UNIVERSIDAD NACIONAL AUTÓNOMA DE HONDURAS

FACULTAD DE CIENCIAS ECONÓMICAS

POSFACE

DIRECCIÓN DE ESTUDIOS DE POSTGRADO

MAESTRIA EN GESTIÓN INFORMATICA

TESIS

“INCIDENCIA DE LAS METODOLOGÍAS LEAN-AGILE EN LA MEJORA DE TIEMPOS DE ENTREGA Y REDUCCIÓN DE COSTOS PARA EL DESARROLLO DE SOFTWARE”

SUSTENTADA POR

CONSTANTINO SORTO REYES

PREVIO A OPTAR AL TÍTULO DE

MÁSTER EN GESTIÓN INFORMATICA

TEGUCIGALPA, HONDURAS

AUTORIDADES UNIVERSITARIAS

LICDA. JULIETA CASTELLANOS RUIZ

RECTORA

ABOG. EMMA VIRGINIA RIVERA MEJÍA

SECRETARIA GENERAL

LICDA. LETICIA SALOMÓN

DIRECTORA DEL SISTEMA

DE ESTUDIOS DE POSTGRADO

LICDA. BELINDA FLORES DE MENDOZA; M.A.

DECANA DE LA FACULTAD

DE CIENCIAS ECONÓMICAS

DR. JORGE ABRAHAM ARITA LEÓN

COORDINADOR GENERAL DE POSTGRADO DE

LA FACULTAD DE CIENCIAS ECONÓMICAS

Dedicatoria

La experiencia me he enseñado que mi vida en realidad no me pertenece, mi vida le pertenece en primer lugar a mi padre celestial, a Dios, en segundo lugar a mi familia que son los que han sufrido todas mis derrotas, mis errores y han gozado con mis logros; en tercer lugar se encuentran mis amigos, los que siempre y de una y otra forma me han acompañado en todo momento, es por eso que este logro y muchos más va dedicada a ellos.

A Dios porque sin el aun estuviera perdido en los laberintos del mundo consumiéndome día a día en los pantanos de la banalidad, del egoísmo, de los vicios y de todos los pecados del mundo, a Dios porque sin él no estuviera aún vivo, a Dios porque él es lo principal en mi vida.

A mi familia, porque incondicionalmente me apoyan, me ayudan a crecer, y por lo que a pesar de la distancia que nos separa, el calor de mi hogar nunca se separa de mí, estas y una infinidad de razones más; mis logros son para mi familia.

A mis amigos que siempre me han aceptado, y que de una u otra forma y me han creado en mi la voluntad de mejorar constantemente.

Agradecimientos

Agradezco a Dios porque siempre ha estado conmigo, aun en los momentos en los que me he perdido, me ha dado la fuerza y la voluntad para seguir adelante y no abandonar la lucha que cualquier actividad conlleva, me ha fortalecido y enseñado a crecer constantemente y aunque duela siempre me ha ensenado a aprender lo mejor de cada lección.

A mi familia porque sin ellos no estaría donde estoy, sin el amor, la fortaleza y la voluntad de mi madre, sin el rigor y las enseñanzas de mi padre, sin el amor y el consejo de mis hermanos; los amo y vivo para ver en sus ojos la felicidad y sufrir con ellos sus caídas.

A mis amigos que siempre han estado para socorrerme en momentos difíciles y para brindarme alegrías en los momentos estresantes.

A mi novia que me ha enseñado a crecer y ver el mundo de una manera distinta, que me ha aceptado y perdonado constantemente mis errores.

A mis maestros, compañeros y asesores que me han brindado los conocimientos necesarios para llegar al fin de una nueva etapa en la vida.

A Inversiones la Paz por brindarme la ayuda económica para financiar este estudio, y por brindarme el tiempo para llevar a cabo la misma.

A Daniel Dávila mi jefe que me ha brindado su ayuda en todo tiempo aun y cuando no fuera de carácter laboral se ha portado como un amigo, un compañero, un maestro.

Gracias a Dios porque todo estuvo en el momento y lugar adecuado, porque mi camino lo ha elegido de manera que me guste y que no sufra tanto, ¡Gracias a Dios por mi vida!

Resumen Ejecutivo

La presente investigación se centra en dar solución a la problemática general de desarrollo de software:

¿Cómo hacer frente a un proyecto de desarrollo de software, que requiere tiempos cortos de entrega, sin elevar los costos?

Por esta razón el estudio se centra en nuevos métodos de gestionar los proyectos de software, siendo estos nuevos métodos las metodologías Lean-Agile para el desarrollo de software (Liker & Morgan, 2006) (Figueroa, Solís, & Armando A., 2007), esta alternativa de producción de software relativamente nueva, cuenta con documentación sustentadora de su utilización en diferentes proyectos de software, tanto específicos como lo es el desarrollo para software con servicios Saas (Chavez, Martín, Rodríguez, Murazzo, & Valenzuela, 2012), como generales (Garzas, 2015).

En el documento se describen características principales de las dos grandes categorías de las metodologías del desarrollo como ser Metodologías de desarrollo tradicional o pesadas y Metodologías Lean-Agile.

En esta investigación se fundamenta, con base en las experiencias de desarrolladores, que la incidencia de las metodologías Lean-Agile en el desarrollo de software, produce un efecto de mejora de tiempos y costos en la creación de software.

Palabras Clave: Metodologías de desarrollo, Lean, Agile, Mejora, Tiempos, Costos.

Abstract

This research is focused on solving general problem of software development:

How to make a software development project that requires shorter deliveries times without increased costs?

For this reason the study is focusing in new methods of managing software projects, this new method is Lean-Agile Methodology for software development (Liker & Morgan, 2006) (Figueroa, Solís, & Armando A., 2007).,This alternative of software production is relatively new, has supportive documentation of their use in many software projects, specific projects such a software as a services Saas (Chavez, Martín, Rodríguez, Murazzo, & Valenzuela, 2012), as a general software (Garzas, 2015).

This document describes the principal features about two big categories of methodology of software, a traditional development methodology and Lean-Agile software development methodology.

This research is based in the experiences of developers, and they affirm it the use of Lean-Agile methodology in software development, produces an effect in improvement of times and costs in making software.

Keywords: Software Methodology, Lean, Agile, Improvement, Times, Costs.

Índice **Página**

Índice de Ilustraciones

Índice de Formulas

Índice de Tablas

Índice de Gráficos

Introducción

La motivación primordial para el desarrollo de la presente investigación, se basa en la necesidad inherente de la industria del software de obtener productos de manera rápida y de bajo costo, esto debido a la presencia creciente de dinamismo en los requerimientos de software.

En esta investigación se describen diferentes metodologías, propias de las metodologías tradicionales o pesadas y metodologías Lean-Agile, con ello se prepara un panorama de características generales y diferenciadoras entre cada una de las metodologías, así mismo se abordan las bondades que presentan y posibles limitantes.

La investigación se centra en la incidencia de la Metodologías Lean-Agile y sus efectos en el desarrollo de software, la cual se realiza con participantes dotados de experiencias en la utilización de metodologías tradicionales o pesadas y metodologías Lean-Agiles, de lo cual se obtiene su valoración con respecto al efecto que percibieron, haciendo un enfoque en las variables de tiempos de entrega y costos de desarrollo, con esta valoración se sustentan las respuestas a las preguntas de investigación e hipótesis planteadas.

Los capítulos abordados en el presente documento se describen a continuación.

CAPÍTULO I. PLANTEAMIENTO DEL PROBLEMA

En este capítulo se aborda el antecedente de la temática, así como también la justificación de la investigación y el planteamiento del problema, así mismo se definen los objetivos a cumplir de la presente investigación, como también las preguntas y limitantes.

CAPÍTULO II. MARCO CONCEPTUAL Y TEÓRICO

En este capítulo se realiza una valoración respecto a la bibliografía utilizada, de la misma forma se detalla un marco conceptual el cual define mediante conceptos, términos importantes utilizados en la presente investigación, seguidamente se encuentra el marco teórico; como su nombre lo indica engloba la teoría de la temática, esta teoría se describe de manera puntual, haciendo referencia a literatura de fuente valida y verificable.

CAPÍTULO III. ENFOQUE Y TIPO DE INVESTIGACIÓN

En este capítulo se detalla el tipo de investigación y el enfoque que se le dará a la misma

CAPÍTULO IV. HIPOTESIS Y VARIABLES

Este capítulo detalla las Hipótesis que se pretenden evaluar, así como también las variables que servirán para la evaluación de las mismas.

CAPÍTULO V. ESTRATEGIA METODOLÓGICA

El capítulo V detalla el tipo de investigación a realizar, así mismo se detalla la población que será sujeta a estudio, de la cual se obtendrá una muestra; misma que será definida en este capítulo.

CAPÍTULO VI. PLAN DE ANÁLISIS

Con la recolección de los datos procedemos a desarrollar una planeación de cómo se realizará el análisis respectivo, en este capítulo se aborda ese plan; plan que contempla los puntos importante a tratar en el análisis.

CAPÍTULO VII. ANÁLISIS DE RESULTADOS.

Cuando se define el plan de análisis procedemos a la ejecución del mismo, este capítulo contempla los resultados obtenidos después de realizar cada una de las tareas descritas en el capítulo anterior.

CONCLUSIONES

Con los resultados analizados no queda más que concluir sobre la presente investigación, este capítulo contempla esas conclusiones y además alberga posibles enfoques a nuevas investigaciones sobre esta temática.

CAPÍTULO I. PLANTEAMIENTO DEL PROBLEMA

1.1 Antecedente

El desarrollo de software es el pilar fundamental para toda empresa que desea competir en cualquier tipo de mercado. Las metodologías Lean-Agile para el desarrollo de software, tiene sus raíces en la filosofía Lean, propia de las Empresa Automotriz TOYOTA (Liker J. , 2003), la cual fundamenta sus principios en la eliminación de desperdicios, aprendizaje y desarrollo continuo ya sea de personal, productos o servicios brindando así calidad y satisfacción al cliente.

El desarrollo de software Lean es una adaptación de cada uno de estos principios industriales al desarrollo de software, el termino de desarrollo de software es implantado por Mary Poppendieck y Tom Poppendieck en 1990 en su libro *Lean Software Development* (Poppendieck & Poppendieck, 2003).

Bajo este creciente mundo tecnológico y realizando una interacción entre Lean y el desarrollo de software surgen distintas variaciones de las mismas, denominadas Metodologías Agiles (Calderón & Rebaza, 2007), las cuales son aplicadas muchas veces, sin saber las ventajas o desventajas de las mismas y constantemente tomadas como modas y no como metodologías capaces de brindar potencialidades al desarrollo de software.

Las metodologías de desarrollo de software estándar o tradicionales, son denominadas las metodologías pesadas, las cuales son desde de hace de más de 40 años los lineamientos a seguir cuando de desarrollo de software se trata (SommerVille, 2005) , estas metodologías pesadas no han presentado cambios.

Dentro de los estudios anteriormente realizados y referenciados por esta investigación, resaltan las características, descripción, conceptualización y formas de aplicación de las metodologías Lean-Agile en investigaciones como (Calderón & Rebaza, 2007), (Figueroa, Solís, & Armando A., 2007), (Letelier & Penadés, 2006), (Poppendieck & Poppendieck, Lean Software Development: An Agile Toolkit, 2003). Dentro de las investigaciones con un enfoque en particular destacan las de su aplicación a áreas en específico como es el caso de (Chavez, Martín, Rodríguez, Murazzo, & Valenzuela, 2012) en el cual se aborda la aplicación de estas metodologías a servicios Saas, dentro

de esta investigación se crea un énfasis en la aplicación de estas metodologías debido al dinamismo del software en la nube y a la variante de que el mismo debe funcionar para una variedad de clientes, estos clientes; cada uno con características particulares son los que presentan requerimientos cambiantes, así mismo es importante mencionar que poco de la investigación abordada se desarrolla sobre el efecto causado en variables directas del software; variables como calidad, tiempos y costos, dentro de estas investigaciones se destaca el trabajo realizado por (Gonzalez, 2008), en el cual se aborda la aplicación de las metodologías agile para la evolución del software, la investigación se fundamenta en seguir de cerca la metodología SCRUM y su aplicación al software, como resultado obtiene conclusiones con enfoques cualitativos para el personal que se utilizó, así como también conclusiones sobre la aplicación de la metodología SCRUM como medio de gestión para la creación de software.

En el abordaje de las metodologías de desarrollo ya sean pesadas o agiles, se crean múltiples debates a cerca de; que usar, porqué usar y porqué dejar de usar, por lo cual el área de investigación es aun amplia y de alto impacto, como lo menciona (Garzas, 2013) quien se establece como un gurú de la gestión Ágil, estas metodologías no son desordenadas más bien requieren en su totalidad de disciplina y carácter de obligatoriedad en cuanto a ciertos aspectos muy dependiente de la variante agile a utilizar.

1.2 El Problema de Investigación

Actualmente el crecimiento empresarial va muy de la mano con el cambio tecnológico, este cambio no solo es de hardware también implica creación de software, o mejoras a software existentes. Estos requerimientos o nuevos proyectos de software van enfocados a suplir las necesidades de los usuarios, los cuales en la actualidad presentan demandas muy variables con cambios muy dinámicos, características que tienen consecuencias en el manejo de dicho proyecto, al ser los requerimientos dinámicos y variables (Gonzalez, 2008), implica que los proyectos de software tengan que realizarse en tiempos cortos y a costos aceptables para los clientes.

Para el manejo de los requerimientos propios de la creación de software, desde sus inicios se manejan metodologías de trabajo, estas metodologías de trabajo son denominadas Metodologías Tradicionales (Figueroa, Solís, & Armando A., 2007).

Las metodologías Tradicionales o pesadas, surgen con los inicios del desarrollo de software, el cual era manejado bajo adaptaciones de metodologías existentes de otras áreas industriales. Dentro de las características principales de estas metodologías está su fuerte vínculo al cumplimiento de un plan, lo cual genera que el cumplimiento del mismo sea obligatorio; en consecuencia a esta rigurosidad estas metodologías son robustas, de poco cambio y poca flexibilidad.

Este manejo de los proyectos de software ha venido evolucionando conforme se desarrollan soluciones a estos requerimientos o proyectos, así mismo como es de esperarse no todo es un éxito ya que según (McConnell, 1996) los proyectos o requerimientos que presentan problemas de entregas fuera de tiempo, incremento del tamaño del proyecto, incremento del costo y solicitud de requerimientos fuera de planeación rondan entre el 25% y 50% del total de proyectos finalizados.

La gestión actual de un proyecto de software debe presentar capacidades de repuesta a cambios, en tiempos cortos y costos bajos (Gonzalez, 2008), esto de acuerdo a las particularidades del rubro al cual se aborde. En un tiempo corto debido a que posiblemente el cambio genere ganancias para el cliente, y muy posiblemente este cambio brindara la ventaja competitiva que el mismo requiere sobre su competencia; con costos controlables para que los presupuestos de dichos proyectos no se vean afectados y sea factible económicamente realizar dichos cambios.

En la relación del manejo actual de estos proyectos de software con requerimientos de tiempos cortos y de costos controlables y las Metodologías Tradicionales surge el cuestionamiento:

¿Cómo hacer frente a un proyecto de desarrollo de software que requiere tiempos cortos de entrega sin elevar los costos?

1.3 Objetivos de la Investigación

1.3.1 Objetivos Generales

- Evaluar el efecto en los costos y tiempos de entrega que la incidencia de las Metodologías Lean-Agile generan en un proyecto de desarrollo de software.

1.3.2 Objetivos específicos

- Evaluar el efecto que la incidencia de las Metodología Lean - Agile generan en los costos de desarrollo de software.

- Evaluar el efecto que la incidencia de las Metodología Lean - Agile generan en los tiempos de entrega de desarrollo de software.

1.4 Preguntas de la Investigación

¿Cuál es el efecto que la incidencia de las Metodología Lean - Agile generan en los costos de desarrollo de software?

¿Cuál es el efecto que la incidencia de las Metodología Lean - Agile generan en los tiempos de entrega de desarrollo de software?

1.5 Justificación Del Estudio

Los proyectos de software que presentan un esquema de variabilidad y dinamismo en cuanto a los requerimientos de desarrollo de software se refiere (Gonzalez, 2008), tienen implicaciones en los tiempos de entrega y los costos en los que se incurre, esto debido a que si se requiere un software con extrema urgencia, se deberá gestionar el proyecto de manera que el mismo cumpla con el tiempo requerido, y para que logre ser factible su costo deberá estar dentro de lo presupuestado para dicha tarea; para hacer frente a estas características se han aplicado Metodologías tradicionales o pesadas, siendo estas de un carácter riguroso, poco cambiante e inflexible.

El desarrollo de los sistemas tradicionales de ciclo de vida o metodologías pesadas se originó en la década de 1960 para servir de lineamientos en el desarrollo a gran escala de sistemas de negocio (SommerVille, 2005), en una época de grandes conglomerados empresariales. La idea principal era continuar el desarrollo de los sistemas de información en una estructurada rígida, de una manera metódica, y reiterando en cada una de las etapas del ciclo de vida.

Los proyectos de software con requerimientos dinámicos no se sujetan a una metodología rígida como las metodologías tradiciones, esto debido a que se requieren productos de software en tiempos cortos sin alterar presupuestos asignados a su desarrollo. Es aquí, donde debido a la necesidad que se fundamenta en los requerimientos cambiantes, es que nacen las Metodologías Lean – Agile (Garzas, 2015) las cuales se proponen la cercanía con el usuario, la integración del equipo de desarrollo con la operación misma y la pronta puesta en marcha del software que sustentara un requerimiento.

Estas Metodologías Lean- Agile, en su teoría (Figueroa, Solís, & Armando A., 2007) describen la capacidad de ser flexibles, cambiantes, creando una atmosfera de trabajo en equipo, sin incurrir en implicaciones de calidad en el producto final, siendo de esta forma una solución a las problemáticas planteadas del manejo de proyectos de software.

1.6 Delimitación del Problema

La problemática en cuestión, se basa en la evaluación del efecto que la incidencia de las Metodologías Lean-Agile en los proyectos de desarrollo de software generan, con la afirmación de la premisa anterior se podría definir que estas Metodologías Lean-Agile representa una competencia para afrontar los proyectos de software donde los requerimientos de los mismos sean dinámicos, por lo cual requerirán soluciones en tiempos cortos a costos bajos.

Debido al tiempo que conlleva la realización de una comparación entre metodologías, la investigación se dirige a la captura de experiencias de profesionales del desarrollo en la temática abordada, para esta recolección de profesionales, la investigación se llevara a cabo en la ciudad de Tegucigalpa M.D.C, dentro de esta limitante geográfica se dispondrá a abordar a los profesionales que se desempeñen actualmente en un puesto que lo vincule a la gestión de proyectos de software, cabe agregar que no todas las empresa que se encuentran en dicha ciudad poseen un departamento

de tecnología con una sección de desarrollo, por lo cual de todas las empresas de la ciudad en cuestión se tomaron solo las catalogadas como “Grandes Contribuyentes” por parte de Dirección Ejecutiva de Ingresos D.E.I. y de estas se abordara a los líderes de dicha área o afines.

1.7 Posibles Deficiencias en el Proceso de la Investigación

Una de las deficiencias más grandes en la investigación es el recurso del tiempo ya que debido a esto no se podrá abordar un proyecto de mayor impacto que el que se seleccionó.

1.8 Viabilidad del Estudio

El proyecto a investigar es viable porque se cuenta con toda la documentación necesaria, así mismo se cuenta con el personal y el equipo necesario para realizar dicha investigación.

El asesor de dicha investigación posee un grado de experiencia sobre la temática abordada, esta experiencia lo califica para realizar el trabajo de investigación sin inconvenientes en esta área.

CAPITULO II. MARCO CONCEPTUAL Y TEORICO

2.1 Revisión de la Literatura

De las Metodologías de desarrollo de software son muchos los autores que se destacan, así mismo de esta temática se han fundamentado muchas investigaciones capaces de brindar una amplia descripción sobre la temática.

La literatura referente a esta temática podría decirse que inicia con los primeros libros enfocados en el método de producción de Toyota, de esta literatura se genera mucho conocimiento, tanto que basado en esto se genera el movimiento Ágil, obteniendo el "Manifiesto Ágil" el cual rige las características principales de una Metodología Ágil, dentro de las referencias abordadas en el presente documento, destacamos las siguiente referencias, las cuales son altamente reconocidos a nivel mundial y que en su mayoría son utilizadas para la formación de profesionales; estas referencias son : (Liker & Morgan, 2006), (Liker J. , 2003), (Liker & Meier, 2005), (Sampieri, Collado, & Lucio, 2010) (Joyanes Aguilar, 2012) (Womack, Jones, & Roos, 1990), (SommerVille, 2005), (Poppendieck & Poppendieck, 2003) (Roger S. Pressman).

Para desarrollar la temática en cuestión se tomaron como referencias: libros, tesis, artículos de revistas, y documentos de investigación científica (papers).

Entre los autores de la diferente literatura abordada, se consideró solo a quienes tuviesen una publicación con datos fidedignos y verificados, así mismo dentro de sus documentos se debería hacer uso de referencias válidas y verificables, esto con el objeto de fortalecer el fundamento del marco teórico.

Bajo la revisión del Asesor Técnico se verifico la literatura usada.

2.2 Marco Conceptual

En la presente sección se realizará una exposición de los principales términos asociados a la teoría abordada de la presente investigación, esta descripción se desarrollará de manera cronológica en cuanto a la aparición y uso de conceptos.

2.2.1 Software

La definición del término Software tiene diversas fuentes debido a la variedad temática a la que el término se vincula; pero en su gran mayoría se refieren a software como un programa de computadora.

Dentro de la gran variedad de profesionales en el tema, la definición de software más amplia y completa para fines de este estudio, se le atribuye a (SommerVille, 2005) quien lo define como:

"Todos los documentos asociados y la configuración de datos que se necesitan para hacer que estos programas operen de manera correcta."

Por lo general, un sistema de software consiste en: diversos programas independientes, archivos de configuración que se utilizan para ejecutar estos programas, un sistema de documentación que describe la estructura del sistema, la documentación para el usuario que explica cómo utilizar el sistema y sitios web que permitan a los usuarios descargar la información de productos recientes.

Los desarrolladores de software se enfocan en ver el software como un producto, es decir, un producto que se vende a un cliente.

Según este enfoque de producto y su producción (SommerVille, 2005) califica el software en dos tipos:

1. Productos genéricos. Son sistemas aislados producidos por una organización de desarrollo y que se venden al mercado abierto a cualquier cliente que le sea posible comprarlos.

Ejemplos de este tipo de producto son el software para PCs tales como bases de datos, procesadores de texto, paquetes de dibujo y herramientas de gestión de proyectos.

2. Productos personalizados (o hechos a la medida). Son sistemas requeridos por un cliente en particular. Un contratista de software desarrolla el software especialmente para ese cliente.

Ejemplos de este tipo de software son los sistemas de control para instrumentos electrónicos, sistemas desarrollados para llevar a cabo procesos de negocios específicos y sistemas de control del tráfico aéreo.

2.2.2 Metodología de Desarrollo

Las metodologías de desarrollo son un concepto popularizado en el ámbito de la ingeniera del software, estas metodologías según (INTECO, 2009) se definen como:

"La metodología para el desarrollo de software en un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo por el cual fue creado. "

Según la definición las metodologías de desarrollo de software, al igual que en la industria tratan la forma de obtener un producto, estas metodologías definen herramientas, mejores prácticas, experiencias, entregables, roles, tareas, y actividades, de las cuales se obtiene el producto final, en este caso software.

2.2.3 Requerimiento

Para realizar una conceptualización sobre requerimiento es necesario situarnos en la definición general de la palabra como tal; la real academia española (Real Academia Española, 2011) define requerimiento como:

"Acción y efecto de requerir ".

De esta denotación tomamos el concepto de requerir (Real Academia Española, 2011) el cual se definen como:

"Intimar, avisar o hacer saber algo con autoridad pública."

Mismo que posee como verbo transitivo "Necesitar", el cual es más claro para nuestro uso, utilizando la definición de requerimiento y realizando un cambio por la palabra necesitar obtenemos que requerimiento se define como:

"Acción y efecto de necesitar "

Dejando claro nuestro concepto de requerimiento, podemos utilizar dicha conceptualización en temas referentes al desarrollo de software, el cual generaliza para expresar una necesidad como requerimiento de software.

La IEEE (IEEE, 1998) define los requerimientos de software como:

"Una descripción para un producto software particular, programa o conjunto de programas, que realizan ciertas funciones en un ambiente específico. Debe expresar las funciones a ser realizadas, en qué situación y para quién, los resultados a obtener, como así también centrarse en los servicios a llevar a cabo".

Esta definición será la que se asumirá en la presente investigación.

2.2.4 Desperdicios

Referente a los desperdicios expuestos en el método de producción Toyota (Rother & Shook, 1999) los cuales están descritos de la siguiente forma:

- **Defectos:** Fabricar productos defectuosos incluyendo servicios (Rother & Shook, 1999)
  - Errores al principio.
  - Datos erróneos al sistema.
  - Error de diseño.
  - Cambio de las especificaciones en los mismos.
  - Errores de facturación.
  - Rotación de personal entrenado.
  - Consecuencias, reparaciones, remplazos y aumento de costos.

- **Exceso de producción:** Producción precipitada o en cantidades mayores a las necesarias (Rother & Shook, 1999).

- **Transporte:** Mover el trabajo en proceso de un sitio a otro (Rother & Shook, 1999), Ejemplo: mover materiales y producto terminado, entre el almacenado y otros procesos.

- **Esperas:** Conocido generalmente por producir tiempo muerto (Rother & Shook, 1999), es donde el personal no se encuentra produciendo ningún valor, por ejemplo: cuando el personal espera por razones como:
  - Máquinas automatizadas que aún no han terminado su trabajo.
  - Espera por el siguiente proceso.
  - Herramientas, suministros, piezas, documentación que no están listos y se tiene que esperar.
  - No se tiene trabajo por falta de inventario.
  - Caídas y altos tiempos de respuesta de los sistemas.

- **Inventarios :** Grandes cantidades innecesarias de inventario (Rother & Shook, 1999) por ejemplo
  - Materia prima, producto terminado.
  - Bandejas llenas de trabajo pendiente de lo cual podemos obtener diferentes tipos de consecuencias.
  - Tiempo de espera más largos.
  - Obsolescencia.
  - Producto dañado.
  - Puesto de almacenamiento y transporte.
  - Ocultamiento de los problemas

- **Movimientos:** Cualquier movimiento de personal que no añada valor (Rother & Shook, 1999).
  - Alcanzar.
  - Buscar.

- Traer herramientas.
- Caminar de un sitio otro en el lugar de trabajo.

- **Procesos innecesarios:** Pasos innecesarios para procesar los productos (Rother & Shook, 1999).
  - Herramientas inadecuadas.
  - Mal diseño del producto causa movimientos innecesarios.
  - Calidad más alta de la necesaria
  - Múltiple introducción de los mismos datos.
  - Trabajar para que no se note que no hay nada que hacer.

2.1Marco Teórico

2.2.2 Filosofía Lean

La filosofía Lean (Liker J. , 2003) tiene su origen en el Japón, cuando Sakichi Toyoda tenía su fábrica de telares, ahí aplicó la filosofía a largo plazo basado en la mejora continua y creo muchas herramientas para que la calidad de sus productos fuera la mejor, basados en un modelo de prueba y error. El hijo de Sakichi, Kiichiro fue quien decidió dejar los telares e inició la fábrica de autos de donde surgieron los principios te JIT y Kanban.

Para su estudio, la filosofía Lean (Liker J. , 2003) (Liker & Morgan, 2006) se subdividió en cuatro partes importantes denominadas las 4P

- Filosofía (Philosophy)
- Proceso (Process)
- Personas y socios (People/Partners)
- Solución de problemas (Problem Solving)

Cada subcategorías fue fundamentada en cada uno de los principios siguientes (Liker J. , 2003) :

1. Basar las decisiones gerenciales en una filosofía largo plazo a expensas de los resultados financieros a corto plazo.
2. Crear un proceso de flujo continuo para llevar los problemas a la superficie.
3. Usar sistemas Pull para evitar la sobre producción.
4. Nivelar la carga de trabajo.
5. Crear una cultura que se detenga solucionando problemas para lograr la calidad correcta a la primera.
6. La estandarización de las tareas y los procesos es base para el mejoramiento continuo y el poder de decisión de los empleados.
7. Usar controles visuales para que los problemas no queden ocultos.
8. Usar únicamente tecnología confiable y concienzudamente probada para que esté al servicio de las personas y el proceso.

9. Formar líderes que entiendan concienzudamente el trabajo, vivan la filosofía de la compañía y la enseñen.
10. Desarrollar gente excepcional y equipos que sigan la filosofía de la compañía.
11. Respetar a sus socios y proveedores retándolos y ayudándolos a mejorar.
12. Ir uno mismo a la situación para entenderla concienzudamente.
13. Tomar las decisiones lentamente y por consenso, considerando concienzudamente todas las opciones a implementar rápidamente.
14. Convertirse en una organización de aprendizaje a través de la reflexión y la mejora continua.

El gran auge de la industria automotriz en Japón (Womack, Jones, & Roos, 1990) en cuanto al mercado occidental causo gran interés en Estados Unidos; el cual envío diferentes recursos de la universidad MIT a realizar estudios de las empresas Toyota.

De acuerdo a estos estudios de la universidad de la MIT (Womack, Jones, & Roos, 1990) surgieron diferentes enfoques del porqué la calidad de los autos de Japón superaba a todos los diferentes autos del mercado occidental, aparte de que los autos de Japón representaba una mejor calidad también representaba menos costos y eran los autos más vendidos durante su primer año de incursión en el mercado occidental, dejando grandes bajas en ventas al gigantes de la industria como General Motors, Mercedes-Benz entre otros. De este estudio surgió la introducción de la Filosofía Lean y el sistema T.P.S. que abordaremos a continuación.

2.2.3 Sistemas de Producción TOYOTA T.P.S.

El Sistemas de Producción TOYOTA T.P.S. es un sistema de producción y administración (Liker & Morgan, 2006) surgido y madurado en la empresa automotriz Toyota. Como se había mencionado anteriormente el sistema de producción de Toyota viene fundamentado de la fábrica textil de su creador Sakichi Toyoda, el cual creía que el sistema de producción debía alejar al empleado de las tareas repetitivas basándose en innovaciones y crecimiento a través de las soluciones del empleado.

De este tipo de metodología (Rother & Shook, 1999) de producción surgen diferentes conceptos como ser JIDOKA (automatización), POKA JOKE (a prueba de fallos), Just in time (Justo a tiempo) y Muda (desperdicios).

La meta principal de este tipo de sistemas (Rother & Shook, 1999) es eliminar los desperdicios mismos que puede ser englobados en diferentes categorías que mencionamos a continuación:

- Defectos
- Exceso de producción
- Transporte
- Esperas
- Inventarios
- Movimientos
- Procesos innecesarios

El modelo T.P.S. busca reducir cada uno de los desperdicios anteriormente mencionados. Para lo cual necesita la integración de toda la empresa, para la representación de este modelo se usa un diagrama, mismo que es representado por los cimientos de una casa, específicamente en construir una casa de lo cual obtenemos el siguiente gráfico.

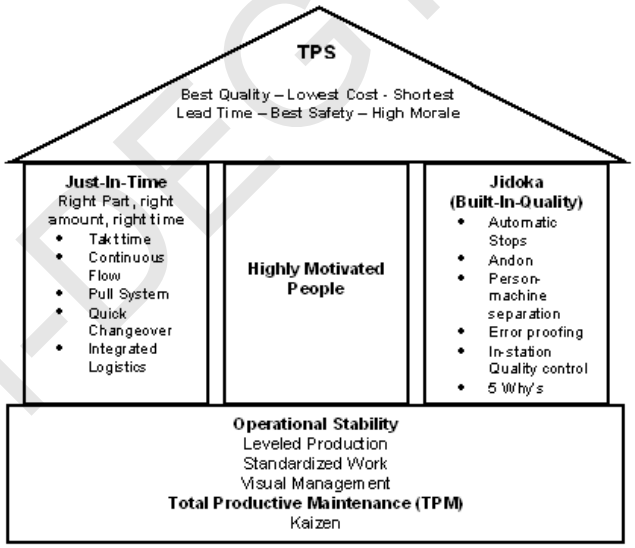


Ilustración 1 Sistema de Producción TOYOTA Fuente (Liker J. , 2003)

Como podemos ver la casa se fundamenta en las cuatro 4p.

