PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

UNA EVALUACIÓN EXPERIMENTAL PARA COMPARAR LA CALIDAD DE UN SOFTWARE APLICANDO O NO TDD DENTRO DEL MODELO CASCADA

Tesis para optar el grado de Magíster en Ingeniería Informática que presenta

ANTONY MAYKOL GOICOCHEA PUERTAS

Dirigido por

HÉCTOR ANDRÉS MELGAR SASIETA

San Miguel, 2015

Resumen

El presente trabajo de investigación brinda un enfoque general de la aplicación de la técnica *Test Driven Development (TDD),* o Desarrollo Guiado por Pruebas, dentro de la metodología tradicional con enfoque Cascada, y cómo su implicancia proporciona resultados favorables durante el proceso de implementación y en consecuencia la mejora de la calidad del producto. La investigación se llevó a cabo mediante una evaluación experimental en donde se crearon cuatro (4) grupos de desarrollo, cada uno de ellos estaba conformado por once (11) estudiantes del octavo ciclo de la especialidad de Ingeniería Informática. El experimento consistió en que dos (2) grupos apliquen la técnica de TDD dentro de la metodología Cascada y los otros (2) grupos no la apliquen.

La inclusión de la técnica TDD se llevó a cabo en las primeras fases del modelo Cascada (Definición de requerimientos y Diseño del sistema) a través de la definición de los Casos de Prueba (Test Cases) y mediante ellos se estableció la línea inicial para el comienzo de la implementación del código fuente del sistema a realizar.

Mediante la aplicación de este experimento se logró obtener resultados estadísticos iniciales que confirman que la inclusión de la técnica TDD en el proceso de implementación y pruebas unitarias permite identificar una mayor cantidad de errores, lo cual se ve reflejado al final del proceso en un producto de mayor calidad.

Finalmente, al concluir el proceso de desarrollo del software, se aplicó una encuesta para medir la percepción / intención de uso de los participantes respecto a las técnicas TDD y Cascada.

Palabras Clave: *Desarrollo Guiado por Pruebas; TDD; Evaluación Experimental; Metodologías Tradicionales; Cascada; Ingeniería del Software.*

Abstract

This research provides a general approach to the application of the technique Test Driven Development (TDD) within the traditional waterfall methodology and how its implication provides favorable results during the implementation process and consequently the improvement of product quality. The research was carried out by an experimental evaluation in which four (4) development groups were created, each of which was conformed by eleven (11) students of the eighth cycle of the specialty of Computing Engineering. The experiment consisted of two (2) groups apply the technique of TDD within the waterfall methodology and the other groups do not apply.

The inclusion of the TDD technique was carried out in the early stages of the cascade model (definition of requirements and system design) through the definition of test cases (Test Cases) and through them the initial line was established for the beginning of the source code implementation of the system to perform.

With the implementation of this experiment it is possible to obtain initial statistical results confirming that the inclusion of the TDD technique in the process of implementation and unit testing can identify a greater number of errors, which is reflected at the end of the process on a product higher quality.

Finally, at the end of the software development process, a survey was applied to measure the perception / intention to use of the participants regarding the TDD and Waterfall techniques.

Keywords: *Test Driven Development; TDD; Experimental Evaluation; Traditional Methodologies; Waterfall; Software Engineering*.

A mis padres y a toda mi familia, gracias por su presencia y consejos, los cuales me brindaron la fuerza y el impulso necesarios para seguir siempre adelante y no perder de vista mis objetivos.

A Dios, por su bendición y la fortaleza espiritual que me otorga día a día.

Un agradecimiento especial para todos los catedráticos de la Maestría en Informática de la Escuela de Posgrado, por sus enseñanzas y apoyo constante para cumplir el objetivo de mejorar profesionalmente cada día.

A la Pontificia Universidad Católica del Perú, por permitirme formar parte de su excelente plana de profesionales, mucho orgullo de pertenecer a esta magnífica casa de estudios.

ÍNDICE

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

ÍNDICE DE CUADROS

ÍNDICE DE IMÁGENES

Capítulo 1: Presentación del trabajo de investigación

1.1. Introducción

Un Proceso de Desarrollo de Software se define como un conjunto de actividades, métodos, prácticas, y transformaciones que se utilizan para desarrollar y mantener el software y sus productos asociados [Cugola, 1998]. Estos procesos cuando son bien diseñados y ejecutados conducen a lo que se conoce como éxito metodológico [Perrin, 2008].

Existen diversas metodologías en la Industria del Software las cuales permiten a las organizaciones optar por aquella que más les convenga al momento de desarrollar un nuevo producto, ya sea propia, tradicional o ágil [Gharaibeh, 2009]. Todas éstas presentan ventajas y desventajas en la gestión del proceso, adopción de cambios en los requisitos y capacidad para volver a etapas previas del desarrollo sin mayores problemas [Pressman, 2002].

Independientemente de la metodología elegida, el rápido ritmo del mundo de los negocios de hoy en día, donde la competencia y la tecnología están en su apogeo, las empresas de desarrollo de software necesitan mejorar sus estándares de calidad, además de reducir los costos en sus operaciones [Karamat, 2006].

Se define la calidad del software como: *"la concordancia con los requerimientos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente"* [Pressman, 2002]. Existe algo importante que se debe tener presente, la calidad del software debe ser construida desde el comienzo, no es algo que puede ser añadido después. Para que el producto final sea de calidad, el proceso por medio del cual éste es elaborado debe ser también de calidad [Pressman, 2002]. Finalmente, la calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor [ISO25000, 2005].

En la actualidad, donde existe mayor competencia entre las empresas por el desarrollo de software, los clientes optan por aquellas que prestan software

fiable y de calidad [Karamat, 2006]. Es ahí donde el aseguramiento y el control de calidad, basados en las pruebas de software, resultan ser el más fuerte ataque en la lucha por conseguir productos de alta calidad.

El desarrollo del presente trabajo describe, como tema de investigación, el uso de la técnica *Test Driven Development (TDD)* o Desarrollo Guiado por Pruebas, en la cual el desarrollador escribe las pruebas unitarias de un conjunto de requisitos de usuario antes de escribir el código y luego se ejecutan las pruebas para verificar que fallen, posteriormente se implementa el código necesario para conseguir superar las pruebas y finalmente se lleva a cabo la refactorización del código implementado. El objetivo del presente trabajo es lograr aplicar la técnica mencionada en un proceso de desarrollo de software que sigue una metodología tradicional en Cascada, a fin de lograr medir la calidad del producto, en función de la cantidad de errores detectados.

Como parte del contexto, el presente trabajo de investigación se llevó a cabo con estudiantes del octavo ciclo de la especialidad de Ingeniería Informática de la Pontificia Universidad Católica del Perú, específicamente del curso de Ingeniería del Software, en el cual se desarrolló un sistema de información para una empresa comercializadora del rubro ferretero (véase Anexo A) y en el cual se debía aplicar la técnica TDD dentro del enfoque Cascada.

Se tomó de muestra 4 grupos, de los cuales 2 de ellos aplicaron TDD dentro de su proceso de desarrollo de software tradicional y los otros 2 grupos no lo aplicaron. Al final, los 4 sistemas obtenidos fueron evaluados mediante un conjunto de pruebas que se enfocaron en las funcionalidades definidas en el Catálogo de Requisitos. Los resultados obtenidos permitieron medir la calidad del producto en función de la cantidad de errores detectados.

Finalmente, los datos estadísticos obtenidos brindaron una base para afirmar que la correcta aplicación de la técnica TDD fortalece la metodología tradicional aplicada (Cascada) y brinda, desde el inicio del proceso de desarrollo, un enfoque para la obtención y mejora de la calidad, logrando que los productos software creados bajo este enfoque sean considerados de mayor calidad.

1.2. Definición del problema

Debido a que el software ha sido parte de la sociedad moderna por más de 50 años, la elección de una metodología es un tema muy complejo, puesto que algunas organizaciones diseñan su propio proceso personalizado para el desarrollo de su software. A pesar de esta personalización la gran mayoría habla sobre las dos clases de metodologías existentes: tradicionales o pesadas y ágiles o ligeras [Gharaibeh, 2009].

