultados que se obtienen de dicha entrevista se usan para comparar la representación que se realiza mediante el núcleo de Semat a partir de la teoría consultada en la literatura. Dicha comparación

permite emitir conclusiones acerca de cómo se implementa el enfoque ATDD en la industria de software de la ciudad de Medellín. El nombre de los desarrolladores y, de las empresas en las cuales se desempeñan no se dará a conocer, guardando la confidencialidad, con la finalidad que la información obtenida no afecte su prestigio y buen nombre.

## PROBLEMA

Las pruebas de software constituyen una de las fases de mayor importancia durante el proceso de desarrollo de software. Sin embargo, cuando el software enfrenta procesos de mantenimiento debe ser probado nuevamente, lo cual demanda la adopción de pruebas de regresión, entre otras, que incrementan el costo de mantener el software y los tiempos de respuesta [1]. Para mitigar el impacto de lo anterior, existe la automatización de pruebas, además de nuevos enfoques de desarrollo dirigido por pruebas, tales como TDD, BDD y ATDD [2]. Para la adopción de dichos enfoques se presentan inconvenientes referentes a su divulgación, diferenciación y representación, ya que no se usa una manera idónea de representar dichos enfoques de forma que se facilite su entendimiento por parte de los interesados. Normalmente, las descripciones que se encuentran no son estructuradas y se basan en lenguaje natural. Pero en trabajos como [3] se propone la representación de la práctica TDD usando el núcleo de Semat. En [4] se representa la práctica BDD mediante el uso de un diagrama de clases de UML. Lamentablemente, no se evidencia en la literatura una propuesta para la representación del enfoque ATDD, de forma que ésta facilite su comprensión por parte de los interesados y así su posterior adopción de manera práctica.

Lo anterior, da origen a la formulación de la hipótesis de investigación que se define a continuación:

## HIPÓTESIS

- ✓ El enfoque de Desarrollo Dirigido por Pruebas de Aceptación (ATDD), carece de una base teórica ampliamente divulgada y de una forma de descripción y representación que facilite su comprensión por parte de los interesados.
  
- ✓ No existe conocimiento detallado del enfoque ATDD en la industria, lo cual dificulta su adopción y crea una brecha entre lo que se implementa y lo que se describe en la teoría

Para confirmar las anteriores hipótesis, se procede a formular las preguntas de investigación que se indican a continuación:

### PREGUNTAS DE INVESTIGACIÓN:

1. ¿Existe un vacío teórico respecto a la descripción y representación de la práctica ATDD, de forma que esto cause la no implementación o implementación errónea en la industria?
  
2. ¿Es el lenguaje de Semat una herramienta idónea para representar la práctica ATDD?

## METODOLOGIA

El presente trabajo consistió en realizar una revisión de literatura exploratoria acerca de ATDD para el logro del objetivo específico número 2 “Explorar a través de la literatura científica cómo debe realizarse ATDD, según lo indique la teoría consultada”. Para ello se consultaron fuentes como revistas indexadas, libros, artículos de conferencias. Para realizar búsqueda en dichos recursos bibliográficos se usaron términos descritos en el título y palabras claves, así como la combinación y sinónimos de ellos, para posteriormente filtrar sobre los resultados y seleccionar artículos que describieran parcial o completamente ATDD. A partir de dichos artículos, se hizo una revisión de las fuentes que estos citaban, con el objetivo de devolverse en el tiempo e identificar la cadena de artículos que trataban el tema previamente, hasta identificar autores predominantes en el tema. Esta revisión permitió describir a partir de su análisis y conclusiones, los pasos para implementar ATDD.

Para el logro del objetivo específico 1 “Identificar en un grupo de empresas de la ciudad de Medellín, cómo adoptan la práctica ATDD en su proceso de desarrollo de software”, se seleccionó un grupo de 10 empresas candidatas de software de la ciudad de Medellín. Mediante referidos por parte de académicos se logró contactar 7 personas con perfil de desarrollador de software con nivel de experiencia intermedia y senior, a los cuales luego de realizar entrevista general telefónica, se les pidió responder una serie de preguntas, luego de manifestar su aceptación e informar sobre garantía de confidencialidad. De los anteriores 7, uno no concluyó las respuestas, y el otro manifestó que en su empresa no definitivamente no se usaba la práctica ATDD ni una similar, por lo cual su opinión no era de nuestro interés. Así, se contó con 5 participantes, cada uno de empresa distinta. Las preguntas planteadas se describen en la sección diseño de entrevistas, las cuales se formularon a partir de la revisión bibliográfica.

Para el logro de los objetivos 3, 4 y 5, se realizó modelado usando los los elementos del núcleo de Semat, basado en el estándar Essence de OMG, el cual permitió modelar los pasos descritos en términos de Espacios de Actividades, Actividades, Competencias, Productos de Trabajo, Alfes y Responsables. Esto permitió establecer las comparaciones pertinentes entre lo descrito verbalmente y lo representado en Semat, además del contraste entre teoría y práctica.

Este trabajo se organiza de la siguiente manera: En la sección 2 se describen los objetivos general y específicos que se desean abordar. Posteriormente, se realiza en la sección 3 un marco teórico referente a los conceptos relevantes para el desarrollo del trabajo, seguido de la descripción detallada de Semat en la sección 4. Luego, en la sección 5, se presenta una revisión de antecedentes respecto de trabajos relacionados. En la sección 6, se realiza una descripción de ATDD acorde a la teoría consultada, donde se describen los pasos para implementar ATDD, además de la comparación y validación de estos mediante entrevistas a desarrolladores practicantes. En la sección 7 se discuten resultados y su respectivo análisis. Por último, las secciones 8 y 9 presentan las conclusiones y bibliografía respectivamente.

## 2. OBJETIVOS

### 2.1 OBJETIVO GENERAL

Analizar cómo algunas empresas de desarrollo de software de la ciudad de Medellín usan las prácticas ATDD, con la finalidad de identificar diferencias entre la implementación y la definición teórica

### 2.2 OBJETIVOS ESPECÍFICOS:

- Identificar en un grupo de empresas de la ciudad de Medellín, cómo adoptan la práctica ATDD en su proceso de desarrollo de software.
- Explorar a través de la literatura científica cómo debe realizarse ATDD, según lo indique la teoría consultada.
- Modelar en Semat las prácticas ATDD según lo descrito en la teoría y según la adopción que se realiza en las empresas que se seleccionen.
- Comparar las representaciones gráficas obtenidas al realizar la representación en Semat de la adopción de ATDD realizada en un grupo de empresas de la ciudad de Medellín vs la descripción de ATDD encontrada en la literatura consultada.

- Establecer resultados, hallazgos y conclusiones posterior al modelado y comparación de la práctica ATDD en un grupo de empresas de software de la ciudad de Medellín y en la teoría de la literatura científica.
- Elaborar un artículo de publicación apoyado en el grupo de investigación ITOS, de la universidad de Antioquia, que se entienda por ATDD y como se ejecutan estas prácticas en un grupo de empresas de la ciudad de Medellín.



### 3. MARCO TEÓRICO

A continuación, se describen conceptos relevantes al presente trabajo, concernientes al proceso de desarrollo y pruebas de software y Semat:

**Pruebas de software:** son el proceso de ejecución de un programa bajo condiciones controladas, evaluando y registrando los resultados con la intención de descubrir un error, el objetivo fundamental es diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores haciéndolo con la menor cantidad de tiempo y esfuerzo [5]. Es un proceso, para evaluar la funcionalidad de una aplicación de software con la intención de determinar si el software desarrollado cumple con los requisitos especificados o no e identificar los defectos para garantizar que el producto esté libre de defectos para producir un producto de calidad. Su objetivo es detectar las diferencias entre las condiciones existentes y requeridas (es decir, defectos) y evaluar las características del elemento de software.

**Ciclo de vida del software:** Es un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la operación y el mantenimiento de un producto software, abarcando la vida del sistema desde su definición hasta su retirada (ISO 12207). El ciclo de vida de un proyecto especifica el enfoque general del desarrollo, indicando los procesos, actividades y tareas que se van a realizar y en qué orden, y los productos que se van a generar, los que se van a entregar al cliente y en qué orden se van a entregar. En cascada consta de las siguientes fases: análisis, diseño, implementación, pruebas y mantenimiento [6].

**Mantenimiento de software:** Es la modificación de un producto de software después de la entrega, para corregir errores, mejorar el rendimiento u otros atributos. El mantenimiento del software es también una de las fases del ciclo de vida del desarrollo del sistema (SDLC), que se aplica al desarrollo de software. La fase de mantenimiento es la fase que sigue al despliegue (implementación) del software en el campo [7].

**Semat:** es una iniciativa propuesta por Ivar Jacobson, Bertrand Meyer y Richard Soley. Este marco de pensamiento muestra que, a partir del análisis de todos los métodos desarrollo, se pueden definir elementos comunes, es decir, se puede definir un núcleo con “las cosas con las que se trabaja” y “las cosas que siempre se hacen” en el desarrollo de software [8].

**Essence:** Es un estándar, el cual propone un núcleo para trabajar con métodos en ingeniería de software que permite identificar y describir los elementos esenciales de métodos y prácticas existentes o futuras para comparar, evaluar, elaborar, adaptar, usar, simular prácticas de la ingeniería de software para ayudar a los equipos de trabajo a medir el progreso y la salud de sus esfuerzos o proyectos de desarrollo facilitando la comunicación entre practicantes, investigadores, estudiantes mediante un vocabulario común [9].

**TDD:** Es una metodología que se enfoca específicamente en “pruebas unitarias”. El desarrollador toma un requisito y luego lo convierte en un caso de prueba específico. Luego, el desarrollador escribe el código para pasar esos casos de prueba en particular solamente [10]. Una diferencia fundamental de la programación tradicional es la que los desarrolladores escriben pruebas unitarias después de escribir el código [11].

**Pruebas automatizadas:** Es una unidad automatizada donde los marcos de prueba minimizan el esfuerzo de prueba, reduciendo una gran cantidad de pruebas a un clic de un botón. Por el contrario, durante la ejecución de la prueba manual, los desarrolladores y evaluadores deben realizar un esfuerzo proporcional al número de pruebas ejecutadas: implican escribir las pruebas unitarias como código y colocando este código en un marco de prueba como Junit [12].

**BDD:** (Behavior Driven Development). Es en gran medida una extensión de la metodología TDD. El desarrollador define un caso de prueba, prueba el código para verificar que el caso de prueba fallará. A continuación, el desarrollador escribe el código necesario para pasar el caso de prueba y luego prueba el código para garantizar el cumplimiento. Donde BDD difiere de TDD es cómo se especifica el caso de prueba. Los casos de pruebas de BDD existen de una manera que especifica el comportamiento deseado. El lenguaje claro de los casos de prueba de BDD simplifica la comprensión de todos los interesados en un proyecto de desarrollo. Se diferencia por estar escrito en un idioma compartido, lo que mejora la comunicación entre los equipos tecnológicos y no tecnológicos y las partes interesadas. Las pruebas se escriben antes que el código, se centran más en el usuario y se basan en el comportamiento del sistema [11].

**ATDD:** Es un ejercicio de colaboración entre consultores, diseñadores (para construir los criterios de aceptación), probadores y desarrolladores para escribir primero casos de prueba de aceptación (o funcional) para una solución o un producto y luego usar los casos de prueba de aceptación para continuar con el desarrollo [13].

**Núcleo:** Conjunto de elementos usados para formar una base que describe la ingeniería de software [9]. En Semat el núcleo se refiere a los elementos de tipo alfa que son propuestos como elementos comunes a todos los esfuerzos presentes en la ingeniería de software.

**Método:** Conjunto de prácticas que forma la descripción de los esfuerzos que se realizan en una empresa. Los esfuerzos se visualizan mediante instancias como alfas, productos de trabajo, actividades y similares [9].

**Práctica:** Es un esfuerzo que se puede repetir y se realiza con un propósito específico [9].

**Alfa:** Acrónimo de (Abstract Level Progress Health Attribute). Se caracteriza por un conjunto de estados que representan su progreso y salud. Cada estado tiene una lista de chequeo que especifica los criterios necesarios para alcanzar un estado en particular [9].  
representa un indicador crítico de todo lo que es más importante para monitorear y que sigue una línea de progreso.

**Estado:** Expresa una situación donde algunas condiciones se proponen [9].

**Espacios de actividad:** Son las “cosas que siempre hacemos” en ingeniería de software. Agrupan conjuntos de actividades que se realizan mientras se desarrolla un producto de software [9].

**Competencia:** Una característica del interesado o equipo que refleja la habilidad de hacer un trabajo. Se puede detallar en diferentes “niveles de competencia” [9].

**Productos de trabajo:** Artefactos que se utilizan o se generan en una práctica [9].

**Actividad:** Se define como una o más clases de puntos de trabajo y la guía sobre cómo realizarlas [9].

#### 4. DESCRIPCIÓN DE SEMAT (*Software Engineering Method and Theory*)

SEMAT se funda en septiembre de 2009 por Ivar Jacobson, Bertrand Meyer y Richard Soley, quienes sintieron que había llegado el momento de cambiar fundamentalmente la forma en que las personas trabajan con el desarrollo de software. Escribieron una declaración de llamado a la acción, que en pocas líneas identifica una serie de problemas críticos, explica por qué es necesario actuar y sugiere lo que hay que hacer. Apoya un proceso para redefinir la ingeniería de software basada en una teoría sólida, probada de principios y mejores prácticas que:

- Incluya un núcleo de elementos ampliamente acordados, extensible para usos específicos
- Abordar cuestiones tecnológicas y de personas.
- Están respaldados por la industria, la academia, los investigadores y los usuarios.
- Extensión de soporte frente a requisitos y tecnología cambiantes.

Se enfoca en encontrar un núcleo de elementos que acepta la comunidad de desarrollo de software. La propuesta pretende refundar la ingeniería de software mediante la definición de una teoría sólida, principios comprobados y mejores prácticas. Semat busca determinar un núcleo para describir acciones y elementos universales, usando un lenguaje y las prácticas de desarrollo de software, para poderlas aplicar, evaluar y medir. El núcleo contiene un pequeño número de “cosas con las que siempre se trabaja” y “cosas que siempre se hacen” cuando se desarrollan sistemas de software [14].

Semat suministra tres características únicas del núcleo: es accionable, es extensible y es práctico. El núcleo es accionable por la forma en que maneja las “cosas con las que siempre se trabaja”. Ellas se capturan como alfas en lugar de productos de trabajo. Un alfa es un elemento esencial del esfuerzo de ingeniería de software, siendo relevante para la evaluación del progreso y la salud. Semat identifica siete alfas: oportunidad, interesados, requisitos, sistema de software, trabajo, forma de trabajo y equipo.

Semat permite estandarizar la ingeniería de software incluyendo un conjunto de elementos comunes que sirven de referencia para relacionar y diferenciar métodos de desarrollo de software. Se organiza en tres áreas de interés que se centran en aspectos específicos de la ingeniería de software: clientes (verde), solución (amarillo) y esfuerzo (azul). Para lograrlo, utiliza alfas, espacios de actividad, actividades, prácticas, métodos, patrones y productos de trabajo.

#### 4.1 ELEMENTOS DE SEMAT

Los elementos que componen cada una de las áreas de conocimiento y que, a su vez, definen el núcleo son: alfas, espacios de actividad y competencias

**Alfas:** Representaciones de las cosas esenciales con las que trabajar. Los alfa proporcionan descripciones del tipo de cosas que un equipo administrará, producirá y usará en el proceso de desarrollo, mantenimiento y soporte de software y, como tales, son relevantes para evaluar el progreso y la salud de un esfuerzo de software. También actúan como el ancla para cualquier sub-alfa adicional y productos de trabajo requeridos por las prácticas de ingeniería de software.

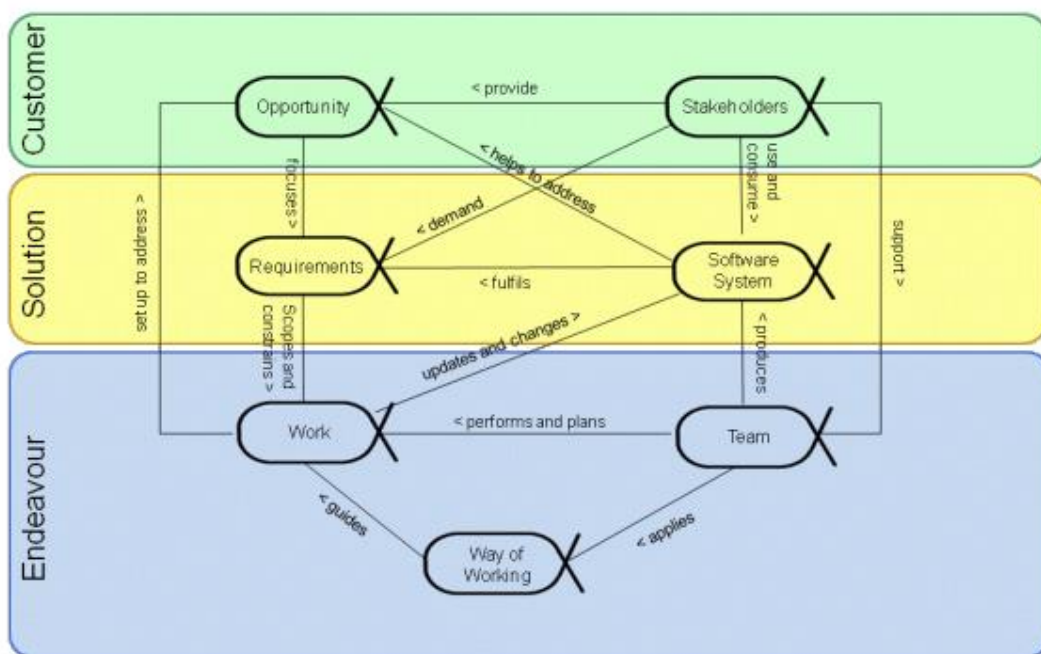


Figura 1. Representación de alfas en núcleo de SEMAT [9].