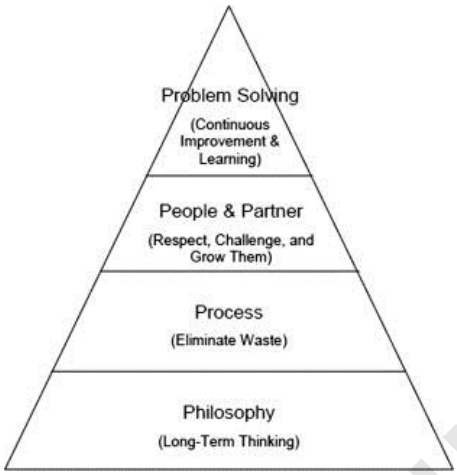


Ilustración 2 Modelo 4P Fuente (Liker & Meier, 2005)

En general podemos resumir que para aplicar el T.P.S. es necesario tener de fundamento la filosofía Lean (Poppendieck & Poppendieck, 2003) , la cual se basa en agregar valor a cada uno de los productos que nosotros vamos a desarrollar, y este valor es agregado en cada paso que se realiza en la fabricación de cualquier producto.

La siguiente parte es llevar a cabo el proceso correcto para llegar al resultado correcto (Poppendieck & Poppendieck, 2003), se podría decir que la parte fácil sería realizar ingresos inmediatos de dinero y la parte difícil sería las inversiones a largo plazo. Luego tenemos la escala de personas y socios, a la organización le implica inculcar en su personal y a en sus proveedores la filosofía y no hablarles de que se van a hacer millonarios de la noche la mañana, sino enfocarlos en el respeto, crecimiento bajo ambientes retadores apegados a la realidad y la filosofía a largo plazo.

En la parte final pero no menos importante, es donde se realiza la eliminación de los desperdicios y la solución de los problemas; la disciplina continua de solventar problemas dirige al aprendizaje organizacional en torno de la mejora en calidad, menores costos y mejora continua.

A la aplicación de todo este proceso disciplinado, organizado y estricto, llamaremos una Producción Lean (Liker & Morgan, 2006).

El T.P.S. al igual que la filosofía Lean se basa en conceptos claves a lo cual se le llama sistema socio – técnico.

Ilustración 3 Sistema Socio – Técnico Fuente (Liker & Morgan, 2006)

Este sistema se describe con 13 principios fundamentales (Liker & Morgan, 2006)

1. **Establecer el valor definido por el cliente para separar el valor agregado del desperdicio.**

El objetivo de este Principio es estimar el valor definido por el cliente lo más exacto posible y eliminar o reducir el desperdicio.

El principio nos obliga a tener un proceso que identifique el producto en específico y sea capaz de encontrar los valores que el cliente requiere y tener la capacidad de transmitirlo a su equipo de trabajo. (Liker & Morgan, 2006)

2. **Concentrar esfuerzos al comienzo del proceso de desarrollo de productos, para explorar soluciones mientras hay un máximo de espacio de diseño.**

Al inicio de todo proyecto es cuando tenemos un ambiente amplio de trabajo, el cual es necesario explotarlo al máximo, para llevar a cabo este principio es necesarios que la gente más experimentada proporcione soluciones que anticipen y resuelvan problemas, diseñar contramedidas de calidad y lograr minimizar la variación del fin convenido. (Liker & Morgan, 2006)

3. **Crear un flujo nivelado en el proceso de desarrollo de productos.**

Para este principio se requiere la estabilidad en el proceso a estudiar, se requiere un proceso libre de desperdicio, y así acelerar el desarrollo del producto; aunque aun así se pueden tener diferentes retos de diseños específicos y únicos la tarea es llevar la secuencia con similares proyectos, en este sentido se pueden mejorar el desarrollo utilizando herramientas adaptadas para eliminar el desperdicio y sincronizar actividades múltiples de departamentos funcionales. (Liker & Morgan, 2006)

4. **Utilizar una estandarización rigurosa para reducir la variación y crear flexibilidad y resultados predecibles.**

La estandarización es muy importante, debido a esto podemos desarrollar de una mejor manera cada uno de los productos y podríamos tener la flexibilidad y velocidad necesaria para eliminar el desperdicio y obtener como consecuencia resultados más predecibles obteniendo una mínima variación del fin de un proyecto. (Liker & Morgan, 2006)

5. **Desarrollar un sistema de ingeniero jefe para integrar el desarrollo de principio a fin.**

El proceso como tal debe de ser guiado por un ingeniero jefe excepcional, con la habilidad de liderar la integración del producto e integrando las personas que están trabajando en el programa.

Este ingeniero jefe es la voz del cliente y el responsable del éxito del programa, desde su concepción hasta su finalización. (Liker & Morgan, 2006)

6. **Organizar para balancear la experiencia funcional integración multifuncional.**

En esta parte vemos una de las funciones del ingeniero jefe, el cual debe de mantener las diferentes posiciones o funciones de cada uno de los que trabajan en el Proyecto.

Se realiza para mantener el enfoque de que el cliente es primero y que se debe dar al máximo en cada uno de las funciones para lograr el objetivo deseado, trabajando desde posiciones, funciones y áreas; los desarrolladores pueden ir probando y mejorando la calidad de cada uno de los elementos. (Liker & Morgan, 2006)

7. Desarrollar una elevada competencia técnica en todos los ingenieros.

Para alcanzar una elevada competencia técnica en todos los desarrolladores, es necesario partir desde su contratación, en la cual debe efectuarse un riguroso proceso de selección.

Se le sumara un sistema de capacitación técnica con evaluaciones regulares, y valoraciónes la competencia técnica demostrada.

La aplicación de una selección rigurosa y una capacitación técnica elevada nos ayudará a desarrollar productos de una gran calidad debido a la confianza profesional que se tiene, a una variación menor en cuestión de desperdicios y a una velocidad de producción que reduce costos. (Liker & Morgan, 2006)

8. Integrar completamente los proveedores al sistema desarrollo de productos.

Un cliente ve un producto como un todo, a él no le importa si los proveedores trajeron elementos de mala calidad, por lo cual es necesario que nuestros proveedores compartan los mismos principios para lograr una calidad óptima. (Liker & Morgan, 2006).

9. Edificar en aprendizaje y mejora continua.

El arma más poderosa posiblemente en cuanto las empresas se refiere, es al aprendizaje y mejora continua. El aprendizaje no es un agregado, una actividad extracurricular, es el núcleo de un desarrollo a través de capacitación, solución a problemas, caídas y otras experiencias; todas las cuales se enfocan en el mejoramiento y aprendizaje continuo. (Liker & Morgan, 2006)

10. Crear una cultura de apoyo a la excelencia y a la mejora incesante.

Una de las herramientas más efectivas que utiliza esta metodología es la cultura de apoyo, ya que gracias al balance de carga que existe en el desarrollo de productos, podemos tener técnicos en diferentes áreas con la misma calidad que el que se dedica especialmente a eso, por lo cual la cultura de apoyo es un aporte de excelencia y de la mejora incesante, ya que diferentes ojos pueden estar viendo en el mismo ángulo. (Liker & Morgan, 2006)

11. Adaptar tecnología acoplable a las personas y procesos.

La tecnología debe ser transparente a la integración, debe dar soporte a cada uno de los procesos, debe mejorar el rendimiento de las personas; no remplazarlos, debe dar una solución específica, debe ser del tamaño correcto, no más grande de lo que se necesita.

Queremos recordar que la competencia puede desarrollar el mismo sistema, ya que tiene las bases para realizarlo, la diferencia será en que nuestro sistema será capaz de acelerar los procesos y promover mejoras continuas, recordando siempre que las personas siguen siendo responsables por la realización del trabajo. (Liker & Morgan, 2006)

12. Alinear la organización por medio de simple comunicación visual.

En todo proyecto la comunicación debe ser una prioridad, esta comunicación debe ser objetiva, suficiente, precisa y enfocado en hechos esenciales.

La disciplina y estandarización mejora la efectividad en todos los tipos de comunicación, especialmente en comunicación acerca de solución de problemas. (Liker & Morgan, 2006)

13. Utilizar herramientas poderosas para la estandarización y el aprendizaje organizacional.

Se deben utilizar herramientas simples y especificas con métodos que puedan nivelar el conocimiento y estandarización de productos, procesos y habilidades. (Liker & Morgan, 2006)

Estos 13 principios van de acuerdo con cada una de las herramientas que se utilizan para desarrollar cualquier tipo de producto, ya sea industrial, de servicio o en este caso un desarrollo de software.

Luego de conocidos cada uno de los 13 principios necesarios para aplicar la metodología T.P.S. Es necesario conocer diferentes herramientas que se utilizarán en cada uno de los desarrollos de productos.

2.2.4 Herramientas utilizadas en T.P.S.

Muchas de las herramientas utilizadas en T.P.S. (Liker & Meier, 2005) Son herramientas utilizadas en diferentes áreas, solo que enfocadas y disciplinados por cada uno de los principios Lean y los principios T.P.S.

Hablar de todas las herramientas de la filosofía Lean, es hablar sobre un mundo de herramientas, por lo cual nosotros sólo tomaremos tres de las herramienta más importante necesarias para la producción de un producto.

2.2.4.1 Mapa de Corriente de Valor de Estado Actual

(Rother & Shook, 1999) Se define como el levantamiento de un proceso; o el conocimiento de un proceso, en este caso no se abordan una optimización; sino que se hace un desglose detallado de cómo está funcionando actualmente un proceso dentro de la empresa.

En este proceso es necesaria la medición de tiempos, ya que está basado en ver donde se está desperdiciando el tiempo, donde estamos desperdiciando personal u otro recurso o los lugares donde se esté aplicando alguno de los desperdicios anteriormente mencionados.

Ejemplo.

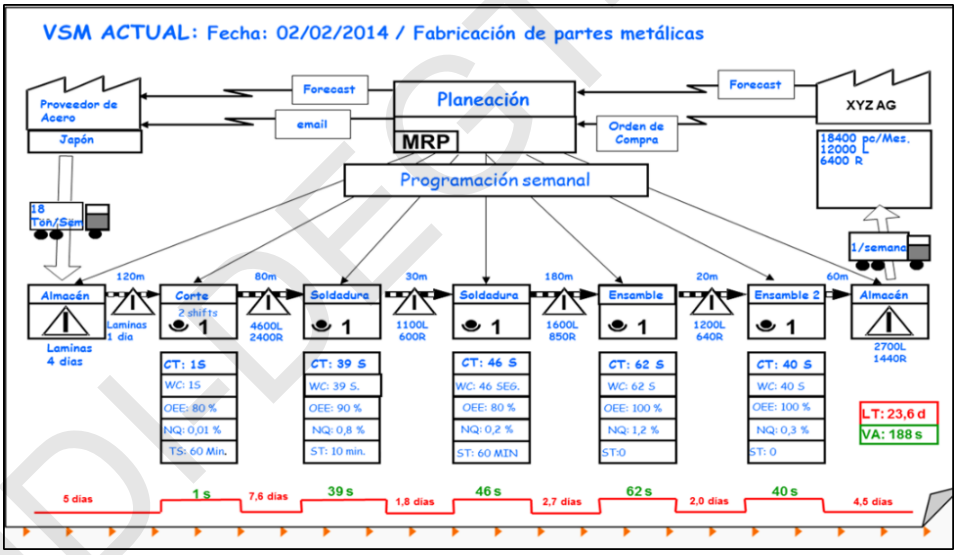


Ilustración 4 MCV Estado Actual, Fuente (Lean Solutions, 2015)

2.2.4.2 Mapa de Corriente de Valor de Estado Futuro

El Mapa de Corriente de valor de estado futuro (Rother & Shook, 1999) es una mejora a partir del Mapa de corriente de valor de estado actual, en el cual nosotros podremos ver ciertas situaciones definidas con tiempo o movimiento de personal y en este caso se realiza la optimización de procesos, integrando la tecnología del software.

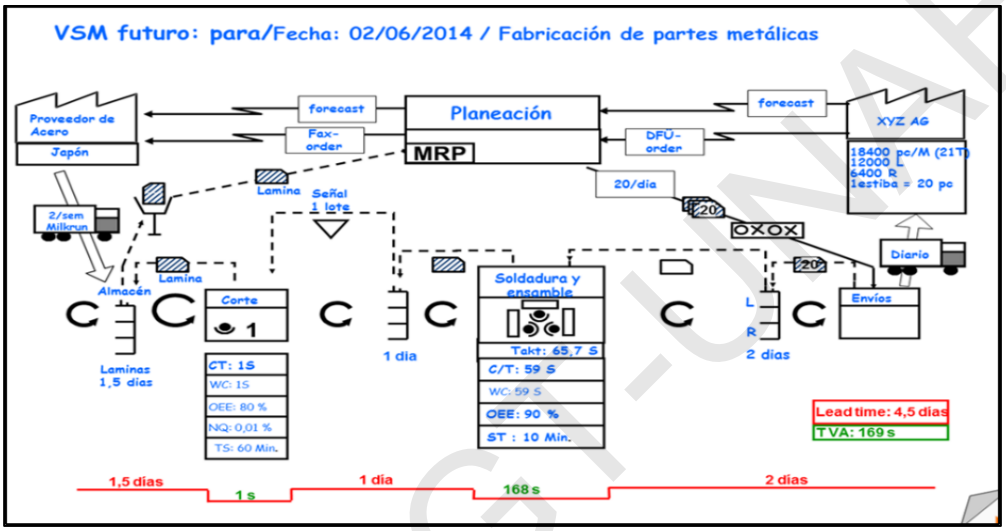


Ilustración 5 MCV Estado Futuro Fuente (Lean Solutions, 2015)

En el Mapa de Corriente de valor de estado futuro (Rother & Shook, 1999) podremos identificar diferentes tipos de optimización de recursos; como lo mencionamos antes, posiblemente una optimización de recursos sería movimiento de un personal o el no traslado de algún tipo de material. En nuestro caso nos interesa saber dónde está la automatización basada en software, una vez identificada la parte donde se va optimizar con software, podemos empezar a desarrollar basados en los tiempos que el mapa de estado actual muestra.

Iniciado este proceso podemos empezar a aplicar dentro de nuestro desarrollo, otra de las herramientas necesarias para un desarrollo de software.

2.2.4.3 Desarrollo de Software con Herramientas Lean.

Según la experiencia detallada en (Poppendieck & Poppendieck, 2003), después de docenas de años de trabajo desarrollando software y muchos años trabajando en plantas manufactureras, decidieron fusionar ambas y como resultado de esto surgió la metodología de desarrollo de software con herramientas Lean.

Según (Oppenheim, 2011) cuando ya se tiene el análisis del Mapa de corriente de valor de estado futuro, es necesario iniciar con el desarrollo del software, donde comenzamos aplicando uno de los principios Lean "Ir y ver por uno mismo".

Identificado el proceso que vamos a desarrollar, lo mejor que podemos hacer es ir y ver por nosotros mismos (Poppendieck & Poppendieck, 2003) para identificar y en base a experiencias, deducir que es lo que necesita realmente el usuario o el cliente, para evitar desperdicios de valor agregado y esfuerzo innecesario.

La experiencia obtenida por el desarrollador y el análisis del ingeniero en jefe, nos ayudará a identificar con más detalle cada uno de los elementos que debemos abordar y cada uno de los elementos en los cuales no nos debemos entretener a la hora de desarrollar el programa.

2.2.4.4 5S Desarrollo y Código.

En el desarrollo es necesario una de las herramientas más populares de la filosofía Lean las 5S.

Las 5S es una herramienta de ordenamiento (Liker & Meier, 2005) lo cual evita el despilfarro o el mal gasto de recursos en cosas innecesarias, misma que al traerla a un desarrollo de software lo aplicamos de la siguiente manera (Bell, 2006) :

1. **Seleccionar:** Vamos a seleccionar todo lo que es necesario para que nuestro desarrollo se realice, en este punto seleccionaremos todo lo fundamental de cada uno de los requerimientos. (Bell, 2006)

2. **Situar:** Vamos a balancear la carga, se realiza con la identificación correcta y asignación de los módulos a desarrollar entre el equipo de trabajo. (Bell, 2006)

3. **Suprimir:** En este paso se suprime todos los procesos innecesarios que se identificaron, debido a que esto nos quitará tiempo y posiblemente sólo nos dejará desperdicio. (Bell, 2006)

4. **Estandarizar:** Cada uno de los desarrollos o cada uno de los procesos necesarios para el desarrollo del programa deberá ser estándar, capaz de ser utilizado por alguien más, elemento básico del desarrollo orientado objetos. (Bell, 2006)

5. **Sustentar:** Cada uno de los módulos realizados por nuestro equipo de producción debe poder irse mejorando continuamente, sin comprometer la calidad. (Bell, 2006)

Luego de la definición de cómo se realizará cada uno de los procesos, se comienza a realizar una segunda capa del 5S (Bell, 2006) orientado al código, aplicando de igual forma: seleccionar situar suprimir estandarizar y sustentar.

Esta segunda capa se realizará a nivel de código, del cual lo que requiere es

1. **Seleccionar:** Sólo lo necesario del código, código correcto y de escritura estándar (Codificación de cada empresa). (Bell, 2006)

2. **Situar:** Si el desarrollo se realizó en módulos, situar cada uno de manera lógica y coherente. (Bell, 2006)

3. **Suprimir:** Eliminar todo el código innecesario, comentarios innecesarios y elementos innecesarios. (Bell, 2006)

4. **Estandarizar:** Definido por la metodología de escritura, declaración de variables, tabulación de etc. (Bell, 2006)

5. **Sustentar:** Por último y muy importante es que cada uno de los elementos agregados después de una entrega, deben de ser desarrollados y estandarizados de la misma forma que lo anterior mencionado. (Bell, 2006)

Por ultimo se llega a la etapa de entrega del producto, la cual se debe realizar aplicando los principios Lean, más Los principios T.P.S. (Liker & Morgan, 2006) logrando la integración de Procesos, Personas y Tecnología.

2.2.5 Metodología de Desarrollo Agile

Las metodologías de desarrollo tradicionales, nos brindan un trabajo robusto y poco flexible; ante los cambios que surgen en el desarrollo de software estas metodologías de desarrollo nos resultan engorrosas, en el caso de que se requiera un cambio de fondo en el desarrollo de software.

En los años 90 surge una relación entre la flexibilidad y calidad de los productos terminados de software, por lo cual se integran los procesos Lean al desarrollo de software (Martin, 1991), esta unión ligada fuertemente a la reducción de costos y a evitar el desperdicio, en este caso de tiempo y recursos, da como resultado nuevas metodologías y enfoques al desarrollo de software.

La relación formada entre la flexibilidad, agilidad y la temática de la filosofía Lean, da como resultado las nuevas metodologías de desarrollo Agile, aunque no se atribuye directamente el enfoque a la filosofía Lean, muchos escritores aducen el surgimiento de estas metodologías a Lean Software Devolopment, es por eso que también son llamada Metodologías de Desarrollo Lean – Agile.

El enfoque fue planteado por primera vez por (Martin, 1991) y se dio a conocer en la comunidad de Ingeniería de Software con el nombre de RAD o Rapid Application Development. RAD, el cual consistía en un entorno de desarrollo altamente productivo, en el que participaban grupos pequeños de programadores utilizando herramientas que generaban código en forma automática tomando como entradas; sintaxis de alto nivel.

La historia de las Metodologías Agiles y su apreciación como tales en la comunidad de la Ingeniería de Software, tiene sus inicios en la creación de una de las metodologías utilizada como arquetipo: XP - eXtreme Programming, que nace de la mente de Kent Beck, tomando ideas recopiladas junto a Ward Cunningham.

Estas metodologías presentaban cambios rápidos y de gran valor por lo cual fueron llamadas Metodologías Ligeras y a las metodologías tradicionales se las llama Metodologías Pesadas.

2.2.6 Manifiesto Agile

El desempeño de las metodologías agiles tiene mucho auge en el mundo de la ingeniería del software, tanto así que se vio la necesidad de tener un manifiesto (Méndez, 2010) , el cual definiría; que es una metodología ágil, que hace, como lo hace y qué se necesita para que una metodología sea tomada como metodología ágil. El manifiesto (Manifiesto por el Desarrollo Ágil de Software, 2001) comienza citando los entornos de desarrollo que se obtienen con la metodología:

- **Al individuo y las interacciones del equipo de desarrollo, sobre el proceso y las herramientas**. (Manifiesto por el Desarrollo Ágil de Software, 2001) La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo; que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.
- **Desarrollar software que funciona, más que conseguir una buena documentación.** (Manifiesto por el Desarrollo Ágil de Software, 2001) La regla a seguir es "no producir documentos, a menos que sean necesarios de forma inmediata, para tomar una decisión importante". Estos documentos deben ser cortos y centrarse en lo fundamental.
- **La colaboración con el cliente, más que la negociación de un contrato.** (Manifiesto por el Desarrollo Ágil de Software, 2001) Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos, será la que marque la marcha del proyecto y asegure su éxito.
- **Responder a los cambios, más que seguir estrictamente un plan.** (Manifiesto por el Desarrollo Ágil de Software, 2001) La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo; por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Los valores anteriormente descritos generan ciertas reglas o principios que se deben considerar para poder lograrlos, estos principios son enumerados en el manifiesto (Manifiesto por el Desarrollo Ágil de Software, 2001) los cuales citamos a continuación:

I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software, que le aporten un valor.
II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
III. Entregar frecuentemente software que funcione desde un par de semanas, a un par de meses, con el menor intervalo de tiempo posible entre entregas.
IV. La gente del negocio y los desarrolladores, deben trabajar juntos a lo largo del proyecto.
V. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
VI. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información, dentro de un equipo de desarrollo.
VII. El software que funciona, es la medida principal de progreso.
VIII. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
IX. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
X. La simplicidad es esencial.
XI. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

2.2.7 Xp – Extreme Programming

(Letelier & Penadés, 2006)La XP Extreme Programming es conocida como la iniciadora de las metodologías Agile, la particularidad con la que inició esta metodología, es la integración del usuario final al equipo de desarrollo, esta particularidad brinda al equipo el entendimiento total de la operación para la cual se está desarrollando el software. Podemos encontrar que esta metodología está basada en:

Pruebas Unitarias: Se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. (Letelier & Penadés, 2006)

Re fabricación: Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio. (Letelier & Penadés, 2006)

Programación en pares: Una particularidad de esta metodología, es que propone la programación en pares; la cual consiste en que dos desarrolladores participen en un proyecto, en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. (Letelier & Penadés, 2006)

Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores
- La simplicidad, al desarrollar y codificar los módulos del sistema
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

La Metodología (Letelier & Penadés, 2006) pretende que con la interacción del usuario final, el análisis o las fallas en el análisis, se vallan cubriendo a medida se desarrolle, se prueba y se re fábrica el software, esto hace que nuestro tiempo de recuperación ante un cambio sea diluido o sea muy corto, ya que el usuario final ha estado en toda la operación y los cambios ya se han venido realizando conforme se desarrolló el software.

2.2.8 Fdd – Feature Driven Development

(Calderón & Rebaza, 2007) La metodología FDD o metodología por rasgos forma parte de la gran gama de metodologías agile, su nacimiento se le atribuye a Jeff de Luca, con la contribución de Peter Coat. Su enfoque es en la realización de iteraciones cortas, las cuales duran dos semanas, durante este tiempo se generan diferentes actividades (Calderón & Rebaza, 2007) las cuales se enumeran a continuación:

- Desarrollar un Modelo Global
- Construir una Lista de los Rasgos

- Planear por Rasgo
- Diseñar por Rasgo
- Construir por Rasgo

La últimas dos actividades son las que se realizan en cada iteración que se requiera. En esta metodología se busca agilizar el trabajo, dividiendo el equipo de trabajo en dos partes:

- Dueños de las clases
- Programadores Jefes

Los cuales tienen actividades muy diferentes pero al final ligadas. Los programadores jefes actúan como líderes de los rasgos; pero no se encargan de la programación, ellos más que todo se encargan del diseño y logística de cada rasgo. Los Dueños de las clases se encargan del desarrollo de las clases que sustentan los rasgos.

2.2.9 Rapid Application Development (Rad)

El RAD iniciado en 1992 por James Martin (Martin, 1991) , es una metodología basada en iteración, entregas y prototipos, la cual se basa en diferentes principios básicos, como ser:

- Objetivo clave: es para un rápido desarrollo y entrega de una alta calidad en un sistema de relativamente bajo coste de inversión.
- Intenta reducir el riesgo inherente del proyecto, partiéndolo en segmentos más pequeños y así proporcionar más facilidad de cambio durante el proceso de desarrollo.
- Orientación dedicada a producir sistemas de alta calidad con rapidez, principalmente mediante el uso de iteración por prototipos (en cualquier etapa de desarrollo), promueve la participación de los usuarios y el uso de herramientas de desarrollo computarizadas. Estas herramientas pueden incluir constructores de Interfaz gráfica de usuario (GUI), Computer Aided Software Engineering (CASE) las herramientas, los sistemas de gestión de bases de datos (DBMS), lenguajes de programación de cuarta generación, generadores de código, y técnicas orientada a objetos.
- Hace especial hincapié en el cumplimiento de la necesidad comercial, mientras que la ingeniería tecnológica o la excelencia es de menor importancia.

- Control de proyecto: implica el desarrollo de prioridades y la definición de los plazos de entrega. Si el proyecto empieza a aplazarse, se hace hincapié en la reducción de requisitos para el ajuste, no en el aumento de la fecha límite.
- En general incluye Joint application development (JAD), donde los usuarios están intensamente participando en el diseño del sistema, ya sea a través de la creación de consenso estructurado en talleres, o por vía electrónica.
- La participación activa de los usuarios es imprescindible.
- Iterativamente realiza la producción de software, en lugar de colgarse de un prototipo.
- Produce la documentación necesaria para facilitar el futuro desarrollo y mantenimiento.

2.2.10 Scrum

Scrum (Calderón & Rebaza, 2007) da sus primeros pasos a la luz en 1986, de un artículo de la Harvard Business Review titulado "The New Product Development Game" de Hirotaka Takeuchi e Ikujiro Nonaka, que introducía las mejores prácticas más utilizadas en 10 compañías japonesas altamente innovadoras, estas en su mayoría ligadas a la filosofía Lean. A partir de ahí y tomando referencias al juego de rugby, Ken Scwaber y Jeff Sutherland formalizan el proceso conocido como Scrum en el año 1995.

Scrum es una metodología fuertemente disciplinada por el PM Project managment, por sobre todas las demás disciplinas de desarrollo.

Al inicio de cada proyecto (Calderón & Rebaza, 2007) se realiza un Product Back Log el cual contiene los requerimientos funcionales y no funcionales deseados.

Al igual que sus hermanos de metodología Agile el Scrum (Calderón & Rebaza, 2007) realiza iteraciones para minimizar los riegos y generar los cambios a cada una de estas iteraciones, estas iteraciones se les conoce como sprint, dentro del cual surgen diferentes cambios, los cuales se llevan en un Sprint BackLog, los cuales son una subdivisión de los Product BackLog que se levantaron previamente con los stakeholders. La duración recomendada de cada Sprint es de un mes.

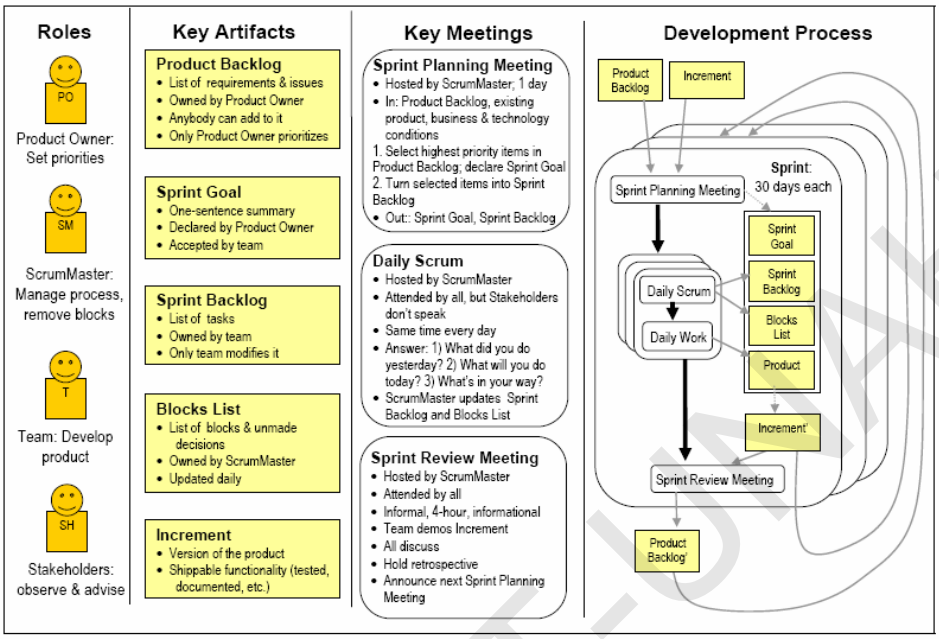


Ilustración 6 Esquema Scrum Fuente (Calderón & Rebaza, 2007)

La ilustración anterior nos muestra la interacción de cada una de las subdivisiones del Scrum, el cual en su desarrollo es dividido de la siguiente forma:

1. Pre-Juego: Planeamiento. El propósito es establecer la visión, definir expectativas y asegurarse la financiación. Las actividades son la escritura de la visión, el presupuesto, el registro de acumulación o retraso (backlog) del producto inicial y los ítems estimados, así como la arquitectura de alto nivel, el diseño exploratorio y los prototipos. El registro de acumulación es de alto nivel de abstracción. (Calderón & Rebaza, 2007)

2. Pre-Juego: Montaje (Staging). El propósito es identificar más requerimientos y priorizar las tareas para la primera iteración. Las actividades son planificación, diseño exploratorio y prototipos. (Calderón & Rebaza, 2007)

3. Juego o Desarrollo. El propósito es implementar un sistema listo para entrega, en una serie de iteraciones de treinta días llamadas "corridas" (sprints). Las actividades son un encuentro de

planeamiento de corridas en cada iteración, la definición del registro de acumulación de corridas y los estimados, y encuentros diarios de Scrum. (Calderón & Rebaza, 2007)

4. Pos-Juego: Liberación. El propósito es el despliegue operacional. Las actividades, documentación, entrenamiento, mercadeo y venta. (Calderón & Rebaza, 2007)

2.2.11 Desarrollo de Software Tradicional.

Existen diferentes modelos de desarrollo de software tradicionales, para fines de estudio tomaremos los tres principales que a continuación describimos según (SommerVille, 2005).

2.2.11.1 El Modelo Desarrollo en Cascada

Éste modelo de desarrollo de software (SommerVille, 2005) es el inicio de diferentes procesos de ingeniería más generales, data de los años 70.

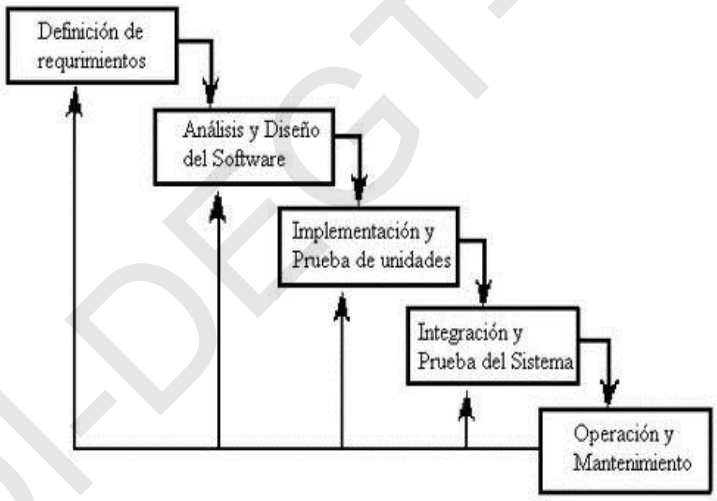


Ilustración 7 Diagrama de Modelo Cascada, Fuente (SommerVille, 2005)

Este es un modelo de desarrollo (SommerVille, 2005) es sencillo y útil; aun y cuando el personal no disponga de mucha experiencia en el área de desarrollo. Es aplicable cuando el problema a abordar es de fácil comprensión y no requiere demasiado discernimiento. El seguimiento es realmente sencillo, se trabaja en base a una definición de requerimientos previos, luego se procede

al diseño del Software, seguidamente se procede con la implementación y pruebas, se prodede con la integración del sistema y por último la operación y mantenimiento.

