UNIVERSIDAD TÉCNICA DEL NORTE
INSTITUTO DE POSTGRADO

MAESTRÍA EN INGENIERÍA DE SOFTWARE

“IMPLEMENTACIÓN DE UNA METODOLOGÍA DE DESARROLLO ÁGIL PARA EL MEJORAMIENTO DE LA GESTIÓN DE LOS PROYECTOS DE SOFTWARE EN LA UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY”

Trabajo de Investigación previo a la obtención del Título de Magister en Ingeniería de Software

AUTOR: Lincon Hermel Chamorro Andrade.

TUTOR: Jose Antonio Quiña Mera.

IBARRA – ECUADOR

2018

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del proyecto Repositorio Digital Institucional, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1002512398		
APELLIDOS Y NOMBRES:	Chamorro Andrade Lincon Hermel		
DIRECCIÓN:	Azaya, San Salvador 346 y Latacunga		
EMAIL:	lincoln_10025@yahoo.es		
TELÉFONO FIJO:	2545475	**TELÉFONO MÓVIL**	0982663699

DATOS DE LA OBRA	
TÍTULO:	"IMPLEMENTACIÓN DE UNA METODOLOGÍA DE DESARROLLO ÁGIL PARA EL MEJORAMIENTO DE LA GESTIÓN DE LOS PROYECTOS DE SOFTWARE EN LA UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY"
AUTOR (ES):	Chamorro Andrade Lincon Hermel
FECHA: AAAAMMDD	2019/02/12
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	☐ **PREGRADO** ☒ **POSGRADO**
TITULO POR EL QUE OPTA:	Título de Magister en Ingeniería de Software
ASESOR /DIRECTOR:	Ing. Jose Antonio Quiña Mera

2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Chamorro Andrade Lincon Hermel, con cédula de identidad Nro. 1002512398, en calidad de autor (es) y titular (es) de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en formato digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión; en concordancia con la Ley de Educación Superior Artículo 144.

3. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 12 días del mes de febrero de 2019

EL AUTOR:

(Firma).....
Nombre: Chamorro Andrade Lincon Hermel
C.C. 1002512398

UNIVERSIDAD TÉCNICA DEL NORTE

Resolución N° 001-073 CEAACES -2013-13

INSTITUTO DE POSTGRADO

Ibarra, 23 de Noviembre del 2018

Dra. Lucía Yépez V MSc.
Directora
Instituto de Postgrado

ASUNTO: Conformidad con el documento final

Señora Directora:

Nos permitimos informar a usted que revisado el Trabajo de Grado *"Implementación de una metodología de desarrollo ágil para el mejoramiento de la gestión de los proyectos de software en la Universidad de Investigación de Tecnología Experimental Yachay"* del maestrante Lincon Chamorro, de la Maestría de Ingeniería de Software, certificamos que han sido acogidas y satisfechas todas las observaciones realizadas.

Atentamente,

	Apellidos y Nombres	Firma
Tutor/a	Quiña Mera Antonio	
Asesor/a	García Santillan Iván	

AUTORÍA

Yo, Lincon Hermel Chamorro Andrade con cédula de identidad Nro. 1002512398 declaro bajo juramento que el presente trabajo "*Implementación de una metodología de desarrollo ágil para el mejoramiento de la gestión de los proyectos de software en la Universidad de Investigación de Tecnología Experimental Yachay*", es de mi autoría, así como los resultados de la investigación y que ha sido presentado previamente para ningún grado ni calificación profesional y se ha respetado las fuentes de información.

El Autor:

Ing. Lincon Hermel Chamorro Andrade

CI: 1002512398

DEDICATORIA

Este proyecto de tesis lo dedico a DIOS, que es el dador de todo en la vida, a mis padres que con esfuerzo y sacrificio me educaron y cultivaron en mí el deseo de superarme y alcanzar todas las metas que me proponga.

A mis compañeros de estudio, quienes se volvieron parte de este arduo pero significativo proceso. A mi tutor, profesores y consejeros por apoyarme y corregirme con paciencia y dedicación, con el fin de obtener excelentes resultados en mi investigación.

A mi novia por estar a mi lado a cada momento brindándome su apoyo para continuar con esta tarea.