**Espacios de actividad:** Representaciones de las cosas esenciales que hacer. Los espacios de actividad proporcionan descripciones de desafíos a los que se enfrenta un equipo al desarrollar, mantener y respaldar sistemas de software y el tipo de cosas que el equipo hará para cumplirlos. complementan los alfas, ya que proporcionan el conjunto de actividades esenciales que normalmente se hacen en ingeniería de software. Definidos como “Cosas que siempre hacemos” [14] (Por ejemplo: Comprender las

necesidades del interesado, comprender los requisitos, implementar el sistema, probar el sistema, usar el sistema, rastrear el progreso, etc.)

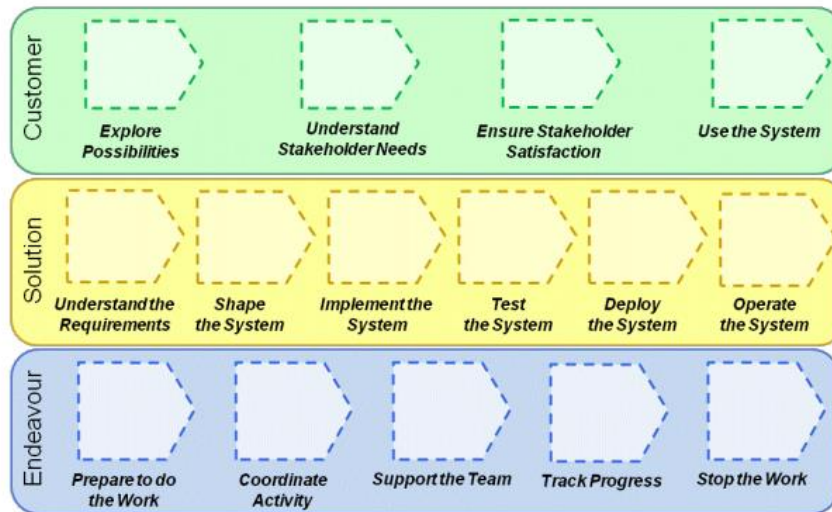


Figura 2. Representación de los espacios de actividad en el núcleo de SEMAT [9]

**Competencias:** representaciones de las capacidades claves necesarias para llevar a cabo el trabajo de ingeniería de software.



Figura 3. Representación de competencias en SEMAT [9]



## 4.2 ORGANIZACIÓN DEL NÚCLEO DE SEMAT

Está organizado en las siguientes tres áreas, cada se enfoca en un aspecto específico de la ingeniería de software:

- **Cliente:** Ésta área de preocupación contiene todo lo que tiene que ver con el uso real y la explotación del software.
- **Solución:** Ésta área de preocupación contiene todo para hacer la especificación y el desarrollo del software.
- **Esfuerzo:** Ésta área de preocupación contiene todo lo que tiene que ver con el equipo y la forma en que se acercan a su trabajo [9].

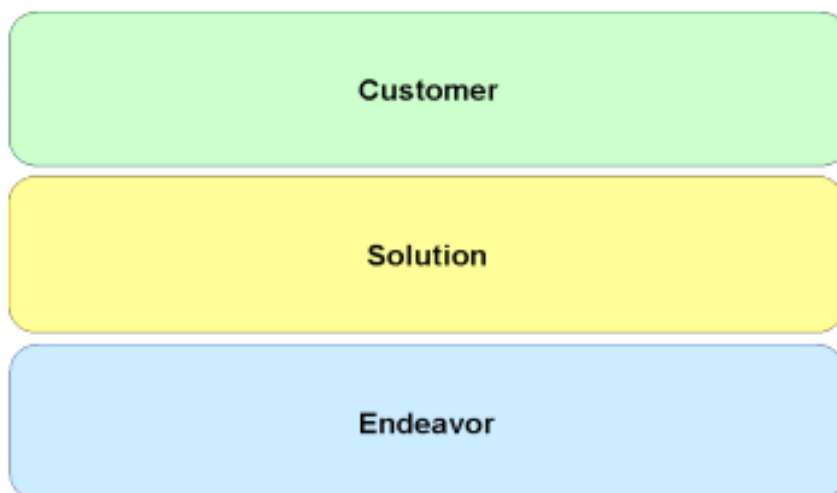



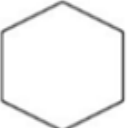
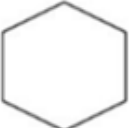


Figura 4. Áreas de organización núcleo SEMAT [9].

El núcleo de Semat emplea elementos tales como:

Elemento	Descripción
 <p data-bbox="483 640 537 667"><b>Alfa</b></p>	<p data-bbox="824 527 1385 716">Describe las cosas que el equipo debe administrar, producir y usar en el proceso de desarrollo, mantenimiento y soporte. Semat identifica siete alfas: oportunidad, interesado, requisitos, sistema de software, trabajo, equipo y forma de trabajo</p>
 <p data-bbox="430 913 602 961"><b>Espacio de actividades</b></p>	<p data-bbox="824 814 1385 905">Representa las cosas esenciales que se deben hacer para desarrollar el software. Semat define 15 espacios de actividad</p>
 <p data-bbox="430 1165 581 1192"><b>Actividades</b></p>	<p data-bbox="824 1050 1385 1171">Define uno o más tipos de producto de trabajo y uno o más tipos de tarea, además de dar orientación sobre cómo utilizarlos en el contexto de una práctica</p>
 <p data-bbox="456 1396 553 1423"><b>Prácticas</b></p>	<p data-bbox="824 1276 1385 1381">Es un grupo de elementos necesarios del núcleo de Semat para expresar la guía de trabajo con un objetivo específico.</p>
 <p data-bbox="446 1606 544 1633"><b>Métodos</b></p>	<p data-bbox="824 1503 1385 1591">Es un conjunto de elementos utilizados para formar una base común que describe el esfuerzo de ingeniería de software.</p>






 <p><b>Patrón</b></p>	<p>Es una descripción de la estructura de una práctica.</p>
 <p><b>Productos de trabajo</b></p>	<p>Es un artefacto de valor y relevancia para el esfuerzo de la ingeniería de software. Un producto de trabajo puede ser un documento, una parte de software, una prueba de software o la preparación de un curso.</p>
 <p><b>Competencia</b></p>	<p>Una característica del interesado o equipo que refleja la habilidad de hacer un trabajo. Se puede detallar en diferentes “niveles de competencia”.</p>
 <p><b>Asociación de patrón</b></p>	<p>Permite conectar un patrón con otros elementos en una práctica.</p>
 <p><b>Estado</b></p>	<p>Expresa una situación donde algunas condiciones se proponen.</p>

Figura 5. Representaciones graficas SEMAT [9]

Tabla 1. Tabla representación de elementos de SEMAT. Elaboración propia del autor

El núcleo en acción tiene diversas aplicaciones en la vida diaria de los profesionales de software como: correr una iteración, correr el desarrollo completo desde la concepción de la idea hasta la entrega del producto y escalar hacia grandes organizaciones y esfuerzos complejos de desarrollo de software. Semat presenta beneficios para los profesionales y equipos de trabajo de software, por permitir evaluar el progreso y salud

de los esfuerzos de desarrollo de software, evaluar las prácticas actuales, mejorar la forma de trabajo, mejorar la comunicación entre equipos integrando nuevas ideas y mejorar la interoperabilidad entre equipos, proveedores y organizaciones de desarrollo de software.

Los elementos del núcleo SEMAT forman la base de un vocabulario, un mapa del contexto de ingeniería de software, sobre el cual podemos definir y describir cualquier método o práctica existente o prevista en un futuro próximo. Se definen de una manera que les permite ser extensibles y personalizables, admitiendo una amplia variedad de prácticas, métodos y estilos de desarrollo [9].

## 5. ANTECEDENTES

A continuación, se describen trabajos previos, donde se realiza la descripción de métodos, prácticas o paradigmas de desarrollo usando algún tipo de lenguaje de modelado o representación gráfica:

Según [15], las representaciones gráficas permiten evidenciar la composición de la estructura de un modelo, sus componentes y la forma como interactúan y se integran sus elementos. El núcleo de Semat contiene los elementos esenciales comunes en todos los métodos de ingeniería de software. A través del artículo se describe la representación gráfica del modelo CMMI – DEV exponiendo sus componentes, usando Semat que facilita la manipulación del modelo, con la descripción de las prácticas y sub prácticas utilizando el lenguaje universal proporcionado permitiendo la evaluación del esfuerzo y la calidad. La representación de las prácticas de CMMI-DEV permite identificar los elementos que hacen parte de la definición de las prácticas en sí mismas y que, a su vez, facilitan su

entendimiento y guían su aplicación. Esto representa una ventaja para las personas que requieran implementar las prácticas de CMMI-DEV de manera individual, de acuerdo con sus proyectos o necesidades particulares. Este artículo muestra la importancia del núcleo de Semat y su lenguaje para representar a nivel de mayor detalle los elementos y componentes de la práctica CMMI-DEV, pero a su vez no sirve como elemento a este documento ya que se busca una referencia para representación del desarrollo dirigido por pruebas de aceptación a través del núcleo Semat y en este caso el trabajo se enfoca en prácticas de áreas de proceso de CMMI.

En [8], se busca la representación de métodos basados CDM, RUP (Rational Unified Process) y UNCM, a través de la identificación de las prácticas que tienen en común y con la asociación a éstas prácticas ya existentes de otros métodos con sus actividades correspondientes, con la consecución de la unificación y complementación de las prácticas a través de la representación de los alfas, los productos de trabajo y espacios de actividad asociados, representando las etapas del ciclo de vida de un producto de software (requisitos, análisis, diseño, codificación, prueba, implementación y mantenimiento). En el desarrollo se presenta la sintaxis gráfica de los elementos del núcleo de Semat y los métodos, con la posterior descripción de las distintas representaciones gráficas que ya existen en la literatura y posteriormente se evidencia la validación de la representación propuesta usando el núcleo de Semat. Éste trabajo hace una recopilación exhaustiva de los métodos basados CDM, RUP (Rational Unified Process) y UNCM, logrando representar en detalle cada uno de sus elementos, actividades y demás.

En [3] se basa en el análisis de todos los alfas relacionados con TDD, donde el núcleo de Semat, se utiliza para la implementación de prácticas consistentes a través de una introducción sincronizada de todos los nuevos alfa relacionados con TDD. La intención es adquirir un conjunto de recomendaciones precisas, que se llevan a cabo

conjuntamente para establecer la práctica de TDD y mejorar la calidad del software incorporado.

En [16], el uso de representación de SEMAT facilita el entendimiento del flujo de datos de cada actividad que se ejecuta durante la aplicación de las pruebas de gestión y los procesos fundamentales que integran la dinámica de las pruebas de Software. El objetivo de esta representación es evidenciar la trazabilidad y la consistencia en la práctica verificación de la calidad del software con las fases de la norma, donde se encuentran los productos de trabajo y el rol asociado.

Semat busca estandarizar los procesos e identificar un núcleo común de elementos que facilite el desarrollo de software a equipos de trabajo en diferentes fases del ciclo de vida.

En [4] se presenta un conjunto de las principales características de BDD identificadas a través de un análisis de literatura relevante, con una revisión de estudios existentes y a través de estos, se presenta un modelo conceptual con las características que le constituyen. Éste modelo proporciona una forma más explícita y formal de la descripción de los conceptos de BDD y sus relaciones; constituye un diagrama de clase de UML, que al compararlo con los elementos que ofrece Semat, carece de elementos previamente mencionados como alfas, espacios de actividad, actividades, competencias, roles, entre otros que se definen como comunes a los esfuerzos de la ingeniería de software y que facilitan el entendimiento por parte de los interesados.

En [17], busca acoplar los diferentes conceptos teórico/prácticos implícitos en los proyectos de desarrollo, haciendo principal énfasis en los procesos de ingeniería de requerimientos y las buenas prácticas soportadas por las metodologías ágiles. La Guía

propone una serie de acciones para hacer las cosas y una unificación de todos los aspectos teóricos. También propone una amplia investigación sobre aspectos teóricos tales como Ingeniería de Requerimientos, metodologías ágiles, la forma en que se modelan los requerimientos. En este sentido, fueron analizadas metodologías como Kanban, Scrum y XP, con el fin de determinar las principales características propias de los proyectos de desarrollo ágil, se propone su adaptación al núcleo de Semat, con la propuesta que estas se encuentran en constante cambio, por lo que iniciativas como la de Semat son bastante interesantes en cuanto a que no solo se basan en “las mejores prácticas”, sino que también permiten adaptabilidad. La guía que se plantea es coherente a lo sugerido por Semat, por lo que más que una nueva forma de hacer las cosas se considera como una iniciativa libre de ser adaptada y mejorada en función de cumplir con el objetivo principal de cualquier desarrollo de software. Éste planteamiento fue validado por la opinión de un experto en proyectos ágiles, quien concluyó que el desarrollo de la guía y su propuesta “presentan de manera exitosa y rigurosa, varios elementos conceptuales de ingeniería de software como métodos, mejores prácticas y tendencias que integrados podrían tener un éxito gigante”.

En [18], se propone la representación de prácticas de métodos ágiles de desarrollo de software dentro del núcleo de Semat, al realizar esta representación, demuestra los beneficios que se pueden obtener de un método ágil de desarrollo de software al aplicar sus prácticas, plantea la elección del uso de Semat sobre otras formas de representación de prácticas de ingeniería de software, ya que estas formas de representación no poseen elementos que permitan mostrar las prácticas de software en un lenguaje común, ya que lo define la persona que está creando la representación, siendo esto un limitante a la hora de comparar prácticas de software y también de combinar elementos de diferentes prácticas de software. Semat en su núcleo define elementos universales para cualquier esfuerzo de ingeniería de software y que emplean un lenguaje común. Este lenguaje común restringe los elementos y el vocabulario de quien desee representar una práctica

de ingeniería de software, que permite comparar diferentes prácticas de software y compartir elementos entre dichas prácticas, lo que hace las prácticas reutilizables.

En [19], se propone un método para generar automáticamente una representación conceptual en UML de un lenguaje controlado, a través de esquemas intermedios, llamados esquemas preconceptuales, donde se propone un conjunto de reglas que permiten transformar el lenguaje controlado propuesto a estos esquemas y luego a los diagramas, haciéndolo aplicable a cualquier dominio. Bajo esta propuesta se busca la creación automatizada y mejorada en calidad para los diagramas UML.

En [12] indica un estudio que ha demostrado que por usar ATDD los desarrolladores han consolidado una gran cantidad de confianza, hasta el punto de desplegar sin mucho las pruebas de clientes. Los clientes han visto que las pruebas transmiten sus requisitos comerciales de la manera correcta.

Los trabajos que se mencionan previamente se consideran referentes de representaciones de prácticas inmersas en el proceso de desarrollo de software, mediante el uso de diferentes lenguajes o formas de representación, tales como: esquemas preconceptuales, núcleo de Semat, diagramas UML. Sin embargo, ninguno de dichos trabajos hace referencia a la representación de ATDD, de cara a facilitar su comprensión por parte de los interesados, mediante un lenguaje estructurado y cercano a la ingeniería de software, lo cual es justamente en lo que se enfoca esta investigación.

Se elige el núcleo de Semat como alternativa ideal para representar la práctica ATDD, debido a que cuenta con elementos como competencias, espacios de actividad, actividades, alfas, productos de trabajo, prácticas, etc., que permiten armar métodos a



partir de prácticas mediante un lenguaje estandarizado para la ingeniería de software. Si se compara contra esquemas preconceptuales o diagramas UML, éstos carecen de los elementos mencionados.

## 6. DESCRIPCIÓN TEÓRICA Y VALIDACIÓN DE ATDD.

### 6.1 DESARROLLO DIRIGIDO POR PRUEBAS DE ACEPTACIÓN (ATDD)

ATDD Es una forma de TDD en la que se obtiene el producto a partir de las pruebas de aceptación escritas en conjunto con los usuarios. La idea es tomar cada requerimiento, en la forma de una historia de usuario, construir varias pruebas de aceptación, y a partir de ellas construir las pruebas automatizadas, para luego, mediante ciclos de TDD, ir desarrollando el código. De esta manera, las pruebas de aceptación de una historia de usuario se convierten en las condiciones de satisfacción de la misma, y los criterios de aceptación se convierten en especificaciones ejecutables. En definitiva, ATDD se centra en empezar de lo general y las necesidades del cliente, para ir deduciendo comportamientos y generando especificaciones ejecutables [20].

La prueba de aceptación se enfoca primero en definir y automatizar, en lugar de verificar el código que se desarrolla. Al final del ciclo de desarrollo, ATDD garantiza la continuidad, retroalimentación al desarrollador sobre brechas funcionales que resultan en defectos en el código al final del ciclo de desarrollo. La siguiente lista enuncia algunas de las ventajas que ofrece ATDD:

- Asegura que todos los desarrolladores entiendan los requisitos a nivel granular.
- Comentarios tempranos a los desarrolladores con pruebas fallidas durante el ciclo de desarrollo.
- Dado que los problemas se identifican durante el sprint de desarrollo, las soluciones se pueden organizar rápidamente de forma estratégica.
- Completa suite de regresión automatizada ayuda validando la calidad de una aplicación durante cualquier mejora adicional.

Los objetivos generales de ATDD son:

- Mejorar las especificaciones.
- Facilitar el paso de especificaciones a pruebas.
- Mejorar la comunicación entre los distintos perfiles involucrados en el desarrollo: clientes, usuarios, analistas, desarrolladores y testers.
- Mejorar la visibilidad de la satisfacción de requerimientos y del avance.
- Disminuir la incorporación de funcionalidades o cualidades que el cliente no necesite ni haya solicitado (“goldplating”).
- Usar un lenguaje único, más cerca del consumidor.

Las pruebas unitarias se utilizan para probar la última funcionalidad desarrollada. Se centran en la funcionalidad más reducida posible y en los aspectos técnicos. Normalmente, el alcance de las pruebas unitarias está alineadas con una funcionalidad en particular y el escenario de desarrollo. Las pruebas de aceptación están orientadas a capturar los criterios de aceptación que va de extremo a extremo en los productos o servicios implementados.