1.2.1. El modelo tradicional en Cascada

Este modelo se muestra en el Gráfico 1.1. Debido a la forma de este modelo para pasar de una fase a otra, es que se le conoce como el modelo en Cascada. Las principales etapas de este modelo se transforman en actividades fundamentales de desarrollo [Sommerville, 2009]:

1. **Análisis y definición de requerimientos.** Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios, las cuales se definen en detalle y sirven como una especificación del sistema.
2. **Diseño del sistema y del software.** El proceso de diseño del sistema divide los requerimientos en sistemas hardware o software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.
3. **Implementación y prueba de unidades.** Durante esta etapa, el diseño del software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla con su especificación.
4. **Integración y prueba del sistema.** Los programas o las unidades individuales de programas se integran y se prueban como un sistema completo para asegurar que se cumplan los requerimientos del software. Después de las pruebas, el sistema software se entrega al cliente.
5. **Funcionamiento y mantenimiento.** Por lo general (aunque no necesariamente), ésta es la fase más larga del ciclo de vida. El sistema

se instala y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren nuevos requerimientos.

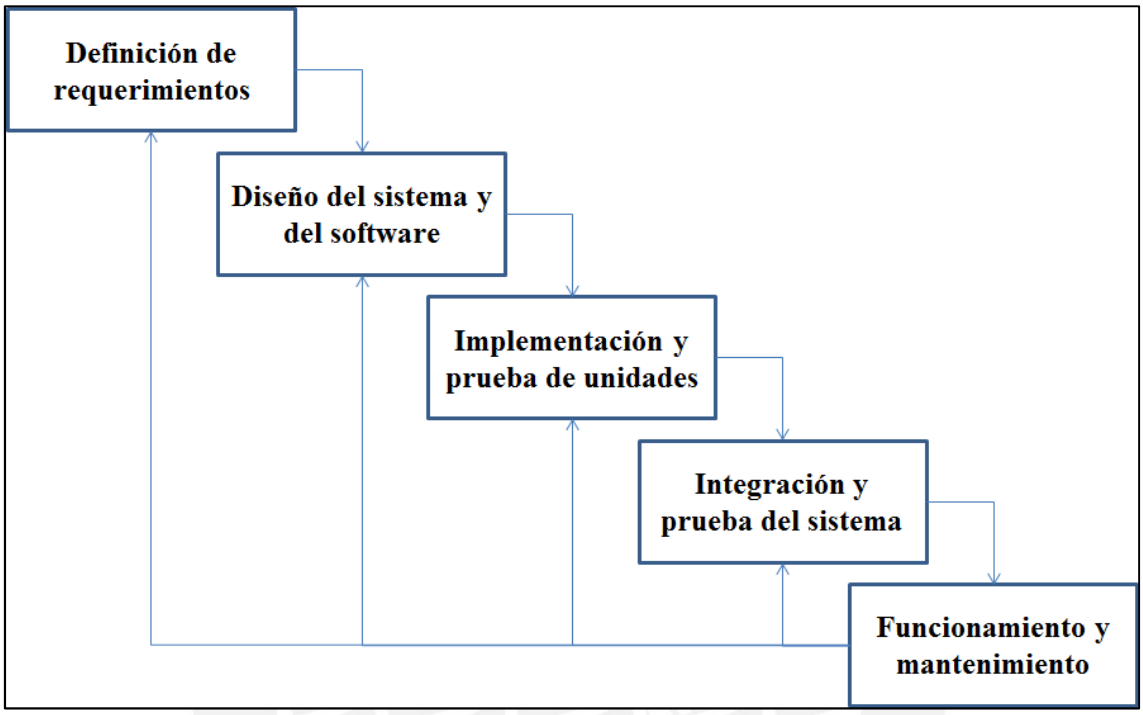

Gráfico 1.1: El modelo en Cascada [Sommerville, 2009]

En principio, el resultado de cada fase es uno o más documentos aprobados (<<firmados>>). La siguiente fase no debe empezar hasta que la fase previa haya finalizado. En la práctica, estas etapas se superponen y proporcionan información a las otras.

Procesos tradicionales (pesados) son especialmente efectivos para la aplicación en sistemas críticos debido a su proceso de desarrollo estructurado [Sommerville, 2009]. Documentación sólida y continua es de gran importancia para evitar malos entendidos entre los actores involucrados [Hildenbrand, 2008]. Sin embargo, la dificultad para modificar un programa operativo y el largo tiempo de lanzamiento de las iteraciones han llevado al desarrollo de metodologías ágiles (ligeras) que soporten el desarrollo de software rápido [Hildenbrand, 2008].

1.2.2. El desarrollo ágil

El desarrollo de software ágil se define como *"la capacidad del equipo de software para responder eficientemente y eficazmente a los cambios e incorporar los requerimientos del usuario durante el ciclo de vida del proyecto"* [Lee, 2010]. Además, se encarga de refinar y perfeccionar los procesos de desarrollo según sea necesario [Henderson, 2005].

La agilidad es más que una respuesta efectiva al cambio. También incluye la filosofía del manifiesto ágil. Estimula las estructuras y actitudes de los equipos para que la comunicación sea más fácil. Resalta la entrega rápida del software operativo y le resta importancia a los productos de trabajo intermedio (lo cual no siempre es bueno); adopta al cliente como una parte del equipo de desarrollo y trabaja para eliminar la actitud del tipo "nosotros y ustedes" que aún perjudica a muchos proyectos de software. Reconoce que la planificación tiene sus límites en un mundo incierto y que el plan de proyecto debe ser flexible [Pressman, 2002].

En la Tabla 1.1 se pueden visualizar los 12 principios que rigen al manifiesto ágil.

#	Principios
1	**Satisfacción del cliente** – "Nuestra mayor prioridad es satisfacer al cliente a través de la entrega temprana y continua de software valioso".
2	**Aceptar el cambio** – "Bienvenidos los cambios en los requerimientos, incluso tarde en el desarrollo. Los procesos ágiles aprovechan el cambio para obtener ventajas competitivas del cliente".
3	**Entregas frecuentes** – "Entregar frecuentemente software que funciona, a partir de un par de semanas a un par de meses, con preferencia a la escala de tiempo más corta".
4	**Trabajar juntos** – "La gente de negocios y los desarrolladores deben trabajar juntos diariamente durante todo el proyecto".
5	**Individuos motivados** – "Construir proyectos alrededor de individuos motivados. Darles el medio ambiente y el apoyo que

	necesitan, y confiar en ellos para hacer el trabajo".
6	**Conversación cara a cara** – "El método más eficiente y eficaz de transmitir información hacia y dentro de un equipo de desarrollo es la conversación cara a cara".
7	**Software funcionando** – "El software funcionando es la principal medida de progreso".
8	**Desarrollo Sostenible** – "Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida".
9	**Excelencia técnica** – "La atención continua a la excelencia técnica y el buen diseño mejora la agilidad".
10	**Simplicidad** – "El arte de maximizar la cantidad de trabajo que no se hace es esencial".
11	**Equipos auto organizados** – "Las mejores arquitecturas, requisitos y diseños emergen de equipos auto organizados".
12	**Auto reflexión** – "A intervalos regulares, el equipo reflexiona sobre cómo ser más eficaz, en consecuencia enfoca y ajusta su comportamiento".

Tabla 1.1: Principios del manifiesto ágil [Blom, 2012]

Las metodologías ágiles (ligeras) han surgido como una de las implementaciones más eficientes dentro del campo del desarrollo de software. Como un ejemplo de estas metodologías tenemos a *Extreme Programming (XP)*, la cual es una metodología que se utiliza para reducir el tiempo de salida al mercado de nuevos productos, y con ella la emergencia de ***Test Driven Development (TDD)***, en donde primero se crea la prueba antes de comenzar con la codificación [Karamat, 2006], lo cual permite el desarrollo de programas más robustos [Ficco, 2011]. De manera general, TDD permite producir software de manera iterativa e incremental, y con un estricto control sobre el proceso, favoreciendo la detección temprana de errores y mejorando el diseño [Ficco, 2011].

1.2.3. Test Driven Development (TDD)

Es una técnica para el desarrollo de software incremental, que se ha utilizado esporádicamente durante décadas, y que volvió a surgir en los últimos años como paradigma de desarrollo en el contexto de las denominadas metodologías ágiles. En TDD el desarrollador escribe las pruebas unitarias de un conjunto de requisitos de usuario antes de escribir el código. Se ejecutan las pruebas unitarias para verificar que éstas fallan. A continuación se implementa el código necesario hasta conseguir superar las pruebas y cuando se detecta un error se trata con prontitud hasta que sea superado. Luego de ello, el desarrollador lleva a cabo la refactorización del código para llevarlo a estándares aceptables, a continuación se repite el ciclo, procediendo a definir y diseñar un nuevo conjunto de casos de prueba para otras funcionalidades [Ficco, 2011]. Ver el detalle en el Gráfico 1.2.