Lincon

AGRADECIMIENTO

Primeramente, doy gracias a Dios por brindarme las fuerzas necesarias para terminar con eficiencia este proyecto de investigación.

A mi novia, padres y hermanos por creer en mí y apoyarme firmemente durante todo el tiempo de estudio.

Agradezco de manera especial a la Dirección de Tecnología de la Información de la Universidad Yachay Tech, por permitirme desarrollar la investigación.

A mis compañeros de maestría, al director de tesis Ing. Antonio Quiña por guiarme y supervisar mi trabajo.

Gracias a todos ustedes.

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO I. PROBLEMA DE INVESTIGACIÓN

1 Tema

IMPLEMENTACIÓN DE UNA METODOLOGÍA DE DESARROLLO ÁGIL PARA EL MEJORAMIENTO DE LA GESTIÓN DE LOS PROYECTOS DE SOFTWARE EN LA UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY TECH.

1.1 Introducción

La importancia del software en el mundo es muy relevante ya que a través de éste se procesan grandes cantidades de información en áreas como la educación, medicina, industria, tecnología, economía etc. Su evolución y crecimiento en las últimas décadas ha sido exponencial convirtiéndose actualmente en común denominador dentro de la sociedad; la relevancia del software es tal, que es considerado un bien intangible con un costo superior al hardware; en un mundo globalizado en constante evolución donde las barreras físicas y culturales prácticamente han desaparecido, esa integración e interrelación con la sociedad genera gran cantidad de información. A partir de la existencia de esta información es que el software adquiere una connotación transcendental ya que su función es gestionar y procesar estos datos. En la actualidad muchas organizaciones tienen la necesidad de automatizar sus procesos, en este contexto el hablar de gestión de proyectos de software es cotidiano, lo cual implica un gran reto para las áreas encargadas del desarrollo de software, las que desean mejorar su productividad (Pressman R. , 2010) .

La Universidad de Investigación de Tecnología Experimental Yachay Tech constituye un reto sin precedentes en la Historia para el Ecuador, con la firme idea de convertir al País en exportador de tecnología, en un referente para la región en investigación, en un prestigioso rincón del mundo donde se encontrarán las mentes más brillantes del planeta y por supuesto ser una nación sustentable a partir de las ideas y no de los recursos no renovables.

La formación de profesionales que contribuyan con el desarrollo de la tecnología, la economía, el espíritu empresarial, la cultura y el futuro del Ecuador; ser la gente que sabe cómo aprovechar este potencial y llevar al país muy lejos son sus objetivos.

1.2 Planteamiento del problema

La Universidad Yachay Tech a partir de su creación en el año 2014 ha experimentado un crecimiento continuo lo cual exige la implementación de procesos enmarcados en la “Normas de Control Interno para las entidades, organismos del sector público y personas jurídicas de derecho privado que dispongan de recursos públicos” (NIC), disposición de la constitución de la república para instituciones públicas; al ser este un proyecto de carácter emblemático requiere estar a la altura y exigencia de estándares de calidad nacionales e internacionales.

La demanda de construcción de software a medida es continua y se solicita de manera urgente; es esto lo que se evidencia y observa desde el área de desarrollo de software de la Dirección de Tecnologías de la Información. Se evidencia retrasos permanentes en la entrega de proyectos; el cliente siempre busca la entrega del producto en el menor tiempo posible. Se observa dificultades para realizar una adecuada planificación de los proyectos solicitados.

Se han creado aplicaciones de software de manera apresurada sin emplear metodologías de desarrollo; presumiblemente sea esta razón por la cual se evidencia incumplimiento de plazos de entrega, inadecuada estimación de costos y además presentan dificultades en el funcionamiento.

La socialización de los proyectos de desarrollo de software con otras áreas de la Universidad es deficiente. Los requerimientos suelen ser ambiguos y poco claros, lo cual dificulta entender la lógica del negocio y a su vez se obtiene como resultado final un software que no se ajusta a las necesidades del cliente.

Son recurrentes los problemas al momento de establecer los roles a los involucrados con proceso de desarrollo de software; se observa falta de compromiso de la alta gerencia de la institución, la participación de los usuarios es esporádica.