En comparación con las pruebas unitarias, las pruebas de aceptación dependen de los requisitos del usuario para facilitar la forma de trabajar. Esto también ayuda a reducir el ciclo de tiempo del proceso general, incluye las interacciones entre diferentes módulos de las aplicaciones, sistemas, base de datos e interfaz de usuario. Es un ejercicio de colaboración entre analistas (para construir los criterios de aceptación), testers y desarrolladores para una solución o un producto [2].

El uso de ATDD tiene un costo definido en el sentido que requiere mucho esfuerzo del cliente y del desarrollador, pero el resultado puede utilizarse directamente para ayudar al desarrollo del código (verificación). Si los requisitos cambian las pruebas deben reflejar esto, agregando un costo mayor al proceso regular de desarrollo. En un proyecto de desarrollo con plena cobertura de pruebas de aceptación automatizadas que no debería tener ningún requisito desarrollado sin primero haberlo descrito como una prueba de aceptación automatizada, y acordar con el cliente que esto es lo que se desea. Si los requisitos resultan estables, las pruebas de aceptación se agregan con un poco de costo, pero grandes beneficios.

ATDD no funciona para todo tipo de proyectos, ATDD que tiene un costo inicial más alto, pero ahora de esfuerzo a largo plazo, ya que las pruebas actúan como un arnés de seguridad para hacer cambios y actúa como documentación de código. Es difícil juzgar cómo los mismos proyectos tendrían éxito sin la disponibilidad de estas pruebas, no se conoce de alguna ninguna investigación sobre su importancia.

Un beneficio de ATDD es la capacidad de mostrar a los clientes las pruebas, el cliente tiene que entender tanto el dominio como la notación para las pruebas. Sin embargo, las

pruebas de ajuste son fáciles de realizar y enseñar, y la gente los entenderá fácilmente [12].

## 6.2 DESCRIPCIÓN DE DESARROLLO DIRIGIDO POR PRUEBAS DE ACEPTACIÓN (ATDD) DESDE LA REVISIÓN BIBLIOGRÁFICA

A partir de una revisión bibliográfica se logra describir las siguientes actividades necesarias para la ejecución de ATDD:

### 6.2.1 Seleccionar y documentar historia de usuario

ATDD es una técnica basada en pruebas, por lo que primero se debe implementar una prueba. El término "prueba" puede ser muy engañoso, especialmente cuando todavía no hay nada que probar. Eso también puede parecer demasiado abstracto al principio y, por lo tanto, puede ser beneficioso (pero no necesario) para escribir primero una breve descripción sobre la nueva prueba: una historia de usuario.

Una historia de usuario, en el contexto de ATDD, es una breve descripción de lo que exactamente va a probar la próxima prueba de aceptación. El propósito de una historia de usuario es capturar el significado esencial de una prueba de aceptación y limitarlo de hacer demasiado. Las historias de usuario están destinadas a ser fáciles e inequívocamente comprensibles, por lo tanto, están escritas en lenguaje natural usando terminología de dominio. Las historias de usuarios deben ser escritas por expertos en dominios para obtener el mayor beneficio fuera de la técnica, pero aún hacen que las pruebas sean más fáciles de entender, incluso si fueran escrito por programadores [13].

La historia de usuario debe ser verificable, significa que el usuario puede confirmar que la historia está hecha. Teniendo las pruebas de aceptación hacen que una historia sea comprobable, y pasar esas pruebas muestra que el sistema satisface las necesidades del cliente.

Puede haber otras razones para dividir una historia en varias historias, incluso si cumple con estos criterios, por ejemplo, no todos los detalles de una historia pueden necesitar ser completados para entregar valor comercial. Por lo tanto, los detalles que no se requieren actualmente, se incorporarán a una nueva historia que se entregará más tarde.

Las características de una historia de usuario deben ser:

- Cada historia de usuario tiene un rol y una acción, y generalmente una razón.
- Las historias deben estar escritas en el idioma del cliente.
- Las historias deben cumplir con los criterios: independientes, negociables, valiosas, estimables, cortas y comprobables.
- Cada historia debe tener criterios de aceptación.
- Los criterios de aceptación pueden ayudar a determinar el tamaño de las historias [20].

### 6.2.2 Escribir escenarios de prueba basados en criterios de aceptación

Como primer paso, el equipo de desarrollo escribe la (s) prueba (s) de aceptación para los casos de uso dados antes de desarrollar la funcionalidad real. Esto garantiza:

- Que el desarrollo satisfaga los criterios de aceptación.
- Todas las expectativas de los clientes son consideradas y cumplidas según los estándares.

Fallar las pruebas de aceptación: Se espera que para la nueva prueba de aceptación mencionada anteriormente los casos deben fallar si la funcionalidad no está desarrollada. Para determinar estas pruebas de aceptación se ejecutan en el código existente. Por lo tanto, la idea detrás de fallar estas pruebas es asegurar la integridad del objetivo una vez que se desarrolla el código y la prueba ha pasado. En caso de que se pasen los casos de prueba sin desarrollar el código, esencialmente significa que no hay necesidad de ningún desarrollo.

### 6.2.3 Implementar prueba de aceptación

Las especificaciones de los ejecutables pueden servir como documentación funcional y almacenamiento de conocimiento que puede ser extremadamente útil, especialmente para software menos experimentado. Los desarrolladores también mejoran sus vidas al facilitar la configuración del sistema en estado deseado durante las actividades de corrección de errores.

Los sistemas complejos tienen muchos módulos interconectados, muchos de los cuales necesitan configurar antes de usar incluso uno de ellos. Sin pruebas automáticas, significa mucho de trabajo manual. Las especificaciones ejecutables manejan el trabajo sucio automáticamente y dejan que los programadores se concentren en lo esencial: localizar y corregir errores en el código. Además, se debe haber disponer de muy buen conocimiento para ejecutar pruebas manuales. En caso de ATDD este conocimiento se almacena en especificaciones ejecutables, así las pruebas se obtienen ejecutadas por

software y no por un humano haciendo el tiempo más efectivo, en segundo lugar, ya no es necesario preguntar a los expertos de dominio por cada pequeño detalle que un desarrollador no sabe porque la mayor parte de la lógica de dominio está incluida en las pruebas automáticas.

ATDD depende en gran medida de las pruebas automáticas. El marco de las pruebas de aceptación automáticas asegura que el sistema nunca se rompa en muchos a la vez, permite a los tester de software concentrarse en su verdadero trabajo: tratar de romper el software, en lugar de pasar su tiempo escribiendo desesperadamente los mismos valores de prueba y otra vez y esperando no toparse con un error más [1].

#### 6.2.4. Ejecución de prueba de aceptación

La ejecución de la prueba proporciona comentarios sobre el trabajo que debe hacerse a continuación. Si la prueba falla, debe hacerse pasar.

ATDD intenta eliminar la ambigüedad presentando los requisitos de software como ejemplos de acciones del usuario que se aplican al software. Estos ejemplos describen cómo exactamente los estados del sistema cambian de un estado a otro, es decir, qué entrada de valores son necesarios para iniciar la transición de estado y qué valores de salida se requieren para validar el resultado.

Un gran problema con la documentación del software es que requiere un esfuerzo significativo y disciplina para mantenerse sincronizado con el código base. El código cambia constantemente durante el proceso de desarrollo, pero los programadores no están dispuestos a dedicar su tiempo a actualizar la documentación. ATDD obliga a sus practicantes a mantener la documentación actualizada. Esto no se aplica a todo tipo de documentación (por ejemplo, manuales de usuario), pero al menos al marco de

especificaciones ejecutables producido a lo largo de la prueba de manejo del desarrollo de la aplicación. Las pruebas de aceptación no se desactualizan mientras están en uso, mientras el proceso de desarrollo obedezca las reglas de ATDD, lo que significa que no se está utilizando ningún otro código escrito que el código para aprobar una prueba que falla, las pruebas permanecen actualizadas.

Si bien la documentación estática tradicional debe actualizarse manualmente, las especificaciones ejecutables, por definición, se rompen cuando divergen del sistema bajo prueba, es decir, cuando el sistema no cumple más con los requisitos, estas también deben actualizarse manualmente, pero a diferencia de la documentación estática, las especificaciones ejecutables proporcionan comentarios instantáneos sobre el estado del sistema que permite a los desarrolladores la posibilidad de solucionar el problema justo después de que surgió. Si una nueva función rompe otras pruebas de aceptación que no sean las pruebas propias de la función, entonces la función en sí o la función rota, las pruebas deben ser ajustadas. En ambos casos, las pruebas y el sistema evolucionan juntos para el nuevo estado del software.

Parcialmente se debe a que ejecutar especificaciones ejecutables permite ver instantáneamente lo que hace un programa mientras inspecciona el código fuente, que solo revela lo que se supone que debe hacer el programa. Un conjunto de pruebas que muestran todo en verde después de un día de trabajo da la maravillosa sensación de control y autoconfianza y proporciona alguna evidencia concreta del trabajo bien hecho.

Las especificaciones ejecutables también proporcionan mucho más valor de documentación para los desarrolladores de software que los documentos en papel tradicionales. Un programador puede realmente comenzar a construir el sistema basándose en especificaciones ejecutables porque a diferencia de la documentación informal que describe un sistema en oraciones ambiguas, las pruebas de aceptación describen el sistema mediante ejemplos con valores concretos de entrada y salida.



Las pruebas de aceptación también ayudan a obtener los valores de datos concretos que generalmente no se indica explícitamente en las historias de usuario abstractas. La interfaz de prueba creada como resultado de dicha obtención mejora la capacidad de prueba general y promueve el acoplamiento flexible del software en módulos [1].

Escribir pruebas unitarias después de definir el objetivo final en forma de casos de prueba funcionales (aceptación), es importante para los desarrolladores con finalidad de escribir las pruebas unitarias antes del código funcional. Esto ayuda a garantizar que cada línea de código sea comprobable y tenga un objetivo definido, asegurando así las mejores prácticas. Esto también ayuda a eliminar cualquier línea de código no deseada. Al escribir pruebas unitarias, los desarrolladores deben asegurarse de la unidad más pequeña se probará para la funcionalidad dada.

Se busca fallar las pruebas unitarias dado que el código funcional aún no está listo, la prueba de la unidad debe fallar.

Hay que tener en cuenta que usar ATDD es una práctica exigente, requiere disciplina. La literatura y algunos estudios han demostrado que los desarrolladores tienden a descuidar la actualización de las pruebas de aceptación automatizadas cuando los plazos están próximos. Al hacerlo todo el arnés de las pruebas se convierte en una carga [12].

#### 6.2.5 Refactorización

Fowler [21] define la refactorización de la siguiente manera: "Refactorizar es el proceso de cambiar un sistema de software de tal manera que no altera el comportamiento externo del código, pero mejora su estructura interna. Es una forma disciplinada de limpiar el código que minimiza las posibilidades de introducir errores. En esencia, cuando refactorizas estas mejorando el diseño del código después de que se haya escrito".

La refactorización es como una inversión que facilita el mantenimiento del software. No afecta la funcionalidad actual, que es solo un reflejo del estado actual del software. Sin embargo, es muy probable que el estado actual sea solo eso, solo un corto momento en la vida de un software. A medida que pasa el tiempo, se anticipa que el software evolucionará y ahí es donde la refactorización juega un papel crucial para garantizar que la evolución sea posible sin acercar el software al cambio de caos por cambio. Sin embargo, la refactorización segura es prácticamente imposible sin una prueba adecuada. En Fowler [21] "Refactorización: Mejorando el diseño del código existente", afirma lo siguiente: "Si desea refactorizar, la condición previa esencial es tener unas pruebas sólidas". Desafortunadamente, en TDD es muy difícil de aplicar al software heredado que no ha sido desarrollado de acuerdo con Principios de TDD desde el principio y no tiene una estructura interna comprobable. Ahí es donde ATDD viene a ayudar: las pruebas de aceptación ejecutables acceden a la interfaz de un software y, por lo tanto, se puede aplicar incluso al código heredado.

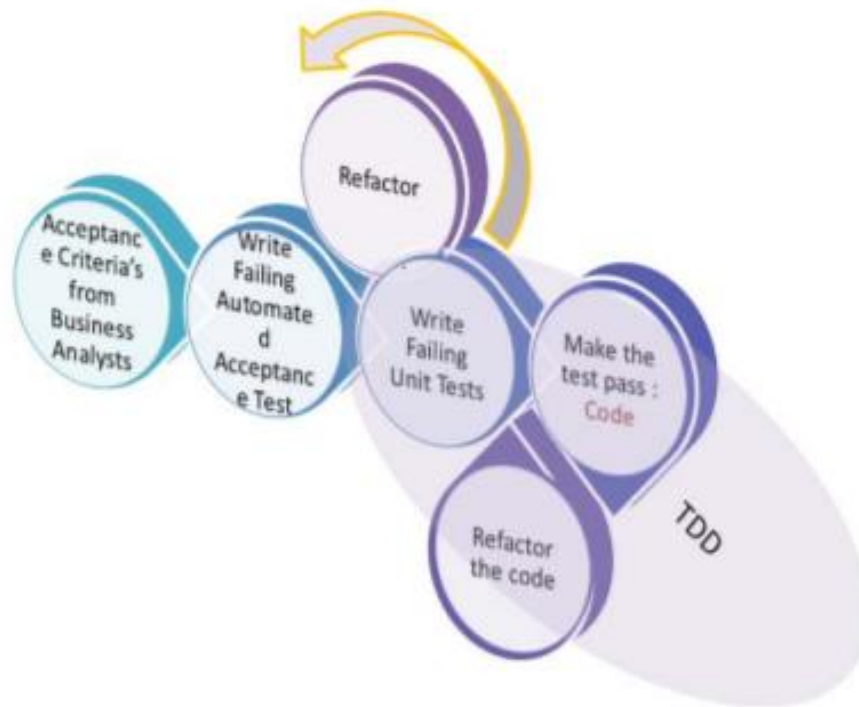
El refactoring, o refactorización de código, es una práctica de la Ingeniería de Software que busca mejorar la calidad del código con miras a facilitar su mantenimiento, pero sin alterar el comportamiento observable del mismo. La refactorización se aplica transformando un sistema una vez que se desarrolló su funcionalidad y la misma ya está codificada. Un problema central del refactoring es poder establecer su corrección, es decir, que el comportamiento observable del programa sea el mismo antes y después de cada refactorización. Mientras que las herramientas de refactoring suelen chequear algunas precondiciones, en muchos casos el análisis necesario es tan costoso que se evita, dejando la decisión en manos del desarrollador. Por otro lado, existen aún muchas refactorizaciones que no pueden ser automatizadas por su complejidad, y por lo tanto el análisis de precondiciones no es suficiente. Otra solución mucho más habitual para asegurar la corrección de un refactoring es utilizar pruebas. En este contexto, decimos que una refactorización es correcta si luego de la misma, las pruebas que antes

funcionaban siguen funcionando sin problema, por lo tanto, la ayuda en el análisis de la corrección de las refactorizaciones es una razón más para desarrollar software siguiendo el protocolo de ATDD, más allá de las ventajas conocidas de contar con distintos niveles de pruebas. Por ello se esperaría una adopción mayor en la industria de la práctica de ATDD.

En definitiva, las pruebas unitarias dan un primer nivel de seguridad. Si este primer nivel falla luego de una refactorización, no significa necesariamente que la misma sea incorrecta; puede ser simplemente que las pruebas unitarias, que están tan adheridas al código, hayan sido afectadas por la refactorización. Por lo tanto, ante una falla en las pruebas unitarias, pasamos a las pruebas de integración, que funcionan como pruebas de un segundo nivel. Estas pruebas de integración, o pruebas de cliente, también pueden ser afectadas por una refactorización que transforma el protocolo de una clase, sin que signifique que la refactorización fuera incorrecta, de manera que tampoco podemos determinar ninguna conclusión. Al contrario, si contamos con pruebas de aceptación que cubren el mismo código, son estas las pruebas que usaremos en última instancia.

Así, ante refactorizaciones pequeñas alcanza con las pruebas unitarias, mientras que refactorizaciones mayores requieren que usemos pruebas de aceptación. A pesar de ser más lentas de ejecutar, debido a su mayor tamaño y a que suelen incluir acceso a recursos externos, las pruebas de aceptación son la red de seguridad de última instancia del comportamiento que se debe preservar.

Las pruebas de aceptación son el último nivel por considerar, dado que una falla en las mismas implicaría que se ha cambiado el comportamiento especificado por los requerimientos y, por lo tanto, la refactorización es incorrecta al no garantizar la preservación del comportamiento [2]



*Figura 6. TDD asociado a desarrollo dirigido por pruebas (ATDD) [13].*

#### 6.2.6 Superar prueba de aceptación

En esta actividad se debe escribir el código con el objetivo de ser una unidad funcional y garantizar que la prueba pase. Se debe refactorizar el código (si es necesario) para optimizarlo aún más, mientras va asegurando que en ningún caso la prueba de la unidad falle. El ciclo de la unidad (TDD) es repetitivo hasta el objetivo final de lograr pasar la prueba de aceptación. Finalmente se integran las actividades de compilación y la ejecución de prueba con el servidor de integración continua, para garantizar que los casos de prueba (tanto las de aceptación, como las unitarias) se ejecutaron con cada compilación [1].

Cuando finalmente pasa, el diseño del software en construcción debe ser revisado y mejorado. La prueba se ocupa de que las características en construcción no se rompan:

cada modificación debe ser seguida por la nueva ejecución de la prueba. Todas las demás pruebas, que pueden ser potencialmente afectadas por los cambios.

#### 6.2.7 Cerrar historia de usuario

Se infiere de la bibliografía consultada que esta actividad es la encargada de finalizar el proceso ATDD. Consiste en posterior a haber logrado superar la prueba de unitaria del código desarrollado, se procede a realizar un commit final de la funcionalidad, dando así por cumplida la(s) funcionalidad(es) objetivo que se definió en la historia de usuario. Se considera una reunión en la cual se entrega hace entrega al cliente de la funcionalidad cumpliendo los criterios de aceptación requeridos.