El objetivo del proceso de diseño de casos de prueba es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y, muestren que el sistema satisface sus requerimientos corroborando que las salidas reales y esperadas son las mismas [Sommerville, 2009].

La refactorización es una técnica de la Ingeniería de Software que sirve para reestructurar un código fuente, trata de mejorar el diseño de código existente alterando su estructura interna sin cambiar su comportamiento externo. Es el proceso de cambio de un sistema software de tal manera que no se altera el comportamiento externo del código, sin embargo mejora su estructura interna. La refactorización se realiza a menudo como parte del proceso de desarrollo del software, los desarrolladores alternan la inserción de nuevas funcionalidades y casos de prueba con la refactorización del código para mejorar su consistencia interna y su claridad. La refactorización es la parte del mantenimiento del código que no arregla errores ni añade funcionalidad. El objetivo, por el contrario, es mejorar la facilidad de comprensión del código o cambiar su estructura y diseño y eliminar el código innecesario para facilitar el mantenimiento en el futuro.

Añadir nuevo comportamiento a un programa puede ser difícil con la estructura dada del programa, así que un desarrollador puede realizar primero la

refactorización (clean code) para facilitar esta tarea y luego añadir el nuevo comportamiento [Beck, 1999].

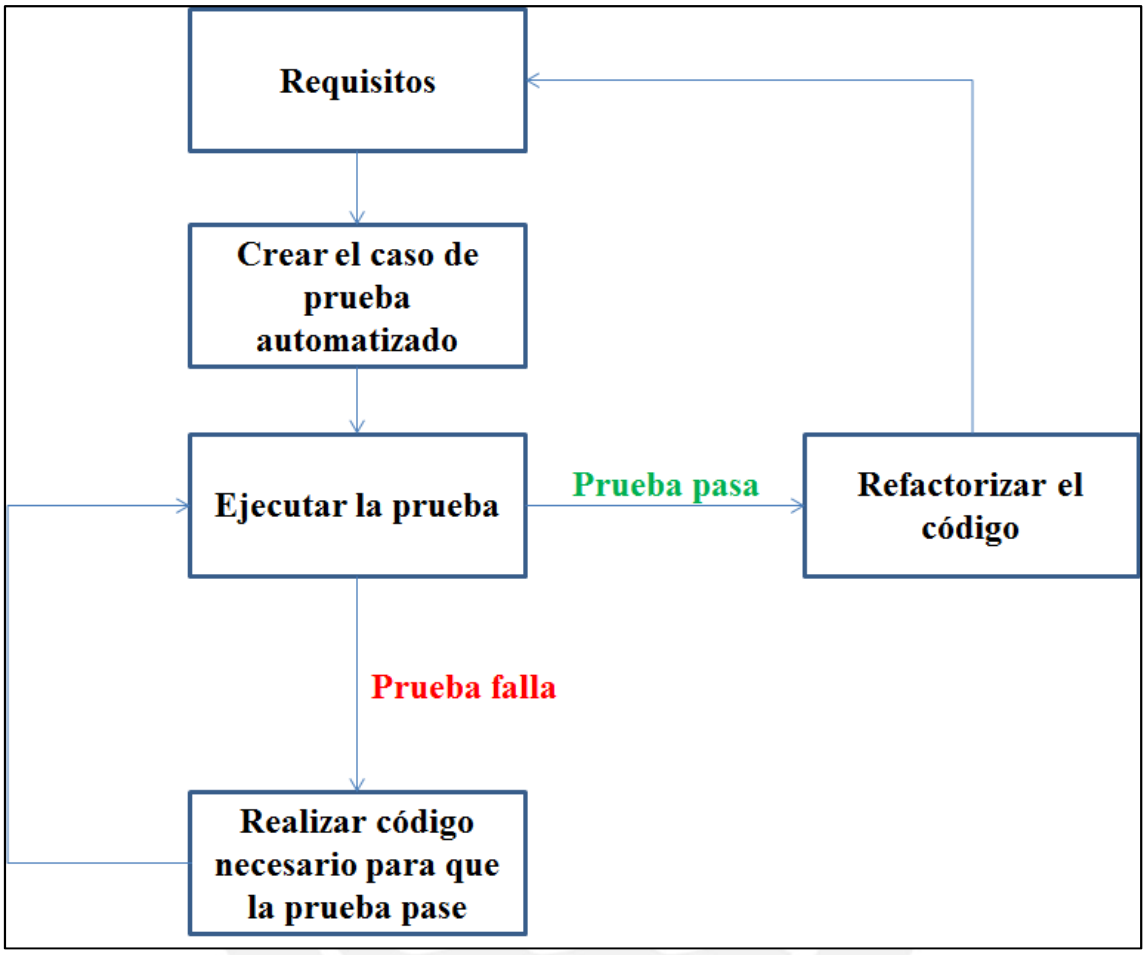

Gráfico 1.2: Principales pasos de la técnica TDD [Karamat, 2006]

Los pasos se describen a continuación:

1. De la lista de requisitos, obtenida de la interacción con el cliente, se selecciona el requisito a implementar.
2. El desarrollador procede a crear un caso de prueba automatizado que define la funcionalidad del requisito a implementar.
3. El desarrollar ejecuta la prueba para verificar que la prueba falla. Hasta este momento solo se tiene definida la prueba y no la implementación (código) de la funcionalidad.
4. A continuación, escribe el código necesario para lograr que la prueba sea exitosa. Es decir, se implementa el requisito.

5. El desarrollador ejecuta la prueba. Si el resultado de la prueba es incorrecto, vuelve a refinar el código hasta que la prueba sea exitoso. Si el resultado de la prueba es positivo, el desarrollador realiza la refactorización del código que implementa el requisito.
6. Se repiten los pasos con nuevas funcionalidades (requisitos) a implementar.
7. El desarrollador realiza repeticiones periódicamente en la ejecución de todos los casos de prueba creados, para asegurarse de que el nuevo código introducido no ha causado fallos en el código anterior (pruebas de regresión).

Finalmente, es importante resaltar que la calidad se puede ver comprometida en aquellos productos, en donde las pruebas se llevan a cabo al final del proceso, es por ello, que la obtención de una metodología tradicional fortalecida tiene como finalidad incorporar la técnica de TDD en etapas iniciales del desarrollo, para así favorecer la detección temprana de errores, logrando producir software más robusto y por consiguiente más confiable para los clientes.

Cabe precisar que por lo revisado en el estado del arte (véase Anexo C), se aprecia que las metodologías ágiles se han convertido en una de las implementaciones más eficientes en el mundo de la Industria del Software y, los equipos de desarrollo que utilizan metodologías ágiles están adoptando cada vez más la técnica TDD. Bajo este enfoque es que el presente estudio se concentra en el ámbito de la técnica TDD en donde se han realizado diversos estudios bajo los siguientes enfoques:

- Localización de errores.
- Reducción del costo de las pruebas.
- Aplicación de TDD en sistemas grandes y complejos, bases de datos, interfaces de usuarios y sistemas distribuidos.
- Comparación de la técnica TDD y Cascada para elaboración de programas.

Por todo lo mencionado anteriormente es que surge la siguiente pregunta que direcciona el presente trabajo de investigación: **¿Cómo se puede aplicar *Test Driven Development (TDD)* en el modelo Cascada a fin de mejorar la calidad del software final, medido en función de la cantidad de errores detectados?**

Para brindar respuesta a la interrogante anterior, es que en el presente trabajo de investigación se realiza una evaluación experimental al proceso de desarrollo del software, en donde, la etapa inicial de la metodología Cascada (Definición de requerimientos y Diseño del sistema) sirve de base para poder incluir la aplicación de la técnica TDD y así definir los casos de prueba a realizar y, a través de ellos, se realice la implementación del código fuente. Es decir, se incluye TDD desde las primeras fases del modelo Cascada.

Mediante la aplicación de este experimento se logró obtener resultados estadísticos iniciales que confirman que la ejecución de la técnica TDD en el proceso de implementación y pruebas unitarias permite identificar una mayor cantidad de errores, lo cual se ve reflejado al final del proceso en un producto de mayor calidad.