Insuficiente administración de los riesgos que ayude a mitigar un evento o condición incierta que, si sucede, tendrá un efecto en por lo menos uno de los objetivos del proyecto. Por esto y más factores el área de desarrollo de software se ve en la necesidad de implementar métodos que ayuden en la gestión de sus proyectos.

Para fundamentar los problemas de desarrollo de software se levantó una encuesta (Ver anexo 1) al equipo de desarrollo de la Dirección de Tecnologías de la Información; en dicha encuesta se corrobora los planteamientos anteriormente descritos.

1.3 Formulación del problema

¿La implementación de una metodología de desarrollo ágil contribuye al mejoramiento de la gestión de los proyectos de software en la Universidad de Investigación de Tecnología Experimental Yachay Tech?

1.4 Justificación

El estudio investigativo es muy importante ya que pretende dejar establecidas las bases metodológicas en el área de desarrollo de software de la dirección de TICs. La Universidad Yachay Tech al estar en un proceso de creación y construcción necesita normar sus procesos; estos se deben enmarcar en las NIC para las áreas de tecnologías de la información de las entidades públicas. La implementación de metodologías para el desarrollo pretende ayudar en el mejoramiento de la gestión de los proyectos de software; y por ende un progreso significativo en la ejecución de los procesos administrativos.

Desde una perspectiva técnica es importante tener un marco metodológico de desarrollo de software que pueda llevar de una manera organizada y estructurada el proyecto en todo su ciclo de desarrollo.

La estimación de los tiempos de ejecución son frecuentemente erróneas lo cual afecta la productividad del cliente y el equipo de trabajo.

1.5 Objetivos

1.5.1 Objetivo general

- Implementar una metodología ágil para el mejoramiento de la gestión de los proyectos de desarrollo de software en la Universidad Yachay Tech.

1.5.2 Objetivos específicos

- Realizar un análisis comparativo de las metodologías ágiles y determinar cuál de estas se adaptan a las necesidades de desarrollo de software de la institución.
- Aplicar la metodología de desarrollo ágil seleccionada mediante un estudio de caso.
- Determinar los factores críticos de éxito en la adopción de la metodología.

1.6 Pregunta de investigación o proposición

La implementación de metodologías de desarrollo ágiles, incide en el mejoramiento de la gestión de proyectos de software en Yachay Tech.

Preguntas directrices

- ¿Qué metodología ágil de desarrollo de software se adapta a las necesidades del área de desarrollo de software de la Dirección de TICs de Yachay Tech?
- ¿En el estudio de caso se aplica todas las fases de la metodología seleccionada?
- ¿Qué factores de éxito se encontraron al implementar la metodología en el área de desarrollo de software de la dirección de TICs de Yachay Tech?
- ¿Luego de la implementación de la metodología ágil, mejoró la gestión de los proyectos de software?

1.7 Variables e indicadores

Variable independiente: Implementación de una metodología ágil para el desarrollo de software.

Variable dependiente: Mejoramiento de la gestión de los proyectos de software.

Indicadores de gestión de proyectos:

- Grado de comunicación con el cliente.
- Grado de satisfacción del cliente en el proceso de desarrollo.
- Porcentaje de cumplimiento de actividades planificadas.
- Porcentaje de las tareas no planificadas en las interacciones.
- Nivel de respuesta ante cualquier incidente durante el proceso de desarrollo.
- Grado de adopción de las metodologías ágiles.

Para establecer los indicadores de gestión se tomó como referencia las recomendaciones de Lynch, R && Cross, K (1995) que hacen énfasis en tres aspectos como la satisfacción del cliente, la productividad y flexibilidad.

(Ver anexo 2) Matrices de Operacionalización de las variables.

CAPÍTULO II. MARCO TEÓRICO

2 Marco teórico – estado del arte

2.1 Introducción al área de estudio.

El crecimiento de la industria del software ha sido favorecido desde inicios de los años 90 por la incursión de países de economías emergentes que centraron sus estrategias en el desarrollo de software a la medida, respondiendo a las necesidades de un nicho de mercado específico (Arora y Gambardella, 2005). El desarrollo del software resultó cada día más complejo debido a su rápida evolución y crecimiento; con este antecedente aparece la ingeniería del software como una disciplina para el desarrollo de sistemas con énfasis en la calidad; su aparecimiento se denota desde hace tres décadas aproximadamente; durante ese tiempo, la ingeniería del software evolucionó desde una oscura idea practicada por un número relativamente pequeño de interesados hasta una legítima disciplina de la ingeniería. En la actualidad, se le reconoce como un área merecedora de investigación seria y estudio complejo.