Cuando el programador queda satisfecho con el diseño, la función puede considerarse lista y se puede seleccionar otra historia de usuario para su procesamiento. Esto hace que el trabajo fluya cíclico e incremental.

### 6.3 DESCRIPCIÓN ENTREVISTAS

#### 6.3.1 Objetivo

Las entrevistas que se proponen buscan abordar el objetivo de investigación que dio origen a este trabajo de grado: identificar las diferencias entre la implementación y la definición teórica encontrada en la exploración de bibliografía científica a través de la

realización de entrevistas dirigidas a desarrolladores de empresas de software de la ciudad de Medellín, donde se indague qué metodología usan, paso a paso para la implementación de ATDD en algún proyecto de software.

### 6.3.2 Población objetivo

Como población seleccionada para las entrevistas, se definen 5 personas con nivel de experiencia superior a 3 años (nivel intermedio y senior) dedicados al desarrollo de software mediante la adopción de técnicas de desarrollo dirigido por pruebas, en su mayoría con rol específico de arquitectos de software o desarrolladores, que hacen parte de equipos de desarrollo de software de empresas medianas y grandes de la ciudad de Medellín. A dicha población se realiza envío de encuesta<sup>1</sup>, con las indicaciones de cómo ser diligenciada, usando Google Forms vía email, previa entrevista general, manifestación de aceptación, explicación y acuerdo de confidencialidad.

### 6.3.3 Metodología

La elaboración de las entrevistas y sus preguntas se realiza de manera descriptiva bajo el modelo teórico de Wohlin *et al* [22], donde se indica que una entrevista es un sistema para recopilar información de o sobre personas para describir, comparar o explicar sus conocimientos, actitudes y comportamiento. Es a menudo una investigación realizada en retrospectiva. El principal medio de recolección de datos cualitativos o cuantitativos son las entrevistas.

---

<sup>1</sup> <https://forms.gle/vnFRPsi4jApm18tb7> Link encuesta compartida a la población objetivo.

Las entrevistas tienen la capacidad de proporcionar una gran cantidad de variables para evaluar, pero es necesario apuntar a obtener la mayor cantidad de comprensión de la menor cantidad de variables ya que esta reducción también facilita la recopilación y el análisis de datos de trabajo. Las entrevistas con muchas preguntas son tediosas para que las respondan los entrevistados, y la calidad de los datos puede, en consecuencia, disminuir. Por otro lado, las encuestas apuntan a proporcionar vistas generales amplias, que pueden requerir preguntas en varios campos.

Los objetivos generales para realizar una encuesta son cualquiera de los siguientes:

- descriptivo
- explicativo
- exploratorio

Se pueden realizar encuestas descriptivas que permitan emitir afirmaciones sobre alguna población específica. Las encuestas exploratorias se utilizan como un estudio previo para una investigación más exhaustiva. La encuesta exploratoria no responde a las preguntas básicas de investigación, pero puede proporcionar nuevas posibilidades que podrían analizarse y, por lo tanto, debe seguirse en la encuesta más centrada o exhaustiva.

De esta manera, la táctica está dirigida y desarrollada utilizando el método cualitativo. Este método de investigación es popular para derivar y explorar aspectos en particular. Los enfoques de investigación cualitativa juegan un papel importante para comprender las percepciones de cómo se desarrolla la práctica de ATDD en algunas empresas de software de la ciudad de Medellín. Ésta metodología también ayuda a definir un contexto del tema que se explora y profundizar en él para adquirir la información importante [23].

Para la selección de los desarrolladores se tiene en cuenta su trayectoria en el área y el aporte a sus equipos de desarrollo. Se procede, inicialmente mediante una conversación con las personas elegidas, donde se explican los objetivos generales de la entrevista, su duración y tiempo estimado para completar, y se destaca cómo los resultados que se obtengan, tienen el propósito de evidenciar cómo se realiza la práctica de ATDD en cada una de las empresas de software donde se emplean y así realizar una comparación entre esta y como se debe realizar según la teoría consultada a través de la bibliografía académica, estableciendo conclusiones. Además de esto se hace énfasis que se mantendrá confidencialidad del nombre de la empresa y el desarrollador, donde en el trabajo de grado se enunciarán los procesos para realizar pruebas de aceptación y su forma de ejecución. Los entrevistados tienen una semana para diligenciar la entrevista y realizar la entrega. Se explica los entrevistados que no es válida la opción de no responder una pregunta.

#### 6.3.4 Diseño de entrevistas

Las preguntas de la entrevista se diseñan a partir del conocimiento que se adquiere de ATDD mediante la realización de búsqueda y análisis de literatura indexada, tales como libros, artículos de revista, conferencias y resultados de investigación; además del conocimiento y experiencia del autor y asesor, donde se recopila información acerca de cómo se realiza la práctica de ATDD, definiendo un orden de las actividades que le conforman.

Se plantean preguntas iniciales abiertas, no estructuradas, a manera de introducción y contextualización, posteriormente se transforman en cerradas, centradas en dar solución



al objetivo de investigación. Las preguntas están numeradas, se dan instrucciones claras y se evitó la elaboración de preguntas ambiguas. Para la versión definitiva de las preguntas, se realizaron 2 versiones previas, que fueron mejoradas a partir de sugerencias dadas por investigadores con conocimiento relacionado a la ingeniería de software.

#### 6.4.5 Estructura entrevista

La estructura consta de un encabezado donde se indica el tipo de instrumento y el título del trabajo de grado. Posteriormente, se agrega una nota donde se explica el tipo de instrumento, la finalidad, se indica el título del trabajo de grado, el autor(a) y asesor, además se informa el programa académico y la universidad. Se explica de nuevo que la información recolectada a través de éste instrumento será con fines académicos y se mantendrá bajo confidencialidad. Luego de esto se solicita a la persona que está diligenciando la entrevista, que proceda a indicar nombre, rol, empresa en la que labora. A continuación, se encuentran las preguntas que fueron seleccionadas para integrar la entrevista, con la estructura descrita:

**Instrumento de recolección de información acerca del uso de la práctica de  
Desarrollo Dirigido por Pruebas de Aceptación (ATDD) por parte de  
desarrolladores de software de la ciudad de Medellín.**

Nota: El siguiente formulario es un instrumento de recolección de información con fines netamente académicos, en el marco del trabajo de grado titulado REPRESENTACIÓN DE LA

## PRÁCTICA DE DESARROLLO DIRIGIDO POR PRUEBAS DE ACEPTACIÓN (ATDD)

USANDO EL NÚCLEO DE SEMAT, realizado por la estudiante Ingrid Jakeline Garcia Mesa, de Ingeniería de Sistemas - Universidad de Antioquia, bajo la asesoría del docente Juan Ricardo Cogollo Oyola.

La información que se recolecte es totalmente confidencial y se usará para fines estadísticos o concluyentes de forma general; su nombre o el de su organización no se mencionarán en el reporte final para garantizar la confidencialidad.

Nombre completo: \_\_\_\_\_

Empresa donde labora: \_\_\_\_\_

Rol/Cargo: \_\_\_\_\_

1. La siguiente es una lista de pasos secuenciales para llevar a cabo el desarrollo dirigido por pruebas de aceptación. Califique en una escala de 1 a 5 cada una de ellas. Donde 5 es totalmente de acuerdo, 4 de acuerdo, 3 parcialmente acuerdo, 2 en desacuerdo, 1 totalmente en desacuerdo. Al final puede manifestar sus opiniones al respecto:

- A. Seleccionar y documentar historia de usuario
- B. Escribir escenarios de prueba basado en criterios de aceptación

- C. Implementar prueba de aceptación
- D. Ejecución de prueba de aceptación
- E. Refactorización
- F. Superar prueba de aceptación
- G. Cerrar historia de usuario

1. Opinión al respecto:

2. Con relación a la lista de pasos de la pregunta 1, ¿qué pasos considera innecesarios?

3. Con relación a la lista de pasos de la pregunta 1, ¿qué pasos adicionales incorporaría?

4. Considera que la práctica de Desarrollo Dirigido por Pruebas de Aceptación (ATDD) es:

- a. Ampliamente usada en la industria de software de la ciudad de Medellín
- b. Parcialmente usada en la industria de software de la ciudad de Medellín
- c. No usada en la industria de software de la ciudad de Medellín

5. ¿Cómo percibe o califica usted el nivel de conocimiento de los profesionales de desarrollo de software de la ciudad de Medellín respecto de la práctica de Desarrollo por Pruebas de Aceptación (ATDD)?

- a. Alto
- b. Medio
- c. Bajo
- d. Nulo

6. ¿Qué razones considera que influyen en la no utilización de la práctica de Desarrollo Dirigido por Pruebas de Aceptación (ATDD)? (seleccione una o varias)

- a. Alto costo de implementación en términos de tiempo y capital humano.
- b. Desconocimiento del proceso de implementación.
- c. No se evidencia el retorno a la inversión por usar dicha práctica.
- d. No es claro cómo contribuye a la calidad del producto final.
- e. No es claro cómo contribuye a la calidad del proceso de desarrollo.

7. ¿Conoce las diferencias entre las prácticas Desarrollo Dirigido por Pruebas (TDD), Desarrollo Dirigido por Pruebas de Aceptación (ATDD) y Desarrollo Dirigido por Comportamiento (BDD)?:

a. Sí

b. No

8. Si su respuesta a la anterior pregunta fue afirmativa, escriba cuáles son las diferencias

9. Entre las prácticas Desarrollo Dirigido por Pruebas (TDD), Desarrollo Dirigido por Pruebas de Aceptación (ATDD) y Desarrollo Dirigido por Comportamiento (BDD), ¿Cuál considera mejor a la hora de adoptarla en su proceso de desarrollo de software y por qué?

10. ¿Qué ventajas y desventajas considera que tiene la práctica de Desarrollo Dirigido Por Pruebas de Aceptación (ATDD)?

11. ¿Cómo ha adquirido su conocimiento acerca de la práctica de Desarrollo Dirigido por Pruebas de Aceptación (ATDD)?

a. Estudio teórico y luego implementación práctica

b. Formación práctica en entorno empresarial

c. Asistencia a eventos de la comunidad de desarrollo

d. Otro, ¿Cuál? \_\_\_\_\_

12. Mencione algunas herramientas o frameworks que usted conozca o haya usado para implementar TDD, ATDD o BDD:

13. ¿Cuál tipo de metodología considera que es la idónea para la implementación de la práctica ATDD?

a. Metodologías tradicionales

b. Metodologías ágiles

c. Las dos anteriores

14. Si usted fuese el encargado de tomar la decisión en su empresa:

a. Adoptaría la práctica de desarrollo dirigido por pruebas de aceptación, consciente de que hacer una inversión adicional en tiempo durante el proceso de desarrollo, traerá recompensa facilitando el proceso de mantenimiento futuro.

b. No adoptaría dicha práctica debido a la necesidad de terminar el proyecto en el menor tiempo posible.

c. Sí adoptaría la práctica

d. No adoptaría la práctica por otra razón

## 7. RESULTADOS Y ANÁLISIS

### 7.1 REPRESENTACIÓN DE DESARROLLO DIRIGIDO POR PRUEBAS DE ACEPTACIÓN EN SEMAT (TEORÍA)

En éste capítulo se desarrolla el objetivo principal planteado en este trabajo de grado donde se plantea identificar diferencias entre la implementación y la definición teórica de la práctica ATDD en algunas empresas de la ciudad de Medellín.

Se inicia realizando una búsqueda de material bibliográfico disponible en revistas científicas, a través de motores académicos, donde se halle documentación disponible acerca de ATDD y cómo se lleva a cabo esta práctica, posteriormente se establece un orden y se definen cómo las etapas para realizar ATDD. Con la descripción de cada una de las actividades se procede a representar en el núcleo de *Semat*.

La práctica ATDD en la teoría y en la práctica es difícil de establecer un modelo que permita su comparación. *Semat* permite a través de representaciones gráficas evidenciar las prácticas empleadas en el desarrollo de software; al hacer esto, se pueden comparar las características de la práctica ATDD y su relación con otros elementos del desarrollo de software, como las personas, procesos y productos de trabajo.

A continuación, se muestran las representaciones obtenidas en SEMAT de cada una de las actividades que conforman ATDD explicado desde la teoría bibliográfica recolectada:

A. Seleccionar y documentar historia de usuario

Espacio de actividad	Actividades	Competencias
Comprender las necesidades del interesado	Revisar historia de usuario con el cliente	Representación de los interesados
Comprender las necesidades del interesado	Refinar historia de usuario posterior reunión con el cliente	Representación de los interesados Análisis
Comprender las necesidades del interesado	Socialización y análisis historia de usuario con equipo de desarrollo	Análisis
Comprender los requisitos	Identificar requerimientos	Análisis
Comprender los requisitos	Documentar reglas del negocio	Análisis
Comprender los requisitos	Realizar análisis de requisitos	Análisis
Comprender los requisitos	Modelar casos de uso	Análisis

Tabla 2: Espacios de actividad, actividades y competencias en: Seleccionar y documentar historia de usuario. Elaboración propia del autor

Espacio de actividad	Alfa	Producto de trabajo	Actividad	Rol
Comprender las necesidades de los interesados	Oportunidad	Historia de usuario	Emplear vocabulario común	Analista Líder o Gerente de proyecto
Comprender los requisitos	Requisitos	Análisis de requisitos	Detallar requisitos	Analista Líder o Gerente del proyecto
Comprender requisitos	Requisitos	Casos de uso	Detallar casos de uso	Analista Líder o Gerente del proyecto

Tabla 3: Espacios de actividad, alfas, productos de trabajo y roles en: Seleccionar y documentar historia de usuario. Elaboración propia del autor



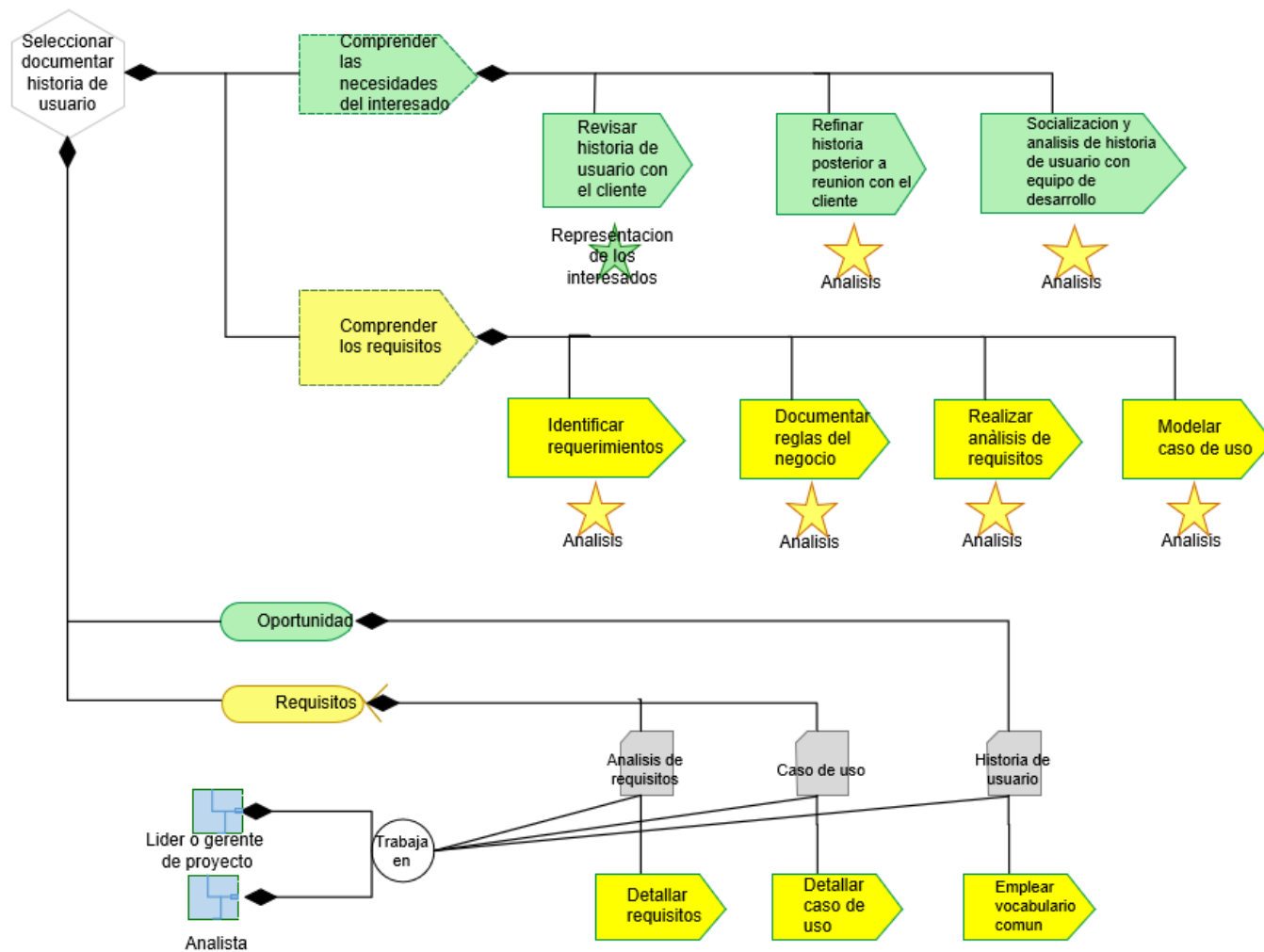


Figura 7. Representación teórica desarrollo dirigido por pruebas en Semat: Seleccionar y documentar historia de usuario. Elaboración propia del autor

## B. Escribir escenarios de prueba

Espacio de actividad	Actividades	Competencias
Asegurar la satisfacción del interesado	Definir criterios de aceptación con el product owner	Representación de los interesados Análisis
Asegurar la satisfacción del interesado	Socializar con equipo criterios de aceptación de los escenarios de prueba	Análisis Gestión
Dar forma al sistema	Definir entorno de ejecución de prueba	Análisis
Dar forma al sistema	Determinar lenguaje de desarrollo	Análisis
Dar forma al sistema	Definir el diseño del lenguaje de prueba	Análisis