Se supone que sólo se baja en la cascada; pero también se puede dar marcha atrás y subir los escalones, pero esto requerirá una dificultad en el desarrollo (SommerVille, 2005). El modelo de cascada es utilizado cuando el producto es pequeño, debido a que hasta muy tarde en el proceso se puede ver un producto terminado y un error grave sólo puede ser detectado en las últimas fases. La especificación de requisitos en este modelo debe ser específica y no puede variar debido a que si varía en la mitad del proceso hay que retroceder o volver a iniciar.

Las revisiones de cada uno de los proyectos emprendidos con este modelo requieren de una gran complejidad y son muy difíciles de realizar.

2.2.11.2 El modelo Prototípado

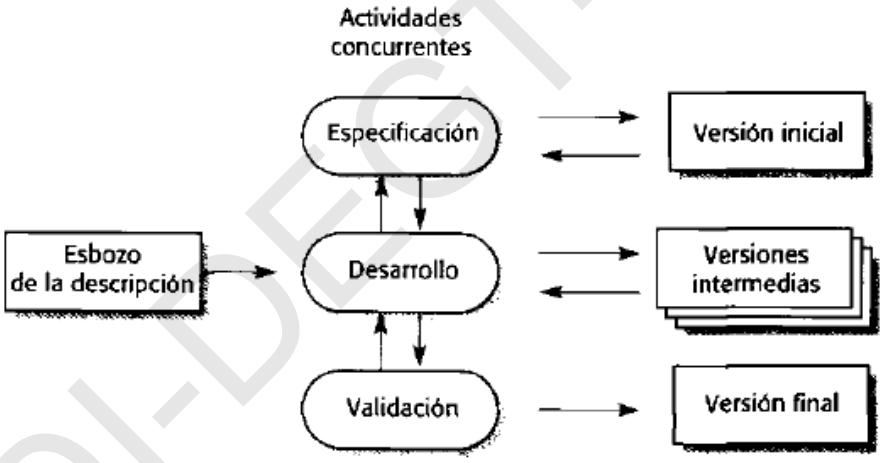


Ilustración 8 Diagrama Modelo Prototipado Fuente (SommerVille, 2005)

Este modelo es un modelo de prueba y error (SommerVille, 2005), en este modelo se parte de un requerimiento previo, luego se hace una entrega al usuario para que este lo revise y anote las correcciones necesarias.

Una vez que se han obtenido estas correcciones se procede a realizar un nuevo prototipo y se entrega de nuevo al usuario, esto se aplica siempre y cuando el usuario final presente nuevos requerimientos.

Este modelo (SommerVille, 2005)se divide en dos grandes ramas:

- Prototipado evolutivo: Que parte de un prototipo inicial y se va afinando progresivamente hasta convertirse en una versión final.
- Prototipado desechable: Es aquel en el cual se parte de un prototipo y cada vez que el usuario entregue nuevas requerimientos sobre el prototipo, este se desecha y se crea un prototipo nuevo.

El modelo prototipado comienza (SommerVille, 2005) con una recolección de requisito. Hay una reunión previa entre desarrolladores y cliente y se fijan los requisitos generales para partir al primer prototipo y luego se van afinando objetivos específicos, a medida que se van entregando programas. Una de las ventajas de este tipo de modelo, es que se van identificando requisitos específicos a medida entregan prototipos, así como también presenta la flexibilidad de probar diferentes tipos de caminos hacia la finalidad del proceso.

El modelo presenta diferentes inconvenientes, ya que como es sabido el cliente nunca sabe lo que quiere, esto puede conllevar a que el prototipado se vuelva un ciclo infinito y el usuario siempre quiera realizar diferentes cambios, debido a esto existe un riesgo de que muchos cambios en el mismo software produzcan que esté bajo de calidad y su objetivo final no se haya cumplido, y este mismo se ramifique en pequeños objetivos que no tienen ninguna relación con el objetivo para el cual se desarrolló.

2.2.11.3 Modelo de proceso e**n** espiral

El modelo del proceso en espiral se origina en 1986 por Barry Boehm.

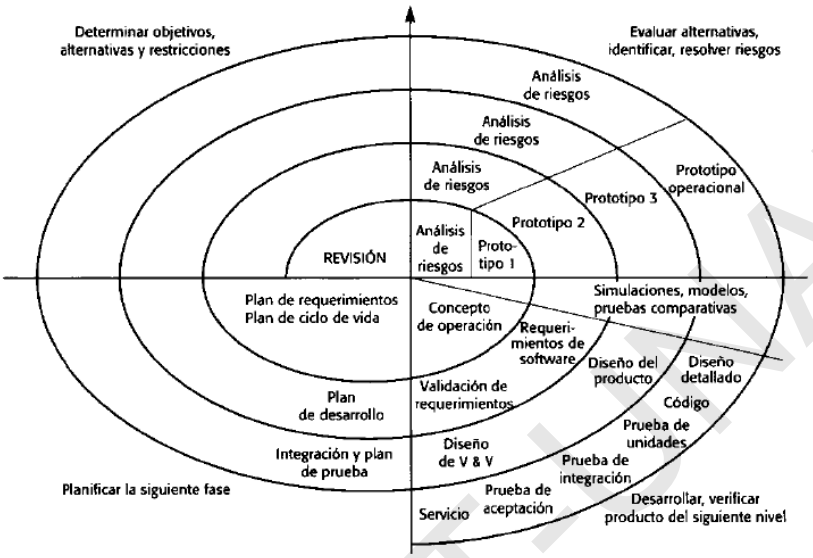


Ilustración 7 Diagrama de Modelo Espiral Fuente (SommerVille, 2005)

Éste modelo trata primero el requerimiento y las actividades de mayor riesgo (SommerVille, 2005), luego de eso, bajo múltiples recorridos trabajan diferentes áreas de cada una de las tareas. Gracias a su forma se pueden cambiar todo tipo de actividades en el desarrollo y asimismo entregar cada uno de los prototipos hasta que éste se ha completado.

Este tipo de modelo representa bastantes fortalezas, en cuanto la gestión de riesgo se refiere, de hecho el modelo original de Boehm se define como:

- Fijar objetivos
- Gestionar y reducir riesgos
- Desarrollo y validación
- Planificar siguiente ciclo

Este mismo proceso puede tornarse evolutivo, creando cada vez un nuevo espiral y al finalizar cada uno de los espirales se desarrolla la revisión del proyecto, seguidamente se toma la decisión de crear una nueva vuelta o finalizar el proyecto. Una de las ventajas al aplicar este proceso es que su enfoque casi siempre va de la mano con lo que está sucediendo en la empresa o con el proyecto.

Centra su atención en usar componentes ya creados, e ir eliminando errores en cada una de las vueltas, mejorando la calidad del producto.

Gran parte de los inconvenientes de utilizar esta metodología es que a veces se torna un enfoque incontrolable y requiere de mucha experiencia para la identificación de riesgos, asimismo requiere un gran discernimiento para lograr definir cúando el espiral debe determinar.

2.2.12 Desarrollo de Software Lean - Agile vs. Desarrollos de Software Tradicional

Según (Poppendieck & Poppendieck, Lean Software Development: An Agile Toolkit, 2003) la metodología de desarrollo de software lean - agile representa mucha mejoras, en relación al desarrollo de software tradicional.

Los desarrollos de software lean - agile se basan en la industria, lo cual refiere un producto de bajos costos, mucha calidad y agilidad o velocidad en producción, a diferencia del desarrollo de software tradicional, el cual se basa simplemente en sacar un producto sin importar tiempo o costos.

Según (Poppendieck & Poppendieck, Lean Software Development: An Agile Toolkit, 2003) el potencial de este hibrido como lo es el Desarrollo de software lean – agile es una metodología basada en conocimientos experiencias y la mejora continua, lo cual lo hace una metodología robusta y capaz de ir cambiando a medida se presente diferentes requerimientos en diferentes plataformas, el desarrollo de software tradicional, es una metodología que podría decirse se quedó obsoleta, ya que su forma de trabajo no cambió desde sus inicios y al mundo cambiante de la tecnología de hoy poco o nada pueden hacerle frente.

En (Poppendieck & Poppendieck, Lean Software Development: An Agile Toolkit, 2003) se hace mención de un ejemplo claro del enorme potencial del desarrollo de software con herramientas lean:

"Tradicionalmente el ingeniero se estresa por una pequeña decisión en algún proyecto y no puede continuar, pierde tiempo y el objetivo no se cumple; enorme diferencia cuando se tiene mucha información de un proceso, si es visto como un todo, si se cuenta con la experiencia de todo el proceso y no solo de una parte en particular y a todo esto le suma la robustez industrial"

El desarrollo de software Lean - Agile (Poppendieck & Poppendieck, Lean Software Development: An Agile Toolkit, 2003) obliga al desarrollador a conocer todo el proceso, e integrar a los usuarios en cada una de sus iteraciones o desde el inicio hasta fin del proyecto, con esto se logra obtener su experiencia para obtener la mayor información posible. En el desarrollo de software tradicional solo se conocen ciertas partes del proceso, por lo cual su alcance y solvencia de este tipo de problemas es limitado. En los tiempo de respuesta o finalización de un proyecto, cuando se desarrolla software tradicional, siempre se tienen comentarios como los siguientes: "Nunca está a tiempo", "Siempre tarde" etc. cada uno de los cometarios, recalcando que el tiempo que se estimó no era el adecuado, que siempre salieron los productos tarde, además de agregar comentarios como: "esto no funciona como debería", "nada de esto me sirve", lo cual hace constar que la calidad no es buena y el objetivo a cubrir no se cumplio. En los desarrollos de software con lean - agile se tiene un panorama y una administración del desarrollo igual que la de una maquina industrial (Garzas, 2015) , esto debido a la robustez adquirida por su herencia industrial, por lo cual estos comentarios quedan en el olvido, además de que el usuario se vuelve parte del proyecto; por lo cual el objetivo siempre se cumplirá.

Según (Poppendieck & Poppendieck, Lean Software Development: An Agile Toolkit, 2003) los métodos de desarrollo estándar quedaron como fundamento y ya es la época de utilizar todo el conocimiento que generaron, y darle fuerza con las aplicaciones industriales y utilizar metodologías capaces de brindar calidad, bajos costos y tiempos de entrega altamente cumplibles para lo cual nacieron las Metodologías de desarrollo Lean Agile.

CAPITULO III. ENFOQUE Y TIPO DE INVESTIGACIÓN

3.1 Enfoque de investigación

La investigación a desarrollar basados en (Sampieri, Collado, & Lucio, 2010) es cuantitativa, entendiéndose que se realizará bajo un conjunto de procesos secuenciales que serán probatorios, donde cada una de las actividades o etapas serán ordenadas con el objetivo de que una etapa preceda a otra, manteniendo una obligatoriedad de realización en cada actividad, esta investigación cuantitativa delimita un problema del cual se realizan preguntas, objetivos, construcción de un marco o perspectiva teórica y finalizando con la creación y comprobación de hipótesis, mediante la medición de variables definidas y analizadas por una estrategia o plan establecido.

La investigación se realizará enfocados en comprobar hipótesis, para lo cual se aplicara un instrumento mediante el cual se obtendrán los datos, seguidamente se desarrollará un análisis sobre los mismos, estos análisis se utilizara como recursos sustentador para aceptar o rechazar las hipótesis.

3.2 Tipo de Investigación

El tipo de investigación que se abordara será correlacional-causal, debido a que lo que se pretende estudiar es la relación entre dos variables, como lo describe (Sampieri, Collado, & Lucio, 2010) la investigación correlacional-causal es aquella donde una variable X está relacionada directamente con una variable Y, ejerciendo sobre esta una causa o efecto de cambio.

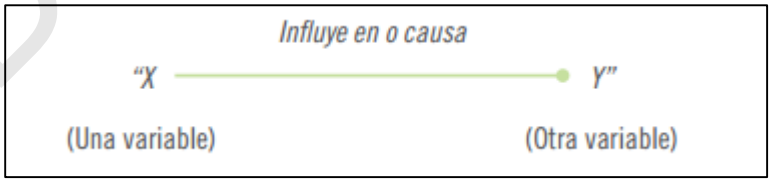


Ilustración 8 Diagrama Correlacional - Causal Fuente (Sampieri, Collado, & Lucio, 2010)

En la presente investigación se tomaran como variables; la Metodología de desarrollo, el tiempo y los costos, formando una relación causa efecto entre las mismas, de la siguiente forma:

Variable X	Relación	Variable Y
Metodología de Desarrollo	Influye o causa	Tiempo de Desarrollo
Metodología de Desarrollo	Influye o causa	Costos de Desarrollo

Tabla 1 Relación entre variables Fuente (Construcción Propia)

CAPITULO IV. HIPOTESIS Y VARIABLES

4.1 Hipótesis

Las hipótesis planteadas para la presente investigación son de tipo correlacional-causal debido a que investigación es del mismo tipo, así mismo cabe mencionar que en cada hipótesis planteada se realiza una relación entre dos o más variables.

1. La incidencia de las metodologías Lean – Agile en el desarrollo de software produce mejoras en el tiempo de desarrollo.

2. La incidencia de las metodologías Lean – Agile en el desarrollo de software produce mejoras en el costos de desarrollo.

4.2 Variables

Nombre	Descripción	Tipo	Indicador	Instrumentación
Metodología de Desarrollo	Metodología de desarrollo utilizada en un proyecto de Desarrollo de software.	Independiente	Metodología	Se definirá mediante las preguntas 3,4 y 5.
Tiempo de desarrollo.	Tiempo que se tardó todo un proyecto de desarrollo de software.	Dependiente	Tiempo en horas.	Se definirá mediante la pregunta 6, en la cual se expondrán diferentes ítems, de los cuales se definirá por la selección una de las opciones de una variantes de la escala de Likert:

				Desmejoro Notablemente, Desmejoro, No Hubo Cambios, Mejoro, Mejoro Notablemente.
Costos de desarrollo.	Costos en los que se incurre en un proyecto de desarrollo de software.	Dependiente	Costos Monetarios	Se definirá mediante la pregunta 7, en la cual se expondrán diferentes ítems, de los cuales se definirá por la selección una de las opciones de una variantes de la escala de Likert: Desmejoro Notablemente, Desmejoro, No Hubo Cambios, Mejoro, Mejoro Notablemente

Tabla 2 Definición de Variables Fuente (Construcción Propia)

4.3 Relación entre variables

4.3.1 Diagrama Sagital

El diagrama sagital que a continuación se muestra, detalla la relación que existe en las variables de la presente investigación

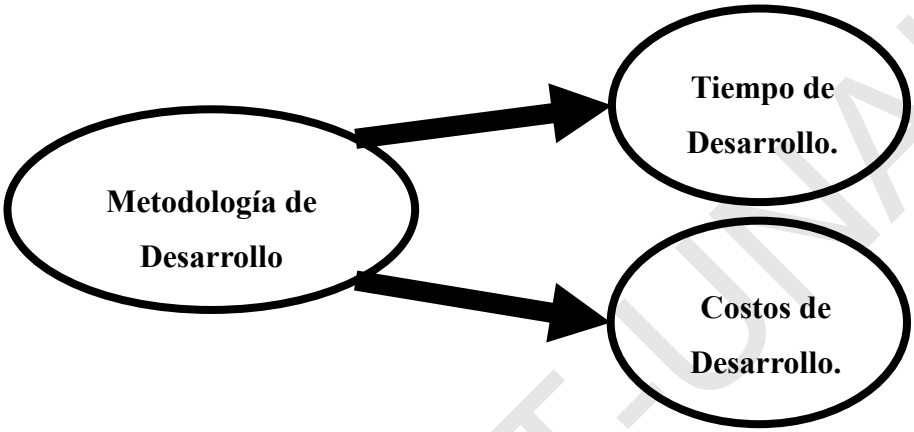


Ilustración 9 Diagrama Sagital, Correlacional-Causal Fuente (Construcción Propia)

4.4 Operacionalización de las Variables

Nombre	Operacionalización
Metodología de Desarrollo	Metodología de desarrollo utilizada. Para el presente documento se establecieron dos posibles metodologías: Metodologías tradiciones o pesadas y Metodologías Lean-Agile
Tiempo de desarrollo.	Tiempo en el que se incurre en la creación de un programa, siguiendo las etapas de un proyecto de software descritas en el instrumento de investigación.
Costos de desarrollo.	Costos Monetarios en los que se incurre en la creación de un programa, siguiendo las etapas de un proyecto de software descritas en el instrumento de investigación.

Tabla 3 Operacionalización de Variables Fuente (Construcción Propia)

CAPITULO V. ESTRATEGIA METODOLÓGICA

5.1 Diseño de la Investigación

El diseño de la investigación según (Sampieri, Collado, & Lucio, 2010) es un plan o estrategia que se realiza bajo el enfoque de cumplir con los siguientes objetivos:

- Responder las preguntas de investigación.
- Someter a prueba cada una de las hipótesis.
- Cumplir con los objetivos de dicha investigación.

El diseño de la investigación será del tipo Transeccionales correlaciónales/causales, debido a que lo que se pretende es correlacionar dos variables en un evento determinado, según (Sampieri, Collado, & Lucio, 2010) Transeccional o transversal, es el diseño de investigación que se aplica cuando se recolecta información en un momento dado, este tipo de diseño se sub divide en dos grandes ramas; los descriptivos y los correlacionales/causales, siendo los descriptivos, los que como su nombre lo indica, describen las variables y los correlacionales/causales; la relación que existe entre las variables.

El evento para la presente investigación se genera en la incidencia de las Metodologías Lean-Agile en el desarrollo de software, causando un efecto o cambio en los costos y tiempos del mismo, definiendo este evento, la relación entre las variables de estudio y los efectos de esta incidencia.

5.2 Población Muestra y Muestreo

5.2.1 Delimitación de la Población

La población para esta investigación como lo aborda (Sampieri, Collado, & Lucio, 2010) serán aquellas entidades; ya sean personas, sucesos o comunidades de estudios, los cuales posean las capacidades de brindar información relevante para la investigación, la cual sujeta a análisis poseerá la capacidad de brindar respuestas confiables a cada una de las preguntas de investigación abordadas.

Fundamentados en lo descrito anteriormente, definimos nuestra población de acuerdo a la conveniencia de esta investigación, situando los participantes como todos aquellos profesionales que han estado vinculados al desarrollo de software, de estos específicamente todos los que se encuentren en tareas de gestión o manejo de proyectos de software, de lo cual definiremos como población todos aquellos Jefes, Lideres o Gerentes de Desarrollo de software o puestos a fines, esto debido a que es necesario que posean experiencia en desarrollo y además que se encuentren actualmente manejando procesos similares.

Como se definió en la sección anterior estos profesionales deben de estar actualmente empleados, por lo cual se tomaran empresas que actualmente se encuentren activas; se ha elegido como población meta las empresas de la ciudad de Tegucigalpa M.D.C, dentro de esta limitante geográfica cabe agregar que no todas las empresa que se encuentran en dicha ciudad, poseen un departamento de tecnología con una sección de desarrollo o con capacidades de contratar este tipo de trabajos, por lo cual de todas las empresas de la ciudad en cuestión, se tomaron solo las catalogadas como "Grandes Contribuyentes" por parte de Dirección Ejecutiva de Ingresos (D.E.I.).

Se limita la población hacia este tipo de empresas, porque son las que conforman el grupo de grandes empresas de Honduras y en este caso de Tegucigalpa, con lo cual estamos seleccionando empresas que posean un departamento de Tecnología o similar, dentro de su organigrama, con lo cual aseguraremos que exista el manejo de requerimientos de software, ya sea para manejo interno o externo en la organización, y así mismo se asegura de que cada requerimiento como tal, sea manejado por una entidad interna, que será la encargada de gestionar su desarrollo.

5.2.2 Tamaño de la Muestra

La muestra que se tomara de acuerdo a la población seleccionada se calculara basados en (Roberto Hernandez Sampieri, 2010), el cual define para el tipo de investigación abordada, un muestreo probabilístico simple, el cual según la recomendación de (Sampieri, Collado, & Lucio, 2010) los tamaños mínimos para dicha investigación se definen a continuación:

Tipo de Estudio	Tamaño Mínimo de Muestra
Transeccional descriptivo o correlacional	30 casos por grupo o segmento del universo.
Encuesta a gran escala	100 casos para el grupo o segmento más importante del universo y de 20 a 50 casos para grupos menos importantes
Causal	15 casos por variable independiente
Experimental o cuasi experimental	15 por grupo

Tabla 4 Tamaños Mínimos de Muestra por Tipo de Estudio Fuente (Sampieri, Collado, & Lucio, 2010)

Como describe la tabla anterior, se tomaran 15 casos por variable independiente, en el caso de la presente investigación, solo se cuenta con una variable independiente la cual es Metodología de Desarrollo, por lo cual para calcular el tamaño de la muestra que utilizaremos, definiremos el tamaño de muestra adecuado, como aquel que nos dé un resultado de 15 o más participantes.

A continuación se aborda el cálculo de la muestra que se utilizara para el desarrollo de la investigación.

Tamaño de la población: Según la Dirección Ejecutiva de Ingresos en el acuerdo N. DEI-SG-043-2011 la cantidad de Grandes Contribuyentes en Honduras está conformado por 621 empresas, para el departamento de Francisco Morazán se conforman un total de 299 empresas.

Error Máximo aceptable: 5%

El error máximo aceptable fue seleccionado basados en (Sampieri, Collado, & Lucio, 2010) el cual define que los errores más utilizados en este tipo de investigaciones se encuentran en el rango de 1 a 5%, además el cálculo que se realizó con el presente error, genera la cantidad de participantes mayores a lo mínimo establecido. En la Tabla 4 Tamaños Mínimos de Muestra por Tipo de Estudio Fuente (Sampieri, Collado, & Lucio, 2010).

Probabilidad de Ocurrencia: 95%

La probabilidad de ocurrencia según (Sampieri, Collado, & Lucio, 2010), es el complemento al error esperado, por consiguiente el error máximo aceptable es de 5% su complemento lo conforma el 95% restante.

Por lo que, según (Sampieri, Collado, & Baptista, 1991); la muestra a evaluar se calcula de la siguiente formula:

$$n' = \frac{s^2}{V} = \frac{p(1-p)}{Se^2}$$

Formula 1 Calculo del tamaño provisional Muestra Fuente (Sampieri, Collado, & Lucio, 2010)

Tamaño provisional de la muestra, Donde:
n' = Tamaño provisional de la muestra
s^2 = varianza muestral
V = varianza poblacional
p = porcentaje de confiabilidad
Se = error estándar

Tabla 5 Definición de Formula tamaño provisional Muestra Fuente (Sampieri, Collado, & Lucio, 2010)

$$n = \frac{n'}{1 + \frac{n}{N}}$$

Formula 2 Calculo Tamaño de la Muestra Fuente (Sampieri, Collado, & Lucio, 2010)

Donde
n = tamaño de la muestra
n' = Tamaño provisional de la muestra
N = tamaño de la población

Tabla 6 Definición de la Formula tamaño de la muestra Fuente (Sampieri, Collado, & Lucio, 2010)

Para la presente investigación los datos a utilizar son los siguientes:

$$N = 299$$

$$\bar{y} = 1$$

$$Se = 0.05$$

$$p = 0.95$$

$$V = Se^2$$

Entonces

$$n' = \frac{s^2}{V} = \frac{p(1-p)}{Se^2} = \frac{0.95(1-0.95)}{0.05^2} = \frac{0.0475}{0.0025} = 19$$

y

$$n = \frac{n'}{1 + \frac{n}{N}} = \frac{19}{1 + \frac{19}{299}} = 17.86 \sim 18$$

La muestra a tomar de esta población de empresas será de 18.

5.2.3 Tipo de muestra

Apegados al desarrollo de la presente investigación, sobre la cual se tomaran decisiones en base a las hipótesis planteadas, lo cual por ende afectara los resultados de la investigación, se seleccionara un tipo de muestra probabilística simple, ya que es según (Sampieri, Collado, & Lucio, 2010) la que se apega al presente tipo de investigación.

5.3 Recolección de Datos

5.3.1 Instrumento de medición

La comprobación de las Hipótesis nos obliga a realizar una medición de las variables, las cuales se medirán a través de un instrumento de medición de tipo encuesta, siendo este aplicado a autoridades y líderes en el desarrollo de software en empresas que tengan entre su organigrama una entidad

dirigida al desarrollo de software y que en el mismo se tenga experiencia aplicando las Metodologías de Desarrollo Lean-Agile.

Como se ha venido mencionando en los objetivos de la presente tesis y sobre las cuales se basan las hipótesis y preguntas respectivas, lo que se pretende es medir si la aplicación de Métodos de Desarrollo Lean-Agile mejora los tiempos de respuesta y costos en el desarrollo de software, y en base a esta premisa se realizó el documento en cuestión.

El instrumento que se agrega en el anexo 1, se realiza sobre la estructura de un modelo genérico de desarrollo de software, el cual se apega en su generalidad a las metodologías pesadas de desarrollo y también a las metodologías agiles. Se definen a continuación cada etapa en el desarrollo del software, para definir el impacto de las metodologías agiles desde un enfoque de tiempos y costos.

Este Instrumento está dirigido a profesionales del desarrollo de software, los cuales deben de poseer conocimientos y experiencias, tanto de metodologías pesadas, como agiles y que actualmente se desempeñen en un cargo de jefatura, director o líder de proyectos de software o gerenciamiento de un Departamento de Desarrollo de Software.

En la tabla Instrumento por variables (ver anexo 4), se describe la forma en que las variables, para la comprobación de las hipótesis, se abordan en el instrumento de medición.

5.3.2 Objetivos del Instrumento de Medición

1. Evaluar el efecto que la incidencia de las Metodología Lean - Agile generan en los costos de desarrollo de software.

2. Evaluar el efecto que la incidencia de las Metodología Lean - Agile generan en los tiempos de entrega de desarrollo de software.

5.3.3 Recolección de datos

La recolección de datos se llevara a cabo realizando la aplicación del instrumento a la cantidad estipulada de la muestra, con esto podremos validar que se abordó la totalidad de la muestra, retornándonos una mayor confianza en los datos recolectados.

5.3.4 Validez del instrumento

La validación del instrumento según (Sampieri, Collado, & Lucio, 2010) es el valor de seguridad o exactitud con el que el instrumento evalúa la o las variables de una investigación.

Para la validación del presente instrumento se aborda la revisión por parte de expertos (Sampieri, Collado, & Lucio, 2010) (Anexo 2) en la materia de Desarrollo de Software, los cuales aprobaron y validaron, al mismo tiempo que retroalimentaron sobre los efectos del instrumento en la presente investigación.

Para realizar determinada tarea, se realizó un instrumento donde el experto solo debía marcar si una pregunta o ítem es válida, enfocándose en los criterios de validez para determinada pregunta, según el método de Lawshe, determinando su índice de validez de la siguiente forma:

$$IVC = P/N$$

Formula 3 Índice de Validez por Expertos Fuente (Sampieri, Collado, & Lucio, 2010)

IVC Índice de Validez, Donde :
P = Resultados de Valides o Pertinentes
N = Numero de Expertos o Jueces

Tabla 7 Definición de Formula Validez determinada por Expertos Fuente (Sampieri, Collado, & Lucio, 2010)

La siguiente tabla nos muestra los resultados obtenidos por los expertos en nuestro instrumento.

Validación del Instrumento por Expertos										
Experto/Ítems	**1**	**2**	**3**	**IVC**		**Experto/Ítems**	**1**	**2**	**3**	**IVC**
1	0	1	1	0.666666667		38	1	1	1	1
2	0	1	1	0.666666667		39	1	1	1	1
3	1	1	1	1		40	0	1	1	0.666666667
4	1	1	1	1		41	1	1	1	1
5	1	1	1	1		42	1	1	1	1
6	1	1	1	1		43	1	1	1	1
7	1	1	1	1		44	1	1	1	1
8	1	1	1	1		45	1	1	1	1
9	1	1	1	1		46	1	1	1	1
10	1	1	1	1		47	1	1	1	1
11	1	1	1	1		48	1	1	0	0.666666667
12	1	1	1	1		49	1	1	1	1
13	1	1	1	1		50	1	1	1	1
14	1	1	1	1		51	1	1	1	1
15	1	1	1	1		52	1	1	1	1
16	1	1	1	1		53	1	1	1	1
17	1	1	1	1		54	1	1	1	1
18	1	1	1	1		55	1	1	1	1
19	1	1	1	1		56	1	1	1	1
20	1	1	1	1		57	1	1	1	1
21	1	1	1	1		58	1	1	1	1
22	1	1	1	1		59	1	1	1	1
23	1	1	1	1		60	1	1	1	1
24	1	1	1	1		61	1	1	0	0.666666667
25	1	1	1	1		62	1	1	1	1

26	1	1	1	1		63	1	1	1	1
27	1	1	1	1		64	1	1	1	1
28	1	1	1	1		65	1	1	1	1
29	1	1	1	1		66	1	1	1	1
30	1	1	1	1		67	1	1	1	1
31	1	1	1	1		68	1	1	1	1
32	1	1	1	1		69	1	1	1	1
33	1	1	1	1		70	1	1	1	1
34	1	1	1	1		71	1	1	1	1
35	1	1	1	1		72	1	1	1	1
36	1	1	1	1		73	1	1	1	1
37	1	1	1	1		74	1	1	1	1
						75	1	1	0	0.666666667

Tabla 8 Validación Instrumento por Expertos Fuente: Construcción Propia

Como podemos ver todas las preguntas fueron aceptadas como válidas o pertenecientes, según los expertos abordados, con los resultados obtenidos podemos concluir que el instrumento es válido.

5.3.5 Confiabilidad del instrumento

Loa confiabilidad de un instrumento según (Sampieri, Collado, & Lucio, 2010) se refiere al grado que posee el instrumento para mantener su información, si este se aplicase repetidamente a un mismo participante. Para verificar la confiabilidad del instrumento se realizó el cálculo bajo la metodología de Alfa Cronbach, misma que según (Sampieri, Collado, & Lucio, 2010) es la más utilizada, para su evaluación se realizó la aplicación del instrumento a 5 profesionales del Desarrollo del software los cuales nos reflejaron una muestra de los posibles resultados de la aplicación del mismo.

Para la interpretación del coeficiente del alfa de Cronbach, utilizaremos la escala descrita por (Sampieri, Collado, & Lucio, 2010) la cual se detalla a continuación:

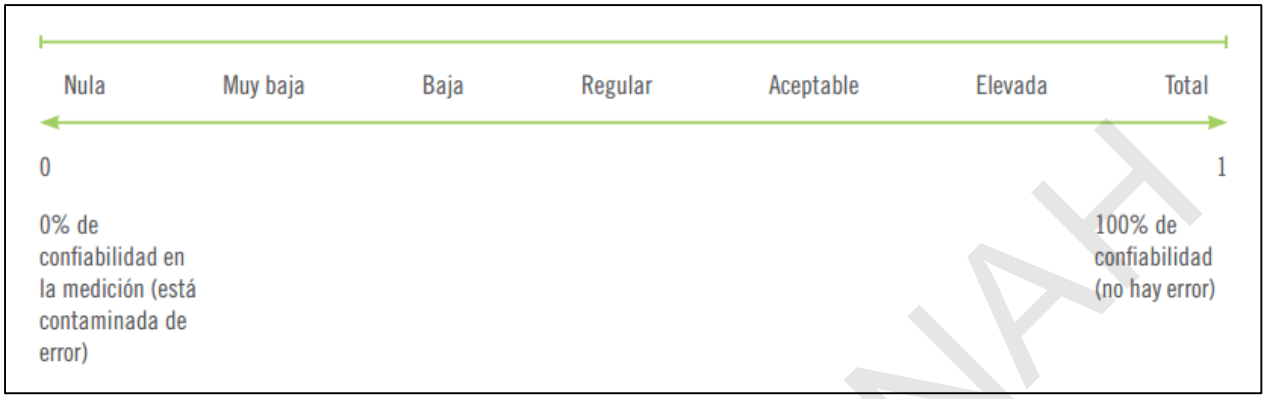


Ilustración 10 Criterios Alfa de Fuente Cronbach (Sampieri, Collado, & Lucio, 2010)

El cálculo se realizó tanto a cada variable, como al instrumento total, obteniendo los siguientes resultados:

La fórmula descrita para el cálculo es la siguiente:

$$\alpha = \frac{K}{K-1}\left[1 - \frac{\sum S_i^{\,2}}{S_T^{\,2}}\right]$$

Formula 4 Alfa de Cronbach Fuente (Sampieri, Collado, & Lucio, 2010)

Aplicando los cálculos a cada una de las variables abordadas por el instrumento, mismas que serán el sustento para concluir sobre las hipótesis planteadas, se obtuvieron los siguientes resultados.