A lo largo de toda la industria, el ingeniero de software sustituyó al programador como el título laboral de preferencia. Los modelos de proceso del software, los métodos de ingeniería de software y las herramientas del software, se adoptaron exitosamente a través de un amplio espectro de segmentos industriales (Pressman R. , 2010). En este contexto la gestión de las instituciones se encuentra cada vez más soportada en sistemas de información intensivos en software, los cuales son fundamentales para apoyar el liderazgo estratégico de la organización en el mercado. Estas condiciones de contexto generan permanentes demandas a la Ingeniería del software, con una premisa fundamental: la calidad (Anaya, 2006).

En el desarrollo del software es indispensable establecer un entorno disciplinado y estructurado que genere una ventaja competitiva frente a las demás organizaciones y en el cual predomine el trabajo en equipo, para evitar resultados impredecibles (Bernardo Quintero, 2005).

La ingeniería del software estudia, analiza y recomienda un conjunto de buenas prácticas para el desarrollo del software; en ese contexto han

aparecido múltiples metodologías con enfoque en la gestión; estas metodologías se presentan con ciclos sistemáticos y poco flexibles, otras se presentan con opciones de retroalimentación y con posibilidades de iteración dentro del desarrollo de los proyectos (Pressman R. , 2010). Estas metodologías tienen diversos enfoques; ya sea para grandes, mediano o pequeños equipos de desarrollo. Como podemos evidenciar por lo anteriormente expuesto, la aplicación de la ingeniería del software en el desarrollo de proyectos, ayuda en el mejoramiento de la gestión. La firma Standish Group publicó los resultados acerca de un estudio realizado sobre diferentes empresas desarrolladoras de software en el Reino Unido en el 2015; se indica que aproximadamente el 50% de los proyectos fueron abandonados por errónea estimación en costo y tiempo.

La tendencia actual es la orientación a procesos del desarrollo de software lo que representa un paso en la dirección correcta, aun así, las expectativas surgidas a partir de la evolución tecnológica y del nacimiento de las herramientas CASE (Computer Aided Software Engineering), no han aportado todos los beneficios esperados. En este sentido la experimentación de tales herramientas en la industria ha permitido probar que las principales razones del fracaso de proyectos de software tienen que ver poco con la tecnología y con las herramientas utilizadas en el desarrollo, pero si mucho con los procesos y los equipos de personas que los llevan a cabo.

En Ecuador, la industria del software está en constante avance y crecimiento, especialmente en el sector privado; en la mayoría de las empresas se han evidenciado problemas en plazos de entrega y estimación de costos. La Revista Tecnológica ESPOL – RTE (2014, pág. 1) en un estudio sobre el aumento de productividad utilizando una metodología ágil indica que de todos los beneficios con los que aportan las metodologías ágiles, producir resultados, el aumento de la productividad y la adaptación, son los beneficios principales que impactan a la competitividad de una organización de manera directa. En otro estudio exploratorio que realizó la ESPOL sobre las empresas de desarrollo en Ecuador, se señala que a

pesar de que la mayoría de éstas conocen de la existencia de estándares internacionales de calidad, no las utilizan. En este contexto La Universidad Yachay Tech al estar atravesando por un proceso de creación y formación; la aplicación de metodologías ágiles de desarrollo de software es importante para su futuro crecimiento como Universidad emblemática referente a nivel latinoamericano.

Pantoja, Collazos, & Penichet (2013) realizaron un estudio el cual estableció que “En la actualidad los temas de calidad y mejora de procesos de software son primordiales para impulsar la industria del mismo”. El artículo escrito por Mejía, Muñoz & Rocha (2013) establece que “La mejora de procesos del software es un mecanismo importante para impulsar la competitividad y eficiencia en las organizaciones de software.