Tabla 4: Espacios de actividad, actividades y competencias en: Escribir escenarios de prueba basados en criterios de aceptación. Elaboración propia del autor

Espacio de actividad	Alfa	Producto de trabajo	Actividad	Rol
Comprender las necesidades de los interesados	Oportunidad	Historia de usuario con criterios de aceptación	Detallar historia de usuario con criterios de aceptación	Analista Líder o Gerente de proyecto
Comprender los requisitos	Requisitos	Elaboración de mockup	Detallar mockup elaborado	Analista de pruebas
Comprender requisitos	Requisitos	Casos de uso	Detallar casos de uso	Analista Líder o Gerente del proyecto

Tabla 5: Espacios de actividad, alfas, productos de trabajo y roles en: Escribir escenarios de prueba basados en criterios de aceptación. Elaboración propia del autor

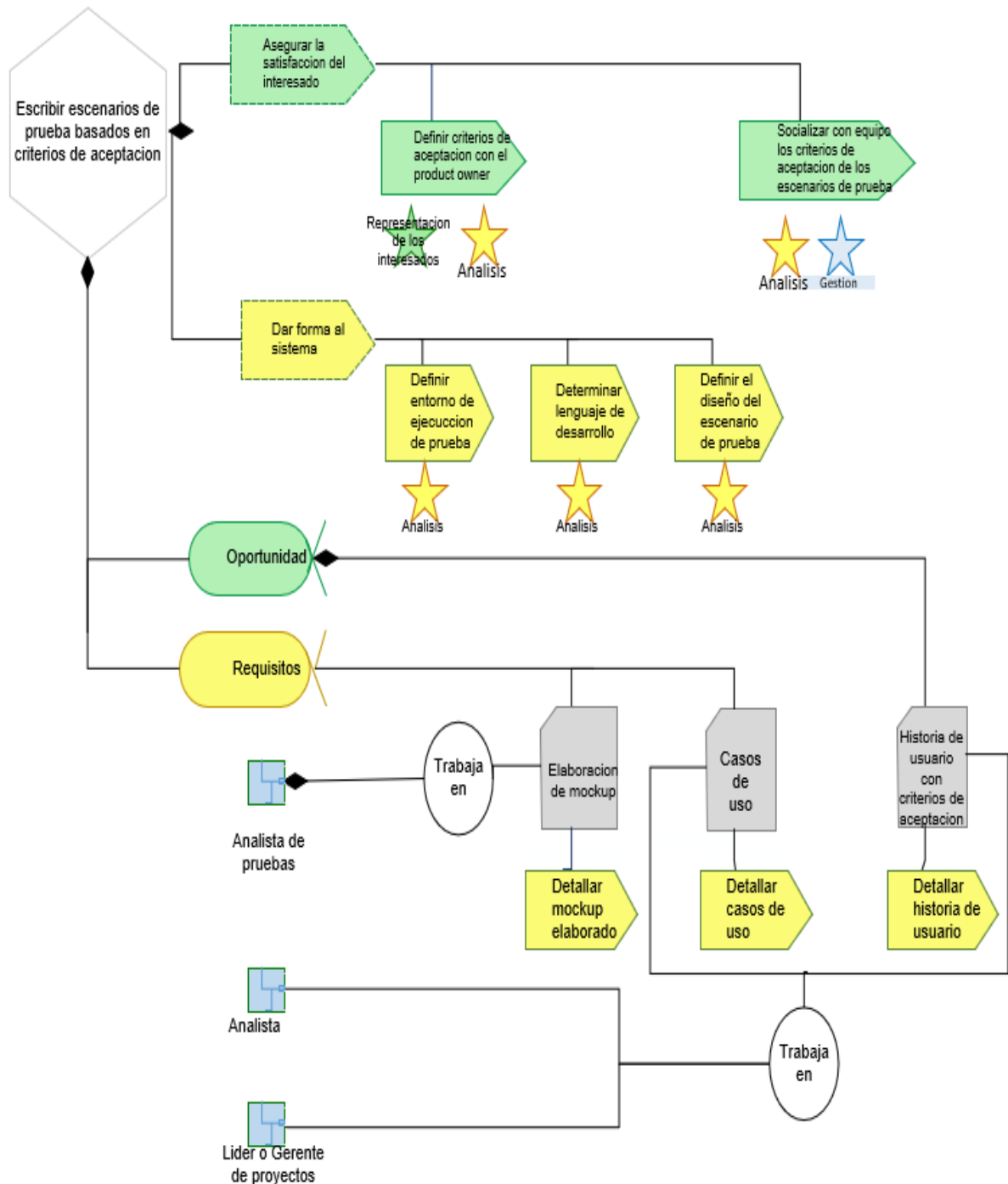


Figura 8. Representación teórica desarrollo dirigido por pruebas en Semat: Escribir escenarios de prueba basados en criterios de aceptación. Elaboración propia del autor

### C. Implementar prueba de aceptación

Espacio de actividad	Actividades	Competencias
Asegurar la satisfacción del interesado	Reunión con cliente para evaluar implementación de prueba	Representación de los interesados Gestión
Implementar el sistema	Escritura manual de prueba de aceptación	Desarrollo
Implementar el sistema	Configurar ambiente de pruebas	Pruebas
Implementar el sistema	Instalación y despliegue de herramientas de pruebas	Pruebas
Prepararse para hacer el trabajo	Asignación de roles	Pruebas
Prepararse para hacer el trabajo	Entrega de tareas a cada rol	Análisis
Prepararse para hacer el trabajo	Asignación técnica y de software	Análisis
Prepararse para hacer el trabajo	Reuniones con equipo de desarrollo	Análisis

Tabla 6: Espacios de actividad, actividades y competencias en: Implementar prueba de aceptación. Elaboración propia del autor

Alfa	Producto de trabajo	Actividad	Rol
Requisitos	Documentación reunión con el cliente	Elaborar documento acta reunión con el cliente	Analista Líder o gerente del proyecto
Sistema de software	Código manual de prueba	Elaborar código manual de prueba	Analista de pruebas Desarrolladores
Sistema de software	Gherkins	Elaborar Gherkin	Analista de pruebas
Sistema de software	Manual técnico	Detallar manual técnico	Analista de pruebas Desarrolladores
Forma de trabajo	Features	Crear feature	Analista de pruebas
Trabajo	Features	Crear feature	Analista de pruebas

Tabla 7: Espacios de actividad, alfas, productos de trabajo y roles en: Implementar prueba de aceptación. Elaboración propia del autor

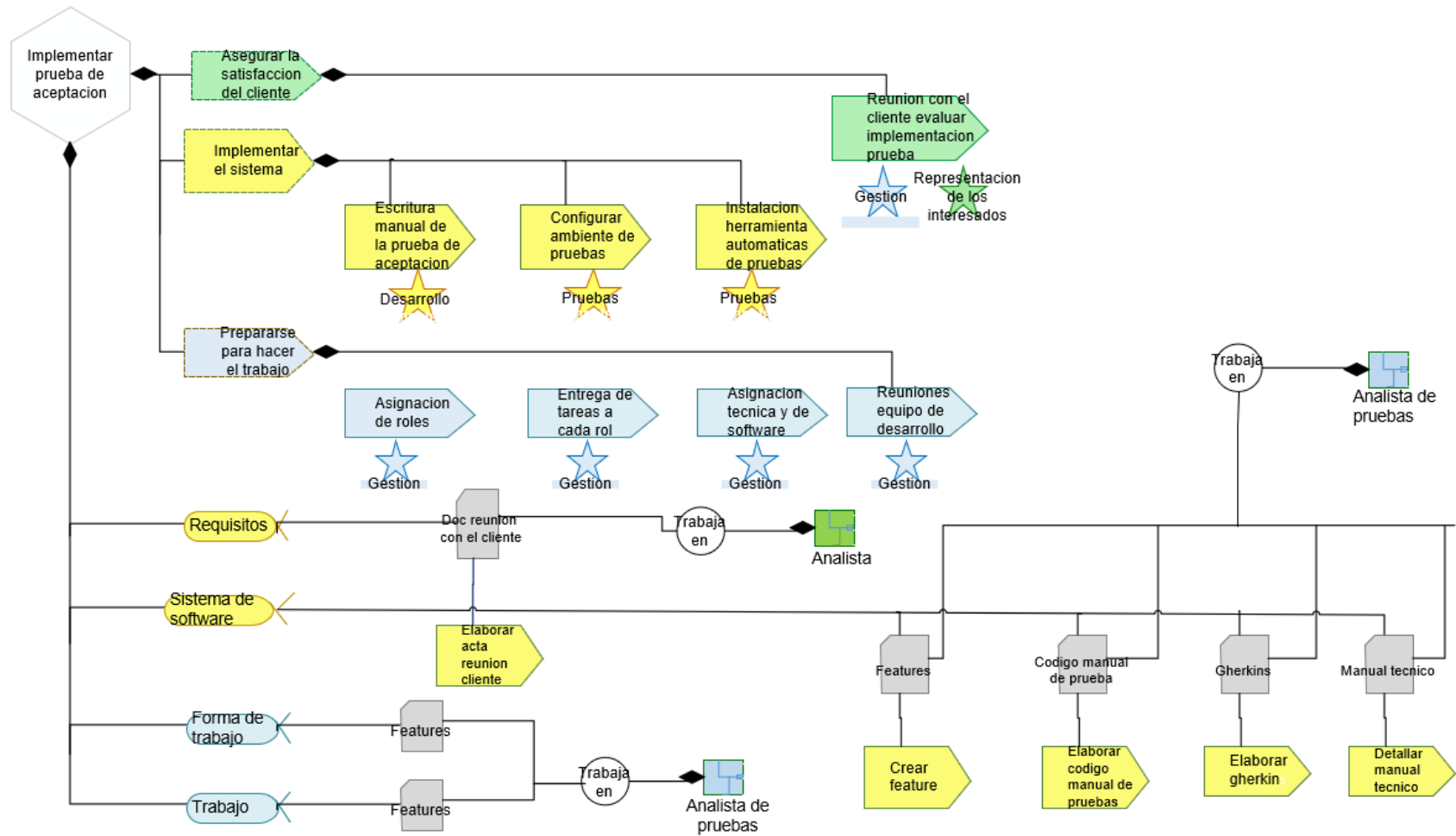


Figura 9. Representación teórica desarrollo dirigido por pruebas en Semat: Implementar pruebas de aceptación. Elaboración propia del autor

#### D. Ejecutar prueba de aceptación

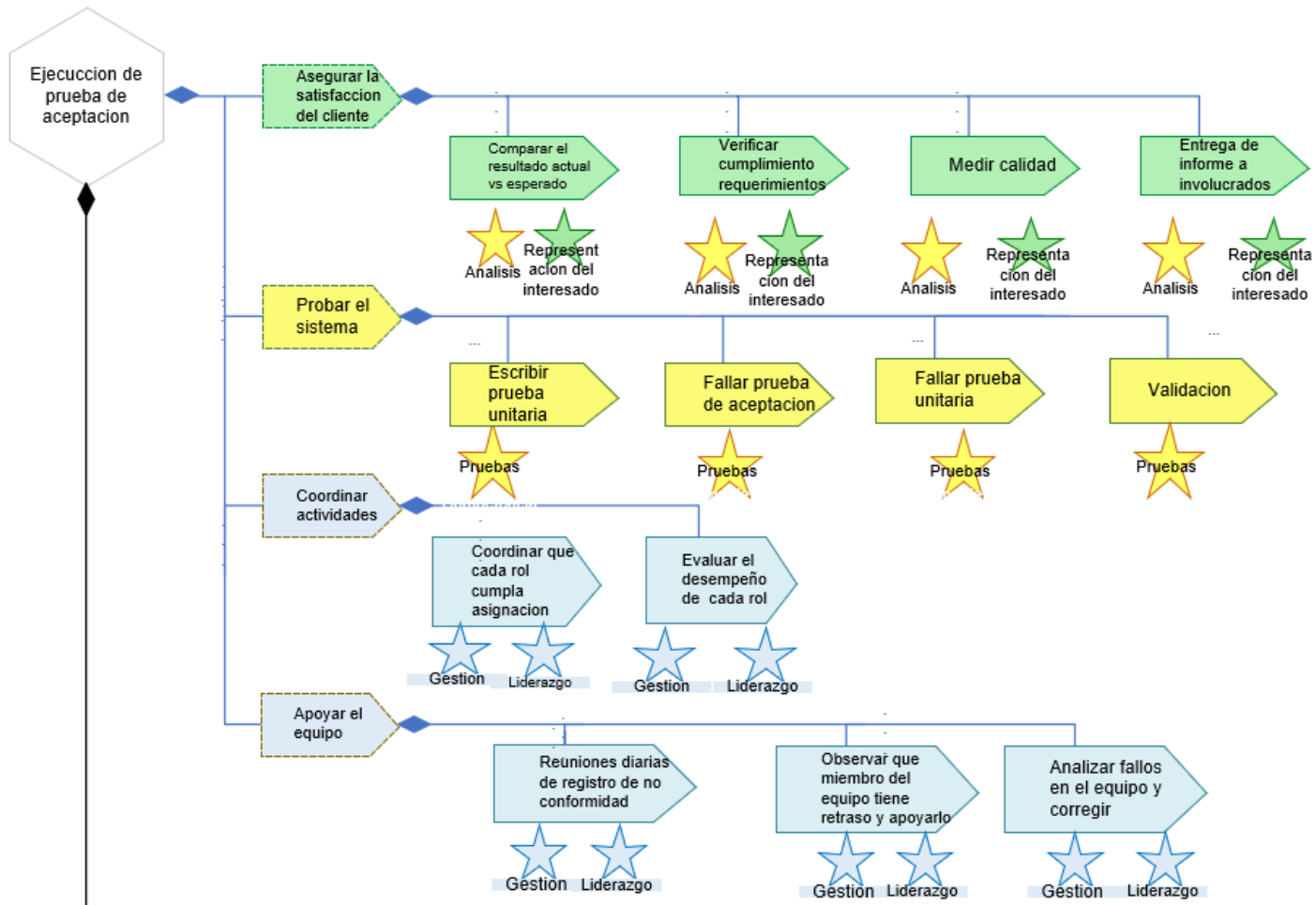
Espacio de actividad	Actividades	Competencias
Asegurar la satisfacción del cliente	Comparar el resultado actual vs esperado	Representación de los interesados Análisis
Asegurar la satisfacción del cliente	Verificar cumplimiento requerimientos	Representación de los interesados Análisis
Asegurar la satisfacción del cliente	Medir calidad	Representación de los interesados Análisis
Asegurar la satisfacción del cliente	Entrega de informe a involucrados	Representación de los interesados Análisis
Probar el sistema	Escribir prueba unitaria	Pruebas
Probar el sistema	Fallar prueba de aceptación	Pruebas
Probar el sistema	Fallar prueba unitaria	Pruebas
Probar el sistema	Validación	Pruebas
Coordinar actividades	Coordinar que cada rol cumpla asignación	Gestión Liderazgo
Coordinar actividades	Evaluar el desempeño de cada rol	Gestión Liderazgo
Apoyar al equipo	Reuniones diarias de registro de no conformidad	Gestión Liderazgo
Apoyar al equipo	Observar que miembro del equipo tiene retraso y apoyarlo	Gestión Liderazgo
Apoyar al equipo	Analizar fallos en el equipo y corregir	Gestión Liderazgo

Tabla 8: Espacios de actividad, actividades y competencias en: Ejecución prueba de aceptación. Elaboración propia del autor

Alfa	Producto de trabajo	Actividad	Rol
Requisitos	Tabla de comparación resultado vs esperado	Elaborar tabla de comparación	Analista de negocios Líder o gerente del proyecto
Requisitos	Documento de verificación de requisitos	Crear documento verificación de requisitos	Analista de negocios
Requisitos	Informe de métricas de calidad	Elaborar informe de calidad	Analista de negocios
Requisitos	Informe de estado al cliente	Elaborar informe de estado	Analista de negocios
Sistema de software	Código fuente prueba fallida	Guardar repositorio prueba fallida	Analista de pruebas
Forma de trabajo	Informe desempeño	Elaborar informe de desempeño	Líder o gerente de proyecto
Trabajo	Registro de no conformidad	Acta de registro de no conformidad	Líder o gerente de proyecto

Tabla 9: Espacios de actividad, alfas, productos de trabajo y roles en: Ejecución prueba de aceptación. Elaboración propia del autor