1.3. Objetivo general

Aplicar la técnica TDD en un proceso de desarrollo de software que sigue una Metodología Tradicional (modelo en Cascada), a fin de medir la calidad del producto obtenida, en función de la cantidad de errores detectados.

1.4. Objetivos específicos

A continuación se describen los objetivos específicos que contribuyen al cumplimiento del objetivo general:

- **OE1:** Analizar las fases de desarrollo de la metodología Cascada y realizar el proceso de inclusión de la técnica TDD.

- **OE2:** Analizar el conjunto de casos de prueba definidos para el sistema.

- **OE3:** Planificar y formular el diseño experimental sobre el cual se basará la presente investigación.

- **OE4:** Preparar y realizar una encuesta en función de las variables basadas en la percepción/intención de uso de la técnica TDD vs Cascada. Las variables a medir son las siguientes:
  - ✓ Facilidad de Uso Percibida *(Perceived Ease Of Use – PEOU)*
  - ✓ Utilidad Percibida *(Perceived Usefulness – PU)*
  - ✓ Intención de Uso *(Intention To Use – ITU)*

1.5. Resultados esperados

Los resultados esperados del presente trabajo de investigación se detallan en la Tabla 1.2:

Objetivo específico	Resultado esperado
OE1	**RE1:** Fortalecimiento de la metodología tradicional Cascada (con TDD), enfocándose, desde el inicio del desarrollo del producto, en las pruebas de software.
	RE2: Obtención de un proceso de desarrollo que brinde mayor probabilidad de éxito en cuanto a la calidad del producto desarrollado.
OE2	**RE3:** Obtención de datos que permitan evaluar experimentalmente el sistema implementado.
	RE4: Comparación de ambas metodologías de desarrollo (Cascada sin TDD vs Cascada con TDD) en función de los datos obtenidos de la ejecución de los casos de prueba.
OE3	**RE5:** Obtención de un proceso experimental que permita comparar estadísticamente ambas metodologías de la presente investigación.
	RE6: Obtención de una base inicial, mediante la ejecución del experimento, que permita argumentar que al comparar las dos metodologías en estudio, una de ellas tiene mayor probabilidad de brindar un producto de mejor calidad.
	RE7: Obtención y análisis de los resultados de las encuestas que permitan brindar una base fundamentada

OE4	sobre la percepción/intención de uso de la técnica TDD en comparación con Cascada.
	RE8: Generalizar la percepción/intención de uso de la técnica TDD en función de las 3 variables medidas: PEOU, PU e ITU.

Tabla 1.2: Resultados esperados

1.6. Justificación

El desarrollo de la presente investigación permitirá:

- Demostrar que la técnica TDD permite realizar acciones preventivas contra el alto costo de desarrollo que se puede incurrir en un proyecto de desarrollo de software que sigue una metodología tradicional, en donde las pruebas se lleven a cabo al final del proceso y en la cual se puede ver comprometida la calidad del producto.
- Brindar a los programadores de software resultados reales que permita incrementar su apreciación sobre la calidad de un producto software, en donde primero se deba pensar en probar. Es decir, entender que, aprender a "primero probar" puede ayudar a comprender verdaderamente la diferencia entre un buen software y un mal software [Doyle, 2011].
- Mejorar la calidad del software haciendo uso de la técnica de Test Driven Development [Karamat, 2006].
- Lograr producir software con un estricto control sobre el proceso, para mejorar así la capacidad de los desarrolladores para localizar errores tan pronto como sea posible [Ficco, 2011].
- Mejorar el análisis de los requisitos y el diseño del software.
- Aportar nuevas exploraciones en el problema de la calidad del software debido a la falta de pruebas o que éstas sean llevadas en etapas finales del desarrollo, lo cual aumenta, en líneas generales, el costo del producto y del proyecto.

1.7. Hipótesis

El paradigma de investigación seleccionado para el desarrollo del presente trabajo es el paradigma cuantitativo, el cual se apoya en el proceso de investigación de un problema basado en la prueba de una teoría compuesta por variables, medida con números y analizada con procedimientos estadísticos, y su objetivo es desarrollar generalizaciones que contribuyan con el desarrollo teórico y que posibiliten una mejor explicación, comprensión y previsión de un fenómeno [Creswell, 2009].

A continuación se detallan las hipótesis evaluadas en el presente trabajo de investigación:

1. **HCALIDAD:** Un modelo Cascada que aplica la técnica TDD brinda una mejor calidad del software, en función de la cantidad de errores detectados, que un modelo Cascada que no aplica la técnica TDD.
2. **HPEOU:** La técnica TDD se percibe como más fácil de usar que la técnica de Cascada.
3. **HPU:** La técnica TDD se percibe como más útil que la técnica de Cascada.
4. **HITU:** La técnica TDD tiene mejor intención de ser utilizado que la técnica de Cascada.

La primera hipótesis tiene por finalidad realizar la comparación del modelo Cascada que aplica TDD versus un modelo Cascada que no aplica la técnica TDD. Esto con la finalidad de poder medir la calidad de un software en función de la cantidad de errores detectados.

Las siguientes 3 hipótesis tienen por finalidad medir, de manera independiente, la percepción/intención de uso de la técnica TDD versus la técnica Cascada.

1.8. Límites del trabajo de investigación

Es importante señalar que desde una perspectiva práctica, la presente investigación proporciona los resultados iniciales de la evaluación, y debe manejarse en futuros experimentos con participantes expertos a fin de

confirmar los resultados preliminares que se obtengan. Por consiguiente la investigación tiene valor como un piloto para fijar el procedimiento experimental, antes de tratar con profesionales desarrolladores de software.

Teniendo como idea lo antes mencionado, se puede indicar que la principal limitación del estudio es el uso de los sujetos participantes, ya que éstos son estudiantes y sin experiencia previa en el uso de la técnica TDD. Sin embargo, el uso de estos participantes no representa una amenaza para la validez interna del estudio, puesto que son estudiantes próximos a culminar su carrera universitaria, y la mayoría de ellos ya se encuentra realizando prácticas pre-profesionales, lo cual puede ser considerado como estar cerca de la población real (profesionales desarrolladores de software). Sin embargo, el uso de estos participantes reduce la validez externa del estudio y por lo tanto no permite generalizar los resultados. Por lo tanto, el estudio necesitará ser repetido en un futuro y con profesionales desarrolladores de software con experiencia en el uso de la técnica TDD y así lograr incrementar la validez externa del presente trabajo de investigación.

Una limitación adicional a considerar es el tamaño de la muestra, ya que es relativamente baja. Se tomó de muestra 4 grupos conformados por 11 estudiantes cada uno. Un número mayor del tamaño de la muestra puede proporcionar datos estadísticos más precisos.

Capítulo 2: Método y procedimiento

Para la presente investigación, se optó por un método deductivo, el cual se caracteriza por llegar a una conclusión a partir de una situación general y genérica, en donde se parte de una teoría, que define las relaciones entre conceptos dentro de un conjunto de suposiciones y restricciones fijadas, para formular hipótesis con el objetivo de confirmar la teoría [Silva, 2005].

2.1. Enfoque de la investigación

Analizar los resultados obtenidos en la implementación de un sistema de información para una empresa comercializadora del rubro ferretero que se desarrolló bajo el enfoque de un proceso tradicional de Cascada, con el propósito de evaluar cuál de las implementaciones (aplicando o no la técnica TDD en un modelo Cascada) brinda mejores resultados en cuanto a la calidad del producto software obtenido, en función de la cantidad de errores detectados.

2.2. Unidades experimentales

Los sujetos seleccionados, sobre los cuales se realizaron las mediciones y se asignaron los tratamientos, fueron estudiantes de la especialidad de Ingeniería Informática de la Facultad de Ciencias e Ingeniería de la Pontificia Universidad Católica del Perú, con conocimiento similar en técnicas y metodologías de desarrollo de software. Los estudiantes tenían edades comprendidas entre 20 y 24 años y estuvieron matriculados en el curso Ingeniería de Software, cursado desde agosto hasta diciembre del 2013. La implementación de la técnica TDD dentro del modelo Cascada fue organizado como parte obligatoria del curso y en la cual 2 de los 4 grupos formados tenían que aplicar la técnica mencionada como parte de la implementación de su sistema de información. Cada grupo estuvo conformado por 11 estudiantes.