Confiabilidad del instrumento Variable Metodología de Desarrollo

K:	El número de ítems	3
S Si2 :	Sumatoria de las Varianzas de los Ítems	0.32
ST2 :	La Varianza de la suma de los Ítems	0.64
a :	Coeficiente de Alfa de Cronbach	0.75

Tabla 9 Coeficiente Alfa de Cronbach Variable Metodología de Desarrollo. Fuente: Construcción Propia

Confiabilidad del instrumento Variable Tiempo de Desarrollo

K:	El número de ítems	36
S Si2 :	Sumatoria de las Varianzas de los Ítems	5.12
ST2 :	La Varianza de la suma de los Ítems	16.96
a :	Coeficiente de Alfa de Cronbach	0.72

Tabla 10 Coeficiente Alfa de Cronbach Variable Tiempo de Desarrollo. Fuente: Construcción Propia

Confiabilidad del instrumento Variable Costos de Desarrollo

K:	El número de ítems	36
S Si2 :	Sumatoria de las Varianzas de los Ítems	6.40
ST2 :	La Varianza de la suma de los Ítems	64
a :	Coeficiente de Alfa de Cronbach	0.93

Tabla 11 Coeficiente Alfa de Cronbach Variable Costos de Desarrollo. Fuente: Construcción Propia

Tomando como fundamento los resultados obtenidos, y utilizando los criterios descritos en la Ilustración 10 Criterios Alfa de Fuente Cronbach (Sampieri, Collado, & Lucio, 2010), se puede concluir, que la confiabilidad del instrumento en cada una de las variables es aceptable, ya que el Coeficiente Alfa de Cronbach es cercano a 1.

Para el instrumento completo se obtuvieron los siguientes resultados.

K:	El número de ítems	75
S Si2 :	Sumatoria de las Varianzas de los Ítems	11.84
ST2 :	La Varianza de la suma de los Ítems	125.76
a :	Coeficiente de Alfa de Cronbach	0.92

Tabla 12 Coeficiente de Alfa Cronbach Instrumento Total. Fuente: Construcción Propia

Según el resultado obtenido del cálculo de coeficiente de Alfa CronBach del instrumento de investigación y en comparación del mismo con los criterios descritos en Ilustración 10 Criterios

Alfa de Fuente Cronbach (Sampieri, Collado, & Lucio, 2010) , concluimos que la confiabilidad del documento es válida.

En anexo 3 se detalla los cálculos y tabulación de las encuestas aplicadas.

5.3.6 Codificación del Instrumento

El instrumento a utilizar se codifico de acuerdo a las preguntas que nos brindaran resultados concretos para la comprobación de la presente investigación, la tabla de codificación se describe en el anexo 5.

CAPITULO VI. PLAN DE ANÁLISIS

El análisis de los datos obtenidos se realizará de forma ordenada y estructurada bajo el siguiente plan:

6.1 Relación de las variables

Para determinar si existe una correlación ente las variables, se recurrirá como lo explica (Sampieri, Collado, & Lucio, 2010) a calcular el coeficiente de correlación de Pearson o coeficiente de correlación lineal, el cual denominaremos por "r", el cual genera valores entre -1 < r < 1,

El cálculo se realizará bajo la siguiente formula:

$$r = \frac{\frac{1}{n} * \sum(x_i - x_m) * (y_i - y_m)}{(\left(\frac{1}{n} * \sum(x_i - x_m)^2\right) * (\frac{1}{n} * \sum(y_i - y_m)^2))^{\frac{1}{2}}}$$

Formula 5 Coeficiente de Pearson Fuente (Sampieri, Collado, & Lucio, 2010)

Donde:
n = número de observaciones.
X_i = valor de la variable x por ítem.
X_m = media de la variable X.
Y_i = valor de la variable y por ítem
Y_m = media de la variable y.

Tabla 13 Definición de la formula Coeficiente de Pearson Fuente (Sampieri, Collado, & Lucio, 2010)

Para su interpretación de acuerdo al signo del resultado se aplica el siguiente criterio:

- Para -1 < r < 0 nos indica que existe una correlación lineal negativa, (Sampieri, Collado, & Lucio, 2010) la define como "A mayor X, menor Y", de manera proporcional. Es decir,

cada vez que X aumenta una unidad, Y disminuye siempre una cantidad constante. Esto también se aplica "a menor X, mayor Y".

- Para 0 < r < 1 nos indica que existe una correlación positiva, (Sampieri, Collado, & Lucio, 2010) la define como "A mayor X, mayor Y" o "a menor X, menor Y", de manera proporcional. Cada vez que X aumenta, Y aumenta siempre una cantidad constante.
- Para r = 0 no existe una correlación lineal entre las variables.

Es Importante hacer notar que para la presente investigación, como lo denota (Sampieri, Collado, & Lucio, 2010) El signo indica la dirección de la correlación (positiva o negativa); y el valor numérico, la magnitud, por tal razón en nuestro caso nos centraremos en la magnitud.

La interpretación de la magnitud del coeficiente de Pearson según (Sampieri, Collado, & Lucio, 2010) es de la siguiente forma:

Coeficiente de Pearson "r"	Interpretación
r = –1.00	Correlación negativa perfecta. ("A mayor X, menor Y", de manera proporcional. Es decir, cada vez que X aumenta una unidad, Y disminuye siempre una cantidad constante.) Esto también se aplica "a menor X, mayor Y".
-1 > r <= –0.90	Correlación negativa muy fuerte.
-0.90 > r <= –0.75	Correlación negativa considerable.
-0.75 > r <= –0.50	Correlación negativa media.
-050 > r <= –0.25	Correlación negativa débil.
-0.25 > r <= –0.10	Correlación negativa muy débil.
-0.10 > r <= -0.01	Correlación negativa muy débil.
r = 0	No existe correlación alguna entre las variables.
0 .01 > r <= 0.10	Correlación positiva muy débil.

0.10 > r <= 0.25	Correlación positiva muy débil.
0.25 > r <= 0.50	Correlación positiva débil
0.50 > r <= 0.75	Correlación positiva media.
0.75 > r <= 0.90	Correlación positiva considerable.
0.90 > r <= 0.99	Correlación positiva muy fuerte.
r = 1	Correlación positiva perfecta. ("A mayor X, mayor Y" o "a menor X, menor Y", de manera proporcional. Cada vez que X aumenta, Y aumenta siempre una cantidad constante.)

Tabla 14 Interpretación Coeficiente de Pearson magnitud Fuente (Sampieri, Collado, & Lucio, 2010)

6.2 Tabulación de los datos

La tabulación de los datos se realizará a través de una hoja de cálculo utilizando el programa Microsoft Excel, en él se recurrirá a la codificación del documento para identificar las preguntas, este código de pregunta representara una columna y delante de la misma se situará la tabulación de los resultados de cada una de las encuestas aplicadas.

Las respuestas de cada encuesta se valoraran de acuerdo al plan siguiente

Categoría	Código
SI	1
NO	2
Metodologías Pesadas (Tradicionales)	1
Metodologías Lean-Agile	2
Desmejoro Notablemente	1
Desmejoro	2
No Hubo Cambios	3
Mejoro	4
Mejoro Notablemente	5

Tabla 15 Plan de Análisis, Fuente: Construcción Propia

6.3 Análisis de los datos por Pregunta

Una vez que se encuentren agrupadas las respuestas, se abordará un análisis por pregunta, el cual denotará los puntos más importantes a tomar en cuenta de cada una, para los casos de las preguntas 6 y 7 el análisis se realizará por categoría y dentro de este análisis se elaborará una descripción por cada ítem que lo conforma, así como un análisis total de la categoría, se elaborarán conclusiones en cada categoría, con fundamento en los resultados expuestos.

6.4 Análisis de los datos por Variables

Al finalizar la tabulación de los datos se procederá a agrupar las respuestas por cada variable, utilizando el documento de codificación del instrumento se identificaran que grupo de preguntas pertenecen a cada variable y seguidamente se procederá a realizar las sumatorias y a realizar un promedio de cada una de las respuestas por pregunta, las cuales se analizaran una a una.

6.5 Presentación de resultados

La presentación de resultados, es la representación de los análisis anteriormente descritos y se realizará considerando cada uno de los análisis planteados, la presentación se elaborará de la siguiente forma:

- Presentación de resultados por pregunta.
    - Presentación de resultados por categoría en los casos de la pregunta 6 y 7
- Presentación de resultados por variable.

CAPITULO VII. ANÁLISIS DE RESULTADOS

7.1 Relación de las Variables

Para fundamentar la relación entre variables, como se describió en el plan de análisis se procederá a calcular el coeficiente de relación de Pearson o coeficiente de relación lineal, del cual su fórmula es la siguiente:

$$r = \frac{\frac{1}{n} * \sum(x_i - x_m) * (y_i - y_m)}{((\frac{1}{n} * \sum(x_i - x_m)^2) * (\frac{1}{n} * \sum(y_i - y_m)^2))^{\frac{1}{2}}}$$

Formula 6 Coeficiente de Pearson Fuente (Sampieri, Collado, & Lucio, 2010)

Donde:
n = número de observaciones.
X_i = valor de la variable x por ítem.
X_m = media de la variable X.
Y_i = valor de la variable y por ítem
Y_m = media de la variable y.

Tabla 16 Definición de la formula Coeficiente de Pearson Fuente (Sampieri, Collado, & Lucio, 2010)

Como lo describimos en el plan de análisis para la presente investigación, solo será necesario la magnitud del valor del coeficiente de Pearson y no su signo, para la interpretación de la magnitud del coeficiente de Pearson según (Sampieri, Collado, & Lucio, 2010) se realizará con los siguientes criterios:

Coeficiente de Pearson "r"	Interpretación
r = –1.00	Correlación negativa perfecta. ("A mayor X, menor Y", de manera proporcional. Es decir, cada vez que X aumenta una unidad, Y disminuye siempre una cantidad constante.) Esto también se aplica "a menor X, mayor Y".
-1 > r <= –0.90	Correlación negativa muy fuerte.
-0.90 > r <= –0.75	Correlación negativa considerable.
-0.75 > r <= –0.50	Correlación negativa media.
-050 > r <= –0.25	Correlación negativa débil.
-0.25 > r <= –0.10	Correlación negativa muy débil.
-0.10 > r <= -0.01	Correlación negativa muy débil.
r = 0	No existe correlación alguna entre las variables.
0 .01 > r <= 0.10	Correlación positiva muy débil.
0.10 > r <= 0.25	Correlación positiva muy débil.
0.25 > r <= 0.50	Correlación positiva débil
0.50 > r <= 0.75	Correlación positiva media.
0.75 > r <= 0.90	Correlación positiva considerable.
0.90 > r <= 0.99	Correlación positiva muy fuerte.
r = 1	Correlación positiva perfecta. ("A mayor X, mayor Y" o "a menor X, menor Y", de manera proporcional. Cada vez que X aumenta, Y aumenta siempre una cantidad constante.)

Tabla 17 Interpretación Coeficiente de Pearson magnitud Fuente (Sampieri, Collado, & Lucio, 2010)

Para realizar este cálculo se utilizó el programa Microsoft Excel, el cual cuenta con la función PEARSON (), esta función calcula el coeficiente de Pearson bajo la fórmula descrita en el plan de análisis. Del cálculo realizado se obtuvieron los siguientes resultados:

Calculo de Coeficientes de Pearson

X=Metodología de desarrollo	
Y=Tiempo de Desarrollo	
X	Y
1	5
1	5
1	5
1	5
1	5
1	4
1	4
1	3
2	2
1	4
1	4
2	3
2	3
1	4
1	4

X=Metodología de desarrollo	
Y=Costos de Desarrollo	
X	Y
1	4
1	4
1	4
1	4
1	4
1	4
1	4
1	3
2	3
1	4
1	4
2	3
2	3
1	4
1	4

1	4
1	4
1	5
Aplicación de Formula r = PEARSON(X,Y)	
r=	-0.73244841

1	4
1	4
1	4
Aplicación de Formula r = PEARSON(X,Y)	
r=	-0.8366600

Tabla 18 Calculo del Coeficiente de Pearson Fuente (Construcción Propia)

Para la relación de las variables

X = Metodología de Desarrollo

Y = Tiempo de desarrollo

Coeficiente de Pearson "r" fue de 0.73

Podemos concluir bajo los datos resultantes que existe una relación directa entre la variable Metodologías de desarrollo y el Tiempo de desarrollo, ya que para el valor de "r" bajo los criterios descritos por (Sampieri, Collado, & Lucio, 2010), señala que la relación entre la variables estudiadas como una correlación positiva media.

Para la relación de las variables

X = Metodología de Desarrollo

Y = Costos de desarrollo

Coeficiente de Pearson r fue de 0.84

Podemos concluir bajo los datos resultantes que existe una relación directa entre la variable Metodologías de desarrollo y Costos de desarrollo, ya que para el valor de "r" bajo los criterios descritos por (Sampieri, Collado, & Lucio, 2010), señala que la relación entre la variables estudiadas como una correlación positiva considerable.

7.2 Análisis de Datos

Como se describió en el plan de análisis, el proceso del mismo se realizó en diferentes etapas que a continuación se describen:

Tabulación de datos

Los datos se tabularon de acuerdo al plan de análisis estipulado en la sección anterior.

Análisis de los datos por Pregunta

El análisis de datos por pregunta se realizó una vez que la tabulación de los datos finalizo, como se mencionó en el plan de análisis la pregunta 6 y 7 se agruparan por categoría, cuando nos referimos a categorías nos referimos a cada una de las etapas necesarias para el desarrollo de software.

Análisis de Datos por Variables

La tabulación de datos se agrupo de tal manera que se obtuvieron resultados visibles para cada una de las variables.

Presentación de resultados

La presentación de los resultados se realizará de forma ascendente, comenzando por la presentación de análisis por preguntas, siguiendo dentro de esta misma sección con el análisis por categorías y finalizando con el Análisis por Variables.

7.2.1 Análisis por pregunta.

Código CI-1 Pregunta: 3. Tiene Conocimiento sobre las Metodologías de Desarrollo de Software Lean-Agile.

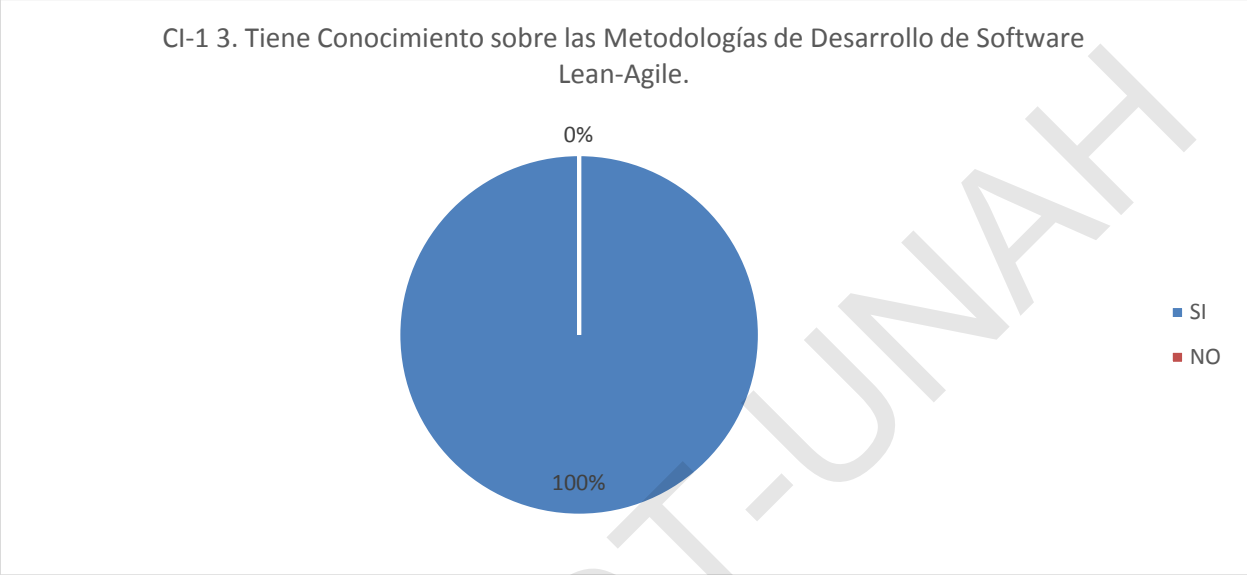


Gráfico 1 Pregunta CI-1, Fuente: Construcción Propia

Debido a que la muestra tomada para la presente investigación, en la cual era requisito previo que los entrevistados conocieran sobre Metodologías de Desarrollo de Software Lean-Agile, los resultados de la Pregunta CI-3 nos muestran en le Gráfico 1 que la regla impuesta a la muestra, se cumplió al 100%, ya que todos los entrevistados tenían conocimientos previos de las Metodologías de Desarrollo de Software Lean-Agile.

Código CI-2 Pregunta: 4. En su empresa se utiliza una Metodología de Desarrollo de Software Lean-Agile

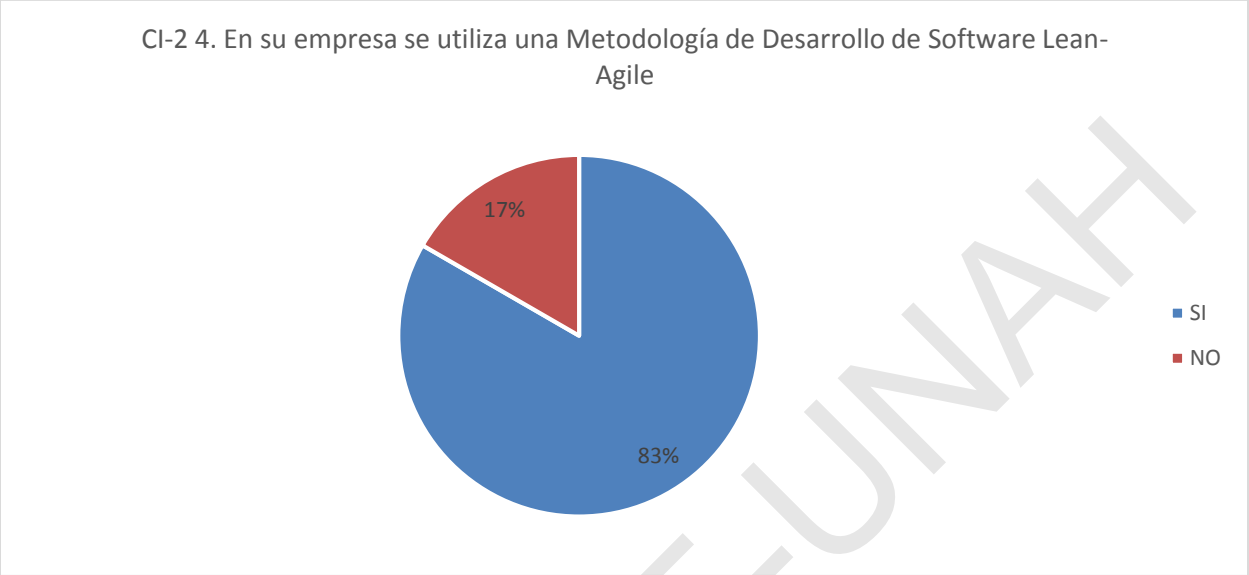


Gráfico 2 Pregunta CI-2, Fuente: Construcción Propia

Los resultados de la pregunta CI-2 nos demuestran (Gráfico 2) que más del 80% de los entrevistados actualmente trabaja bajo un esquema de Metodología Lean- Agile y aunque el porcentaje restante no se encuentre trabajando directamente con la metodología, si relacionamos estos resultados con los resultados de la respuesta CI – 1, podemos concluir con certeza, que los entrevistados tienen conocimientos de la temática y que pueden estimar sobre el uso de la misma.

Código CI-3 Pregunta: 5. Anteriormente a la metodología que se utiliza en su empresa ¿Que metodología de desarrollo de Software se utilizaba?

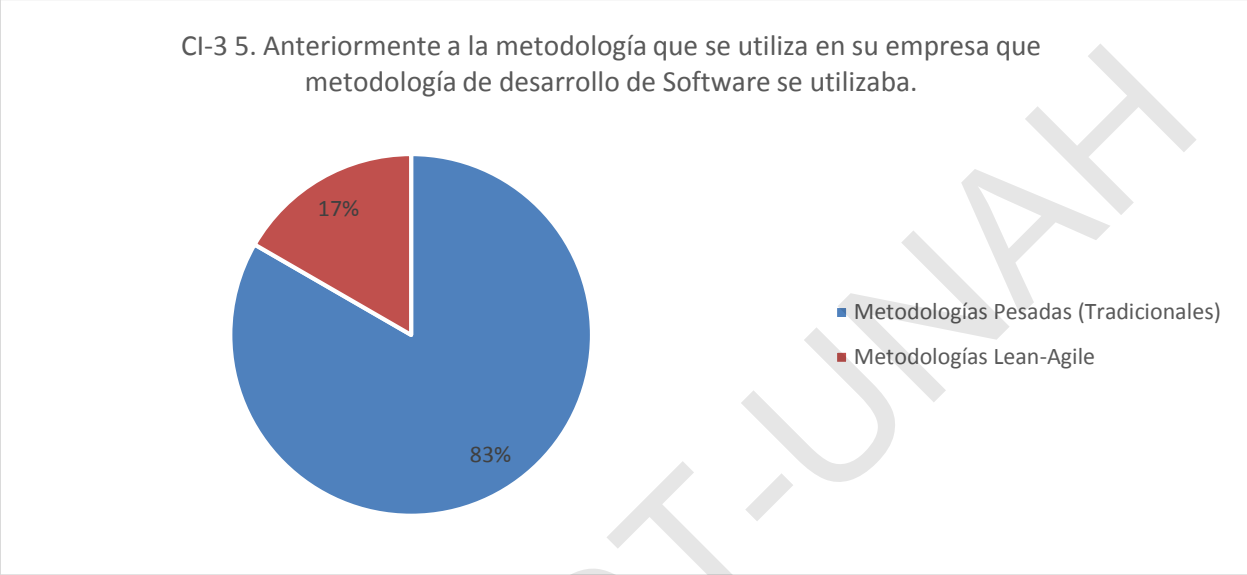


Gráfico 3 Pregunta CI-3, Fuente: Construcción Propia

Según los resultados obtenidos en la aplicación del instrumento la pregunta CI-3, nos muestra en el Gráfico 3 que en la mayoría de las empresas se utilizaba una Metodología Pesada y muy pocas de las mismas utilizaban una Metodología Lean-Agile, al relacionar estos resultados con los resultados de la pregunta CI-2 podemos tener muy claro que las empresas han tenido que venir migrando hacia una Metodología Lean-Agile, y como fuerte fundamento tomamos la pregunta CI-3 la cual muestra claramente que las Metodologías Pesadas vienen siendo sustituidas por las Metodologías Agiles.

Pregunta: 6. A continuación se le presentara un cuadro, en el cual deberá detallar el impacto en el tiempo de duración de cada una de las fases de desarrollo de software utilizando la Metodología Ágil. Se espera que usted realice una comparación en base a su experiencia con las Metodologías Pesadas.

La pregunta: 6 del instrumento de medición, pretende ser una sección en la cual se captura las formas en que la aplicación de una Metodología Lean-Agile influye en el tiempo de desarrollo del software, cada respuesta otorgada por los entrevistados va de acuerdo a su experiencia. Para la captura de las respuestas a esta pregunta, el instrumento se abordó de acuerdo a una planeación genérica del desarrollo de software, el mismo está dividido en categorías. Como para la captura de datos en el instrumento esta pregunta se divide en categorías, para el presente análisis se realizará de la misma forma.

Categoría Requisitos de Usuario

La categoría de requisitos de usuarios según la teoría, es una de las categorías que se ve afectada fuertemente por el uso de la Metodología Lean-Agile, esta metodología hace referencia al fuerte vínculo que debe existir entre el cliente y el equipo de desarrollo, así como también la participación total en las iteraciones de cada proyecto.

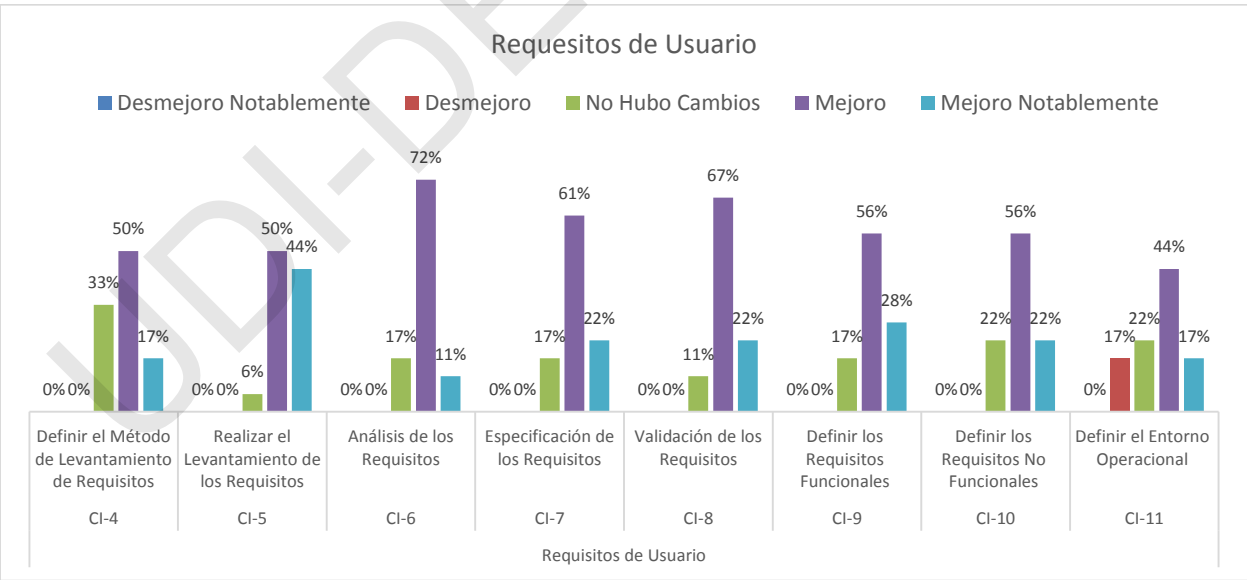


Gráfico 4 Pregunta 6 Cat. Requisitos de Usuario por Ítem, Fuente: Construcción Propia

Como podemos ver en la gráfica anterior (Gráfico 4) la categoría Requisitos de Usuario está conformada por diferentes ítems, de los cuales podemos apreciar que en cada uno a excepción del ítem CI-11 todos los demás ítems recibieron más del 50% como Mejora y todos los ítems recibieron más del 10% de Mejoras Notables, es importante hacer notar que en esta categoría solo el ítem CI-11 recibió una calificación de desmejora la cual es de un 17%, el 83% restante está definido con un 22% de No hubo cambios un 44% de Mejora y un 17% de Mejora notable.

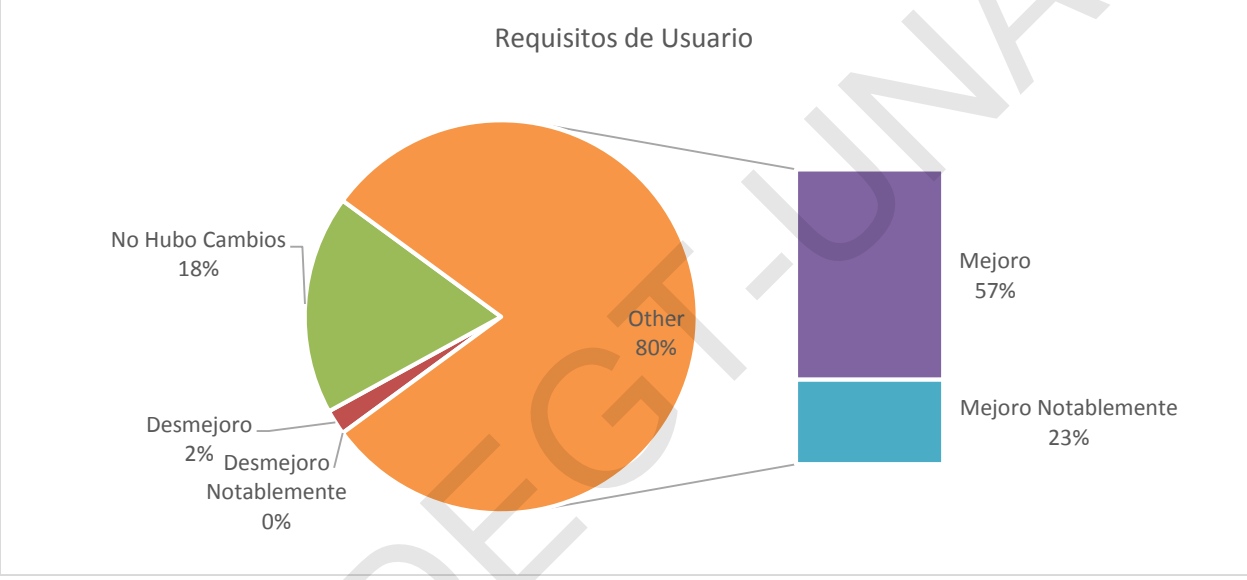


Gráfico 5 Pregunta 6 Cat. Requisitos de Usuario, Fuente: Construcción Propia

Para concluir en esta categoría se realizó un promedio de los resultados obtenidos por ítem con lo cual se realizó el Gráfico 6 y con esta información sustentamos que la categoría Requisitos de Usuario se ve afectada positivamente o representa una mejora en un 80% y en un 18% no presenta cambios. El 2% restante representa una Desmejora la cual se torna casi inexistente si lo comparamos con el porcentaje de mejora, con lo cual la aplicación de las Metodologías Lean-Agile producen una Mejora o Mejora notable en un proyecto de desarrollo de software.

Categoría Requisitos de Hardware

La categoría de requisitos de Hardware es una de las categorías de las cuales la teoría no especifica mucho, pero al tratarse de requisitos y trabajo cerca del cliente, al aplicarse Metodologías que tienen como principio fundamental el cliente y la eliminación de desperdicios, deberían poder apreciarse resultados favorables.