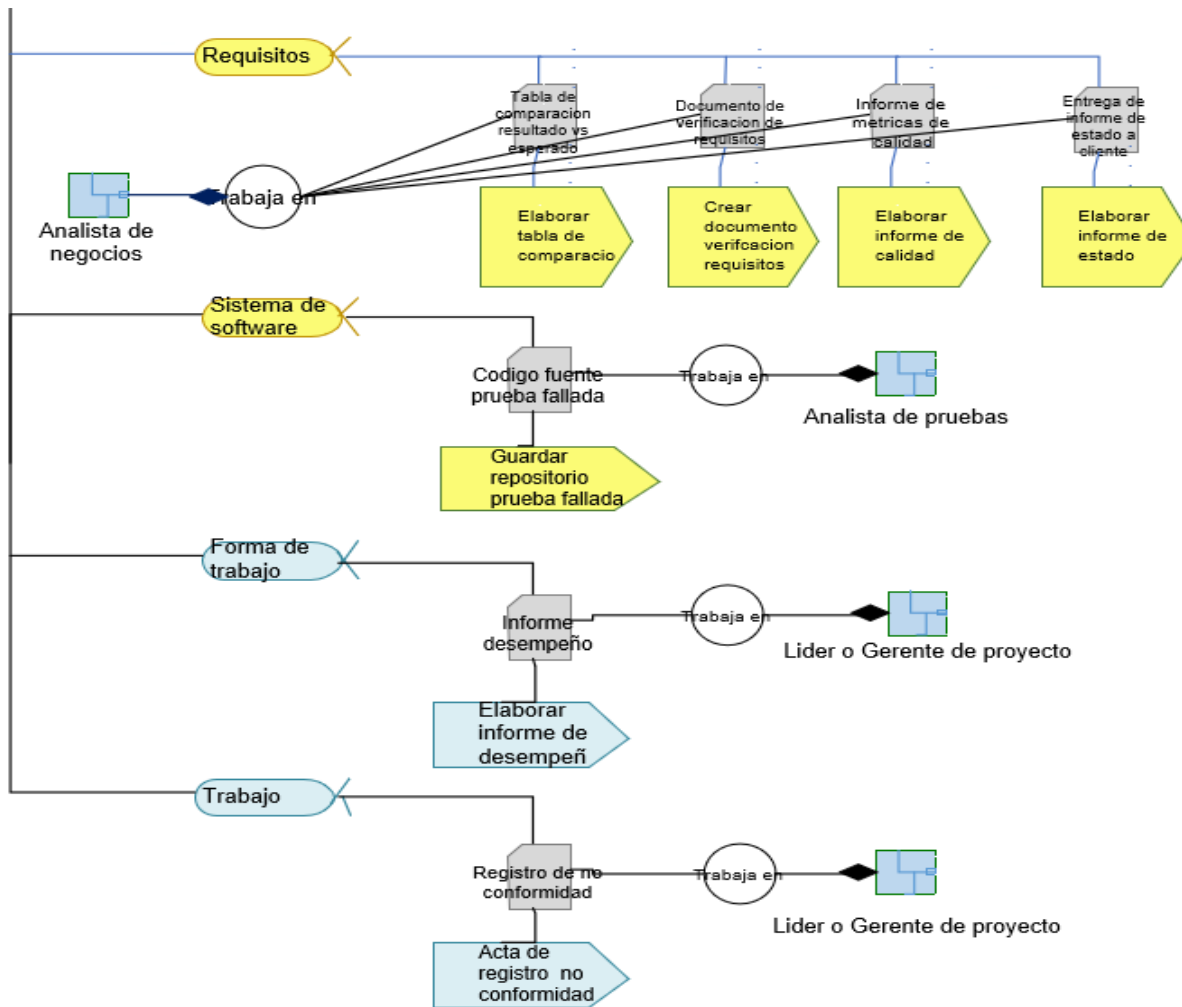


Figura 10. Representación teórica desarrollo dirigido por pruebas en Semat: ejecución prueba de aceptación. Elaboración propia del autor

## E. Refactorización

Espacio de actividad	Actividades	Competencias
Probar el sistema	Reescribir prueba de aceptación	Pruebas
Probar el sistema	Escribir código funcional y superar prueba unitaria	Pruebas
Probar el sistema	Optimizar el código asegurando que ningún caso de prueba falle	Pruebas
Desplegar el sistema	Desplegar entorno	Desarrollador
Desplegar el sistema	Desplegar arquitectura de diseño	Desarrollador
Seguimiento del progreso	Evaluar el desempeño de cada rol	Liderazgo Gestión
Seguimiento del progreso	Medir la calidad e implementación del código	Liderazgo Gestión
Seguimiento del progreso	Evaluar desempeño de cada rol para el sprint	Liderazgo Gestión

Tabla 10: Espacios de actividad, actividades y competencias en: Refactorización.  
Elaboración propia del autor

Alfa	Producto de trabajo	Actividad	Rol
Sistema de software	Código fuente prueba	Elaborar documento acta reunión con el cliente	Analista de pruebas
Sistema de software	Feature prueba	Elaborar código manual de prueba	Analista de pruebas
Forma de trabajo	Evaluación desempeño rol	Evaluación desempeño rol	Líder o gerente de proyectos
Trabajo	Trabajo	Métricas calidad código	Líder o gerente de proyectos

Tabla 11: Espacios de actividad, alfas, productos de trabajo y roles en: Refactorización.  
Elaboración propia del autor

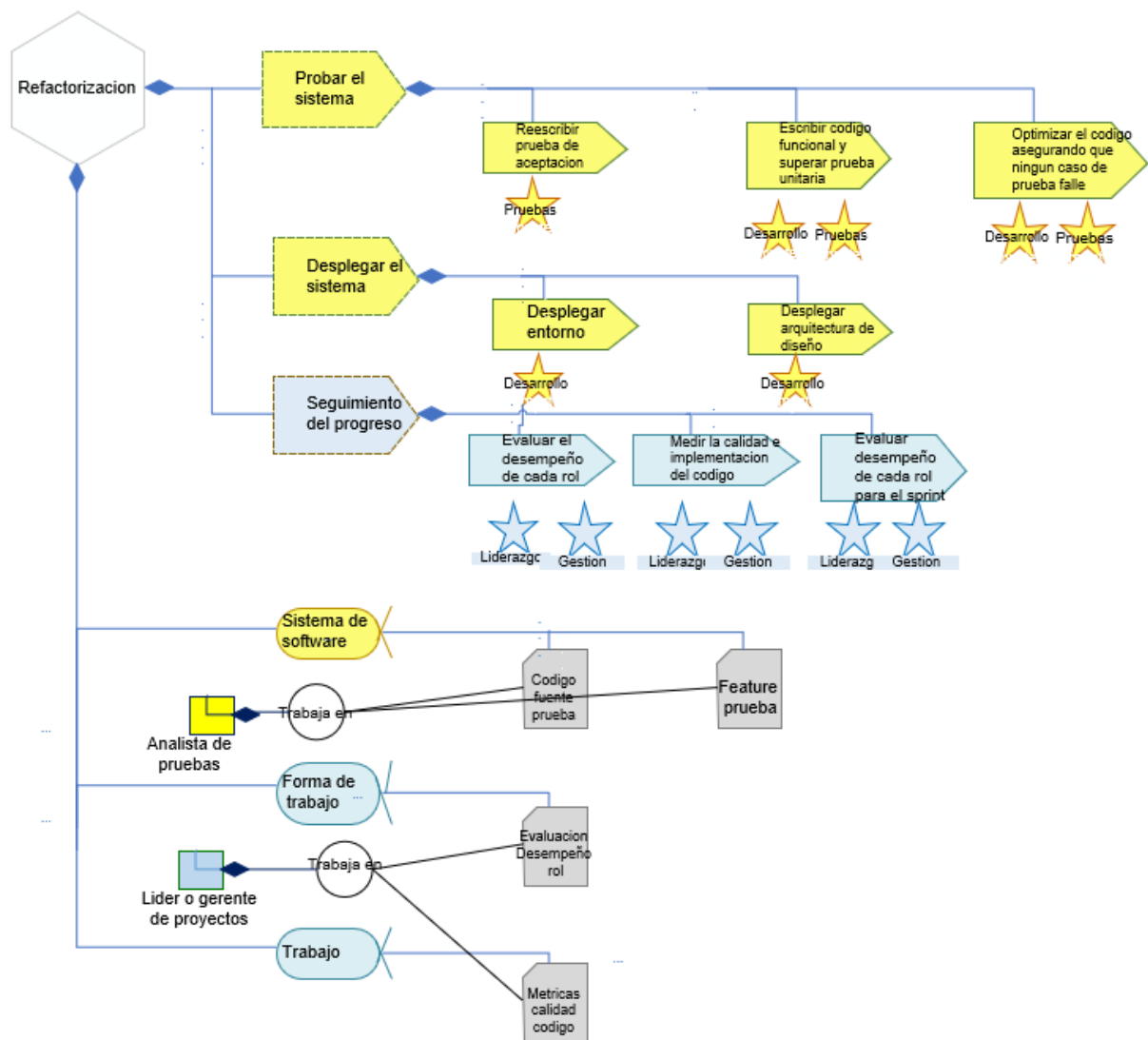


Figura 11. Representación teórica desarrollo dirigido por pruebas en Semat: Refactorización. Elaboración propia del autor

## F. Superar prueba de aceptación

Espacio de actividad	Actividades	Competencias
Asegurar la satisfacción de los interesados	Verificar cumpla con manual de usuario	Análisis
Asegurar la satisfacción de los interesados	Evaluar que la prueba cumpla con lo esperado	Representación de los interesados Análisis
Asegurar la satisfacción de los interesados	Verificar que satisface los requerimientos del cliente	Representación de los interesados Análisis
Probar el sistema	Ejecutar scripts	Pruebas
Probar el sistema	Ejecución de la prueba	Pruebas
Probar el sistema	Despliegue features	Pruebas
Probar el sistema	Validaciones	Pruebas
Probar el sistema	Superar ejecución de la prueba de aceptación y unitaria	Pruebas
Desplegar el sistema	Desplegar el entorno	Desarrollo
Desplegar el sistema	Desplegar arquitectura de diseño	Desarrollo
Seguimiento del progreso	Evaluar que el código sea comprensible, preciso y mantenible	Gestión
Seguimiento del progreso	Medir calidad de la prueba de aceptación	Gestión
Seguimiento del progreso	Evaluar resultado de la prueba vs propósito	Gestión
Seguimiento del progreso	Medir la calidad y mejora en el diseño	Gestión

Tabla 12: Espacios de actividad, actividades y competencias en: Superar prueba de aceptación. Elaboración propia del autor

Alfa	Producto de trabajo	Actividad	Rol
Sistema de software	Repositorio código fuente	Almacenar repositorio	Analista de pruebas
Sistema de software	Scripts	Generar scripts	Analista de pruebas
Sistema de software	Features	Guardar repositorio feature	Analista de pruebas
Forma de trabajo	Evaluación del código	Ejecutar informe de calidad de código	Líder o gerente de proyectos
Forma de trabajo	Métrica prueba de aceptación	Realizar métrica prueba aceptación	Líder o gerente de proyectos
Forma de trabajo	Métrica calidad y diseño	Realizar métrica calidad y diseño	Líder o gerente de proyectos

Tabla 13: Espacios de actividad, alfas, productos de trabajo y roles en: Superar prueba de aceptación. Elaboración propia del autor

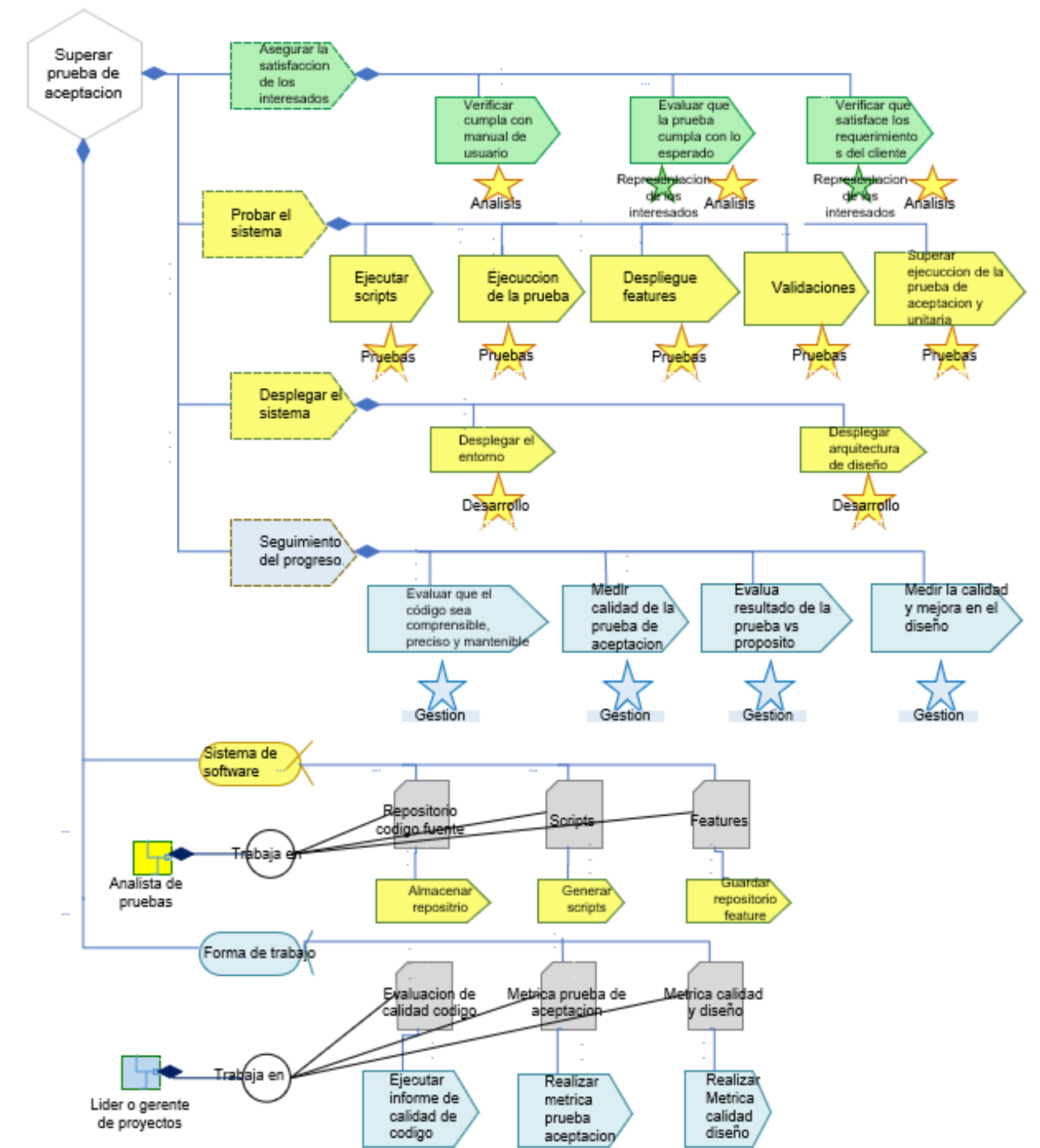


Figura 12. Representación teórica desarrollo dirigido por pruebas en Semat: Superar prueba de aceptación. Elaboración propia del autor

## G. Cerrar historia de usuario

Espacio de actividad	Actividades	Competencias
Necesidad del cliente	Verificación del cliente de la funcionalidad del producto	Representación de los interesados Análisis
Necesidad del cliente	Aprobación del cliente del producto	Representación de los interesados Análisis
Necesidad del cliente	Socialización y análisis historia de usuario con equipo de desarrollo	Representación de los interesados Análisis
Operación de sistema	Despliegue del entorno	Desarrollo
Operación de sistema	Ejecución de la funcionalidad del producto	Desarrollo
Finalizar el trabajo	Evaluación final desempeño de cada rol del equipo	Gestión Liderazgo
Finalizar el trabajo	Cierre final del proyecto con entrega de informes finales	Gestión
Finalizar el trabajo	Reasignación del equipo a otros proyectos	Gestión Liderazgo

Tabla 14: Espacios de actividad, actividades y competencias en: Cierre historia de usuario. Elaboración propia del autor



Alfa	Producto de trabajo	Actividad	Rol
Requisitos	Aprobación cliente	Acta de aprobación funcionalidad	Analista de negocios
Requisitos	Aprobación historia usuario	Finalización historia de usuario	Analista de negocios
Sistema de software	Repositorio código fuente funcionalidad	Almacenar repositorio	Desarrolladores
Sistema de software	Manual técnico	Entrega manuales a cliente	Desarrolladores
Trabajo	Informe evaluación final funcionalidad	Entrega informe final	Líder o gerente de proyecto
Trabajo	Mantenimiento de aplicación y capacitación	Entrega cronograma mantenimiento y capacitación	Líder o gerente de proyecto

Tabla 15: Espacios de actividad, alfas, productos de trabajo y roles en: Cierre historia de usuario. Elaboración propia del autor

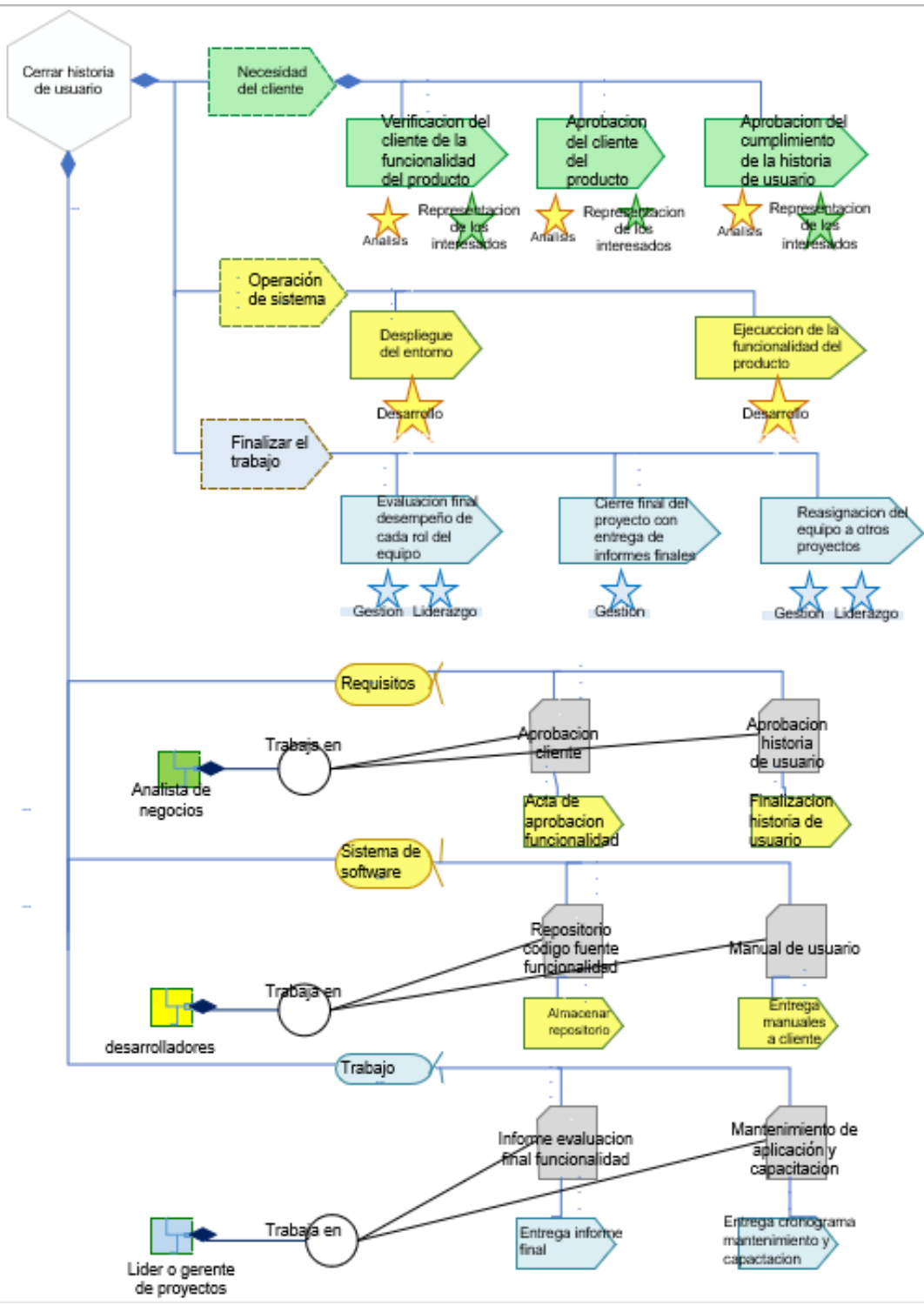


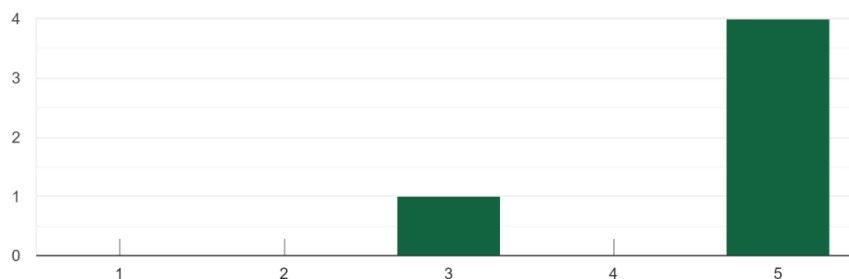
Figura 13: Representación teórica desarrollo dirigido por pruebas en Semat: Cierre historia de usuario. elaboración propia del autor