2.3. Material experimental

El material experimental que sirvió de base para el desarrollo de la presente investigación se lista a continuación:

- Clases teóricas semanales sobre procesos de construcción del software y pruebas de software a cargo del docente del curso de Ingeniería de Software.
- Clase teórica expositiva sobre la técnica TDD a cargo de un especialista.
- Dos (2) laboratorios de la especialidad de Ingeniería Informática con 25 computadoras cada uno y totalmente equipados con lo necesario en cuanto al hardware y software requeridos para llevar a cabo el proceso de desarrollo del sistema solicitado, Adicionalmente, en estos laboratorios se llevaron a cabo las exposiciones que sirvieron para medir el avance de los proyectos de cada grupo.
- Reuniones semanales de 2 horas en los laboratorios de Informática para la medición gradual de sus avances y asesorías respectivas a cada uno de los grupos. Cada grupo estuvo asignado a un asesor especializado para brindar el soporte y revisión necesarios.
- Planillas electrónicas que sirvieron de base para que los grupos puedan definir sus casos de prueba en base a la lista de requisitos definidos que se debían cumplir dentro del sistema solicitado.
- Encuesta compuesta por 12 preguntas, que se utilizó para medir la percepción/intención de uso de los participantes respecto a las técnicas TDD y Cascada.

2.4. Tareas

En el presente punto se detallan las tareas que tuvieron que realizar los sujetos que formaron parte del presente estudio:

1. Asistir a las clases teóricas semanales sobre procesos de construcción del software como parte del curso Ingeniería del Software.
2. Asistir a la capacitación teórica sobre los conceptos básicos de la técnica TDD.
3. Revisar el informe entregado con el tema elegido (sistema de información del rubro ferretero), en donde se detallaban los requisitos que los grupos debían considerar al momento de implementar su sistema, teniendo en cuenta el cronograma de presentaciones definido.

4. Presentar semanalmente el avance de implementación del sistema solicitado.
5. Recibir las pautas, guías y retroalimentación por parte de los asesores asignados para que implementen de la mejor manera el sistema solicitado.
6. Responder la encuesta que sirvió para medir las variables PEOU, PU e ITU del presente trabajo de investigación.

2.5. Recolección de datos

Los instrumentos de recolección de datos utilizados que sirvieron de soporte al procedimiento realizado en el presente estudio, fueron los siguientes:

- **Observación:** Para controlar el avance de cada uno de los grupos y la forma de cómo implementan el proceso de desarrollo de software incluyendo o no la técnica TDD dentro del modelo Cascada.
- **Entrevistas:** Para controlar el avance de cada uno de los grupos y entrevistar de manera abierta a cada uno de los integrantes de cada grupo y así obtener información del proceso de desarrollo del sistema solicitado.
- **Planillas electrónicas:** Que sirvieron para evaluar y controlar la cantidad de errores detectados en cada presentación del avance realizado.
- **Encuesta:** Llevada a cabo previa a la presentación final de los trabajos realizados. Los integrantes de cada grupo brindaron sus experiencias y las compartieron a través de un cuestionario anónimo que permitió evaluar la percepción/intención de aplicar la técnica TDD en comparación con Cascada. Cada sujeto completó la misma encuesta tanto para TDD como para Cascada. Al finalizar la encuesta se verificó que se haya completado en su totalidad.

2.6. Preparación del experimento

Como preparación del experimento, participaron 44 sujetos los cuales fueron divididos aleatoriamente en cuatro (4) grupos experimentales, cada

grupo estuvo conformado por 11 sujetos, los cuales fueron escogidos al azar para la conformación de cada grupo.

A los participantes del experimento se les procedió a capacitar en el modelo Cascada mediante clases expositivas y asesorías permanentes durante todo el ciclo de estudios (aproximadamente 4 meses), tiempo establecido para la implementación del sistema de información solicitado. El objetivo fue brindar una capacitación equitativa a todos los participantes. A partir de la mitad del ciclo hacia el final se brindó una capacitación constante y asesoría a los grupos encargados de aplicar la técnica TDD dentro del modelo Cascada, esto debido a que de la mitad del ciclo hacia adelante se inició la etapa de construcción.

Considerando el tiempo de capacitación y de asesoría permanente se puede afirmar que los participantes tuvieron el tiempo suficiente para poder entender y aplicar adecuadamente los conceptos y procedimientos del modelo Cascada y de la técnica TDD dentro de su proceso de desarrollo.

2.7. Diseño e instrumentos

Con el fin de equilibrar las diferencias en cuanto a la capacidad entre los participantes, así como para la obtención de los datos, el presente experimento utilizó un diseño intra-sujeto. Los 44 estudiantes fueron aleatoriamente asignados en 4 grupos, de los cuales, 2 grupos aplicaron dentro de su modelo Cascada la técnica TDD para la implementación del sistema de información del rubro ferretero y, los otros 2 grupos no la aplicaron.

Para la primera hipótesis definida (hipótesis de variable de calidad), se evaluaron a los 4 grupos implicados en la presente investigación, debido a que se compararía entre los 4 grupos, cuál de ellos obtenía mejores resultados en cuanto a calidad del producto software obtenido.

Para las siguientes 3 hipótesis definidas (hipótesis de variables de percepción/intención), se evaluaron solo a los 2 grupos que aplicaron TDD dentro de su enfoque Cascada, puesto que éstos realizaron la implementación de su código utilizando tanto TDD como Cascada, motivo por el cual, a través de ellos, se pudo realizar la comparación respectiva, en cuanto a la

percepción/intención de uso de ambas técnicas a través de la encuesta definida.

El instrumento inicial que se utilizó en el experimento es el documento de Catálogo de Requisitos de Software, en el cual se refleja la funcionalidad deseada por el cliente y es la base inicial para TDD, a partir de él se definen los Test Cases (Casos de Prueba) a crear. Dicho documento se elaboró en base al informe que se entregó a los alumnos con el tema y cronograma definidos en donde se describieron los principales requisitos a considerar para el sistema de información de rubro ferretero.

Los resultados obtenidos con ambos procedimientos se pudieron comparar a medida que recibieron similares especificaciones como entrada. Por otro lado, a los participantes del experimento se les procedió a capacitar en el modelo Cascada y en la técnica TDD mediante clases expositivas y asesorías permanentes.

Por lo tanto, desde el punto de vista de validez interna del estudio, más importante que la duración de la capacitación es que todos los participantes fueron entrenados por igual en el modelo Cascada y en la técnica TDD, de esa manera este punto fue controlado.

Finalmente como instrumento experimental se adaptó una encuesta a manera de cuestionario para evaluar las variables de la presente investigación. Este cuestionario incluyó 12 preguntas de tipo cerradas valorativas. La escala de valoración utilizada fue la escala de Likert de 5 puntos. Cada pregunta fue formulada en formato “Positivo - Negativo”. Sin embargo, algunas preguntas fueron iniciadas en formato opuesto “Negativo - Positivo”. Dichas preguntas fueron ubicadas al azar para evitar respuestas monótonas en los sujetos (véase Anexo B).

2.8. Variables

La variable independiente (factor) es la aplicación de la técnica TDD o no en un modelo Cascada dentro un sistema de información del rubro ferretero. Por lo tanto, el experimento empleó dos tratamientos o niveles para esta variable independiente:

a. La medición de la calidad obtenida sin la aplicación de la técnica TDD dentro del modelo Cascada.
b. La medición de la calidad obtenida con la aplicación de la técnica TDD dentro del modelo Cascada.

Los datos experimentales recogidos permitieron comparar los efectos de ambos tratamientos. Existen dos tipos de variables dependientes que sirven para comparar los tratamientos (véase Apéndice A):

1. Basadas en el rendimiento (calidad del software medido en función de la cantidad de errores detectados), las cuales se utilizan para evaluar la eficacia real de la implementación de la técnica TDD o no dentro del modelo Cascada.
2. Basadas en la percepción/intención (comparación de TDD vs Cascada), las cuales permiten evaluar la percepción de los participantes sobre el rendimiento y su posterior intención de utilizar la técnica TDD. Estas variables se utilizan para evaluar la eficacia percibida de los procedimientos de medición, así como su posible adopción en la práctica. Las variables basadas en la percepción/intención que se utilizaron fueron las siguientes:

    - Facilidad de Uso Percibida *(Perceived Ease Of Use – PEOU)*
    - Utilidad Percibida *(Perceived Usefulness – PU)*
    - Intención de Uso *(Intention To Use – ITU)*

2.9. Preguntas de investigación

Las preguntas que direccionan el presente trabajo de investigación son las siguientes:

a. ¿Es la eficacia actual (medida en función de calidad del software) del modelo Cascada aplicando TDD más alta que la eficacia actual del modelo Cascada que no aplica TDD?
b. ¿Es la intención de uso y la eficacia percibida de TDD más alta que la intención de uso y la eficacia percibida de Cascada?