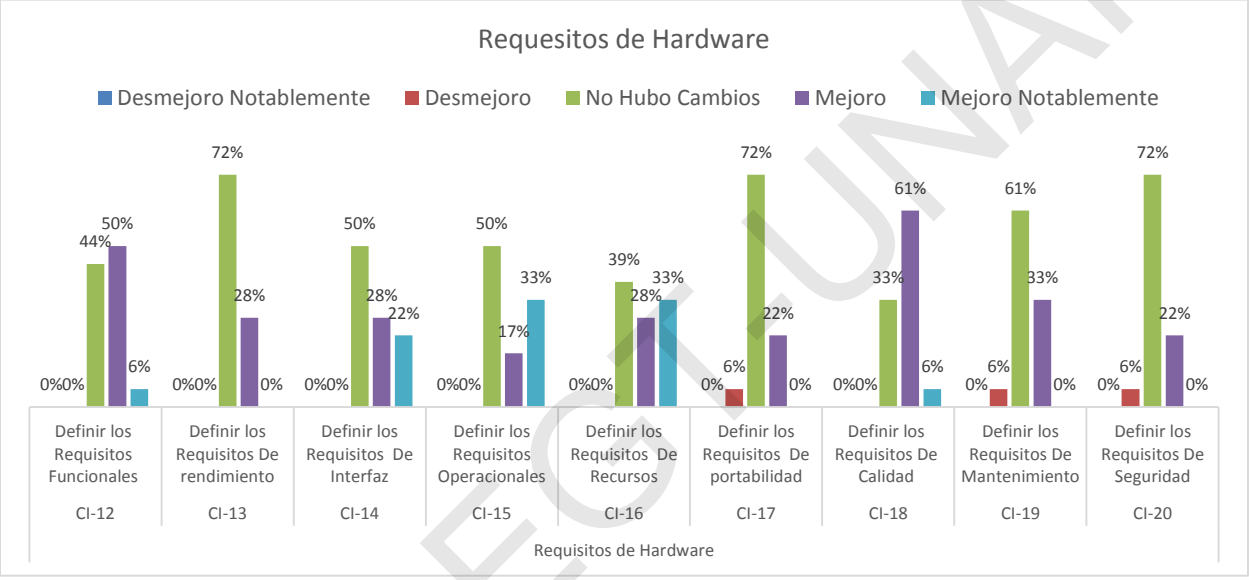


Gráfico 6 Pregunta 6 Cat. Requisitos de Hardware por Ítem, Fuente: Construcción Propia

Esta categoría está conformada por diferentes ítems todos relacionados con la definición de requisitos de Hardware, estos ítems como se puede apreciar en la Gráfica 6 unánimemente no presento Desmejoras Notables, de los mismos los únicos ítems que representaron una desmejora del 6% fueron los ítems CI-17, CI-19, CI-20. En el detalle de los resultados por ítems podemos apreciar que en esta categoría los resultados dominantes son No hubo Cambios, este está representado en cada ítem con más del 50% a excepción del CI-12 el cual tiene una representación del 44%, las mejoras se pueden apreciar con mayor énfasis en los ítems CI-12, CI-18 en las cuales las mejoras son 50% y 61% respectivamente. Aunque la categoría, al parecer no representa mejoras; tampoco representa Desmejoras.

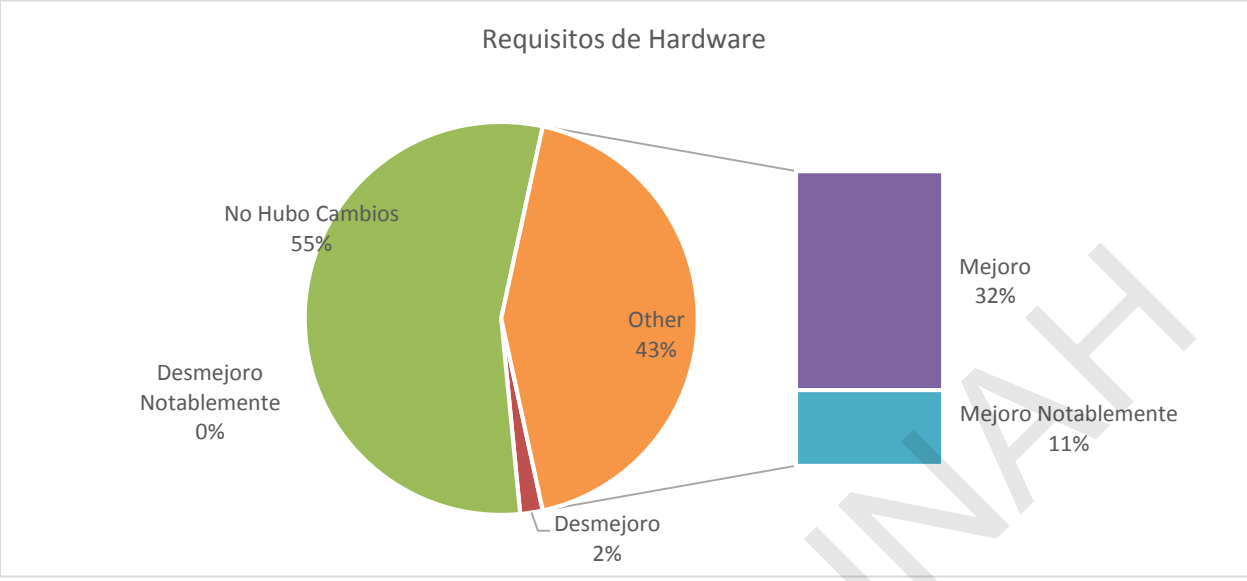


Gráfico 7 Pregunta 6 Cat. Requisitos de Hardware, Fuente: Construcción Propia

Como podemos apreciar en el Grafico 7 el cual representa los resultados totales de la categoría Requisitos de Hardware, el índice de Desmejora es apenas de un 2% es mayor el porcentaje de No Hubo Cambios como se pudo apreciar en el estudio por ítems, en el Grafico 7 No Hubo Cambios representa el 55%, el 43% restante es de Mejoras y Mejoras Notables con un 32% y 11% para cada repuesta.

En el estudio de la presente categoría podemos decir que un 2% de desmejora no es un valor representativo para afirmar que la aplicación de las Metodologías Lean-Agile representa un retraso en el tiempo de ejecución de estas actividades. Podemos afirmar que la aplicación de estas metodologías representara un mantenimiento en los tiempos de ejecución de estas actividades con una favorable posibilidad de mejora.

Categoría Arquitectura

En esta categoría hacemos referencia a las dos categorías anteriores, esta referencia se basa en que los requisitos de usuario y los requisitos de hardware son básicamente lineamientos puros con el cliente. En esta categoría no se trabaja directamente, ya que son representaciones y definiciones técnicas las que se realizan, por lo cual el correcto levantamiento de requisitos afectara directamente la arquitectura del software.

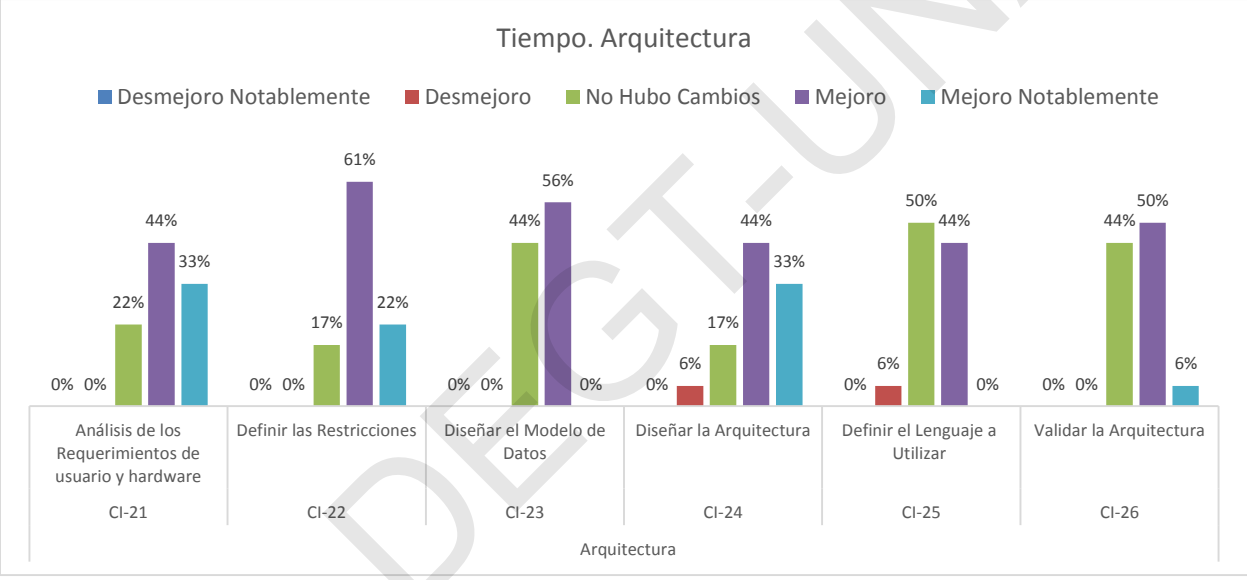


Gráfico 8 Pregunta 6 Cat. Arquitectura por Ítem, Fuente: Construcción Propia

Como se mencionaba anteriormente en esta categoría se realiza una referencia a las categorías anteriores iniciando con el ítem CI-21 el cual realiza un estudio de las categorías anteriores para tener un punto de partida hacia las demás actividades. Como podemos apreciar en la gráfica por ítems (Grafico 8) la Mejora es el resultado predominante con la excepción del ítem CI-25, aunque en este ítem el resultado de Mejora no es el dominante está muy cerca de No hubo cambios el cual es el de mayor representación. En las gráficas por ítem (Grafico 8) es importante notar que solo 2 ítems de los 6 que conforman la categoría tienen un resultado de Desmejora y aunque existe

representación de este resultado el mismo 44% en el ítem CI-25 y para el ítem CI-24 la Mejora total podría decirse que es de un 77% con un 33% de Mejora Notable.

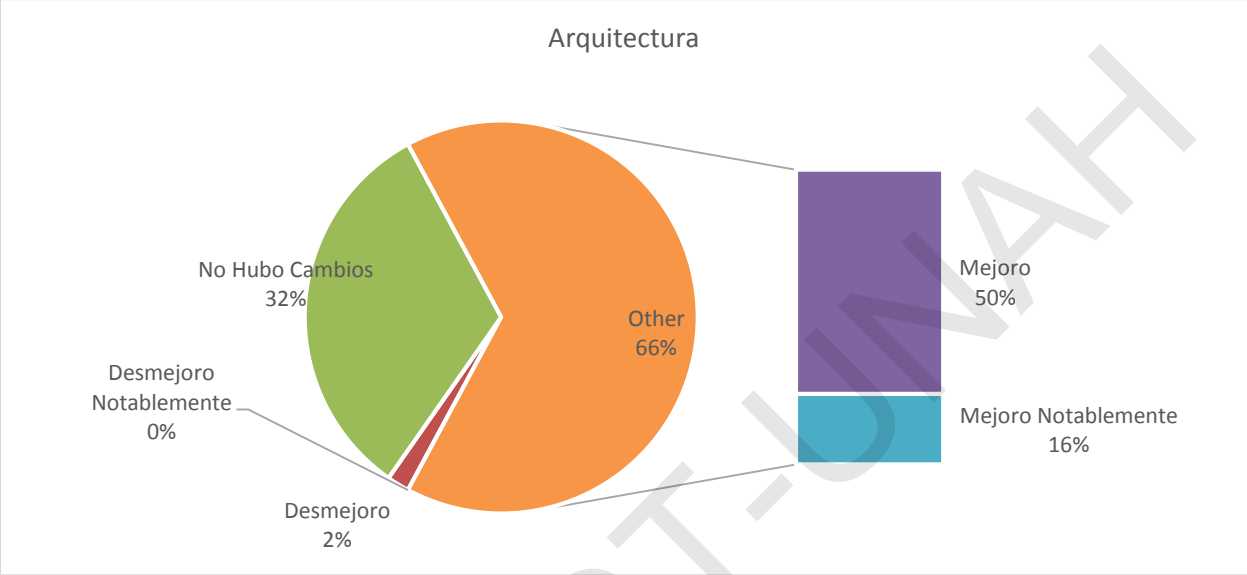


Gráfico 9 Pregunta 6 Cat. Arquitectura, Fuente: Construcción Propia

La arquitectura de software en el desarrollo de un proyecto es un pilar fundamental, como cada uno de sus categorías, pero al estar fuertemente vinculado a las necesidades descritas por los clientes, en las cuales las Metodologías Lean-Agile basan sus principios de trabajo, esperaríamos resultados favorables, bajo esta teoría y fundamentados en los resultados como en el Grafico 9 el cual representa el total de resultados para esta categoría, podemos concluir que la aplicación de estas Metodologías en las actividades de arquitectura de software, son afectadas de manera positiva y representa una mejora en el tiempo de ejecución de esta categoría, la representación de mejora de un 66% compuesto por un 50% de Mejora un 16% de Mejora Notable más el 32% de no Hubo Cambios nos brinda la base sustentable necesaria para realizar tal afirmación, ya que la representación de desmejora es solo de un 2% el cual no es representativo ante el 98% restante.

Categoría Codificación

La codificación de un programa es una tarea totalmente técnica, por lo cual los clientes se mantiene aislados de estas actividades, aun así la teoría recabada sobre las Metodologías de Desarrollo Lean-Agile afirman que la codificación mediante el uso de la eliminación de desperdicio se torna eficiente, con lo cual se realizan mejoras de tiempos.

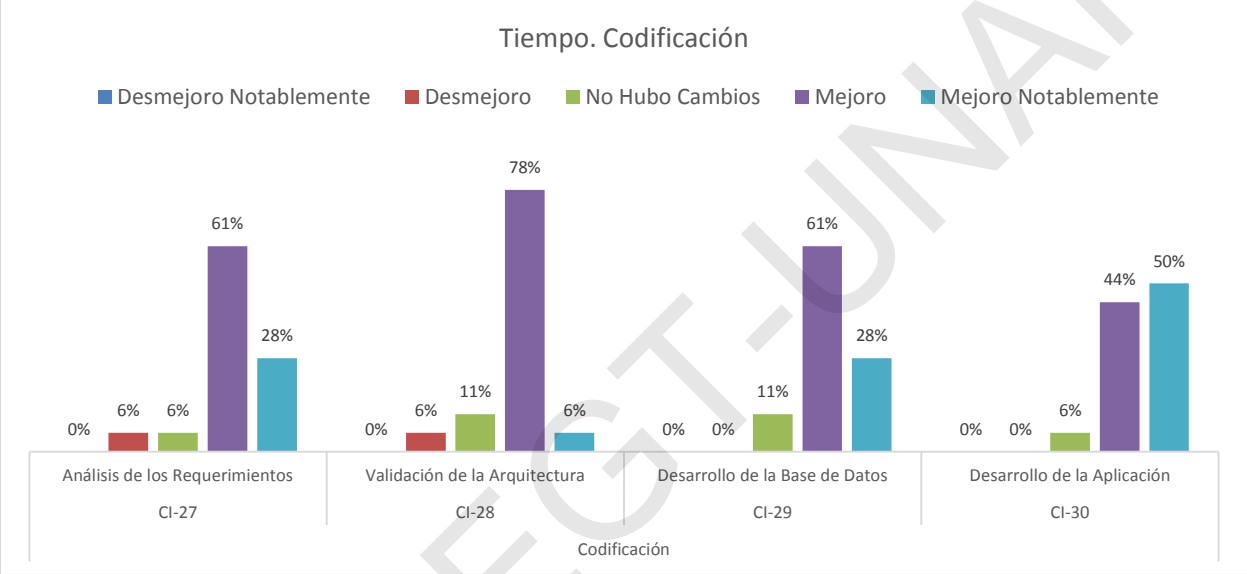


Gráfico 10 Pregunta 6 Cat. Codificación por Ítem, Fuente: Construcción Propia

Los ítems pertenecientes a esta categoría solo son 4, de los cuales podemos apreciar en la Gráfica 10 que en cada uno de ellos los valores dominantes son las mejoras, en los ítems CI-27 y CI-28 se puede ver un valor de 6% para la Desmejora, en cada uno de estos dos ítems la mejora es muy representativa; un 89% para el ítem CI-27 compuesto por un 61% de Mejora y un 28% de Mejora Notable, para el Ítem CI-28 la mejora total es de 84% compuesto por un 78% de mejora y un 6% de Mejora Notable.

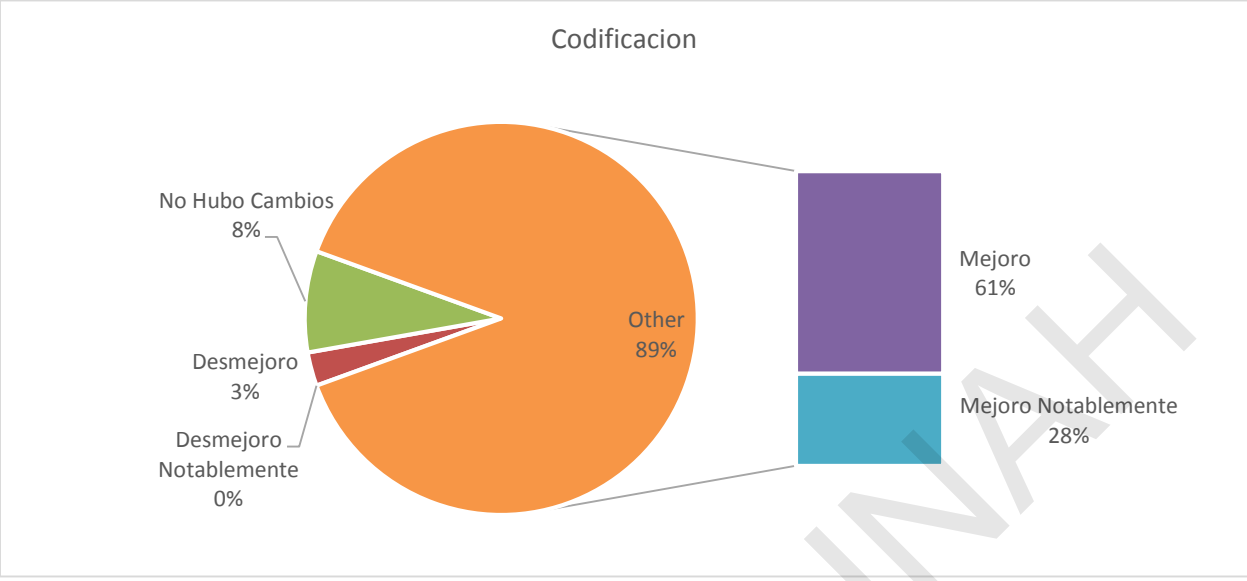


Gráfico 11 Pregunta 6 Cat. Codificación, Fuente: Construcción Propia

En esta categoría la Mejora y Mejora Notable en cada uno de los ítems son las más representativas, esto debido a que la recopilación de la información y la obtención de definiciones en la categorías anteriores se ha realizado muy de la mano del cliente o usuario final, lo cual representa para un desarrollador un entendimiento más claro sobre lo que se requiere como producto final, en la categoría Codificación, el total de sus respuestas está representado en el Gráfico 11, está representación nos detalla que esta categoría presenta una Mejora de 89% compuesta por un 61% de Mejora y un 28% de Mejora Notable el 12% restante está conformado por un 8% de No Hubo Cambios y un 3% de Desmejora. Por lo cual podemos afirmar que en esta categoría la Aplicación de las Metodologías Lean-Agile generan una mejora en los tiempos de ejecución de estas actividades.

Categoría Pruebas

La categoría de pruebas es una de las categorías que se realiza en conjunto, tanto el usuario final, como los programadores y equipo técnico, en esta categoría se espera que la Metodologías Lean-Agile aporten cambios en el tiempo; ya que las iteraciones en su aplicación mejoran la compresión de funcionalidad y de entendimiento del software.

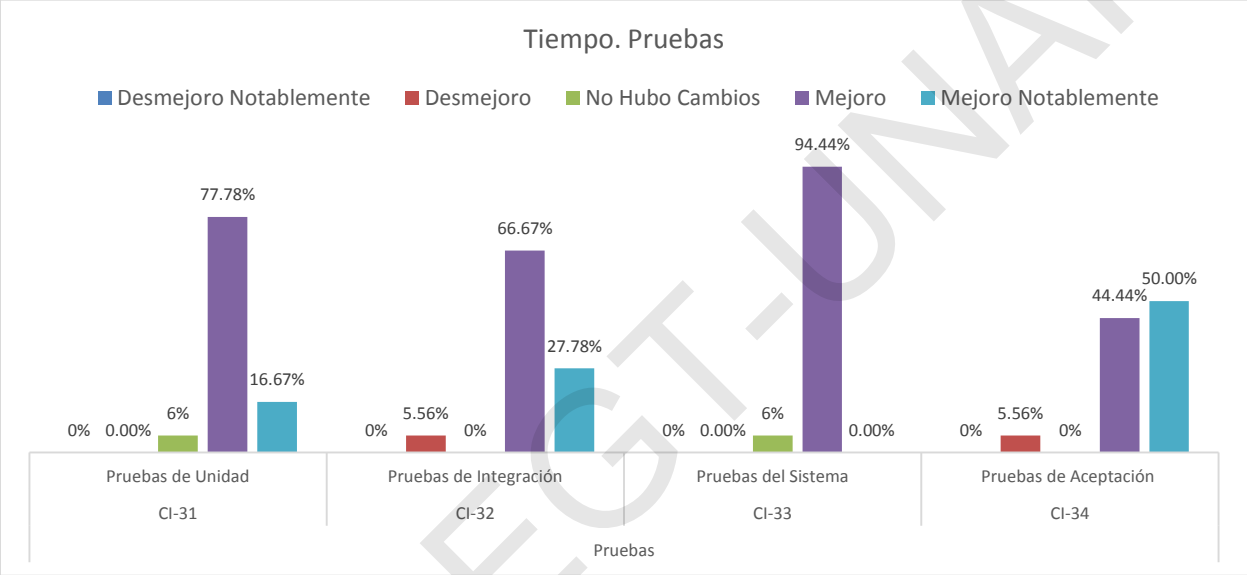


Gráfico 12 Pregunta 6 Cat. Pruebas por Ítem, Fuente: Construcción Propia

En esta categoría, al igual que en la categoría anterior las mejoras son las predominantes como se puede apreciar en el Grafico 12, en los ítems CI-32 y CI-34 presentan una desmejora del 5.56% un valor que no es representativo o influyente, ya que el porcentaje de mejora es de un 94.44% con una apreciación de mejora de casi un 100%.

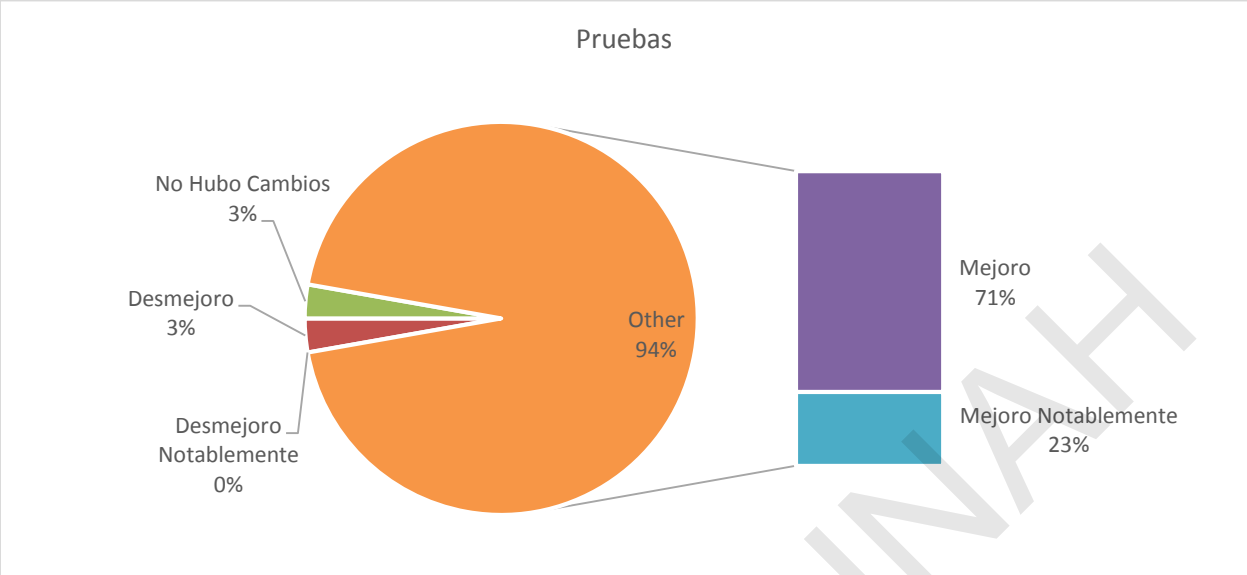


Gráfico 13 Pregunta 6 Cat. Pruebas, Fuente: Construcción Propia

La categoría de pruebas, es una de las categorías que representó un mayor incremento en las mejoras, como se puede apreciar en el Grafico 13, el cual describe el total de resultados para esta categoría, su representación de mejora figura con total de 94% compuesto de un 71% de Mejora y un 23% de Mejora Notable, una representación de No Hubo Cambios de un 3% y una desmejora de un 3%. En esta categoría se ve una gran diferencia de porcentajes por lo cual difiera de las demás categorías, donde los porcentajes de mejora si eran representativos, pero no tan superiores como en este caso, este efecto en la categoría de pruebas basados en la teoría abordada en este documento, se genera por el trabajo de iteración, comunicación constante y cercanía entre el equipo técnico, el cliente y los usuarios finales, con lo cual se logra un producto muy cercano al cumplimiento del 100% de lo que el usuario desea. Con la teoría y los datos como fundamento, podemos concluir que en la categoría de pruebas, la aplicación de la Metodología Lean –Agile generan una mejora en los tiempos de ejecución de estas actividades.

Categoría Entrega

Para finalizar con el plan genérico para desarrollo de software y después de realizadas las pruebas de aceptación por parte del cliente, y estas finalizadas con éxito; es en este punto donde se aborda la categoría de entrega, en esta categoría se abordan las temáticas de cierre del proyecto, instalación y seguimiento del software, así como la recopilación de defectos y corrección de los mismos.

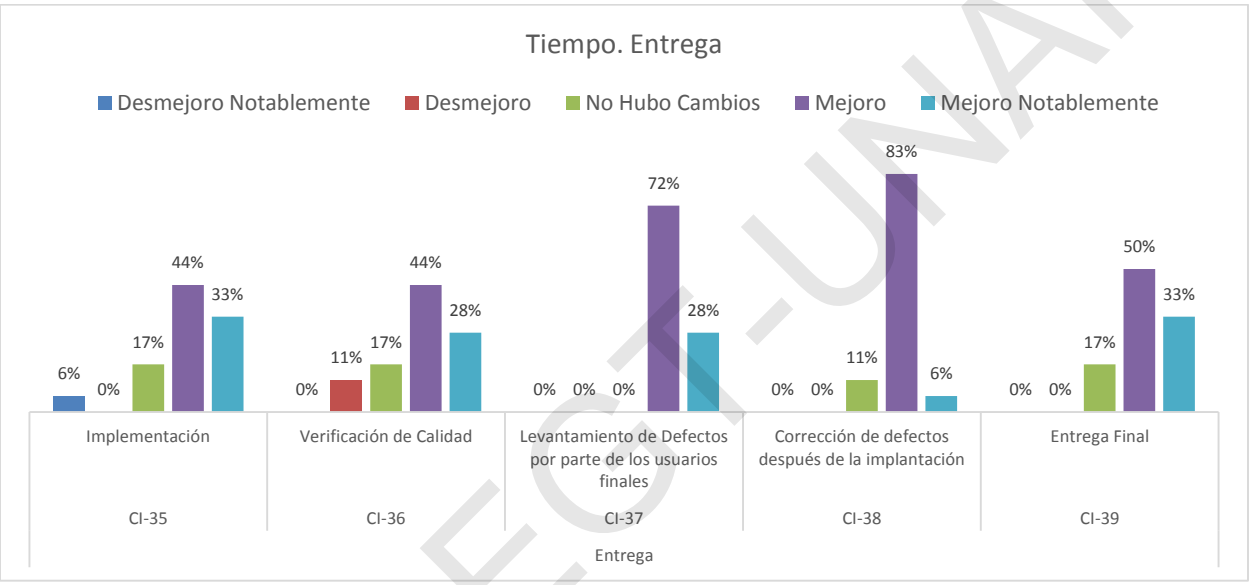


Gráfico 14 Pregunta 6 Cat. Entrega por Ítem, Fuente: Construcción Propia

En esta categoría, como en la categoría anterior, las Mejoras son los valores más representativos, podemos apreciar ese detalle en el Grafico 14, el cual describe que las desmejoras en esta categoría están representadas en el ítem CI-35 con una representación del 6% de Desmejora Notable y en el ítem CI-36 el cual tiene una representación de 11% Desmejora. A diferencia de las anteriores categorías, soló en esta categoría hay una representación de Desmejora Notable; pero su representación es baja en comparación con las Mejoras.

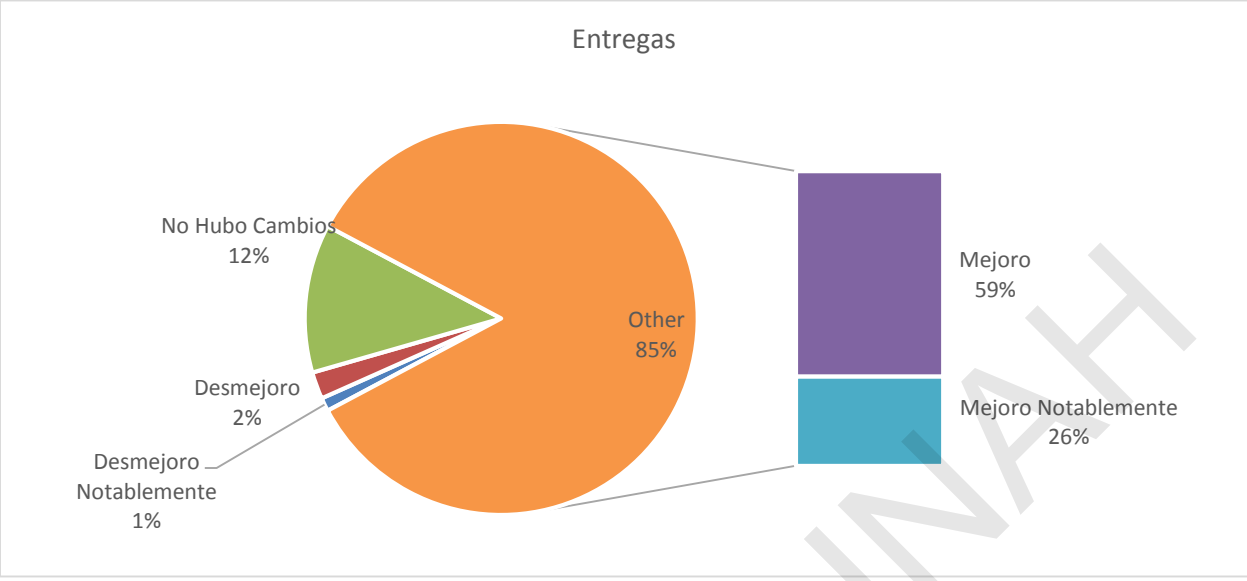


Gráfico 15 Pregunta 6 Cat. Entrega, Fuente: Construcción Propia

Para la representación total de la categoría se realizó el Grafico 15, y como lo vimos anteriormente en el detalle por ítem (Grafico 14) las Mejoras tiene una mayor representación, siendo estas de un 85% compuesto por un 59% de Mejora y un 26% de Mejora Notable, así mismo consta de un 12% de No Hubo cambios y un 3% de Desmejoras las cuales están compuestas por un 2% de Desmejora y un 1% de Desmejora notable. En conclusión podemos afirmar que aplicando las Metodologías Lean-Agile se obtienen mejoras en los tiempos de entrega de un proyecto de software.

Pregunta: 7. A continuación se le presentara un cuadro, en el cual deberá detallar el impacto de los costos incurridos en cada una de las fases de desarrollo de software utilizando la Metodología Ágil. Se espera que tome como referencia la metodología que su usaba anteriormente en su empresa.

La pregunta: 7 del instrumento de medición, es una emulación de la sección anterior, con la única diferencia que esta pregunta se realiza con un enfoque en los costos incurridos en el desarrollo de software y no en el tiempo, vale hacer hincapié que cada respuesta otorgada por los entrevistados va de acuerdo a su experiencia.

Categoría Requisitos de Usuario

Como lo mencionamos en la pregunta anterior, la sección de Requisitos de Usuario es una de las secciones que se apoya más mediante la Metodología Lean-Agile, debido a la cercanía del trabajo del equipo técnico y de los clientes.