## 7.2 RESULTADOS ENTREVISTA

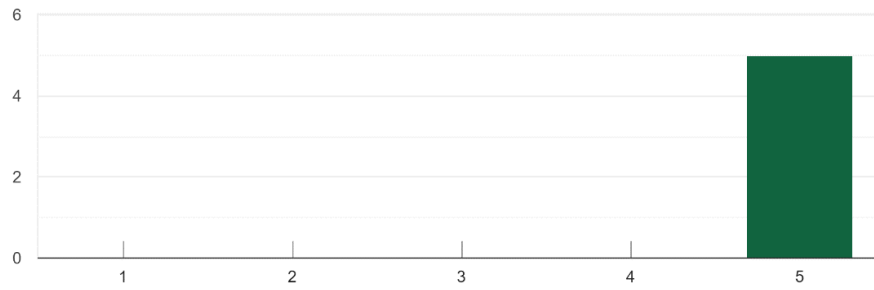
La población entrevistada es un grupo heterogéneo compuesto por personas vinculadas al área de desarrollo en ingeniería de software de la ciudad de Medellín, involucrados con los procesos de pruebas de software, conformado por los siguientes roles: arquitecto de soluciones, líder técnico, analista de desarrollo, analista de pruebas, ingeniero de calidad, quienes aportaron su visión y emitieron sus opiniones respondiendo a un cuestionario previamente elaborado que busca indagar acerca del proceso de la implementación del desarrollo dirigido por pruebas de aceptación desde la práctica en la ciudad de Medellín. Los resultados de las respuestas son las siguientes:

La siguiente es una lista de pasos secuenciales para llevar a cabo el desarrollo dirigido por pruebas de aceptación. Califique en una escala de 1 a 5 cada una de ellas. Donde 5 es totalmente de acuerdo, 4 de acuerdo, 3 parcialmente acuerdo, 2 en desacuerdo, 1 totalmente en desacuerdo. Al final puede manifestar sus opiniones al respecto:

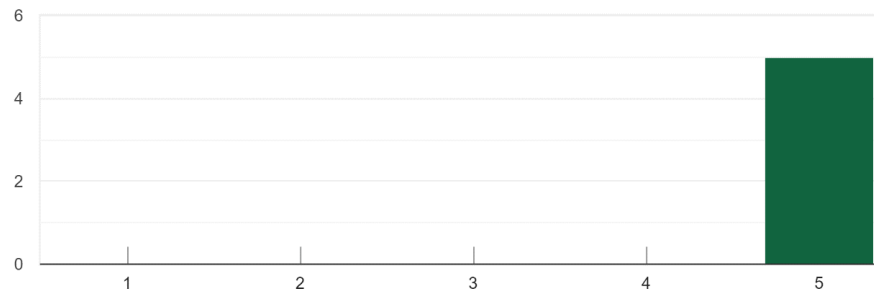
Seleccionar y documentar historia de usuario



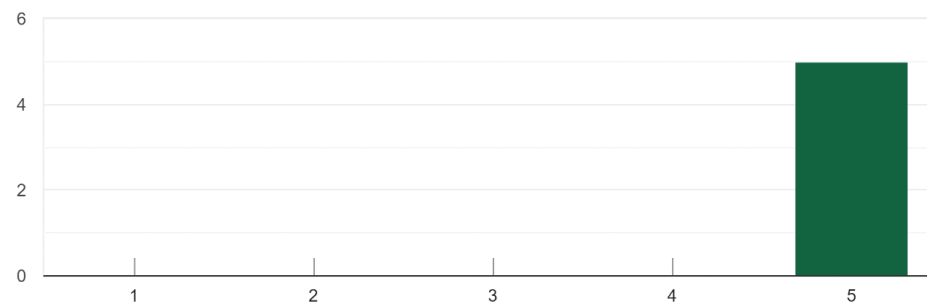
### Escribir escenarios de prueba basados en criterios de aceptación



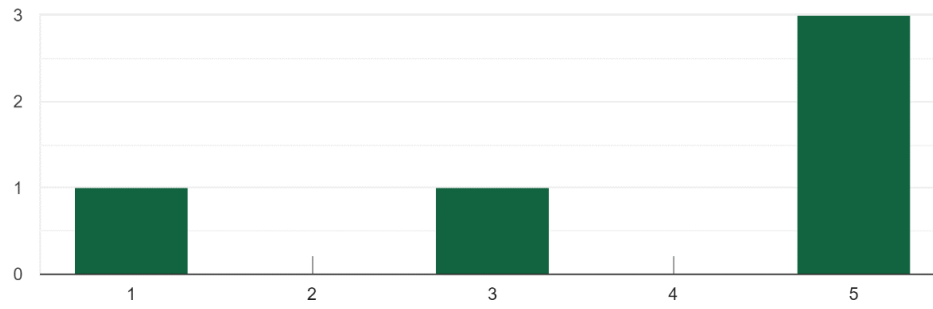
### Implementar prueba de aceptación



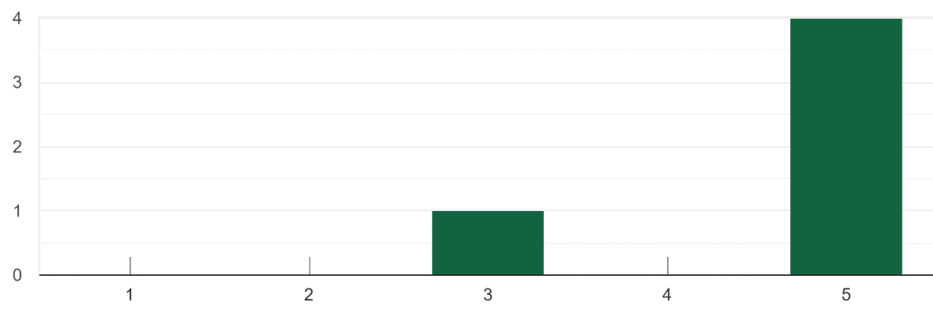
### Ejecución pruebas de aceptación



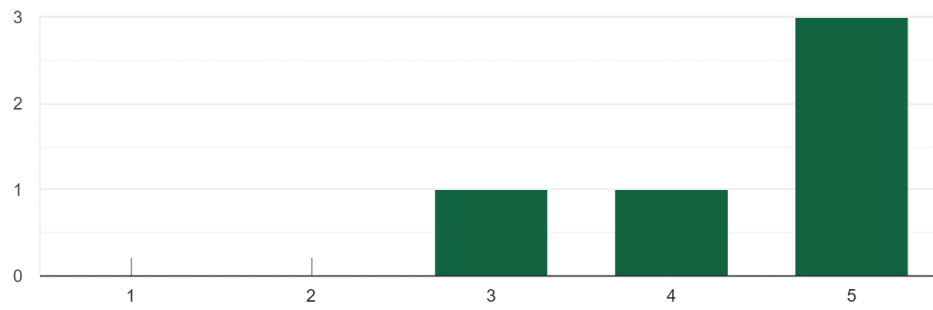
## Refactorización



## Superar prueba de aceptación



## Cerrar historia de usuario



## ¿Qué opinión tiene al respecto?

5 respuestas

La refactorización está más relacionada con la práctica de TDD que con la metodología de ATDD

es una encuesta muy simple

Sería defuado que primero se construya la prueba, luego falle, luego se haga el código más básico para que funcione y por último refactorizar.

BASANDOME EN DESARROLLO DIRIGIDO POR PRUEBAS DE ACEPTACIÓN, AL REFACTORIZAR SI TODO FUNCIONA BIEN, DIRECTAMENTE SE HACE UN DESPLIEGUE, PERO LA HISTORIA DE USUARIO NO SE DEBERÍA CERRAR. SI EN EL SPRINT, PERO EN EL BACKLOG QUEDA PARA POSTERIORMENTE USARLA EN LOS PROCESOS DE SOPORTE O ESCALABILIDAD DE LA MISMA.

Previo al cierre de la historia de usuario se debe hacer alguna entrega al cliente. Adicional el desarrollo lleva internamente TDD

## 2. Con relación a la lista de pasos de la pregunta 1, ¿qué pasos considera innecesarios?

5 respuestas

El paso E

G

Ninguno

NO TENGO MUCHA CLARIDAD DEL PASO F "SUPERAR PRUEBA DE ACEPTACIÓN", PERO PODRÍA DECIR QUE ES INNECESARIO

NA

### 3. Con relación a la lista de pasos de la pregunta 1, ¿qué pasos adicionales incorporaría?

5 respuestas

Incluir prototipos basados en los criterios para validar que se construirá lo que el cliente espera.

Automatización de pruebas

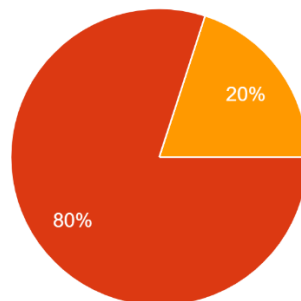
Que primero la prueba falle.

NO INCORPORARÍA PASOS, AL CONTRARIO NO LO LLAMARÍA PASOS SECUENCIALES SINO CÍCLICOS HASTA EL MOMENTO DE LLEGAR AL PASO DE REFACTORIZACIÓN. SI SE REFACTORIZA, SE TIENE QUE MODIFICAR CIERTAS CARACTERÍSTICAS EN CASO TAL QUE NO SE SATISFAGA EL PORCENTAJE TOTAL DE COBERTURA DE PRUEBAS

Demo. Desarrollo aplicando TDD

### 4. Considera que la práctica de Desarrollo Dirigido por Pruebas de Aceptación (ATDD) es:

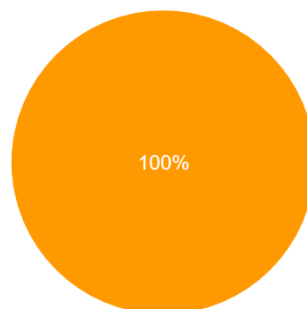
5 respuestas



- Ampliamente usada en la industria de software de la ciudad de Medellín
- Parcialmente usada en la industria de software de la ciudad de Medellín
- No usada en la industria de software de la ciudad de Medellín

### 5. ¿Cómo percibe o califica usted el nivel de conocimiento de los profesionales de desarrollo de software por Pruebas de Aceptación (ATDD)?

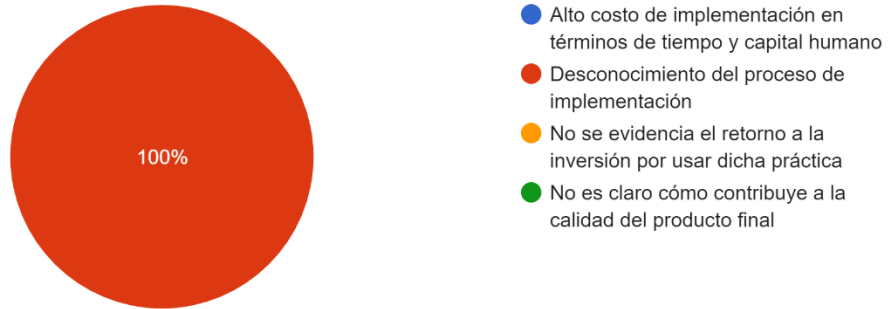
5 respuestas



- Alto
- Medio
- Bajo
- Nulo

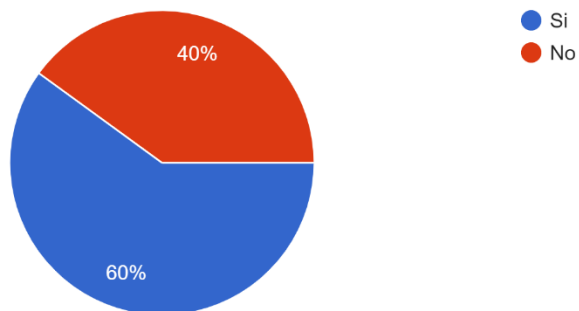
6. Qué razones considera que influyen en la NO utilización de la práctica de Desarrollo Dirigido por Pruebas de Ace...ión (ATDD)? (seleccione una o varias)

5 respuestas



7. ¿Conoce las diferencias entre las prácticas Desarrollo Dirigido por Pruebas (TDD), Desarrollo Dirigido por...lo Dirigido por Comportamiento (BDD)?

5 respuestas





## 8. Si su respuesta a la anterior pregunta fue afirmativa, escriba cuáles son las diferencias

3 respuestas

TDD es una práctica enfocada en el desarrollador para diseñar código a partir de pruebas. ATDD es una metodología (No una práctica) que involucra todo el equipo (desarrolladores, testers, funcionales o PO) que busca desde los criterios de aceptación entregar al cliente lo que solicita. BDD implica un nivel de conocimiento adicional respecto a automatización de pruebas dado que debe diseñar código que superen las pruebas ya no de unidades de código si no de flujos o comportamientos que se den dentro de la aplicación a construir.

PRIMERO QUE TODO, ATDD ES MAS CERCANO A UN PROCESO QUE A UNA ACTIVIDAD QUE SE TENGA QUE DESARROLLAR Y EN ELLA, TODO EL EQUIPO DISCUTE EN COLABORACIÓN A LOS CRITERIOS DE ACEPTACIÓN, EN CAMBIO TDD ES MÁS UNA ACTIVIDAD QUE SE DESARROLLA EN EL PROCESO DE CONSTRUCCIÓN DE UN PRODUCTO DE SOFTWARE, CON EL FIN DE TENER UNAS BASES EN CÓDIGO CON LA CUAL SE HAGA UN DESARROLLO Y SE TENGA UNA COBERTURA TOTAL DE ESTA.

TDD aplica a pruebas unitarias (desarrollo dirigidos por pruebas), ATDD se rige pro pruebas de aceptación y BDD por comportamientos. BDD y ATDD son un poco más parecidos pero cambian en la forma en que llega la historia de usuario y como conciben la a aplicación de pruebas unitarias

## 9. Entre las prácticas Desarrollo Dirigido por Pruebas (TDD), Desarrollo Dirigido por Pruebas de Aceptación (ATDD) y Desarrollo Dirigido por Comportamiento (BDD), ¿Cuál considera mejor a la hora de adoptarla en su proceso de desarrollo de software y por qué?

5 respuestas

Iniciaría con ATDD donde se genera un mejor espacio para identificar y plasmar las necesidades del cliente.

TDD, me parece que es una metodologi que es facil de implementar

TDD, porque es la form ms sencilla de empezar a obtener beneficios de pruebas automatizadas.

SI BIEN TODAS JUNTAS HACEN MARAVILLAS, PERO PARA TODAS SE NECESITA UN NIVEL DE CONOCIMIENTO ALTO Y EN PROYECTOS DE SOFTWARE QUE QUIERAN ADOPTARLAS TODAS, NECESITAN DE UN TIEMPO GRANDE PARA LOGRAR UNA CURVA DE APRENDIZAJE QUE PERMITA LLEVAR EL PROCESO DE MEJOR MANERA. PERO A LA HORA DE QUERER AGILIZAR PROCESOS Y LLEVAR DESARROLLOS EN UN CORTO TIEMPO, IMPLEMENTAR BDD ES LA OPICIÓN MÁS RECOMENDADA PORQUE PENSANDO HACIA UN FUTURO, AHORRARÁ TIEMPOS DE REGRESIÓN DE PRUEBAS Y ESO SE TRADUCE EN COSTOS Y QUE PARA LA INDUSTRIA INFLUYE MUCHO Y ES DECISIVO.

BDD. Tiene un criterio más efectivo sobre la concepción de las pruebas y la forma en que deben desarrollarse para no hacerse repetitivo. Además fomenta que la funcionalidad es parte del código

## 10. ¿Qué ventajas y desventajas considera que tiene la práctica de Desarrollo Dirigido Por Pruebas de Aceptación (ATDD)?