Capítulo 3: Diseño experimental

A continuación se muestra el diseño experimental realizado para el tratamiento de las hipótesis planteadas. Con los resultados obtenidos se pudo definir la aceptación o rechazo de cada una de las variables implicadas. El análisis de los datos se llevó a cabo con la herramienta *IBM SPSS Statistics 21.*

3.1. Procedimiento

A continuación se detalla el procedimiento seguido y la realización de las pruebas estadísticas que se utilizaron para analizar los datos:

a. Se procede a plantear las hipótesis estadísticas para cada una de las variables (calidad, PEOU, PU, ITU).

b. Con los mecanismos definidos para la recolección de datos, se obtienen los datos de las variables.

c. Para cada variable se procede a calcular los descriptivos (media, desviación estándar, valor mínimo y valor máximo).

d. Posteriormente, se procede a calcular la diferencia de los datos obtenidos respecto a los dos tratamientos, con esa nueva variable se procede a determinar la normalidad.

e. Como el número de datos de la muestra es menor que 30, entonces se usa como prueba de Normalidad, la prueba de Shapiro-Wilk a un nivel de significación de $\alpha = 0.05$.

f. Si la diferencia proviene de una distribución normal se procede a realizar una prueba paramétrica (prueba T) de lo contrario se realiza una prueba no paramétrica (Wilcoxon).

g. Del punto anterior, se obtiene el valor del Sig. (bilateral), y se procede a realizar un contraste unilateral para obtener el p-valor. El p-valor = Sig. (bilateral)/2.

h. Se procede a comparar el p-valor con el nivel de significación para decidir la aceptación o no de la hipótesis nula.

   - Si p-valor > α: Se acepta la hipótesis nula
   - Si p-valor < α: Se rechaza la hipótesis nula

i. Con los resultados obtenidos, se procede a aceptar o rechazar las hipótesis planteadas.

3.2. Ejecución

En esta sección se brindan los detalles de la ejecución experimental realizada para cada una de las variables definidas.

3.2.1. Variable Calidad (en función de la cantidad de errores detectados)

Hipótesis:

- **HCALIDAD:** Un modelo Cascada que aplica la técnica TDD brinda una mejor calidad del software, en función de la cantidad de errores detectados, que un modelo Cascada que no aplica la técnica TDD.

Donde:

- PromedioConTDD: Calidad en el modelo Cascada que aplica TDD.
- PromedioSinTDD: Calidad en el modelo Cascada que no aplica TDD.

Definición:

H0: PromedioConTDD = PromedioSinTDD

Vs.

H1: PromedioConTDD > PromedioSinTDD

Descriptivos:

Descriptivos

			Estadístico	Error típ.
PromedioConTDD	Media		17.2500	3.55844
	Intervalo de confianza para la media al 95%	Límite inferior	8.1027	
		Límite superior	26.3973	
	Media recortada al 5%		17.5556	
	Mediana		18.2500	
	Varianza		75.975	
	Desv. típ.		8.71636	
	Mínimo		1.50	
	Máximo		27.50	
	Rango		26.00	
	Amplitud intercuartil		10.25	
	Asimetría		-1.238	.845
	Curtosis		2.598	1.741
PromedioSinTDD	Media		13.0833	3.40934
	Intervalo de confianza para la media al 95%	Límite inferior	4.3193	
		Límite superior	21.8473	
	Media recortada al 5%		13.0648	
	Mediana		13.0000	
	Varianza		69.742	
	Desv. típ.		8.35115	
	Mínimo		1.00	
	Máximo		25.50	
	Rango		24.50	
	Amplitud intercuartil		12.88	
	Asimetría		.071	.845
	Curtosis		.362	1.741

Cuadro 3.1: Descriptivos para la variable Calidad

Estadísticos descriptivos

	N	Mínimo	Máximo	Media	Desv. típ.
PromedioConTDD	6	1.50	27.50	17.2500	8.71636
PromedioSinTDD	6	1.00	25.50	13.0833	8.35115
N válido (según lista)	6				

Cuadro 3.2: Resumen de descriptivos para la variable Calidad

Normalidad:

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Diferencia	.326	6	.046	.831	6	.110

a. Corrección de la significación de Lilliefors

Cuadro 3.3: Pruebas de normalidad para la variable Calidad

Como el nivel de significación obtenido es 0.110 y este valor es mayor a 0.05, la condición (0.110 > 0.05) es VERDADERA, entonces podemos afirmar que la Diferencia (PromedioConTDD - PromedioSinTDD) sí proviene de una distribución normal. Por lo tanto, se realiza una prueba paramétrica.

Prueba Paramétrica (Prueba T):

Prueba de muestras relacionadas

		Diferencias relacionadas					t	gl	Sig. (bilateral)
		Media	Desviación típ.	Error típ. de la media	95% Intervalo de confianza para la diferencia				
					Inferior	Superior			
Par 1	PromedioConTDD - PromedioSinTDD	4,1667	4.07022	1.66166	-.10476	8.43810	2.508	5	.054

Cuadro 3.4: Prueba paramétrica para la variable Calidad

Cálculos realizados:

1. Sig. (bilateral) = 0.054
2. p-valor = 0.054/2 = 0.027
3. Se obtiene que p-valor < α, ya que (0.027<0.05)
4. Por lo tanto, se rechaza la hipótesis nula.

En conclusión:

- Se comprueba empíricamente que *"Un modelo Cascada que sí aplica la técnica TDD otorga mejores resultados, en cuanto a la calidad de software obtenida, que un modelo de Cascada que no aplica la técnica TDD".*

3.2.2. Variable Facilidad de Uso Percibida (Perceived Ease Of Use - PEOU)

Hipótesis:

- **HPEOU:** La técnica TDD se percibe como más fácil de usar que la técnica de Cascada.

Donde:

- PEOU_TDD: Facilidad de Uso Percibida de TDD.
- PEOU_CASCADA: Facilidad de Uso Percibida de Cascada.

Definición:

H0: PEOU_TDD = PEOU_CASCADA

Vs.

H1: PEOU_TDD > PEOU_CASCADA

Descriptivos:

Descriptivos

			Estadístico	Error típ.
PEOU_TDD	Media		3.6636	.10802
	Intervalo de confianza para la media al 95%	Límite inferior	3.4390	
		Límite superior	3.8883	
	Media recortada al 5%		3.6495	
	Mediana		3.6000	
	Varianza		.257	
	Desv. típ.		.50667	
	Mínimo		3.00	
	Máximo		4.60	
	Rango		1.60	
	Amplitud intercuartil		1.00	
	Asimetría		.321	.491
	Curtosis		-1.172	.953
PEOU_CASCADA	Media		3.3364	.10141
	Intervalo de confianza para la media al 95%	Límite inferior	3.1255	
		Límite superior	3.5473	
	Media recortada al 5%		3.3505	
	Mediana		3.4000	
	Varianza		.226	
	Desv. típ.		.47564	
	Mínimo		2.40	
	Máximo		4.00	
	Rango		1.60	
	Amplitud intercuartil		.65	
	Asimetría		-.354	.491
	Curtosis		-.784	.953

Cuadro 3.5: Descriptivos para la variable PEOU

Estadísticos descriptivos

	N	Mínimo	Máximo	Media	Desv. típ.
PEOU_TDD	22	3.00	4.60	3.6636	.50667
PEOU_CASCADA	22	2.40	4.00	3.3364	.47564
N válido (según lista)	22				

Cuadro 3.6: Resumen de descriptivos para la variable PEOU

Normalidad:

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
DiferenciaPEOU	.158	22	.162	.965	22	.591

a. Corrección de la significación de Lilliefors

Cuadro 3.7: Pruebas de normalidad para la variable PEOU

Como el nivel de significación obtenido es 0.591 y este valor es mayor a 0.05, la condición (0.591 > 0.05) es VERDADERA, entonces podemos afirmar que la DiferenciaPEOU (PEOU_TDD - PEOU_CASCADA) sí proviene de una distribución normal. Por lo tanto, se realiza una prueba paramétrica.