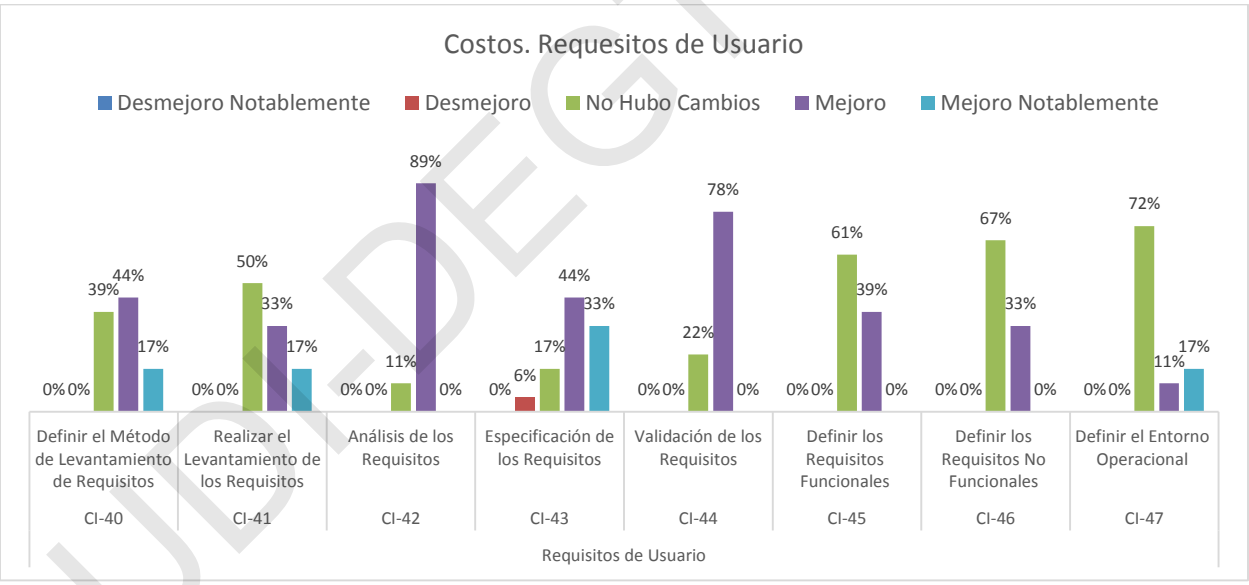


Gráfico 16 Pregunta 7 Cat. Requisitos de Usuario por Ítem, Fuente: Construcción Propia

Referente a esta categoría, podemos apreciar en la gráfica por ítems (Grafico 16) que es una categoría que sí tiene mejora; por no en un grado de valoración superior a las demás respuestas, aun así el único ítem que representa una valoración para Desmejora es el Ítem CI-42, en la mayoría

de los ítems según los resultados obtenidos podemos afirmar que en esta categoría los costos se mantiene y en los ítem CI-42 y CI-44 tienen una mejora alta.

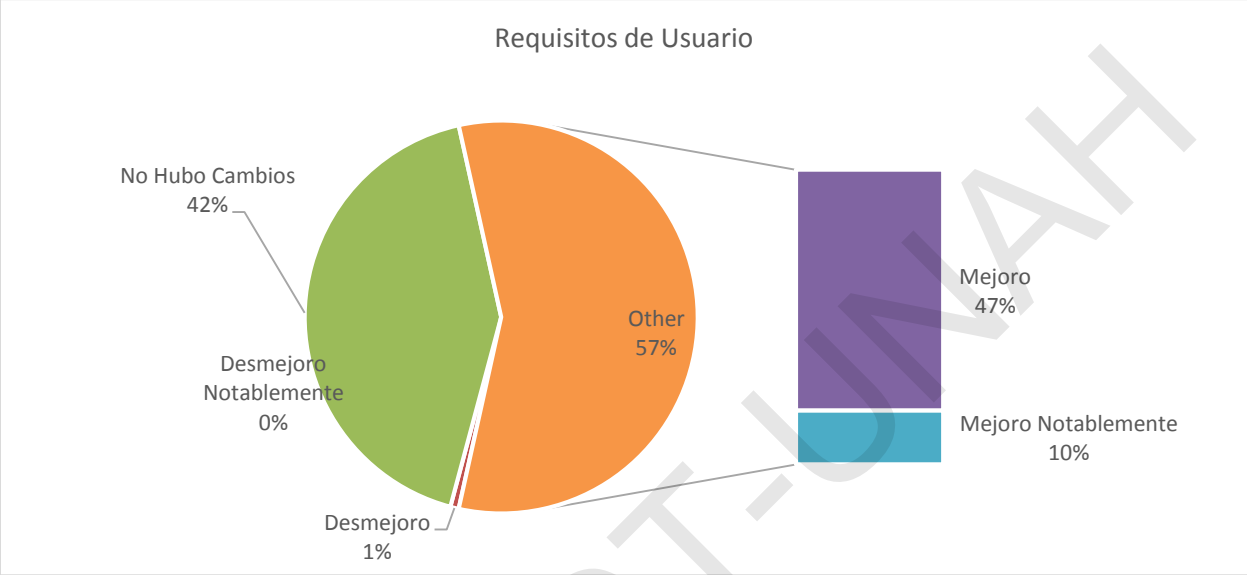


Gráfico 17 Pregunta 7 Cat. Requisitos de Usuario, Fuente: Construcción Propia

Al analizar los resultados totales de esta categoría representados en el Grafico 17, podemos apreciar que la Mejora con una representación de 47% es superior a los demás resultados conformados por un 42% de No Hubo Cambios y un insignificante 1% de Desmejora, en esta categoría podemos afirmar que la aplicación de la Metodologías Lean – Agile representa un aporte positivo, con lo cual se mejoran los costos en los que se incurre en el desarrollo de software.

Categoría Requisitos de Hardware

En la pregunta 6 se mencionaba que teóricamente las Metodologías Lean- Agile no muestran un gran aporte en cuanto a los Requisitos de Hardware, pero tampoco define esta categoría como un atraso o fuente de desmejora.

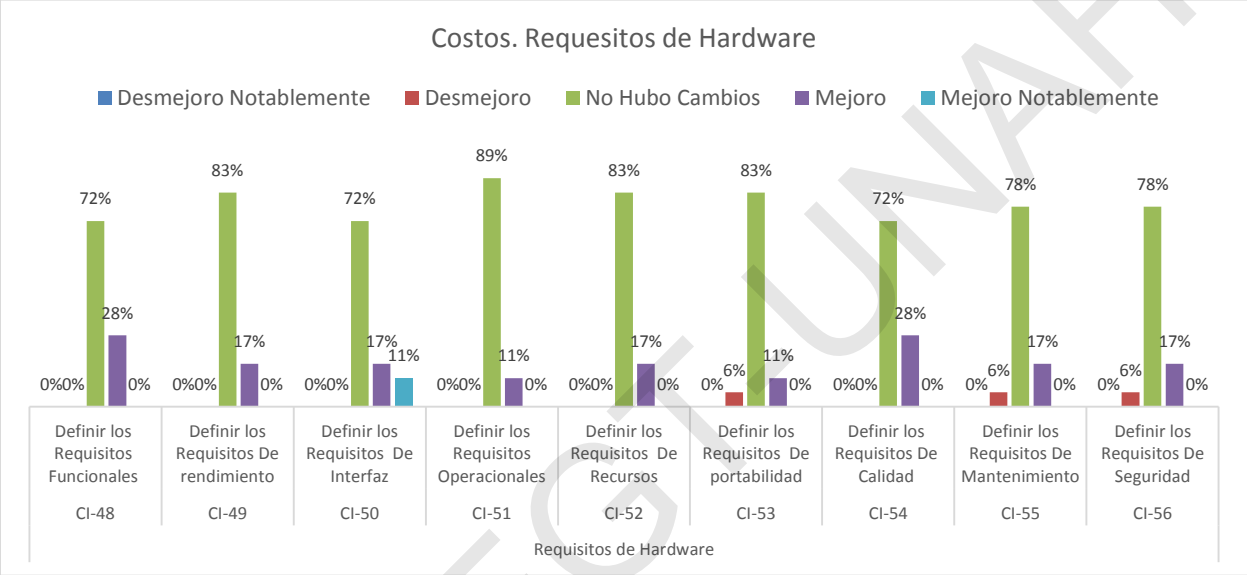


Gráfico 18 Pregunta 7 Cat. Requisitos de Hardware por Ítem, Fuente: Construcción Propia

En esta categoría a diferencia de la categoría anterior, podemos apreciar en la gráfica por ítems (Grafico 18) que la repuesta No Hubo Cambios es la más valorada, mostrando más de un 70% en cada uno de los ítems, la mejora es mucho menor, pero superior a un 10% en cada uno de los ítems y superior a la representación de Desmejora.

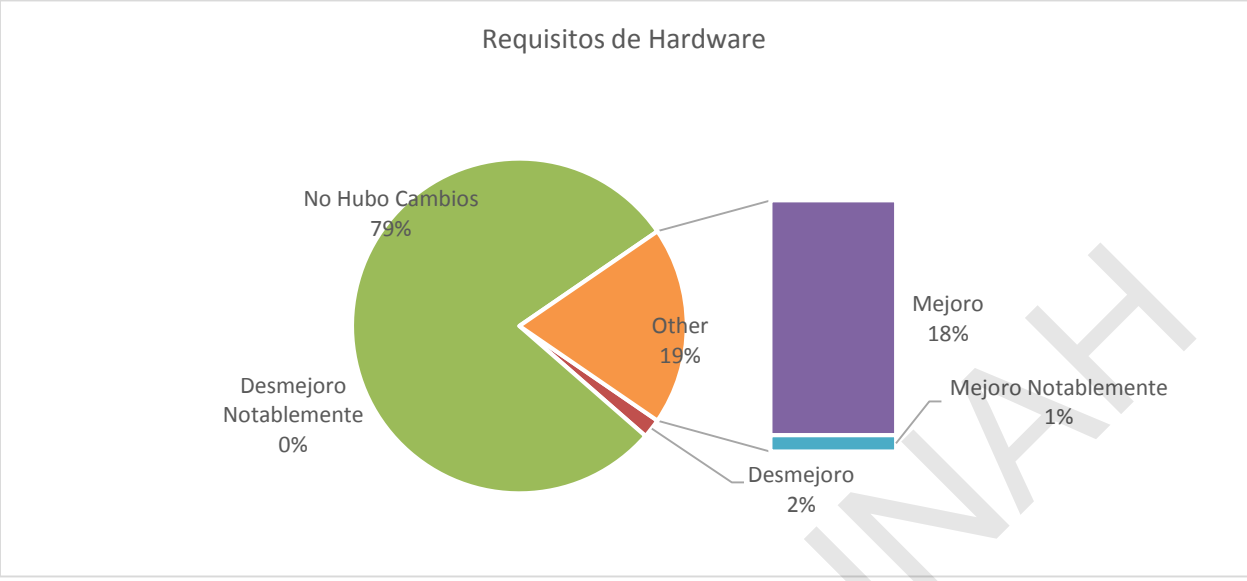


Gráfico 19 Pregunta 7 Cat. Requisitos de Hardware, Fuente: Construcción Propia

En el grafico total de la categoría Requisitos de Hardware representado por el Grafico 19, podemos apreciar lo que la gráfica por ítems (Grafico 18) nos describe, la respuesta No Hubo Cambios es la dominante en esta categoría con una representación de 79% el 21% restante está conformado por un 19% de Mejora y un 2% de Desmejora, en esta categoría está muy claro que los entrevistados basados en su experiencia, no percibieron mejoras sustanciales por lo cual podemos concluir que la aplicación de las Metodologías Lean – Agile no representa una Mejoras considerable en cuanto a costos, pero tampoco representa una Desmejora, en síntesis su aplicación no produce cambios sustanciales.

Categoría Arquitectura

La Categoría es una de las categorías técnicas del desarrollo de software, en el cual se pueden aplicar diferentes principios de los mencionados, tanto en la Filosofía Lean como en el manifiesto Agile.

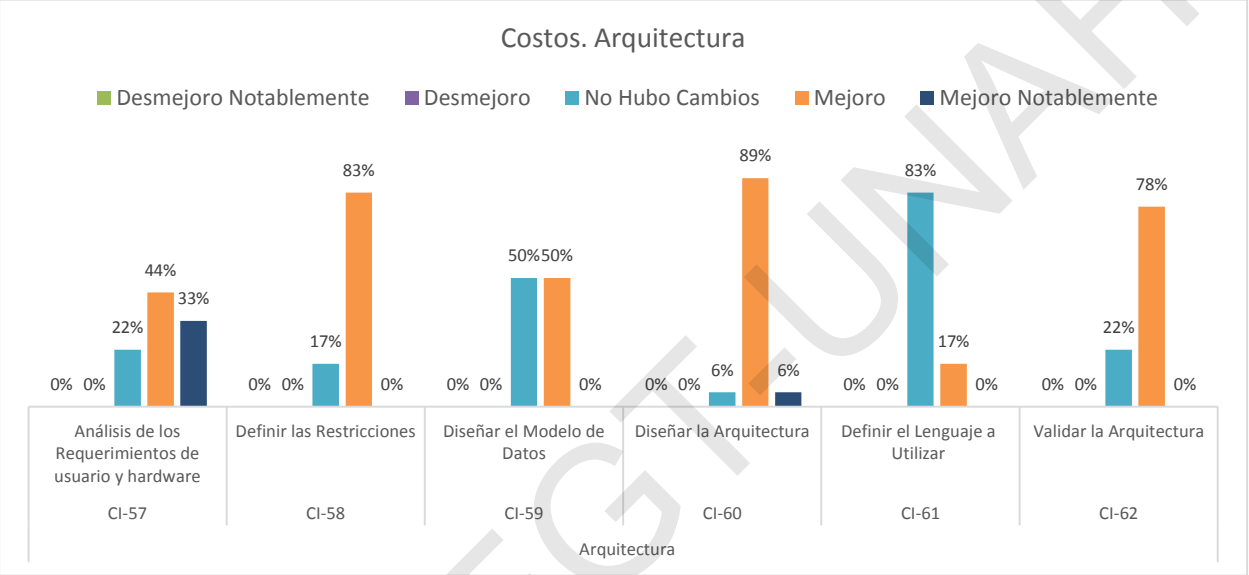


Gráfico 20 Pregunta 7 Cat. Arquitectura por Ítem, Fuente: Construcción Propia

En la gráfica por ítems de la presente categoría (Grafico 20) como podemos apreciar, la respuesta Mejoró es la más valorada, a diferencia de la categoría anterior en la cual la respuesta No Hubo Cambios fue la más valorada. En la gráfica se nos describe que en su totalidad ningún ítem muestra una desmejora, sí hay presencia de la respuesta No Hubo Cambios, especialmente en el ítem CI-61, si nos percatamos; el ítem hace referencia al lenguaje a utilizar, lo cual es una parte importante del desarrollo de software y realmente que no represente una desmejora, es fundamental, en relación a la Mejora podemos ver sus mayores valoraciones en los ítem CI-58, CI-60 y CI-62 los cuales son pilares fundamentales de esta categoría ya que en cada uno de estos ítems se definen restricciones y la arquitectura como tal.

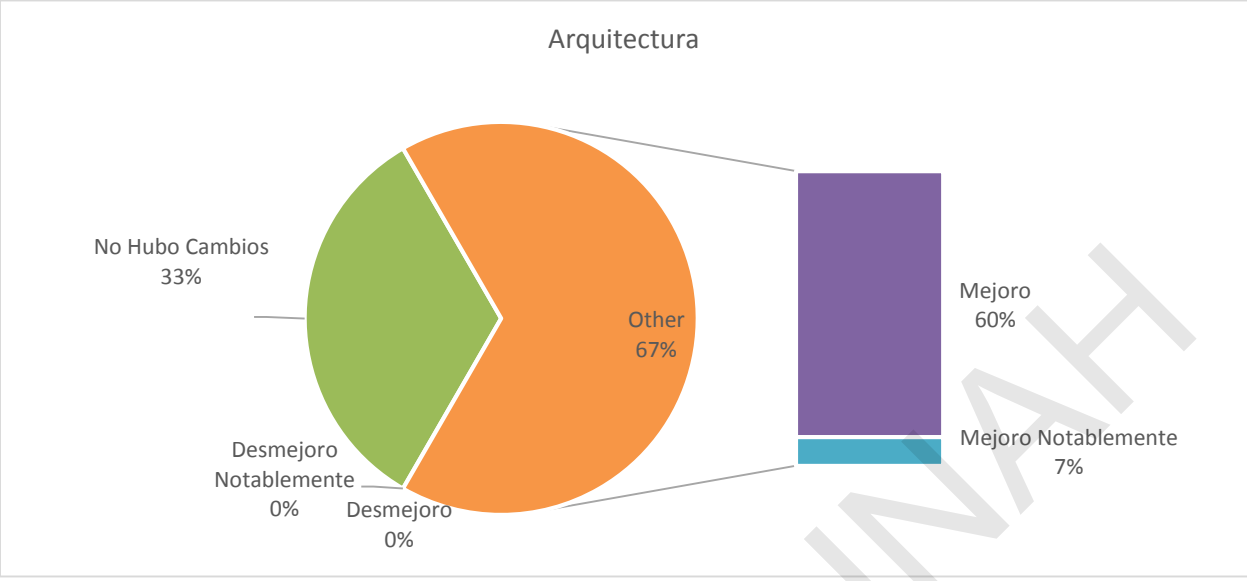


Gráfico 21 Pregunta 7 Cat. Arquitectura, Fuente: Construcción Propia

En el grafico total de la categoría representado por el Grafico 21 podemos apreciar que la respuesta de Mejora tiene un 67% del total de respuestas, conformado por un 60% de Mejoro y un 7% de Mejora Notable, el 33% restante es en su totalidad de No Hubo Cambios. Es importante hacer notar que no existe desmejora de ningún tipo en esta categoría, por cual podemos concluir que en esta categoría al aplicar la Metodología Lean – Agile los costos no desmejoraran, los mismos mejoraran o se mantendrán en el peor de los casos.

Categoría Codificación

La codificación en los proyectos de desarrollo de software, es el núcleo de toda la operación, donde se puede estancar un proyecto por el no entendimiento de los requerimientos, o donde se puede avanzar rápidamente por la comprensión de los mismos. Las Metodologías Lean-Agile se fundamentan en el trato con el cliente, la eliminación de desperdicio o acciones que no generen valor al producto final, así como la simplicidad.

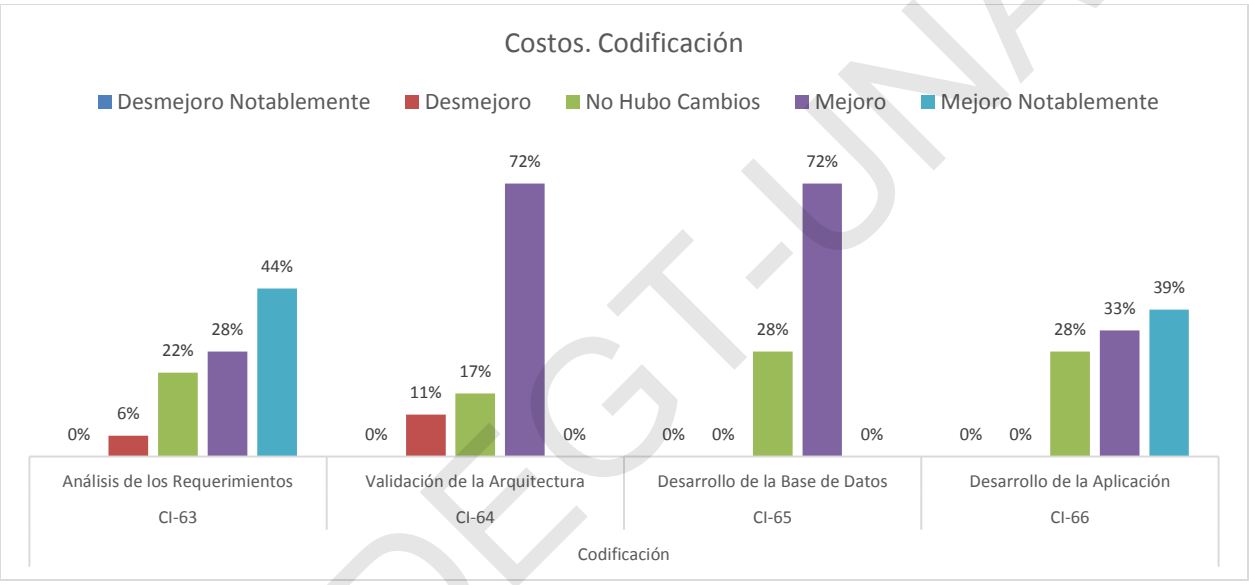


Gráfico 22 Pregunta 7 Cat. Codificación por Ítem, Fuente: Construcción Propia

En esta categoría los ítems que lo componen tiene respuestas variadas como podemos apreciar en el Grafico 22, en los ítems CI-63 y CI-64 se tiene una representación de desmejora y vale recalcar que estos ítem son, según sus características, ítems de entendimiento y verificación de las categorías anteriores, en los ítems restantes no se presenta una desmejora; por el contrario se puede apreciar una mejora considerable y un sostenimiento de costos aceptable.

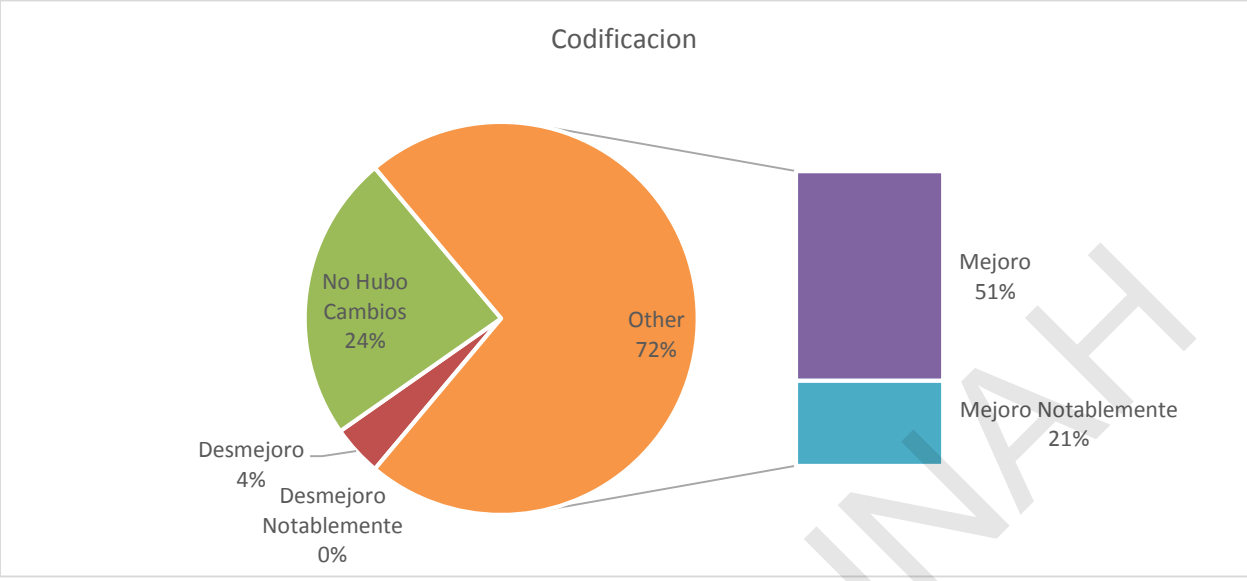


Gráfico 23 Pregunta 6 Cat. Codificación, Fuente: Construcción Propia

En el grafico total de la categoría de codificación representado por el Grafico 23, se puede aprecia que a pesar de que en cada uno de los ítems no hubo, como en las categorías anteriores, una respuesta dominante. Podemos apreciar que el 72% de representación es de Mejora compuesta por un 51% de Mejora y un 21% de Mejora Notable, el restante porcentaje de 28% está conformado por un 24% de No Hubo Cambios y un 4% de Desmejora, es notable que la aplicación de las Metodologías Lean – Agile en la Codificación es favorable, cabe destacar que los datos obtenidos son sustento de la teoría que describe estas metodologías.

Categoría Pruebas

En la categoría de pruebas se abordan tanto pruebas de unidad, de funcionalidad y de aceptación. Según la teoría, al utilizar las Metodologías Lean-Agile se presentan diferentes entregas que son revisadas en cada una de las iteración, en las cuales se debe obligar la participación, tanto del equipo técnico, como del cliente y los usuarios finales, con esta premisa se brinda un soporte muy fuerte para que las tareas de pruebas se puedan realizar de una manera sencilla y de validez, por la participación de todo el equipo implicado en el desarrollo de software.

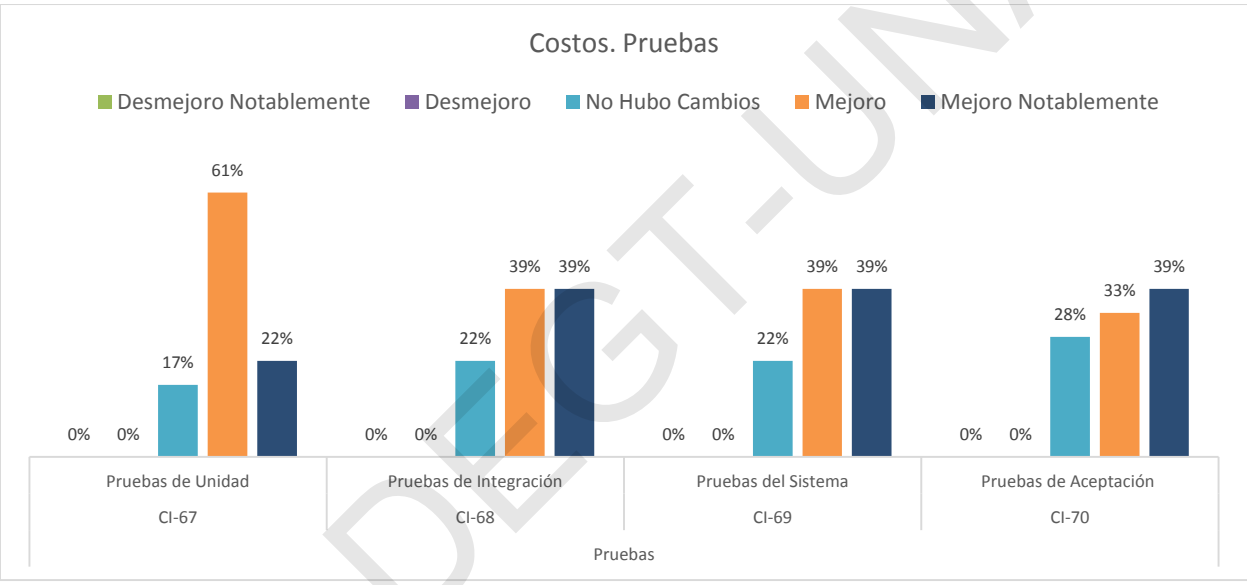


Gráfico 24 Pregunta 7 Cat. Pruebas por Ítem, Fuente: Construcción Propia

En la gráfica por ítems de la presente categoría (Grafico 24) podemos apreciar que ninguno posee una representación de desmejora, en su mayoría los ítems tienen más de un 60% de representación de mejora total, conformado por un porcentaje de Mejora Notable considerablemente alta.

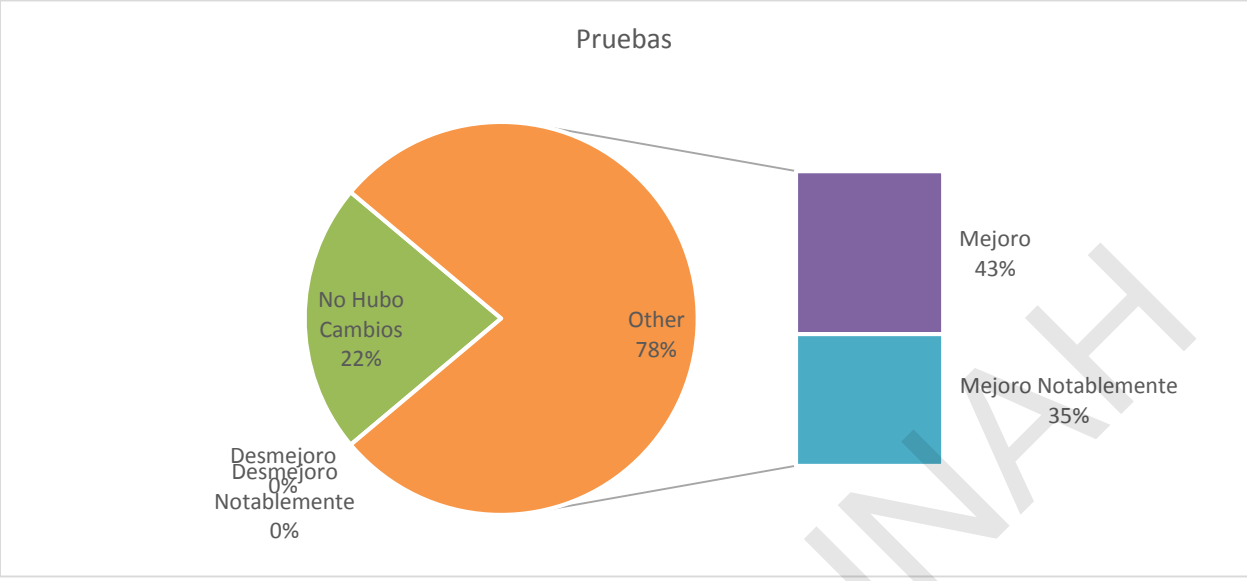


Gráfico 25 Pregunta 7 Cat. Pruebas, Fuente: Construcción Propia

Esta categoría como podemos apreciar en el grafico total representado por el Grafico 25 y en el grafico por ítems (Grafico 24) se carece de una desmejora, cabe recalcar que según la teoría, las pruebas se realizan de manera periódica, tomando como ciclo la iteración entre todo el equipo implicado en el desarrollo de software, la mejora de la categoría total es de un 78% compuesto por un 43% de Mejoro y un 35% de Mejora Notable, el 22% restante está compuesto en su totalidad por la respuesta No Hubo Cambios, con los descrito anteriormente podemos afirmar que la aplicación de las Metodologías Lean-Agile generan un efecto positivo en los costos, mostrando en base a porcentajes Mejoras sustanciales, y de la misma manera dando constancia que su aplicación no producirá ninguna desmejora.

Categoría Entrega

La entrega de un software es un paso crucial en el desarrollo de software, ya que es donde el equipo técnico se desliga del mismo y pasa solo, a formar para de un soporte que se espera sea eventual. En esta categoría los clientes y usuarios finales deben ser capaces de empoderarse del software final y lograr utilizarlo en base a los objetivos para el cual se realizó.

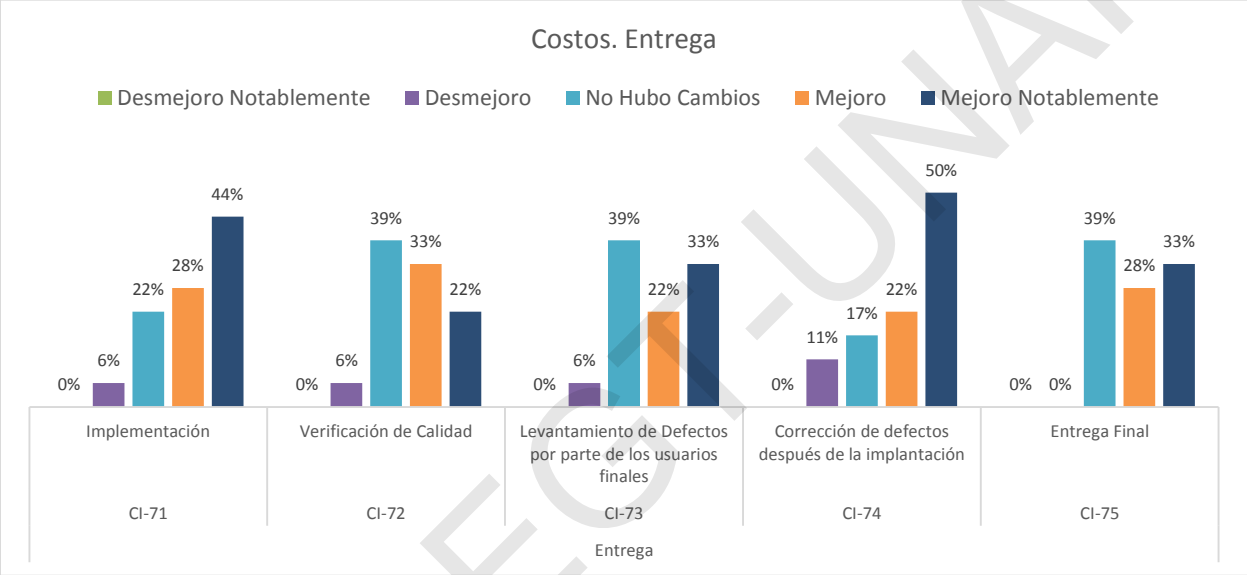


Gráfico 26 Pregunta 7 Cat. Entrega por Ítem, Fuente: Construcción Propia

En la gráfica por ítem de esta categoría (Grafico 26) podemos apreciar que la mejora notable es una de las respuestas mas valoradas, esto a pesar de ser también la única categoría en la que la mayoría de sus ítems representan una desmejora, a excepción del ítem CI-75, es relevante notar que el ítem CI-75 referente a la corrección de errores antes de proceder a la entrega final, tiene una representación del 50% de Mejora Notable; aunque este mismo ítem tiene la representación más alta de desmejora, con un 11% en relación a los demás ítems que componen esta categoría.