La Revista Tecnológica ESPOL – RTE (2015, p.476) De acuerdo a los resultados obtenidos en el estudio realizado sobre la implementación de metodologías ágiles en pequeñas empresas concluye que “ha sido posible observar un creciente interés sobre las metodologías ágiles, tanto en el aumento de su aplicación e implementación en empresas desarrolladoras de software, así como en la cantidad de estudios relacionados a su adopción. Por otra parte, ha sido interesante poder identificar cómo el interés por integrar prácticas ágiles con modelos tradicionales, convencionales, de facto y estándares internacionales ha aumentado considerablemente. Esto, quizás a necesidades particulares y también generalizadas, las cuales buscan solucionar problemas relacionados a: procesos complejos, retardos en las entregas, insatisfacción del cliente, costos, entre otros”.

En el artículo publicado por Kaur, Jajoo, & Manisha (2015) sobre “Aplicación de metodologías ágiles en proyectos de la industria: ventajas y desafíos” concluye que las metodologías agiles podrían no ser un ajuste perfecto para proyectos de desarrollo a gran escala, de prueba o de distribución, como lo es para proyectos de desarrollo pequeños y de

ubicación conjunta, pero de hecho es una metodología poderosa que puede permitir a los equipos mejorar la productividad. Mejorar la visibilidad y lograr una mayor satisfacción del cliente. Esencialmente, da libertad a los equipos de proyectos para la planificación adaptativa y el desarrollo evolutivo.

2.2 Evolución cronológica de las metodologías de desarrollo de software

Los primeros indicios del software datan desde los inicios de la década de 1940, el desarrollo del software ha evolucionado hasta convertirse en una profesión que se ocupa de cómo crear software y garantizar su calidad. Su ámbito puede referirse a la capacidad de maximizar varios factores como la mantenibilidad del software, su estabilidad, velocidad, usabilidad, legibilidad, tamaño, costo, seguridad y número de fallas, así como, entre muchos otros atributos, a características menos medibles como la concisión y satisfacción del cliente.

Aunque la Ingeniería del software (IS) aparece a finales de los años sesenta y principios de los setenta, se puede considerar todavía una disciplina muy nueva, sobre todo si la comparamos con otros tipos de ingenierías. Su evolución ha ido muy relacionada al avance de los lenguajes de programación usados en la implementación del software. Al igual que los lenguajes de programación han pasado de ser estructurados a orientados a objetos. La IS también en su evolución tiene fases marcadas, que son abordadas en los distintos estudios sobre la misma. Ésta distinción hace referencia a las técnicas utilizadas en cada una de las fases del desarrollo (principalmente en las fases de análisis y diseño) y no tanto a cómo se organizan estas fases entre sí. En lo relativo a esta organización, en un principio, las fases se organizaban secuencialmente y rápidamente fue evolucionando hasta los métodos iterativos e incrementales, que son los dominantes hoy en día.

2.2.1 Primera fase (1940 - 1960): Programación o técnicas de programación

Según (Peña, 2006) los inicios del software comienzan por la década de los 40, donde prevalecía el hardware sobre el software, el objetivo principal era programar algoritmos para que los computadores hagan los cálculos, procesos y reportes, todavía no se evidenciaba una metodología clara para realizar programas o sistemas, de esta manera se mantuvo hasta finales de la década de los 60, donde se origina la necesidad del desarrollo de sistemas funcionales de negocio a gran escala para los grandes conglomerados empresariales.

El desarrollo más importante fue que nuevos computadores salían aproximadamente cada uno o dos años, haciendo obsoletos los ya existentes. Una característica importante fue que el código de los programas del software se tenía que escribir en cada una de las máquinas. Los programadores no tenían equipos en sus escritorios y tenían que ir a la sala de máquinas. Las tareas eran corridas al inscribirse para tiempo de máquina o por el personal operativo; adicionalmente se ponían tarjetas perforadas como entrada en el lector de tarjetas y se esperaban por resultados devueltos en la impresora.