Prueba Paramétrica (Prueba T):

Prueba de muestras relacionadas

		Diferencias relacionadas					t	gl	Sig. (bilateral)
		Media	Desviación típ.	Error típ. de la media	95% Intervalo de confianza para la diferencia Inferior	Superior			
Par 1	PEOU_TDD - PEOU_CASCADA	.32727	.61271	.13063	.05561	.59893	2.505	21	.021

Cuadro 3.8: Prueba paramétrica para la variable PEOU

Cálculos realizados:

1. Sig. (bilateral) = 0.021
2. p-valor = 0.021/2 = 0.0105
3. Se obtiene que p-valor < α, ya que (0.0105<0.05)
4. Por lo tanto, se rechaza la hipótesis nula.

En conclusión:

- Se comprueba empíricamente que *"TDD se percibe como más fácil de utilizar que Cascada".*

3.2.3. Variable Utilidad Percibida (Perceived Usefulness - PU)

Hipótesis:

- **HPU:** La técnica TDD se percibe como más útil que la técnica de Cascada.

Donde:

- PU_TDD: Utilidad Percibida de TDD.
- PU_CASCADA: Utilidad Percibida de Cascada.

Definición:

H0: PU_TDD = PU_CASCADA

Vs.

H1: PU_TDD > PU_CASCADA

Descriptivos:

Descriptivos

			Estadístico	Error típ.
PU_TDD	Media		3.1250	.06732
	Intervalo de confianza para la media al 95%	Límite inferior	2.9850	
		Límite superior	3.2650	
	Media recortada al 5%		3.1111	
	Mediana		3.0000	
	Varianza		.100	
	Desv. típ.		.31576	
	Mínimo		2.75	
	Máximo		3.75	
	Rango		1.00	
	Amplitud intercuartil		.50	
	Asimetría		.780	.491
	Curtosis		-.507	.953
PU_CASCADA	Media		2.8977	.08958
	Intervalo de confianza para la media al 95%	Límite inferior	2.7114	
		Límite superior	3.0840	
	Media recortada al 5%		2.8990	
	Mediana		2.7500	
	Varianza		.177	
	Desv. típ.		.42017	
	Mínimo		2.00	
	Máximo		3.75	
	Rango		1.75	
	Amplitud intercuartil		.75	
	Asimetría		.125	.491
	Curtosis		-.165	.953

Cuadro 3.9: Descriptivos para la variable PU

Estadísticos descriptivos

	N	Mínimo	Máximo	Media	Desv. típ.
PU_TDD	22	2.75	3.75	3.1250	.31576
PU_CASCADA	22	2.00	3.75	2.8977	.42017
N válido (según lista)	22				

Cuadro 3.10: Resumen de descriptivos para la variable PU

Normalidad:

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
DiferenciaPU	.249	22	.001	.869	22	.007

a. Corrección de la significación de Lilliefors

Cuadro 3.11: Pruebas de normalidad para la variable PU

Como el nivel de significación obtenido es 0.007 y este valor es menor a 0.05, la condición (0.007 > 0.05) es FALSA, entonces podemos afirmar que la DiferenciaPU (PU_TDD - PU_CASCADA) no proviene de una distribución normal. Por lo tanto, se realiza una prueba no paramétrica.

Prueba No Paramétrica (Prueba Wilcoxon):

Estadísticos de contraste[a]

	PU_CASCADA - PU_TDD
Z	-2.621[b]
Sig. asintót. (bilateral)	.009

a. Prueba de los rangos con signo de Wilcoxon

b. Basado en los rangos positivos.

Cuadro 3.12: Prueba no paramétrica para la variable PU

Cálculos realizados:

1. Sig. Asintót. (bilateral) = 0.009
2. p-valor = 0.009/2 = 0.0045
3. Se obtiene que p-valor < α, ya que (0.0045<0.05)
4. Por lo tanto, se rechaza la hipótesis nula.

En conclusión:

- Se comprueba empíricamente que *"TDD se percibe como más útil que Cascada".*

3.2.4. Variable Intención de Uso (Intention To Use - TU)

Hipótesis:

- **HITU:** La técnica TDD tiene mejor intención de ser utilizado que la técnica de Cascada.

Donde:

- ITU_TDD: Intención de Uso de TDD.
- ITU_CASCADA: Intención de Uso de Cascada.

Definición:

H0: ITU_TDD = ITU_CASCADA

Vs.

H1: ITU_TDD > ITU_CASCADA

Descriptivos:

Descriptivos

			Estadístico	Error típ.
ITU_TDD	Media		2.5000	.19230
	Intervalo de confianza para la media al 95%	Límite inferior	2.1001	
		Límite superior	2.8999	
	Media recortada al 5%		2.4983	
	Mediana		2.3300	
	Varianza		.814	
	Desv. típ.		.90197	
	Mínimo		1.00	
	Máximo		4.00	
	Rango		3.00	
	Amplitud intercuartil		1.50	
	Asimetría		.287	.491
	Curtosis		-.962	.953
ITU_CASCADA	Media		2.8186	.16521
	Intervalo de confianza para la media al 95%	Límite inferior	2.4751	
		Límite superior	3.1622	
	Media recortada al 5%		2.8185	
	Mediana		2.8350	
	Varianza		.600	
	Desv. típ.		.77490	
	Mínimo		1.67	
	Máximo		4.00	
	Rango		2.33	
	Amplitud intercuartil		1.67	
	Asimetría		-.083	.491
	Curtosis		-1.601	.953

Cuadro 3.13: Descriptivos para la variable ITU

Estadísticos descriptivos

	N	Mínimo	Máximo	Media	Desv. típ.
ITU_TDD	22	1.00	4.00	2.5000	.90197
ITU_CASCADA	22	1.67	4.00	2.8186	.77490
N válido (según lista)	22				

Cuadro 3.14: Resumen de descriptivos para la variable ITU

Normalidad:

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
DiferencialTU	.184	22	.051	.948	22	.287

a. Corrección de la significación de Lilliefors

Cuadro 3.15: Pruebas de normalidad para la variable ITU

Como el valor de significación obtenido es 0.287 y este valor es mayor a 0.05, la condición (0.287 > 0.05) es VERDADERA, entonces podemos afirmar que la DiferenciaITU (ITU_TDD - ITU_CASCADA) sí proviene de una distribución normal. Por lo tanto se realiza una prueba paramétrica.

Prueba Paramétrica (Prueba T):

Prueba de muestras relacionadas

		Diferencias relacionadas					t	gl	Sig. (bilateral)
		Media	Desviación típ.	Error típ. de la media	95% Intervalo de confianza para la diferencia Inferior	Superior			
Par 1	ITU_TDD - ITU_CASCADA	-.31864	1.11597	.23793	-.81343	.17616	-1.339	21	.195

Cuadro 3.16: Prueba paramétrica para la variable ITU

Cálculos realizados:

1. Sig. (bilateral) = 0.195
2. p-valor = 0.195/2 = 0.0975
3. Se obtiene que p-valor > α, ya que (0.0975>0.05)
4. Por lo tanto, se rechaza la hipótesis alternativa.

En conclusión:

- Se comprueba empíricamente que *"TDD no tiene mejor intención de ser utilizado que Cascada"*.

3.3. Observaciones y apreciaciones personales

La definición de los casos de prueba, previa a la etapa de implementación, fue un proceso en el cual los asesores asignados trabajaron mucho con los sujetos de cada grupo, se les brindó una planilla electrónica y la asesoría respectiva para que puedan plasmar en un documento el conjunto de casos de prueba que se encargue de corroborar el correcto funcionamiento del sistema solicitado.

De esta manera, desde un principio se indujo a los sujetos para que piensen primero en probar, es decir, conceptualizar sobre cómo debe funcionar el sistema, antes de empezar la codificación. Esto marcó el inicio del camino para la obtención de un proyecto de calidad. Una vez definidos los casos de prueba, se dio inicio al diseño de los casos de prueba automatizados utilizando la técnica TDD.

Como se recuerda, el presente estudio consistió en evaluar a 4 grupos con el objetivo de que cada uno pueda implementar un sistema de información del rubro ferretero y evaluar, al final del proceso de implementación, la cantidad de errores detectados en el producto realizado. 2 de estos 4 grupos aplicaron TDD y los otros dos no lo hicieron. Al final del proceso, de los 2 grupos asignados para aplicar TDD solo uno pudo aplicar la técnica correctamente, esto hizo que los resultados obtenidos al final del proceso sean muy alentadores para este grupo, puesto que durante la exposición final de su sistema, no tuvieron error alguno durante la demostración ni durante las pruebas exigidas por los especialistas en el rubro del sistema solicitado.