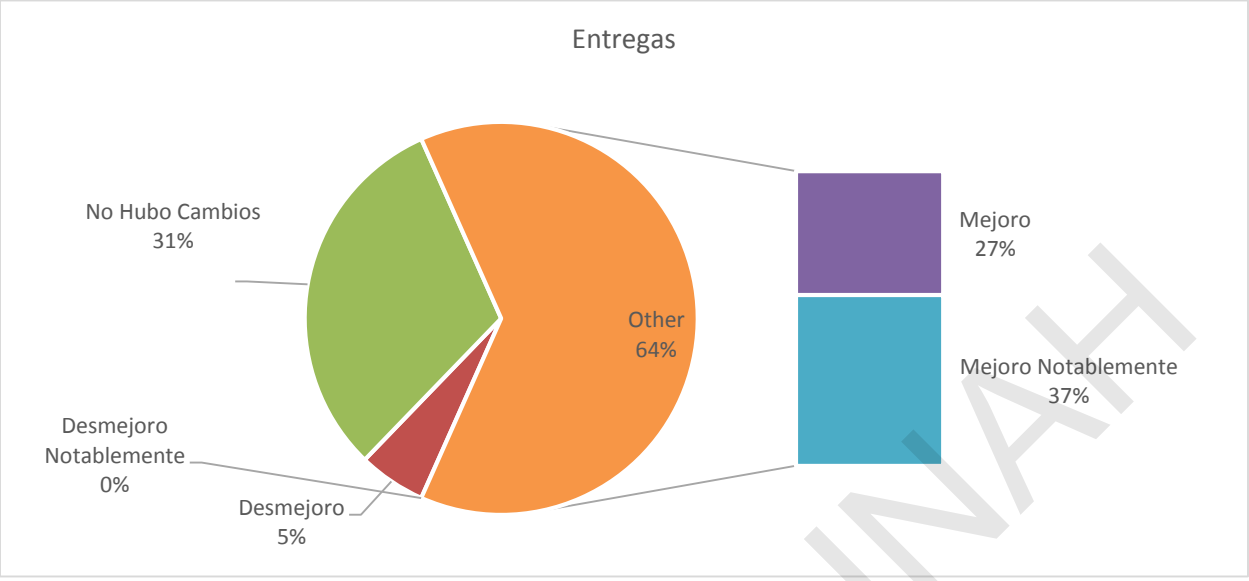


Gráfico 27 Pregunta 7 Cat. Entrega, Fuente: Construcción Propia

En la gráfica total de la categoría representada por el Grafico 27 podemos apreciar que la mejora es de un 64% compuesto por un 27% de Mejora y un 37% de Mejora Notable, es notable que la respuesta con más valoración dentro de esta categoría es la Mejora Notable, en base a lo expuesto en la gráfica anterior (Grafico 27) podemos concluir que el cierre o la entrega de un proyecto de desarrollo de software, cuando aplicamos las Metodologías de desarrollo Lean – Agile recibe un cambio positivo, con el cual se mejoran los costos en los que se incurre.

7.2.2 Análisis por Variables

Metodología de Desarrollo

La metodología es, según la definición de variables, nuestra variable independiente; esta variable será la que impacte en la variable de costos y la variable de tiempo.

La variable metodología está compuesta por tres preguntas, como se describió en la codificación del instrumento y en la creación del mismo, estas preguntas según son código son CI-1, CI-2 y CI-3.

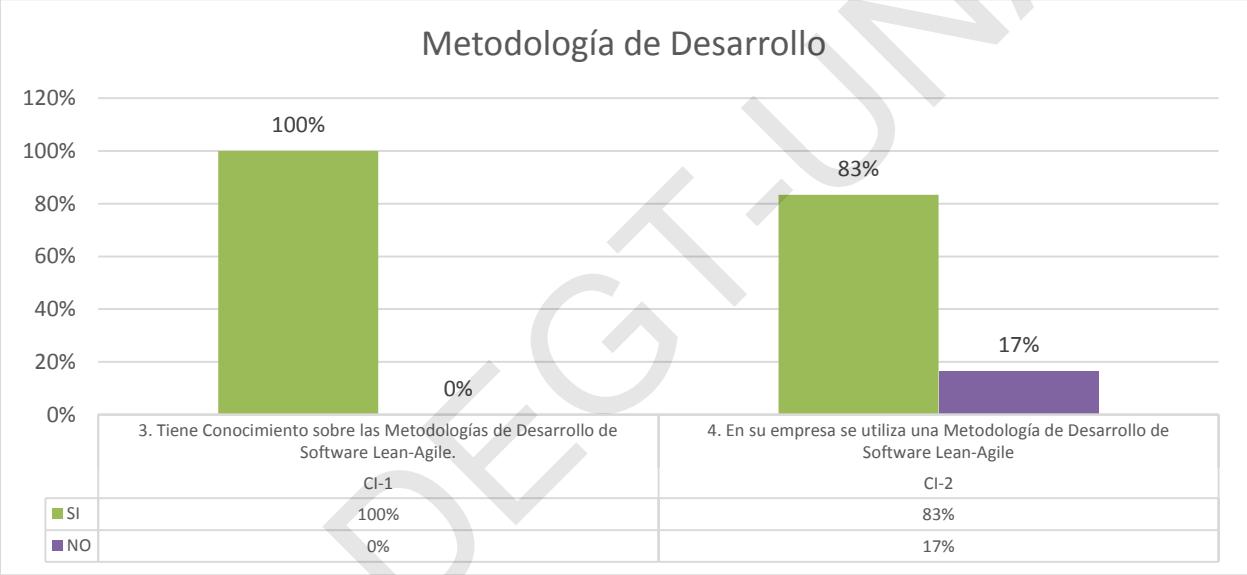


Gráfico 28 Variable Metodología de Desarrollo CI-1, CI-2

En la gráfica anterior (Grafico 28) podemos ver los resultados de las preguntas CI-1 y CI-2, la pregunta CI-1 nos muestra que en efecto todos los entrevistados conocen sobre las Metodologías Lean-Agile, la gráfica CI-2 nos detalla que en su mayoría están utilizando las Metodologías Lean – Agile.

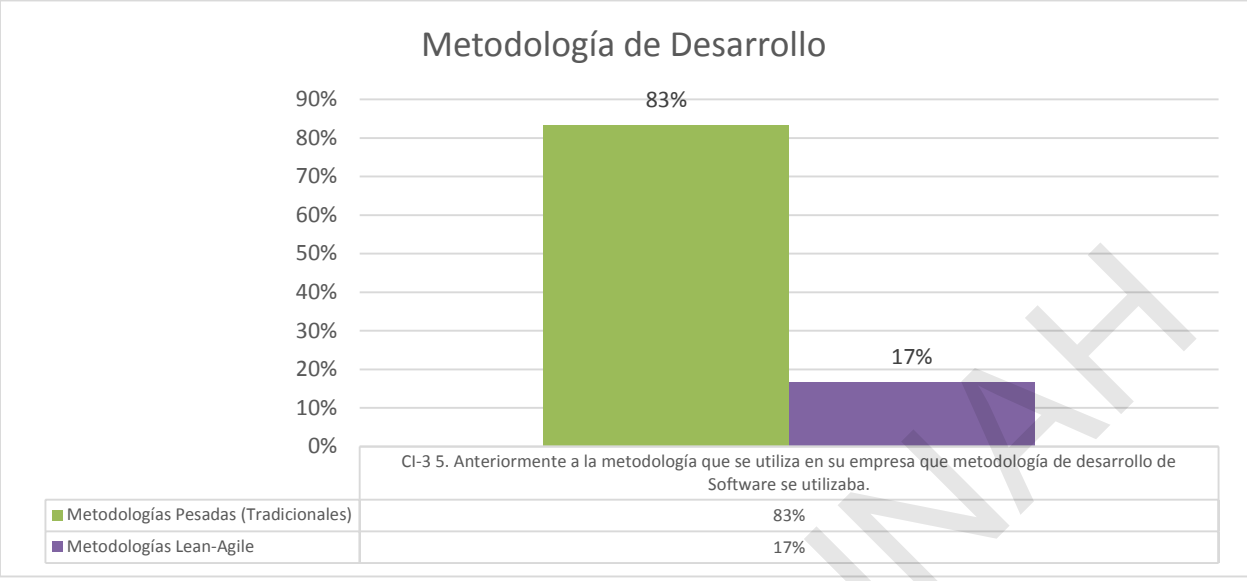


Gráfico 29 Variable Metodología de Desarrollo CI-3

La grafica de CI-3 (Grafico 29) nos detalla que estas empresas en su mayoría migraron de una Metodología Pesada a una Agile.

En forma de consolidación de respuesta de la presente variable, podemos afirmar que todos los entrevistados son conocedores de las Metodologías Lean-Agile, de estos en su mayoría trabajan actualmente con las mismas, lo cual nos brinda el conocimiento para poder abordar las preguntas referentes a las variables de estimación tiempo y costos. Así mismo los entrevistados en su mayoría hacen referencia a que la empresas en las que laboran, migraron de una Metodología Pesada a una Lean - Agile.

Lo que podemos concluir sobre la presente variable basados en los resultados anteriormente descritos, es que los entrevistados son una fuente fidedigna para poder realizar estimaciones sobre la aplicación de las Metodologías Lean – Agile, en los proyectos de desarrollo de software.

Tiempo de Desarrollo

La variable Tiempo de desarrollo es una variable dependiente, su dependencia es hacia la variable Metodología de Desarrollo, las pregunta definida para esta variable es la pregunta 6: la cual explica que lo que se pretende es que el entrevistado realice una comparativa basado en su experiencia del efecto, en cuanto al tiempo se refiere, cuando se aplican las Metodologías Lean – Agile al desarrollo de software.

La pregunta 6 que es la que contempla la presente variable, está compuesta por diferentes categorías y cada categoría por diferentes ítems, todos relacionados con las tareas de un desarrollo de software, o lo que es lo mismo, la planeación de un proyecto de desarrollo de software.

Para el análisis de la variable Tiempo de Desarrollo, se realizó un promedio de los resultados obtenidos de cada ítem perteneciente a la pregunta 6, los ítems según su código son los siguientes:

Código	Código	Código
CI-4	CI-16	CI-28
CI-5	CI-17	CI-29
CI-6	CI-18	CI-30
CI-7	CI-19	CI-31
CI-8	CI-20	CI-32
CI-9	CI-21	CI-33
CI-10	CI-22	CI-34
CI-11	CI-23	CI-35
CI-12	CI-24	CI-36
CI-13	CI-25	CI-37
CI-14	CI-26	CI-38
CI-15	CI-27	CI-39

Tabla 19 Preguntas Variable Tiempo de Desarrollo. Fuente: Construcción Propia

De la agrupación y promedio de los 36 ítems pertenecientes a esta variable, se logró obtener la siguiente gráfica.

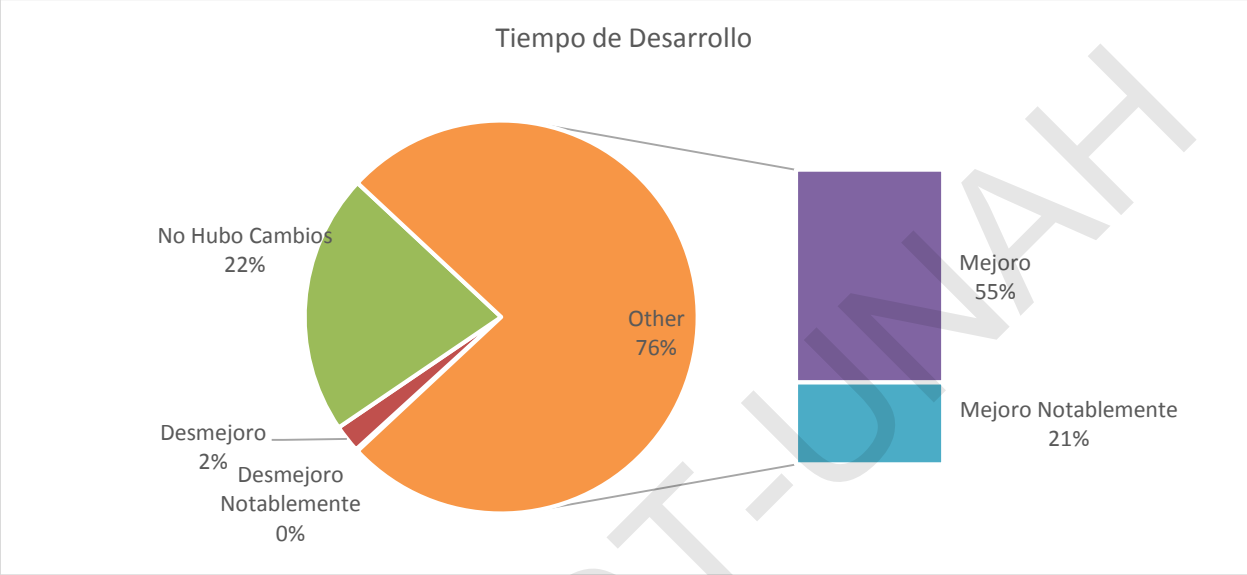


Gráfico 30 Variable Tiempo de Desarrollo. Fuente: Construcción Propia

La grafica 30 es la representación de los datos obtenidos para la variable Tiempo de Desarrollo, esta grafica nos detalla que los resultados que suman el 76% de las respuesta son mejoras, este 76% está compuesto por un 55% de mejora y un 21% de Mejora Notable, el 24 % restante está compuesto por un 22% de No Hubo Cambio y un 2% de Desmejora.

En el análisis por preguntas, se detalló que cada una de las categorías presento Mejoras con una disminución en la categoría de Requisitos de Hardware, en la cual solo se logró un 43% de mejora; pero en las demás categorías las mejoras superaron el 60% de los resultados, teniendo como la categoría que presenta mayores mejoras la Categoría de Pruebas, la cual tuvo un 94%.

La mejoras representadas en la variable Tiempo de Desarrollo, son en gran medida como lo menciona la teoría, la comunicación y trabajo en equipo, esas iteraciones de retroalimentación y control de calidad tienen efectos positivos en el desarrollo de software, ese control de calidad al que se hace mención en la teoría y esas entregas frecuentes de software garantizan el producto final,

así mismo esas entregas obligan a la constante evaluación del software. En los datos que se obtuvieron se observó que la mejora superior se vio en las pruebas, eso no quiere decir que las demás categorías no recibieran mejoras; pero en gran medida el hecho de que la variable Tiempo de Desarrollo represente los datos mostrados en la gráfica, se deben las mejoras realizados en esta categoría.

Podemos afirmar que según la teoría y sustentada por los datos obtenidos, la aplicación de las Metodologías Lean – Agile producen efectos positivos, que generan mejoras en el tiempo de ejecución de un proyecto de software.

Costos de Desarrollo

La variable Costos de Desarrollo es una variable dependiente, su dependencia al igual que la variable de Tiempo de Desarrollo, es hacia la variable Metodología de Desarrollo, esta variable como se detalló en la codificación del documento, está constituida por la pregunta 7, esta pregunta al igual que la pregunta 6 está basada en la actividades o plan para el desarrollo de software, estas actividades están definidas por categoría y cada categoría está conformada por diferentes ítems o actividades, lo que se pretende en esta pregunta es que el entrevistado realice una comparativa basado en la experiencia que posee; cual es el efecto en los costos que incurre en el desarrollo de software cuando se aplica una Metodología de desarrollo Lean – Agile.

Los ítems que conforman esta pregunta según su código son los siguientes:

Código	Código	Código
CI-40	CI-52	CI-64
CI-41	CI-53	CI-65
CI-42	CI-54	CI-66
CI-43	CI-55	CI-67
CI-44	CI-56	CI-68
CI-45	CI-57	CI-69
CI-46	CI-58	CI-70
CI-47	CI-59	CI-71

CI-48	CI-60	CI-72
CI-49	CI-61	CI-73
CI-50	CI-62	CI-74
CI-51	CI-63	CI-75

Tabla 20 Preguntas Variable Costos de Desarrollo. Fuente: Construcción Propia

De la agrupación y promedio de los 36 ítems pertenecientes a esta variable, se obtiene la siguiente gráfica.

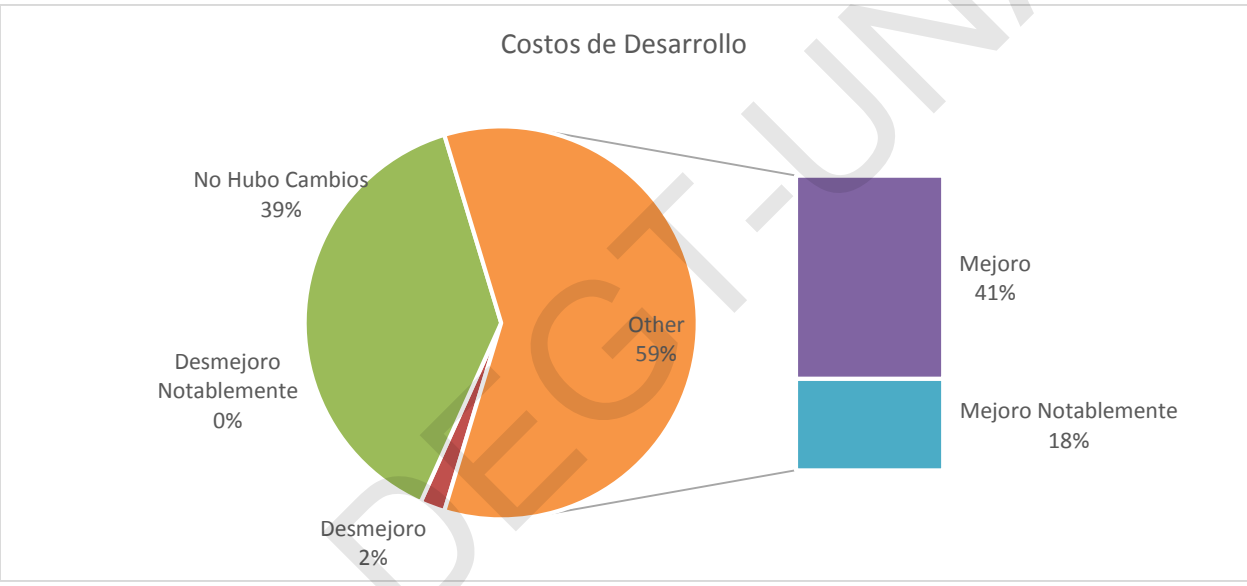


Gráfico 31 Variable Costos de Desarrollo. Fuente: Construcción Propia

Según la gráfica de Costos de Desarrollo representada por la Grafico 31, las mejoras ocupan el 59% del total de resultados compuesto por un 41% de Mejoró y un 18% de Mejora Notable, el 41% restante se compone de un 39% de No Hubo Cambios y un 2% de Desmejoro, como podemos ver, el porcentaje de mayor representación le pertenece a las mejoras, las cuales como en la variable tiempo son las que dominan los resultados.

Según el análisis por preguntas abordado anteriormente en este mismo documento, de las categorías pertenecientes a la pregunta 7 la categoría Pruebas, es la que aporto mayores mejoras con un 78%, como podemos ver coincide con la variable Tiempo de Desarrollo.

En el caso de los costos de desarrollo existe una categoría en la cual la mejora no supero el 20% de los resultados, esta categoría fue Requisitos de Hardware, con un 19% de mejoras, esta categoría, si bien es cierto no fue un aportante notable en las mejoras, tampoco generó desmejoras ya que la mayor representación de resultados los tuvo en No Hubo Cambios con un 79% , lo que quiere decir que aunque no represento mejoras, tampoco genero desmejoras o crecimiento de costos, simplemente se mantuvo como anteriormente se venía manejando.

Retomando lo enmarcado en la teoría perteneciente a las Metodologías Lean – Agile, esta metodología se fundamente en el cliente, el usuario final, la comunicación, las iteraciones y la frecuente presentación de prototipos funcionales de software, esto tiene un efecto en el desarrollo total del software generando cambios agiles y de costos mínimos, así como también calidad en el software. La gran relación que debe existir entre el cliente, los usuarios finales y el equipo técnico genera eficiencia en el proceso.

Con sustento en la teoría y en los resultados obtenidos de la variable costos, podemos afirmar que la utilización de las Metodologías Lean – Agile en un proyecto de software producen mejoras en los costos.

7.3 Análisis de los resultados a la luz de las hipótesis

En la investigación que se desarrolló en este documento se plantearon dos hipótesis, de las cuales concluiremos a continuación:

1. La incidencia de las metodologías Lean – Agile en el desarrollo de software produce mejoras en el tiempo de desarrollo.

En la presente hipótesis se planteaba la relación de las variables metodología y tiempo de desarrollo; por lo cual haremos referencia al análisis de la variable tiempo de desarrollo, misma que nos describe cual es el efecto que la incidencia de la metodología de desarrollo lean- agile

generan en un proyecto de software. En el análisis de la variable se establece que al aplicar dichas metodologías los encuestados percibieron mejoras de un 76% en los tiempo de desarrollo, este 76% está dividido en un 55% de Mejora y un 22% de Mejora notable, el restante 24% se encuentra dividido en un 22% de No hubo cambios y un 2% de Desmejora, por lo cual damos como aceptada la hipótesis mencionada.

2. La incidencia de las metodologías Lean – Agile en el desarrollo de software produce mejoras en el costos de desarrollo.

En la presente hipótesis al igual que la hipótesis anterior, lo que se busca realizar es una relación entre variables, en este caso de las variables metodología de desarrollo y costos de desarrollo, por lo cual haremos referencia al análisis de la variable costos de desarrollo; análisis que se encuentra en la sección de análisis por variables de este mismo documento, este análisis nos describe cual es el efecto que la incidencia de la metodología de desarrollo lean- agile generan en un proyecto de software. En el análisis de la variable se establece que al aplicar dichas metodologías los encuestados percibieron mejoras de un 59% en los costos de desarrollo, este 59% está dividido en un 41% de Mejora y un 18% de Mejora notable, el restante 41% se encuentra dividido en un 39% de No hubo cambios y un 2% de Desmejora en los costos incurridos en el desarrollo software, por lo cual damos como aceptada la hipótesis mencionada.

CONCLUSIONES

En relación al objetivo general de la presente investigación, el cual está establecido de la siguiente forma:

- Evaluar el efecto en los costos y tiempos de entrega, que la incidencia de las Metodologías Lean-Agile generan en un proyecto de desarrollo de software.

Del cual se obtiene los dos objetivos específicos que a continuación se mencionan:

- Evaluar el efecto que la incidencia de las Metodología Lean - Agile generan en los costos de desarrollo de software.

- Evaluar el efecto que la incidencia de las Metodología Lean - Agile generan en los tiempos de entrega de desarrollo de software.

De los cuales concluimos con base en los resultados de la presente investigación, mismos que se utilizan bajo la labor de recurso sustentador para poder afirmar que se logró cumplir con los objetivos específicos, ya que la evaluación de los resultados en su análisis por variables tiempo y costos, nos aporta las herramientas necesaria para evaluar el efecto que la incidencia de las Metodologías Lean-Agile generan en el desarrollo de software.

De la misma forma concluimos que el objetivo general se cumplió en su totalidad, esto debido a que con los resultados obtenidos se logró evaluar la incidencia de las Metodologías Lean-Agile en el desarrollo de software, enfocados en el efecto generado en los tiempos de entrega y costos.

El desarrollo de la presente investigación, es el fundamento para contestar las preguntas planteadas y que se describen a continuación:

- ¿Cuál es el efecto que la incidencia de las Metodología Lean - Agile generan en los costos de desarrollo de software?

Podemos afirmar con base en el análisis de la variable Costos, que con la incidencia de las Metodología Lean – Agile en el desarrollo de software se logran reducir costos, tomando en consideración que la aplicación de la metodología debe ser en todo el transcurso del proyecto.

- ¿Cuál es el efecto que la incidencia de las Metodología Lean - Agile generan en los tiempos de entrega de desarrollo de software?

Podemos afirmar con base en el análisis de la variable Tiempo, que con la incidencia de las Metodología Lean – Agile en el desarrollo de software se logran reducir los tiempos de entrega, tomando en consideración que la aplicación de la metodología debe ser en todo el transcurso del proyecto.

La presente investigación se desarrolló con el enfoque a dar una solución a la problemática abordada en la sección "El problema de investigación" mismo que finaliza con la siguiente pregunta:

¿Cómo hacer frente a un proyecto de desarrollo de software que requiere tiempos cortos de entrega sin elevar los costos?

La pregunta surge de las necesidades actuales, donde los requerimientos de software se ven inundados de dinamismo en cuanto a cambios se refiere, esto obliga a que el manejo de ese requerimiento o proyecto de software entregue en cortos tiempos resultados que otorguen valor al clientes, de la misma forma se espera que esta entrega en tiempos cortos no incurra en un incremento de costos, la investigación que se abordó describe las experiencias de profesionales del área con un perfil de trabajo activo en el momento de su entrevista, y basados en esas experiencias se puede concluir a manera de respuesta para la problemática abordada, que la utilización de metodologías Lean-Agile en el desarrollo de software, aparte de brindar flexibilidad y apego a las necesidades del cliente, nos brinda tiempos cortos de entrega y rebaja en los costos que se incurren.

La presente investigación se desarrolló con limitaciones de tiempo para su elaboración, esta limitante marco el enfoque con el que se abordó el instrumento de medición, ya que en el mismo se consideró las mediciones de metodologías Pesadas o Lean – Agile; no se consideraron híbridos de las mismas o de otro tipo, o aplicaciones parciales de las metodologías antes mencionadas, por lo cual no se formularon hipótesis capaces de profundizar si la aplicación de la metodología en una etapa x de desarrollo, podría mejorar o desmejorar los tiempos y costos, de la misma forma no se pudo llegar a medir la variable calidad, la cual es de vital importancia en el software pero que puede ser objeto de estudio en futuras investigaciones.

En futuras investigaciones, a parte de ingresar la variable calidad, seria de vital importancia lograr formular una metodología basada en la bondades de las metodología pesadas y las metodología Lean – Agile, ya que como se mencionó anteriormente no se cuenta con una medición de algún tipo de hibrido, donde un proyecto maneja varias metodologías, algo que podría llevarnos a una nueva tendencia de metodologías de desarrollo.

Bibliografía

BCIE, B. C. (2011). *Ficha Estadistica de Honduras* .

Bell, S. (2006). *LEAN ENTERPRISE SYSTEMS Using IT for Continuous Improvement.* United States of America.

Calderón, A. R., & Rebaza, V. J. (2007). *Metodologías Ágiles.* Perú: Universidad Nacional de Trujillo.

Chavez, S., Martín, A., Rodríguez, N. R., Murazzo, M. A., & Valenzuela, A. (2012). Metodología AGIL para el desarrollo SaaS. *XIV Workshop de Investigadores en Ciencias de la Computación* (págs. 577-581). Durango: FCEQyN - UNAM.

Figueroa, R. G., Solís, C. J., & Armando A., C. (2007). *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.

Garzas, J. (2013). *Cómo sobrevivir A la planificación de un proyecto ágil.* Mostoles.

Garzas, J. (2015). *Agilidad y Lean. Gestionando los proyectos y negocios del s. XXI (3ª. edición).*

Gonzalez, P. R. (2008). *Estudio de la Aplicacion de Metodologias Agiles para la evolucion de productos de software.* Madrid.

IEEE. (1998). *STD-830-1998 ESPECIFICACIONES DE LOS REQUISITOS DEL SOFTWARE.*

INTECO, L. N. (2009). *INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA* . Madrid: Instituto Nacional de Tecnologias de la Comunicacion.

IS-UNAH, U. N. (05 de 06 de 2015). *Plan de Estudio.* Obtenido de Carrera de Ingeniería en Sistemas UNAH: https://is.unah.edu.hn/plan-de-estudio/

Joyanes Aguilar, L. (2012). COMPUTACIÓN EN LA NUBE Notas para una estrategia española en cloud computing. *Revista del Instituto Español de Estudios Estratégicos*(0), 87-110. Recuperado el 7 de Mayo de 2014, de http://dialnet.unirioja.es/servlet/articulo?codigo=4098278

Juliao Vargas, C. S. (10 de Agosto de 2012). *Universidad Técnica de Manabí.* Recuperado el 25 de Julio de 2013, de Universidad Técnica de Manabí: http://www.sisman.utm.edu.ec/libros/FACULTAD%20DE%20CIENCIAS%20ZOOT%C3%89CNICAS/CARRERA%20DE%20INGENIER%C3%8DA%20ZOOT%C3%89CNICA/02/METODOLOGIA%20DE%20LA%20INVESTIGACION%20CIENTIFICA/metodologia%20de%20investigacion.pdf

Lean Solutions. (2015). *VSM, Value Stream Mapping*. Obtenido de Lean Solutions: http://www.leansolutions.co/

Letelier, P., & Penadés, M. C. (2006). *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Universidad Politécnica de Valencia.

Liker, J. (2003). *The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer*. McGraw-Hill Education.

Liker, J. K., & Morgan, J. M. (2006). *The Toyota Product Development System: Integrating People, Process, and Technology*. New York.

Liker, J. M. (2006). *The Toyota Product Development System: Integrating People, Process, and Technology*. New York.

Liker, J., & Meier, D. (2005). *The Toyota Way Fieldbook.*

Manifiesto por el Desarrollo Ágil de Software. (2001). *Manifiesto por el Desarrollo Ágil de Software*. Obtenido de Manifiesto por el Desarrollo Ágil de Software: http://agilemanifesto.org

María. (s.f.). *METODOLOGÍAS DE DESARROLLO APLICADAS A SOA.* Tesis.

Martin, J. (1991). *Rapid Application Development.* New York: Macmillan Inc.

McConnell, S. (1996). *Rapid Development.*

MEDINA, L. E. (s.f.). *PROPUESTA DE APLICACIÓN DE SCRUM PARA MINIMIZAR LO RIESGOS EN UN PROYECTO DE DESARROLLO DE SOFTWARE.* . Tesis.

Méndez, A. V. (2010). *METODOLOGÍAS DE DESARROLLO DE SOFTWARE.* Mexico: INSTITUTO TECNOLÓGICO SUPERIOR DE APATZINGÁN.

Murazzo, M. A., & Rodríguez, N. R. (2010). Mobile cloud computing. *XII Workshop de Investigadores en Ciencias de la Computación* (págs. 522-526). Buenos Aires: PREBI - SEDICI. Obtenido de http://hdl.handle.net/10915/19570

Murazzo, M. A., Millán, F., Rodríguez, N., Segura, D., & Villafañe, D. A. (2010). Desarrollo de aplicaciones para Cloud Computing. *XVI Congreso Argentino de Ciencias de la Computación* (págs. 941-949). Buenos Aires: PREBI - SEDICI. Recuperado el 8 de Mayo de 2014, de http://sedici.unlp.edu.ar/handle/10915/19374

OEE-UNAH, O. E. (2014). *Tasa de Desmpleo* . Obtenido de Observatorio Economico y de Emprendimiento: http://oee.iies-unah.org/index.php/mercado-laboral/tasa-de-desempleo

Oppenheim, B. W. (2011). *LEAN FOR SYSTEMS ENGINEERING WITH LEAN ENABLERS FOR SYSTEMS ENGINEERING.* Singapore.