5 respuestas

No le veo desventajas, es más una oportunidad para construir aplicaciones con requerimientos definidos y refinados por un equipo y no sólo por un funcional o PO

ventajas : mayor calidad, diseño enfocado en la necesidad. desventajas : errores no identificados, bd, perder la vision

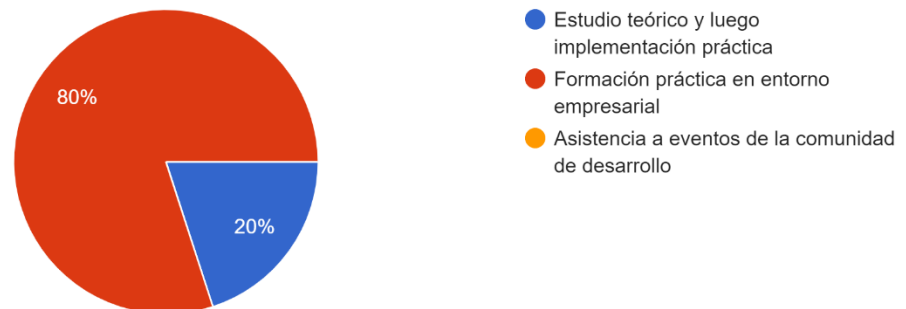
Desventaja: implica un acirva de aprendizaje alta en el equipo.  
Ventaja: permite hacer regresiones muy fácilmente.

LA DESVENTAJA ES QUE TOMA UN TIEMPO PRUDENTE PARA GENERAR UNA CURVA DE APRENDIZAJE DECENTE QUE LLEVE EL PROCESO DE DESARROLLO DE SOFTWARE DE MANERA OPTIMA ENTRE EQUIPOS DE DESARROLLADORES, TESTERS, ETC. PERO ESTA TIENE UNA VENTAJA MUY GRANDE Y SE LLEGA A ELLA CUANDO SE TOMA EL RIESGO DE LA VENTAJA ANTERIORMENTE DESCRITA. Y ES QUE SE HACE UNA PLANIFICACIÓN MÁS SOLIDA DE LO QUE SE HARÁ, NOTIFICANDO ERRORES TEMPRANOS Y HACIENDO CÓDIGOS ESCALABLES, LIMPIOS Y FÁCILES DE REFACTORIZAR POR SUS DEBIDAS DOCUMENTACIONES Y CLARIDADES DEL PROCESO

Desventaja el ciclo de llegada de la historia a Gherkins es un poco lenta y tiene poco uso en la industria. Ventajas código robusto y bien testeado

## 11. ¿Cómo ha adquirido su conocimiento acerca de la práctica de Desarrollo Dirigido por Pruebas de Aceptación (ATDD)?

5 respuestas



## 12. Mencione algunas herramientas o frameworks que usted conozca o haya usado para implementar TDD, ATDD o BDD:

5 respuestas

TDD se puede apoyar en junit para desarrollos sobre java. En ATDD y BDD con Gherkin, cucumber, serenity

SCREEMPLAY

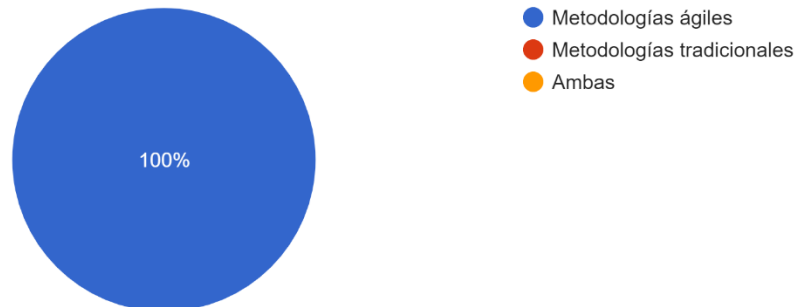
Nunit  
Mstest  
Moq  
JUnit  
Moquito  
Cucumber  
Specflow  
Selenium

JUnit, PHPUnit, Cucumber, Serenity, Mockito, Jacoco, Selenium, Jenkins, Gradle, Maven, etc.

Cucumber, Junit, Karma, Protractor, Selenium

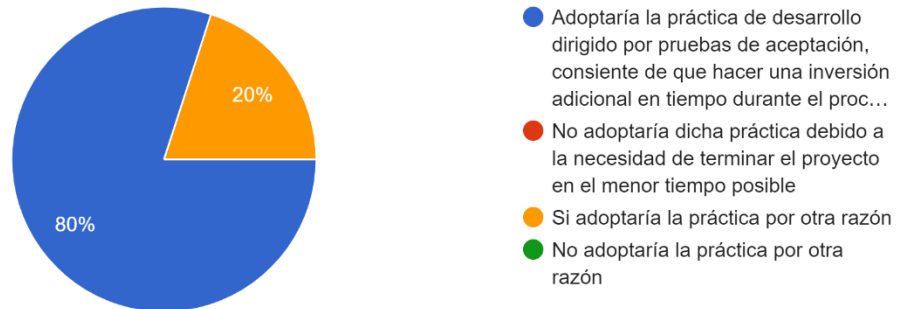
## 13. ¿Cuál tipo de metodología considera que es la idónea para la implementación de la práctica ATDD?

5 respuestas



## 14. Si usted fuese el encargado de tomar la decisión en su empresa:

5 respuestas



### 7.3 ANÁLISIS DE RESULTADOS

Las respuestas obtenidas permitieron identificar la visión que tienen el grupo de entrevistados - quienes pertenecen a cinco de empresas distintas - acerca de la implementación de desarrollo dirigido por pruebas de aceptación, el conocimiento y limitaciones. Lo anterior permitió establecer un referente acerca de cómo se lleva a cabo ATDD en la práctica en algunas empresas de la ciudad de Medellín, desde la perspectiva de estos actores. Esta entrevista permitió analizar lo siguiente:

La estructura por pasos del desarrollo dirigido por pruebas de aceptación, elaborada de acuerdo a como señala la bibliografía académica que debe realizarse, fue construida, solicitando a cada entrevistado asignara una calificación en escala de 1 a 5 de que tan de acuerdo se encontraba con cada uno de ellos hiciera parte secuencialmente de ATDD. En las respuestas se observó que los entrevistados estaban de acuerdo en alto grado con los pasos enunciados, excepto el paso de

refactorización y cierre de historia de usuario donde el grado de satisfacción fue de un 76% y 88%, respectivamente. Al indagar sobre opiniones respecto a la pregunta anterior, lo encuestados manifestaron en un 40% no estar de acuerdo con incluir refactorización en los pasos del modelo, en igual porcentaje lo hicieron con el cierre de historia de usuario manifestando sus consideraciones al respecto. De igual manera manifestaron con cuál de los pasos no estaban de acuerdo, las respuestas fueron diversas, un 60% manifestó inconformidad con alguno de los pasos, en este caso no hubo coincidencia entre los pasos, pero sí se observó que se repetían los mismos pasos que se indicaron no consideraban hicieran parte de los pasos del modelo.

Al indicar que se manifestara que pasos adicionaría al modelo propuesto, los entrevistados propusieron varios, de nuevo sin consenso, las respuestas fueron variadas, entre ellas: fallo prueba, automatización pruebas, etc.

Cuando se consultó acerca del conocimiento del desarrollo dirigido por pruebas, los encuestados señalaron en un 80% que era parcialmente usada en la ciudad y en 20% que no era usada. En la respuesta al nivel de conocimiento de profesionales de desarrollo de software de la ciudad con respecto a ATDD, fue en un 100% que se indicó que es bajo. Al indagar en las razones, el total de entrevistados manifestó que se debe a un total desconocimiento del proceso de implementación.

Se preguntó acerca de si cada uno de ellos conocía la diferencia entre TDD, BDD y ATDD, y en respuesta, en un 60% dijeron que si, y un 40% dijeron no. Posterior a esta pregunta, se solicitó si la respuesta era afirmativa indicara las diferencias, encontrando respuestas no similares entre sí. Al indagar acerca de que práctica de pruebas de desarrollo adoptaría para sus desarrollos de software, un 40% respondió TDD, 20% ATDD y 40% BDD. En la pregunta acerca de las ventajas y desventajas

de implementar ATDD, se obtuvieron diversas repuestas, en el apartado de desventajas existió coincidencia en que se considera como factor, la curva de aprendizaje. En relación cantidad de desventajas vs ventajas, se observa mayor cantidad de desventajas que ventajas.

Al consultar como adquirió conocimiento acerca de ATDD, un 80% de los encuestados, respondió que, a través de formación práctica en entorno empresarial, mientras solo 20% indicó tener estudio teórico y luego implementación técnica. Al indagar sobre las herramientas usadas en la implementación de pruebas TDD, BDD, ATDD, se obtuvo un número considerable de herramientas, algunas de ellas fueron coincidentes. Al indagar acerca de que metodología considera de elección para la implementación de ATDD, hubo un consenso total en elegir las metodologías ágiles como las ideales. Se preguntó, dando opciones de respuesta, si fuera la persona que tomara la decisión de implementar ATDD en su empresa, ante lo cual, la respuesta de elección en 75% fue si la implementarían siendo conscientes de la inversión adicional en tiempo, mientras 20% indicaron que también la elegirían por otras razones, pero en su respuesta no explicó cuál sería el motivo.

Es importante destacar, que los resultados obtenidos en las respuestas permitieron realizar algunos ajustes menores a la representación de ATDD en Semat, ya que al realizar la comparación, se logró asociar mejor las actividades con sus respectivos espacios de actividad, así como algunos productos de trabajo.

#### 7.4 REPRESENTACIÓN DE DESARROLLO DIRIGIDO POR PRUEBAS DE ACEPTACIÓN EN SEMAT (PRÁCTICA)

Al realizar las entrevistas se definen los pasos cíclicos que componen el desarrollo dirigido por pruebas de aceptación que se implementan desde práctica. Los resultados obtenidos en las respuestas permitieron realizar la comparación respecto a la representación de desarrollo dirigido por prueba de aceptación en Semat obtenida a través de la bibliografía, donde se encuentra que existe concordancia con la practica en casi todos los pasos, excepto dos de ellos: refactorizar y cerrar historia de usuario, donde se encontró que existía aceptación en que estos pasos eran coincidentes desde el punto de vista de los entrevistados, pero sugerían añadir algunas de las actividades descritas en las gráficas, que no alteraban significativamente la representación modelada en la teoría de estas dos secuencias.

## 8. CONCLUSIONES

Tal y como lo señala [12], el desarrollo dirigido por prueba de aceptación (ATDD) durante la última década ha ganado mucha atención, pero pocos estudios se han realizado para describir cómo este método afecta los requisitos de la ingeniería.

Allí a manera de resumen, se explica que ATDD proporciona documentación detallada de todos requisitos, pero lo hace de manera iterativa. Esta documentación, construida como pruebas ejecutables, guarda en estrecha comunicación con el cliente. Éstas pruebas emergentes permiten la verificación automática de requisitos comerciales a lo largo de la vida del proyecto, y al mismo tiempo puede actuar como una actualización de documentación para desarrolladores y clientes, reduciendo el tiempo dedicado a probar las piezas automatizadas, así los desarrolladores pueden pasar más tiempo cubriendo aspectos que todavía requieren mano de obra.

No es fácil usar ATDD si se quieren obtener todas sus ventajas. En primer lugar, se requiere más cooperación con el cliente que la agilidad regular de desarrollo. En segundo lugar, no todos los requisitos se pueden describir de la misma manera. Una cuidadosa consideración debe darse a cada requisito antes de elegir si usar la automatización en la aceptación o solo en el nivel de unidad (en cuyo caso las pruebas de aceptación deben ser realizado de otra manera). Independientemente del uso de pruebas de aceptación automatizadas, las pruebas unitarias todavía deben ser usadas. Por último, vemos que ATDD es un método que exige disciplina de los desarrolladores. Si los desarrolladores descuidan la actualización de las pruebas cuando surgen los plazos, el marco de pruebas de aceptación se convierte rápidamente engorroso de mantener y ofrece pocos beneficios [1].



Al realizar la representación en Semat del desarrollo dirigido por pruebas de aceptación según bibliografía, cuando se accedió a conocer los conceptos principales en los cuales se basa el núcleo de Semat, se observó que contrario a como se indica, ATDD dentro de la representación, es un método, ya que se ajusta a cómo la descripción en el manual lo señala, es una composición de prácticas y es dinámico debido a que soporta las actividades diarias durante el desarrollo de software [9].

Las respuestas obtenidas permiten evidenciar que los desarrolladores, aun teniendo conocimiento práctico de ATDD, consideran que su implementación es baja en otras empresas de software la ciudad de Medellín, cuya la principal razón es la curva de aprendizaje, lo cual se considera de alto costo en un proyecto. Se identifica además la necesidad de realizar actividades de divulgación, tanto a nivel académico con el sector empresarial, con el animo de incentivar la cultura de estas formas de desarrollar software, donde todo parte desde la concepción de las pruebas.

## 9. REFERENCIAS BIBLIOGRÁFICAS

- [1] N. Koudelia, *Acceptance Test-Driven Development. Master's Thesis in Information Technology (Software engineering)*, University of Jyväskylä: Jyväskylä, 2011.
- [2] C. Fontela y A. Garrido, «Connection between Safe Refactorings and Acceptance Test Driven Development,» *IEEE Latin America Transactions*, vol. 5, nº 11, pp. 1238-1244, 2013.
- [3] V. Savić y E. Varga, «Extending the SEMAT Kernel with the TDD,» *IET Software*, vol. 12, nº 2, pp. 85 - 95, 2016.
- [4] C. Solis y X. Wang, «A study of the characteristics of behaviour driven development,» de *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, Oulu, 2011.
- [5] Y. F. Pérez, R. L. C. Delgado, C. L. C. Aguila, M. G. Jorrín, D. A. Molina y V. H. Aguilar, «Pruebas de aceptación para un software con la presencia de una entidad certificadora de la calidad,» *Revista Cubana de Ciencias Informáticas 2007*, vol. 1, nº 3, pp. 84-93, 2007.
- [6] L. Castaneda Bueno, «Ingeniería de requerimientos,» 19 Enero 2010. [En línea]. Available: [https://repository.icesi.edu.co/biblioteca\\_digital/bitstream/item/4083/1/Presentacion\\_ciclo\\_vida\\_software.pdf](https://repository.icesi.edu.co/biblioteca_digital/bitstream/item/4083/1/Presentacion_ciclo_vida_software.pdf). [Último acceso: 2 Julio 2019].
- [7] K. E. Kendall y J. E. Kendall, «Análisis y diseño de sistemas,» de *Análisis y diseño de sistemas*, Mexico, Prentice Hall, 2005, pp. 65-74.
- [8] L. D. Jimenez, *Representación en el núcleo de Semat de practicas de métodos de desarrollo basados en planes. Tesis Maestria en ingeniería de sistemas*, Universidad Nacional sede Medellín: Medellín, 2016.
- [9] O. M. Group, «Semat Org,» 1 10 2018. [En línea]. Available: <http://semat.org/documents/20181/57862/formal-18-10-02.pdf/866c80c0-cdc8-488b-bcf8-0c67cb60b5d7>. [Último acceso: 8 Junio 2019].
- [10] K. Beck, *Test Driven Development by example*, Addison Wesley, 2002.
- [11] D. S. Janzen y H. Saiedian, «Test-Driven Development: Concepts, Taxonomy, and Future Direction,» *Computer Science and Software Engineering*, vol. 38, nº 9, pp. 43 - 50, 2005.
- [12] B. H. a. T. Stalhane, «Automated Acceptance Testing as an Agile Requirements Engineering Practice,» de *45th Hawaii International Conference on System Sciences*, Maui, 2012.
- [13] M. S. Vishal Aggarwal, «Acceptance Test Driven Development,» *Journal of Advanced Computing and Communication Technologies*, vol. 2, nº 1, 2014.
- [14] I. Jacobson, P.-W. Ng, P. E. McMahon, I. Spence, S. Lidman y C. M. Zapata, «La Esencia de la Ingeniería de Software: El Núcleo de Semat,» *Revista Latinoamericana de Ingeniería de Software*, vol. 1, nº 3, pp. 71-78, 2017.
- [15] J. Valderrama, L. D. Jimenez Pinzón y C. M. Zapata, «Representation Of CMMI-DEV Practices In The Semat Kernel,» *IEEE Latin America Transactions*, vol. 13, nº 10, pp. 3476 - 3481, 2013.

- [16] C. E. Durango Vanegas, F. A. Vargas Agudelo, D. E. Soto Duran y J. C. Giraldo Mejía, «Representación en el núcleo de SEMAT de la norma ISO/IEC/IEEE 29119-2 para identificar patrones en pruebas de software,» de *4to. Congreso Internacional de Investigación e Innovación en Ingeniería de Software 2016, CONISOFT'16*, Puebla, Mexico, 2016.
- [17] J. D. Murcia Torres y M. E. Torres Moreno, «Guía para la Integración de métodos formales de ingeniería de requerimientos en procesos de desarrollo ágil,» Bogota, 2015.
- [18] J. O. Muñoz Rengifo, *Equipos de desarrollo de software: sus prácticas representadas en Semat*, Tesis de grado. Universidad Nacional Sede Medellín: Medellín, 2015.
- [19] C. M. Zapata Jaramillo, *Definición de un esquema preconceptual para la obtención automática de esquemas conceptuales de UML*, Tesis doctoral, Facultad de Minas: Universidad Nacional Sede Medellín, Medellín, 2006.
- [20] K. Pugh, *Lean-Agile Acceptance Test-Driven*, Indiana: Pearson education, 2011.
- [21] M. Fowler, *Refactoring*, Addison-Wesley, 1999.
- [22] C. Wohlin, P. Runeson, A. Wesslén, B. Regnell, M. C. Ohlsson y M. Host, *Experimentation in software engineering*, Springer, 2000.
- [23] J. Sitzia, K. Kelley, B. Clark y V. Brown, «Good practice in the conduct and reporting of survey research,» *International Journal for Quality in Health Care*, vol. 15, nº 3, pp. 261 - 266, 2003.