Los grupos que no aplicaron TDD dentro del modelo Cascada, tuvieron resultados muy similares, ambos grupos presentaron 1 solo error cada uno.

Por otro lado, el otro grupo que no aplicó correctamente TDD, puesto que tuvo varios errores en el diseño de sus casos de prueba automatizados y en la implementación de éstos, fue víctima de su proceso y por consiguiente los resultados no fueron alentadores. Este grupo presentó 3 errores críticos durante su exposición, siendo al final los que presentaron el sistema de menor calidad.

Si bien uno de los dos grupos que aplicó TDD no obtuvo buenos resultados, éstos no pueden ser atribuidos solamente a la técnica, sino también a fallas en su proceso, tales como la falta de coordinación en el avance progresivo, evidenciado en las presentaciones semanales, la falta de investigación y dedicación general para cumplir el trabajo u objetivo trazado cada semana. Todo esto, junto al esfuerzo adicional de aplicar una técnica nueva y el tiempo limitado, ocasionaron un producto de baja calidad.

La evaluación como técnica en sí, se pudo lograr gracias a las entrevistas personales planificadas cada semana. De esa manera, los asesores de cada grupo, podían evaluar el avance, comprensión e implementación de las funcionalidades y en general, de todo su sistema implementado.

Finalmente en la encuesta, se pudo reflejar y confirmar que a pesar, de que uno de los grupos no obtuvo buenos resultados, sus integrantes tenían la percepción de que la técnica TDD es fácil de implementar, sencilla de entender y aplicar. Esto refuerza la apreciación de que al final, no lograron conseguir el resultado esperado debido a fallas generales que tuvieron durante todo el proceso y no solo se debió a la aplicación de la técnica TDD.

Capítulo 4: Conclusiones y trabajos futuros

4.1. Conclusiones

TDD, de manera teórica, es una técnica sencilla de entender en cuanto a concepto y pasos que la constituyen, éstos son claros y comprensibles, lo que lo hace un proceso bien definido. Lo paradójico es que a la hora de llevarlo a la práctica, sobre todo en su etapa inicial, esto no es tan sencillo como parece, se pudo observar que la implementación de código basado en el diseño de los casos de prueba automatizados fue un proceso arduo y complicado para los sujetos, les tomó buen tiempo y esfuerzo llegar a utilizar la técnica de la manera correcta, aunque como en todo proceso, no todos lo consiguieron.

Con respecto a la percepción/intención que proyecta la técnica TDD, se concluye que en un primer momento la técnica TDD es complicada de utilizar para los desarrolladores que no tienen experiencia en su uso, pero con el avance de las semanas y por consiguiente del proceso de implementación, los sujetos cambian de opinión y comienzan a darse cuenta de las ventajas que ofrece esta técnica.

Si tenemos en consideración los resultados obtenidos, donde se tiene el enfoque de que la percepción de TDD es útil y fácil de usar, se esperaría que se tenga la intención de utilizar la técnica en futuro, pero paradójicamente los resultados obtenidos en la encuesta no corroboran lo esperado. Esto se entiende como que TDD, para los sujetos, aún se ve como un esfuerzo adicional que deben realizar dentro de su proceso de desarrollo de software y que con el modelo Cascada pueden también obtener muy buenos resultados en cuanto a la calidad de un producto software.

Por otro lado, gracias a la correcta aplicación de la técnica TDD dentro del modelo Cascada, se concluye que se obtuvo un proceso de desarrollo fortalecido que permite incrementar las garantías de que el producto software brinde estándares de calidad elevados. De no aplicarse adecuadamente TDD, los resultados obtenidos presentarán deficiencias que harán que el producto software que se obtenga sea de muy baja calidad.

Cabe resaltar que los ocho (8) resultados esperados definidos fueron evaluados y comparados con los resultados obtenidos de la ejecución del presente estudio. Esto a fin de verificar la relación existente entre resultado esperado y resultado obtenido y de esa manera no perder el enfoque de la presente investigación.

Finalmente, en base al cumplimiento de todos los objetivos específicos definidos en el presente trabajo de investigación, se concluye que se pudo conseguir el objetivo general planteado.

La calidad obtenida en el modelo Cascada, con y sin TDD, fue medida en función de la cantidad de errores detectados. TDD definitivamente agrega valor al proceso del modelo Cascada, y brinda el enfoque para que las personas desarrolladoras de software se preocupen más por conseguir productos de alta calidad.

4.2. Trabajos futuros

Se propone realizar el mismo estudio experimental con sujetos profesionales desarrolladores de software y con experiencia en el uso de la técnica TDD. Esto permitirá obtener resultados más representativos e incrementar la validez externa de la presente investigación realizada.

Es posible repetir el experimento con un sistema de información que tenga un rubro diferente y un nivel de complejidad mayor. Esto será posible realizar con personas profesionales desarrolladores de software que cuenten con la experiencia necesaria, lo cual permitirá definir nuevas variables de estudio adicionales a la variable de cantidad de errores encontrados. Por ejemplo: número de líneas de código.

Esto permitirá obtener nuevos resultados y observaciones que permitirán identificar bajo qué características, los diferentes sistemas pueden reaccionar al uso de TDD dentro de una metodología tradicional.

Finalmente, es posible también incrementar el tamaño de la muestra, lo cual permitirá obtener datos estadísticos más precisos.

Capítulo 5: Referencias bibliográficas

[Sommerville, 2009] Ian Sommerville; "Ingeniería del Software", 7ma. Edición, 2009.

[ISO25000, 2005] ISO/IEC 25000:2005; "Software Engineering -- Software product Quality Requirements and Evaluation (SQuaRE)".

[Pressman, 2002] Roger S. Pressman; "Ingeniería del Software. Un enfoque práctico", 5ta Edición, 2002.

[Lee, 2010] Lee, G.; Xia, W.; "Toward agile: an integrated analysis of quantitative and qualitative field data on software development agility", MIS Quarterly 34, 87–114, 2010.

[Henderson, 2005] Henderson-Sellers, B.; Serour, M.K.; "Creating a dual-agility method: the value of method engineering". Journal of Database Management 16, 1–23, 2005.

[Cugola, 1998] Cugola, G.; Ghezzi, C; "Software Processes: a Retrospective and a Path to the Future", In Proc. of the Software Process Improvement and Practice Conference, pp. 101-123, 1998.

[Perrin, 2008] Perrin, R; "Real-world Project Management: Beyond Conventional Wisdom, Best Practices, and Project Methodologies", 2008.

[Hildenbrand, 2008] Hildenbrand, T.; Geisser, M.; Kude, T.; Bruch, D.; Acker, T.; "Agile Methodologies for Distributed Collaborative Development of Enterprise Applications", 2008.

[Creswell, 2009] J. W. Creswell; "Research Design: Qualitative, Quantitative, and Mixed Methods Approaches", 2009.

[Gharaibeh, 2009] Gharaibeh, N.; Abu-Soud, S.M.; Bdour, W.; Gharaibeh, I; "Agile Development Methodologies: Are they suitable for developing Decision Support Systems", 2009.

[Silva, 2005] Silva L.; Menezes E.; “Metodologia da Pesquisa e elaboração de dissertação”. 4ta edición. Florianópolis: Universidad Federal de Santa Catarina, 2005.

[Blom, 2012] Martin Blom; “¿Is Scrum and XP suitable for CSE Development?”, Procedia Computer Science, 2012.

[Karamat, 2006] Taha Karamat; Atif Neuman Jamil; “Reducing Test Cost and Improving Documentation In TDD (Test Driven Development)”, International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2006.

[Ficco, 2011] Massimo Ficco; Roberto Pietrantuono; Stefano Russo; “Bug Localization in Test-Driven Development”, 2011.

[Doyle, 2011] Maureen Doyle; Brooke Buckley; Wei Hao; James Walden; “Work in Progress - Does Maintenance First Improve Student's Understanding and Appreciation of Clean Code and Documentation”, 2011.

[Beck, 1999] M. Fowler; K. Beck; J. Brant; “Refactoring: Improving the Design of Existing Code”, Addison-Wesley, 1999.