En esta etapa aparecen los analistas programadores y analistas de sistemas. Las técnicas de codificación de la época generalmente dejaban a los usuarios insatisfechos ya que sus requerimientos no estaban analizados previamente de manera detenida y con claridad; ante estas circunstancias de evidenció la importancia del análisis y diseño en el desarrollo de un sistema. En tanto la dificultad de las tareas que realizaban las computadoras aumentaba, se vio necesario proponer métodos más sencillos para programar. En aquel tiempo se crearon los lenguajes de tercera generación que se diferenciaban de las anteriores en que sus instrucciones eran de alto nivel, es decir, eran comprensibles para el usuario o programador como si fuera un lenguaje natural; estos lenguajes eran independientes de la máquina, se podían compilar en otras máquinas, su código no estaba asociado al hardware. Estos lenguajes se

denominaron de alto nivel. Entre los más conocidos están FORTRAN que fue creado para el desarrollo de aplicaciones científicas de investigación y de ingeniería; y COBOL que fue creado para el desarrollo de aplicaciones para la gestión administrativa; LISP es el segundo lenguaje de programación de alto nivel de mayor antigüedad entre los que continúan teniendo un uso extendido en la actualidad, desarrollado originalmente en 1958 por John McCarthy y sus colaboradores en el Instituto Tecnológico de Massachusetts (Macluskey, 2009).

La Ingeniería del software apareció por primera vez en la década de 1950 y principios de los años 1960. Los programadores siempre tuvieron conocimiento sobre ingenieros civiles, eléctricos y de computadores; recién se debatía qué podría significar la ingeniería para el software. En 1958, el renombrado matemático estadístico John Tukey acuñó el término software.

Algunas de las características relevantes de esta etapa son las siguientes:

- No existía documentación de ningún tipo.
- Existencia de pocos métodos formales.
- Escepticismo en la comunidad científica de ese entonces.
- Desarrollo a base de prueba y error.

2.2.2 Segunda fase (1960-1970): Modelo de procesos

El modelo de procesos predominaba en los años 60, consistía en codificar y corregir. Entonces codificar y corregir surge como un modelo poco útil, pero sin embargo es la respuesta para muchos programadores al no tener una estructura formal a seguir. Realmente se la considero aplicable ya que de hecho la programación se manejaba de forma intuitiva. A pesar que el desarrollo de software tomo un enfoque de tarea unipersonal y donde el mismo programador era el usuario de la aplicación. Codificar y corregir se consideró como una base inicial para la fabricación del software, en vista de que este modelo se empieza a establecer como idea general de lo que se necesita construir, se utiliza cualquier combinación de diseño, código, depuración y métodos de prueba no formales que se los aplica

hasta que se tiene el producto listo para entregarlo (Garcés Guayta & Egas, 2013). Si al terminar el desarrollo de software no funcionaba correctamente o presentaba falencias se concluía y se iniciaba nuevamente.

En esta etapa aparece la conocida crisis del software; esta surge conforme se iban desarrollando gran cantidad de productos en los que se observa costos excesivos, poca fiabilidad, usuarios descontentos y como si esto fuera poco los plazos de entrega por lo general no se cumplían; esto era originado por la falta de formalismos en la concepción del proyecto. Lo anteriormente señalado, sumado a la falta de metodología para el desarrollo, la carencia de herramientas de soporte y la deficiente administración de los proyectos provocó múltiples pérdidas en la industria del software.

Varios aspectos incidían en el desarrollo del software, uno de ello es que es un proceso relativamente nuevo, del cual no existía personal lo suficientemente capacitado debido a una pobre definición de los procesos de gestión de los proyectos. En otro aspecto debido a que el software es el conjunto de programas o instrucciones de una computadora, y por lo tanto, no es un elemento de carácter físico; es improbable que resulte exitoso el primer intento de elaboración, ya que el personal encargado de su realización no posee conocimientos claros de los requerimientos solicitados por parte del cliente, lo que a su vez, vuelve en muy complicado hacer un diseño a detalle de las especificaciones.

2.2.3 Tercera fase (1970-1985): Proceso de desarrollo de software y modelos tradicionales del ciclo de vida

Durante La década de 1970 y 1980 se identifica muchos de los problemas de desarrollo de software. Muchos proyectos de software sobrepasaron el presupuesto y el tiempo estimados. Algunos proyectos causaron daños a la propiedad. Esta etapa está marcada por proponer soluciones que deban llevarse a cabo para resolver los problemas presentados en la fase anterior. Para dar soluciones a los problemas que enfrentaba el software se definió en termino Ingeniería del software en la

conferencia de la OTAN en 1968, en la década del setenta se empezó a dar más importancia a los datos; y para solucionar proyectos complejos se inició con el análisis por partes; en esta etapa se incorporan conceptos y prácticas de planeación y administración (Sommerville, 2006).