Otero, J. (1990). VARIABLES COGNITIYAS Y METACOGNITiVAS EN LA COMPRENSION DE TEXTOS CIENTIFICOS: EL PAPEL DE LOS ESQUEMAS Y EL CONTROL DE LA PROPIA COMPRENSION. *Enseñanza de las ciencias*, 17-22.

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit.* Addison Wesley.

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit.* Addison Wesley.

R. Bar, A. (2010). La Metodología Cuantitativa y su Uso en América Latina. *Cinta moebio [online]*, 1-14.

Real Academia Española. (2011). *DICCIONARIO DE LA LENGUA ESPAÑOLA - Vigésima segunda edición*. Recuperado el 25 de Julio de 2013, de http://lema.rae.es/drae/?val=Invetsigaci%C3%B3n: http://lema.rae.es

Roberto Hernandez Sampieri, C. F. (2010). Definición de los enfoques Cuantitativos y Cualitativos. En C. F. Roberto Hernandez Sampieri, *Metodología de la Investigación* (págs. 4,5). Mexico. Distrito Federal: Mc Graw Hill.

Roger S. Pressman, P. (s.f.). *Ingeniería del software U N E N F O Q U E P R Á C T I C O.* McGRAW-HILL INTERAMERICANA EDITORES, S.A. DE C.V.

Roger, P. (2002). *Ingenieria del Software un Enfoque Practico Quinta Edicion.* Madrid: McGraw-Hill.

Rother, M., & Shook, J. (1999). *Learning to See: Value Stream Mapping to Add Value and Eliminate MUDA.*

Sampieri, R. H., Collado, C. F., & Lucio, P. B. (2010). *Metodología de la Investigación* (5ta ed.). Perú: Mc Graw Hill.

Sampieri, R. H., Collado, F. C., & Baptista, L. P. (1991). *Metodología de Investigación.* Edo. de México, México: McGRAW - HILL INTERAMERICANA DE MÉXICO, S.A. de C.V.

Serrano S., A. J. (2013). LA INVESTIGACIÓN CIENTÍFICA APLICADO A LOS TRABAJOS DE INVESTIGACIÓN. *I Congreso de Educación, UNAN-Managua.* (págs. 1-12). Managua: UNAN.

SommerVille, I. (2005). *Ingeniería del software Séptima edición.* Madrid: PEARSON EDUCACIÓN. S.A.

STSS, S. d. (2011). *Antecedentes y Diagnostico del Mercado Laboral de Honduras.*

Taylor, S. J., & Bogdan, R. (20 de Junio de 2011). *MANAKIN.* Recuperado el 25 de Julio de 2013, de MANAKIN: http://201.147.150.252:8080/xmlui/handle/123456789/1216?show=full

Trágora: Traducción & Comunicación. (2 de Agosto de 2013). *Trágora: Traducción & Comunicación*. Obtenido de Trágora: Traducción & Comunicación: http://www.tragoratraducciones.com/servicios-de-traduccion/correccion-revision-de-textos-documentos-en-ingles/

Womack, J. P., Jones, D. T., & Roos, D. (1990). *The Machine That Changed The World.*

ANEXOS

Anexo 1 Instrumento para la evaluación de Tiempos y costos en el desarrollo de software utilizando una Metodología Ágil

Instrumento para la evaluación de Tiempos y costos en el desarrollo de software utilizando una Metodología Ágil

Objetivos: 1. Determinar el impacto en el tiempo de desarrollo de software con la integración de una metodología Ágil.

2. Determinar el impacto en los costos de desarrollo de software con la integración de una metodología Ágil.

Acerca del Instrumento: El instrumento que a continuación se aborda, se realiza sobre la estructura de un modelo genérico de desarrollo de software cual se apega en su generalidad a las metodologías pesadas de desarrollo y también a las metodologías agiles. Se definen a continuación cada etapa en el desarrollo del software para definir el impacto de las metodologías agiles desde un enfoque de tiempos y costos.

Dirigido A : Esta Instrumento está dirigida a profesionales del desarrollo de software, los cuales deben de poseer conocimientos y experiencias tanto de metodologías pesadas como agiles y que actualmente se desempeñen en un cargo de jefatura, director o líder de proyectos de software o gerenciamiento de un Departamento de Desarrollo de Software.

Conceptos generales antes de Abordar el Instrumento :

Metodologías de Desarrollo: Se refiere a todo el método utilizado para desarrollar un software desde la recopilación de los requerimientos hasta su entrega final.

Metodologías de Desarrollo Pesadas: Se refiere a metodologías de desarrollo poco flexibles y que no integran al usuario en su totalidad al proceso de desarrollo de software. Estas metodologías se basan en un seguimiento estricto poco flexible y recorren el ciclo de vida del software sin interacción con el usuario final o cliente.

Metodologías de Desarrollo Agiles : Se refiere a la metodologías de desarrollo flexibles y que integran ala usuario en la totalidad del proceso de desarrollo, este método se basa en un seguimiento mediante interacciones entre todos los implicados en el desarrollo del software, presentando cambios agiles en la producción de un software.

Ejemplos de las Metodologías de Desarrollo	
Metodologías Pesadas (tradicionales)	Metodologías Lean-Agiles
Capability Maturity Model (SW-CMM)	Extreme Programming (XP)
Capability Maturity Model Integration for Development (CMMI-DEV)	Scrum
Big Design Up Front (BDUF)	Agile Modeling Adaptive Software Development (ASD)
Cleanroom Software Engineering	Crystal Clear
Rational Unified Process (RUP)	Dynamic Systems Development Method (DSDM)
Essential Unified Process for Software Development (EssUP)	Feature Driven Development (FDD)
Fusebox Lifecycle Process (FLiP)	Lean Software Development (LSD)
Software Process Improvement and Capability dEtermination (SPICE)	Agile Unified Process (AUP)
Métrica	Software Development Rhythms
Jackson System Development (JSD)	Agile Documentation
Joint Application Development (JAD)	ICONIX Process
Open Unified Process (OpenUP)	Microsoft Solutions Framework (MSF)

Desarrollo sin metologia solo usando el ciclo de vida del software	Agile Data Method
	Database Refactoring
	LeanCMMI

Encuesta Ver 1.0

Instrucciones: A continuación se le abordara con una serie de preguntas agradecemos dar su respuesta con la mayor transparencia y veracidad a las diversas preguntas del cuestionario, lo cual nos permitirá un acercamiento científico a la realidad concreta del desarrollo de software. En cada pregunta marque con una X o con un √ en cada respuesta solo podrá elegir un ítem.

1. Nombre de la empresa

2. Cargo dentro de la empresa

3. Tiene Conocimiento sobre las Metodologías de Desarrollo de Software Lean-Agile.
SI__ NO__

4. En su empresa se utiliza una Metodología de Desarrollo de Software Lean-Agile
SI__ NO__

5. Anteriormente a la metodología que se utiliza en su empresa que metodología de desarrollo de Software se utilizaba.
Metodologías Pesadas (Tradicionales)_____
Metodologías Lean-Agile _____

6. A continuación se le presentara un cuadro, en el cual deberá detallar el impacto en el tiempo de duración de cada una de las fases de desarrollo de software utilizando la Metodología Ágil. Se espera que usted realice una comparación en base a su experiencia con las Metodologías Pesadas.

Cat.	Ítems	Desmejoro Notablemente	Desmejoro	No Hubo Cambios	Mejoro	Mejoro Notablemente
Requisitos de Usuario	Definir el Método de Levantamiento de Requisitos					
	Realizar el Levantamiento de los Requisitos					
	Análisis de los Requisitos					
	Especificación de los Requisitos					
	Validación de los Requisitos					
	Definir los Requisitos Funcionales					
	Definir los Requisitos No Funcionales					
	Definir el Entorno Operacional					
Requisitos de Hardware	Definir los Requisitos Funcionales					
	Definir los Requisitos De rendimiento					
	Definir los Requisitos De Interfaz					
	Definir los Requisitos Operacionales					
	Definir los Requisitos De Recursos					

	Definir los Requisitos De portabilidad					
	Definir los Requisitos De Calidad					
	Definir los Requisitos De Mantenimiento					
	Definir los Requisitos De Seguridad					
Arquitectura	Análisis de los Requerimientos de usuario y hardware					
	Definir las Restricciones					
	Diseñar el Modelo de Datos					
	Diseñar la Arquitectura					
	Definir el Lenguaje a Utilizar					
	Validar la Arquitectura					
Codificación	Análisis de los Requerimientos					
	Validación de la Arquitectura					
	Desarrollo de la Base de Datos					
	Desarrollo de la Aplicación					
Pruebas	Pruebas de Unidad					
	Pruebas de Integración					
	Pruebas del Sistema					
	Pruebas de Aceptación					
Entrega	Implementación					
	Verificación de Calidad					
	Levantamiento de Defectos por parte de los usuarios finales					

	Corrección de defectos después de la implantación					
	Entrega Final					

7. A continuación se le presentara un cuadro, en el cual deberá detallar el impacto los costos incurridos en cada de las fases de desarrollo de software utilizando la Metodología Ágil. Se espera que tomo como referencia la metodología que su usaba anteriormente en su empresa.

Cat.	Ítems	Desmejoro Notablemente	Desmejoro	No Hubo Cambios	Mejoro	Mejoro Notablemente
Requisitos de Usuario	Definir el Método de Levantamiento de Requisitos					
	Realizar el Levantamiento de los Requisitos					
	Análisis de los Requisitos					
	Especificación de los Requisitos					
	Validación de los Requisitos					
	Definir los Requisitos Funcionales					
	Definir los Requisitos No Funcionales					
	Definir el Entorno Operacional					
Requisitos de Hardware	Definir los Requisitos Funcionales					
	Definir los Requisitos De rendimiento					
	Definir los Requisitos De Interfaz					

	Definir los Requisitos Operacionales					
	Definir los Requisitos De Recursos					
	Definir los Requisitos De portabilidad					
	Definir los Requisitos De Calidad					
	Definir los Requisitos De Mantenimiento					
	Definir los Requisitos De Seguridad					
Arquitectura	Análisis de los Requerimientos de usuario y hardware					
	Definir las Restricciones					
	Diseñar el Modelo de Datos					
	Diseñar la Arquitectura					
	Definir el Lenguaje a Utilizar					
	Validar la Arquitectura					
Codificación	Análisis de los Requerimientos					
	Validación de la Arquitectura					
	Desarrollo de la Base de Datos					
	Desarrollo de la Aplicación					
Pruebas	Pruebas de Unidad					
	Pruebas de Integración					
	Pruebas del Sistema					
	Pruebas de Aceptación					
Entrega	Implementación					
	Verificación de Calidad					

	Levantamiento de Defectos por parte de los usuarios finales					
	Corrección de defectos después de la implantación					
	Entrega Final					

Gracias por su Colaboración!

Anexo 2 Validación del Instrumento de Medición.

Validación del Instrumento de Medición.

Acerca del Instrumento: El instrumento que a continuación se aborda, se realiza con el objetivo de validar el instrumento que se utilizara en la medición del impacto de las metodologías agiles desde un enfoque de tiempos y costos.

Dirigido A: Esta Instrumento está dirigida a Expertos del desarrollo de software, los cuales deben de poseer conocimientos y experiencias tanto de metodologías pesadas como agiles en el Desarrollo de Software.

Instrucciones: A continuación se le abordara con una serie de preguntas agradecemos dar su respuesta con la mayor transparencia y veracidad a las diversas preguntas del cuestionario, lo cual nos permitirá un acercamiento científico a la realidad concreta del desarrollo de software.

La pregunta general en todo la encuesta es:

¿El ítem que se aborda es válido o perteneciente para medir el del impacto de las metodologías agiles desde un enfoque de tiempos y costos?

En cada pregunta marque con una X o con un √ si el ítem es válido caso contrario dejara la casilla en blanco.

Pregunta	Ítem	Valido o perteneciente
1. Nombre de la empresa		
2. Cargo dentro de la empresa		
3. Tiene Conocimiento sobre las Metodologías de Desarrollo de Software Lean-Agile.	1	

SI__ NO__								
4. En su empresa se utiliza una Metodología de Desarrollo de Software Lean-Agile							2	
SI__ NO__								
5. Anteriormente a la metodología que se utiliza en su empresa que metodología de desarrollo de Software se utilizaba.							3	
Metodologías Pesadas (Tradicionales)_____								
Metodologías Lean-Agile _____								
6. A continuación se le presentara un cuadro, en el cual deberá detallar el impacto en el tiempo de duración de cada una de las fases de desarrollo de software utilizando la Metodología Ágil. Se espera que usted realice una comparación en base a su experiencia con las Metodologías Pesadas.								
Cat.	Ítems	Desmejoro Notablemente	Desmejoro	No Hubo Cambios	Mejoro	Mejoro Notablemente		
Requisitos de Usuario	Definir el Método de Levantamiento de Requisitos						4	
	Realizar el Levantamiento de los Requisitos						5	
	Análisis de los Requisitos						6	
	Especificación de los Requisitos						7	
	Validación de los Requisitos						8	
	Definir los Requisitos Funcionales						9	
	Definir los Requisitos No Funcionales						10	
	Definir el Entorno Operacional						11	

Requisitos de Hardware	Definir los Requisitos Funcionales						12	
	Definir los Requisitos De rendimiento						13	
	Definir los Requisitos De Interfaz						14	
	Definir los Requisitos Operacionales						15	
	Definir los Requisitos De Recursos						16	
	Definir los Requisitos De portabilidad						17	
	Definir los Requisitos De Calidad						18	
	Definir los Requisitos De Mantenimiento						19	
	Definir los Requisitos De Seguridad						20	
Arquitectura	Análisis de los Requerimientos de usuario y hardware						21	
	Definir las Restricciones						22	
	Diseñar el Modelo de Datos						23	

	Diseñar la Arquitectura						24	
	Definir el Lenguaje a Utilizar						25	
	Validar la Arquitectura						26	
Codificación	Análisis de los Requerimientos						27	
	Validación de la Arquitectura						28	
	Desarrollo de la Base de Datos						29	
	Desarrollo de la Aplicación						30	
Pruebas	Pruebas de Unidad						31	
	Pruebas de Integración						32	
	Pruebas del Sistema						33	
	Pruebas de Aceptación						34	
Entrega	Implementación						35	
	Verificación de Calidad						36	
	Levantamiento de Defectos por parte de los usuarios finales						37	
	Corrección de defectos después de la implantación						38	
	Entrega Final						39	

	7. A continuación se le presentara un cuadro, en el cual deberá detallar el impacto los costos incurridos en cada de las fases de desarrollo de software utilizando la Metodología Ágil. Se espera que tomo como referencia la metodología que su usaba anteriormente en su empresa.							
Cat.	Ítems	Desmejoro Notablemente	Desmejoro	No Hubo Cambios	Mejoro	Mejoro Notablemente		
Requisitos de Usuario	Definir el Método de Levantamiento de Requisitos						40	
	Realizar el Levantamiento de los Requisitos						41	
	Análisis de los Requisitos						42	
	Especificación de los Requisitos						43	
	Validación de los Requisitos						44	
	Definir los Requisitos Funcionales						45	
	Definir los Requisitos No Funcionales						46	
	Definir el Entorno Operacional						47	
Requisitos de Hardware	Definir los Requisitos Funcionales						48	
	Definir los Requisitos De rendimiento						49	

	Definir los Requisitos De Interfaz						50	
	Definir los Requisitos Operacionales						51	
	Definir los Requisitos De Recursos						52	
	Definir los Requisitos De portabilidad						53	
	Definir los Requisitos De Calidad						54	
	Definir los Requisitos De Mantenimiento						55	
	Definir los Requisitos De Seguridad						56	
Arquitectura	Análisis de los Requerimientos de usuario y hardware						57	
	Definir las Restricciones						58	
	Diseñar el Modelo de Datos						59	
	Diseñar la Arquitectura						60	
	Definir el Lenguaje a Utilizar						61	

	Validar la Arquitectura						62	
Codificación	Análisis de los Requerimientos						63	
	Validación de la Arquitectura						64	
	Desarrollo de la Base de Datos						65	
	Desarrollo de la Aplicación						66	
Pruebas	Pruebas de Unidad						67	
	Pruebas de Integración						68	
	Pruebas del Sistema						69	
	Pruebas de Aceptación						70	
Entrega	Implementación						71	
	Verificación de Calidad						72	
	Levantamiento de Defectos por parte de los usuarios finales						73	
	Corrección de defectos después de la implantación						74	
	Entrega Final						75	

Anexo 3 Tabulación Prueba de Confiabilidad

	SUJETOS	1	2	3	4	5	S (Varianza)
Variable	Código						
Metodología de Desarrollo	CI-1	1	1	1	1	1	0
	CI-2	1	1	1	1	2	0.16
	CI-3	1	1	1	1	2	0.16
Tiempo de desarrollo.	CI-4	3	3	3	3	3	0
	CI-5	5	5	5	5	5	0
	CI-6	5	5	5	5	5	0
	CI-7	5	5	5	5	5	0
	CI-8	4	4	4	4	4	0
	CI-9	5	5	5	5	5	0
	CI-10	5	5	4	5	4	0.24
	CI-11	4	4	4	3	4	0.16
	CI-12	4	5	4	4	4	0.16
	CI-13	3	4	4	4	4	0.16
	CI-14	5	4	5	4	5	0.24
	CI-15	5	5	4	5	5	0.16
	CI-16	5	5	5	5	4	0.16
	CI-17	3	4	3	3	3	0.16
	CI-18	4	5	4	4	4	0.16
	CI-19	3	4	3	4	3	0.24
	CI-20	3	3	4	3	3	0.16
	CI-21	5	5	5	4	5	0.16
	CI-22	5	4	5	4	5	0.24
	CI-23	3	3	4	4	4	0.24
	CI-24	5	5	4	4	4	0.24
	CI-25	3	3	3	4	4	0.24
	CI-26	4	5	4	4	4	0.16
	CI-27	5	5	5	5	4	0.16
	CI-28	4	4	5	4	4	0.16

	CI-29	5	5	5	5	4	0.16
	CI-30	5	5	5	5	4	0.16
	CI-31	5	4	4	4	4	0.16
	CI-32	4	4	5	5	4	0.24
	CI-33	4	4	4	4	4	0
	CI-34	5	5	5	5	4	0.16
	CI-35	5	5	5	5	4	0.16
	CI-36	5	5	5	5	4	0.16
	CI-37	5	5	5	5	4	0.16
	CI-38	4	4	4	4	4	0
	CI-39	5	5	5	5	4	0.16
Costos de desarrollo.	CI-40	3	3	3	3	3	0
	CI-41	3	3	3	3	3	0
	CI-42	4	4	4	4	4	0
	CI-43	5	5	5	5	5	0
	CI-44	4	4	4	4	4	0
	CI-45	3	3	3	3	3	0
	CI-46	3	3	3	3	3	0
	CI-47	3	3	3	3	3	0
	CI-48	3	3	3	3	3	0
	CI-49	3	3	3	3	3	0
	CI-50	3	3	3	3	3	0
	CI-51	3	3	3	3	3	0
	CI-52	3	3	3	3	3	0
	CI-53	3	3	3	3	3	0
	CI-54	3	3	3	3	3	0
	CI-55	3	3	3	3	3	0
	CI-56	3	3	3	3	3	0
	CI-57	5	5	5	5	5	0
	CI-58	4	4	4	4	4	0
	CI-59	3	3	3	3	3	0
	CI-60	4	4	4	4	4	0

	CI-61	3	3	3	3	3	0
	CI-62	4	4	4	4	4	0
	CI-63	5	5	5	5	5	0
	CI-64	4	4	4	4	4	0
	CI-65	4	4	4	4	4	0
	CI-66	5	5	5	4	3	0.64
	CI-67	5	5	5	4	3	0.64
	CI-68	5	5	5	4	3	0.64
	CI-69	5	5	5	4	3	0.64
	CI-70	5	5	5	4	3	0.64
	CI-71	5	5	5	4	3	0.64
	CI-72	5	5	5	4	3	0.64
	CI-73	5	5	5	4	3	0.64
	CI-74	5	5	5	4	3	0.64
	CI-75	5	5	5	4	3	0.64
Sumatoria de S (Varianza por Ítem)							11.84
ST (Varianza Total)		301	304	302	290	274	125.76

$$\alpha = \frac{K}{K-1}\left[1 - \frac{\sum S_i^{\,2}}{S_T^{\,2}}\right]$$

K:	El número de ítems	75.00
Σ Si²:	Sumatoria de las Varianzas de los Ítems	11.84
S_T^2:	La Varianza de la suma de los Ítems	125.76
α:	Coeficiente de Alfa de Cronbach	

75 [1 - 0.09]

74

1.01351351 [0.91]

α = 0.92

Entre más cerca de 1 está α, más alto es el grado de confiabilidad

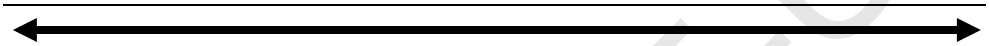
CONFIABILIDAD:

-Se puede definir como la estabilidad o consistencia de los resultados obtenidos

-Es decir, se refiere al grado en que la aplicación repetida del instrumento, al mismo sujeto u objeto, produce iguales resultados

-Ejemplo, si un Test de Inteligencia Emocional se aplica hoy a un grupo de profesores y proporciona ciertos datos; si se aplica un mes después y proporciona valores diferentes y de manera similar en mediciones subsecuentes, tal prueba no es confiable

C O N F I A B I L I D A D



Muy Baja Baja Regular Aceptable Elevada

0 **1**

0% de confiabilidad en la medición (la medición está contaminada de error)

100% de confiabilidad en la medición (no hay error)

Anexo 4 Instrumento por Variables

Variable	Pregunta						Ítem
Contexto general (No Tiene Variable)	1. Nombre de la empresa						
	2. Cargo dentro de la empresa						
Metodología de Desarrollo	3. Tiene Conocimiento sobre las Metodologías de Desarrollo de Software Lean-Agile.						1
	SI__ NO__						
	4. En su empresa se utiliza una Metodología de Desarrollo de Software Lean-Agile						2
	SI__ NO__						
	5. Anteriormente a la metodología que se utiliza en su empresa que metodología de desarrollo de Software se utilizaba.						3
	Metodologías Pesadas (Tradicionales)_____						
	Metodologías Lean-Agile _____						
Tiempo de desarrollo.	6. A continuación se le presentara un cuadro, en el cual deberá detallar el impacto en el tiempo de duración de cada una de las fases de desarrollo de software utilizando la Metodología Ágil. Se espera que usted realice una comparación en base a su experiencia con las Metodologías Pesadas.						
	Cat.	Ítems	Desmejoro Notablemente	Desmejoro	No Hubo Cambios	Mejoro	Mejoro Notablemente	
	Requisitos de Usuario	Definir el Método de Levantamiento de Requisitos						4
		Realizar el Levantamiento de los Requisitos						5
		Análisis de los Requisitos						6
		Especificación de los Requisitos						7
		Validación de los Requisitos						8
		Definir los Requisitos Funcionales						9
		Definir los Requisitos No Funcionales						10
		Definir el Entorno Operacional						11
	Requisitos de Hardware	Definir los Requisitos Funcionales						12
		Definir los Requisitos De rendimiento						13
		Definir los Requisitos De Interfaz						14

		Definir los Requisitos Operacionales						15
		Definir los Requisitos De Recursos						16
		Definir los Requisitos De portabilidad						17
		Definir los Requisitos De Calidad						18
		Definir los Requisitos De Mantenimiento						19
		Definir los Requisitos De Seguridad						20
	Arquitectura	Análisis de los Requerimientos de usuario y hardware						21
		Definir las Restricciones						22
		Diseñar el Modelo de Datos						23
		Diseñar la Arquitectura						24
		Definir el Lenguaje a Utilizar						25
		Validar la Arquitectura						26
	Codificación	Análisis de los Requerimientos						27
		Validación de la Arquitectura						28
		Desarrollo de la Base de Datos						29
		Desarrollo de la Aplicación						30
	Pruebas	Pruebas de Unidad						31
		Pruebas de Integración						32
		Pruebas del Sistema						33
		Pruebas de Aceptación						34
	Entrega	Implementación						35
		Verificación de Calidad						36
		Levantamiento de Defectos por parte de los usuarios finales						37
		Corrección de defectos después de la implantación						38
		Entrega Final						39
Costos de desarrollo.		7. A continuación se le presentara un cuadro, en el cual deberá detallar el impacto los costos incurridos en cada de las fases de desarrollo de software utilizando la Metodología Ágil. Se espera que tomo como referencia la metodología que su usaba anteriormente en su empresa.						
	Cat.	Ítems	Desmejoro Notablemente	Desmejoro	No Hubo Cambios	Mejoro	Mejoro Notablemente	

	Requisitos de Usuario	Definir el Método de Levantamiento de Requisitos						40
		Realizar el Levantamiento de los Requisitos						41
		Análisis de los Requisitos						42
		Especificación de los Requisitos						43
		Validación de los Requisitos						44
		Definir los Requisitos Funcionales						45
		Definir los Requisitos No Funcionales						46
		Definir el Entorno Operacional						47
	Requisitos de Hardware	Definir los Requisitos Funcionales						48
		Definir los Requisitos De rendimiento						49
		Definir los Requisitos De Interfaz						50
		Definir los Requisitos Operacionales						51
		Definir los Requisitos De Recursos						52
		Definir los Requisitos De portabilidad						53
		Definir los Requisitos De Calidad						54
		Definir los Requisitos De Mantenimiento						55
		Definir los Requisitos De Seguridad						56
	Arquitectura	Análisis de los Requerimientos de usuario y hardware						57
		Definir las Restricciones						58
		Diseñar el Modelo de Datos						59
		Diseñar la Arquitectura						60
		Definir el Lenguaje a Utilizar						61
		Validar la Arquitectura						62
	Codificación	Análisis de los Requerimientos						63
		Validación de la Arquitectura						64
		Desarrollo de la Base de Datos						65
		Desarrollo de la Aplicación						66

	Pruebas	Pruebas de Unidad						67
		Pruebas de Integración						68
		Pruebas del Sistema						69
		Pruebas de Aceptación						70
	Entrega	Implementación						71
		Verificación de Calidad						72
		Levantamiento de Defectos por parte de los usuarios finales						73
		Corrección de defectos después de la implantación						74
		Entrega Final						75

Anexo 5 Codificación del Instrumento

Variable	Pregunta							N. Pregunta	Código
Contexto General (No	1. Nombre de la empresa								
	2. Cargo dentro de la empresa								
Metodología de Desarrollo	3. Tiene Conocimiento sobre las Metodologías de Desarrollo de Software Lean-Agile.							1	CI-1
	SI__ NO__								
	4. En su empresa se utiliza una Metodología de Desarrollo de Software Lean-Agile							2	CI-2
	SI__ NO__								
	5. Anteriormente a la metodología que se utiliza en su empresa que metodología de desarrollo de Software se utilizaba.							3	CI-3
	Metodologías Pesadas (Tradicionales)_____								
	Metodologías Lean-Agile _____								
Tiempo de desarrollo.	6. A continuación se le presentara un cuadro, en el cual deberá detallar el impacto en el tiempo de duración de cada una de las fases de desarrollo de software utilizando la Metodología Ágil. Se espera que usted realice una comparación en base a su experiencia con las Metodologías Pesadas.								
	Cat.	Ítems	Desmejoro Notablemente	Desmejoro	No Hubo Cambios	Mejoro	Mejoro Notablemente		
	Requisitos de Usuario	Definir el Método de Levantamiento de Requisitos						4	CI-4
		Realizar el Levantamiento de los Requisitos						5	CI-5
		Análisis de los Requisitos						6	CI-6
		Especificación de los Requisitos						7	CI-7
		Validación de los Requisitos						8	CI-8
		Definir los Requisitos Funcionales						9	CI-9
		Definir los Requisitos No Funcionales						10	CI-10

		Definir el Entorno Operacional						11	CI-11
	Requisitos de Hardware	Definir los Requisitos Funcionales						12	CI-12
		Definir los Requisitos De rendimiento						13	CI-13
		Definir los Requisitos De Interfaz						14	CI-14
		Definir los Requisitos Operacionales						15	CI-15
		Definir los Requisitos De Recursos						16	CI-16
		Definir los Requisitos De portabilidad						17	CI-17
		Definir los Requisitos De Calidad						18	CI-18
		Definir los Requisitos De Mantenimiento						19	CI-19
		Definir los Requisitos De Seguridad						20	CI-20
	Arquitectura	Análisis de los Requerimientos de usuario y hardware						21	CI-21
		Definir las Restricciones						22	CI-22
		Diseñar el Modelo de Datos						23	CI-23
		Diseñar la Arquitectura						24	CI-24

		Definir el Lenguaje a Utilizar						25	CI-25
		Validar la Arquitectura						26	CI-26
	Codificación	Análisis de los Requerimientos						27	CI-27
		Validación de la Arquitectura						28	CI-28
		Desarrollo de la Base de Datos						29	CI-29
		Desarrollo de la Aplicación						30	CI-30
	Pruebas	Pruebas de Unidad						31	CI-31
		Pruebas de Integración						32	CI-32
		Pruebas del Sistema						33	CI-33
		Pruebas de Aceptación						34	CI-34
	Entrega	Implementación						35	CI-35
		Verificación de Calidad						36	CI-36
		Levantamiento de Defectos por parte de los usuarios finales						37	CI-37
		Corrección de defectos después de la implantación						38	CI-38
		Entrega Final						39	CI-39
Costos de desarrollo.		7. A continuación se le presentara un cuadro, en el cual deberá detallar el impacto los costos incurridos en cada de las fases de desarrollo de software utilizando la Metodología Ágil. Se espera que tomo como referencia la metodología que su usaba anteriormente en su empresa.							
	Cat.	Ítems	Desmejoro Notablemente	Desmejoro	No Hubo Cambios	Mejoro	Mejoro Notablemente		

	Requisitos de Usuario	Definir el Método de Levantamiento de Requisitos						40	CI-40
		Realizar el Levantamiento de los Requisitos						41	CI-41
		Análisis de los Requisitos						42	CI-42
		Especificación de los Requisitos						43	CI-43
		Validación de los Requisitos						44	CI-44
		Definir los Requisitos Funcionales						45	CI-45
		Definir los Requisitos No Funcionales						46	CI-46
		Definir el Entorno Operacional						47	CI-47
	Requisitos de Hardware	Definir los Requisitos Funcionales						48	CI-48
		Definir los Requisitos De rendimiento						49	CI-49
		Definir los Requisitos De Interfaz						50	CI-50
		Definir los Requisitos Operacionales						51	CI-51
		Definir los Requisitos De Recursos						52	CI-52
		Definir los Requisitos De portabilidad						53	CI-53

		Definir los Requisitos De Calidad						54	CI-54
		Definir los Requisitos De Mantenimiento						55	CI-55
		Definir los Requisitos De Seguridad						56	CI-56
	Arquitectura	Análisis de los Requerimientos de usuario y hardware						57	CI-57
		Definir las Restricciones						58	CI-58
		Diseñar el Modelo de Datos						59	CI-59
		Diseñar la Arquitectura						60	CI-60
		Definir el Lenguaje a Utilizar						61	CI-61
		Validar la Arquitectura						62	CI-62
	Codificación	Análisis de los Requerimientos						63	CI-63
		Validación de la Arquitectura						64	CI-64
		Desarrollo de la Base de Datos						65	CI-65
		Desarrollo de la Aplicación						66	CI-66
	Pruebas	Pruebas de Unidad						67	CI-67
		Pruebas de Integración						68	CI-68
		Pruebas del Sistema						69	CI-69
		Pruebas de Aceptación						70	CI-70

	Entrega	Implementación						71	CI-71
		Verificación de Calidad						72	CI-72
		Levantamiento de Defectos por parte de los usuarios finales						73	CI-73
		Corrección de defectos después de la implantación						74	CI-74
		Entrega Final						75	CI-75