El termino *ciclo de vida del Software* apareció en esta etapa como consenso para la construcción centralizada del software y además daría las pautas para lograr establecer de manera general los estados por los que pasa el producto de software desde que naces a partir de una necesidad hasta que se deja de utilizarlo.

El modelo de ciclo de vida es el marco de trabajo que ayuda a garantizar que el sistema pueda cumplir con la funcionalidad requerida durante toda su vida útil. Por lo tanto, para definir los requisitos del sistema y desarrollar soluciones de sistema durante las etapas de concepto y desarrollo, se necesitan expertos de otras etapas (por ejemplo, producción, utilización, soporte, retiro) para realizar análisis de compensación y para ayudar a tomar decisiones de diseño y llegar a un solución equilibrada Esto ayuda a garantizar que un sistema tenga los atributos necesarios diseñados lo antes posible. Además, es esencial tener los sistemas habilitadores necesarios disponibles para realizar las funciones de etapa requeridas (IEEE Computer Society Sponsored by the Software & Systems Engineering Standards Committee, 2011). También se detalla como un proceso que comprende la adquisición, suministro, desarrollo, explotación y mantenimiento del software (Everett & McLeod, 2007).

En conclusión, el ciclo de vida de un proyecto especifica el enfoque general del desarrollo, indicando los procesos, actividades y tareas que se van a realizar y en qué orden, y los productos que se van a generar, los que se van a entregar al cliente y en qué orden se van a entregar. Para formalizar la estructura del ciclo de vida del software se establece el **proceso de desarrollo del software.** Con este antecedente es que la ingeniería del software empieza a establecerse como área de estudio e investigación; aparecen una serie de modelos que señalan diferentes fases

y estados que debe seguir para su desarrollo un producto de software; desde su planteamiento inicial, su desarrollo, puesta en marcha y mantenimiento hasta la finalización de su utilización. A esta serie de modelos se los denomina **modelos de ciclo de vida del software;** estos modelos procuran contener todo el proceso del ciclo de vida, estableciendo en cada fase una serie de normativas y medidas que describan como desarrollar un producto de software desde el inicio hasta final alineándolo siempre a que se ajusten a las necesidades y requerimientos de los clientes.

La contribución de dichos modelos al desarrollo del software se centra en proveer una guía para los ingenieros de software con el fin de dirigir y ordenar las diversas actividades involucradas dentro de la ejecución del proyecto, adicionalmente subministran un marco referencial para la administración del proyecto; esto permitirá estimar y administrar los recursos; establecer puntos de control y monitoreo de los avances.

Los modelos más importantes de inician a partir de la publicación del **modelo en cascada** establecido por Winston Royce que se basa en el análisis, diseño, pruebas y mantenimiento. Se define como una secuencia de fases que en la etapa final de cada una de ellas se reúne la documentación para garantizar que cumple con las especificaciones y requisitos antes de pasar a la siguiente fase (Sommerville, 2006).

En Ingeniería de software el desarrollo en cascada, también llamado modelo en cascada, denominado así por la posición de las fases en el desarrollo de ésta, que parecen caer en cascada hacia las siguientes fases, de tal manera que se inicia solamente despues de la finalización de la etapa anterior (Pressman R. , 2010).

El modelo de cascada tiene sus orígenes en la década de 1970, y se define como una secuencia de actividades bien planificadas y estructuradas. El proceso distingue claramente las fases de especificación de las de desarrollo y éstas, a su vez, de las de testing. Es, seguramente, la metodología más extendida y utilizada. Este modelo se basa fuertemente en que cada detalle de los requisitos se conoce de antemano, previo de

comenzar la fase de codificación o desarrollo, y asume, además, que no existirán cambios significativos en los mismos a lo largo del ciclo de vida del desarrollo (Royce, 1970).

Posteriormente **El Modelo en V** propuesto por Alan Davis (1993), surge con la finalidad de corregir problemas evidenciados modelo en cascada. Es un modelo que representa la secuencia de pasos en el desarrollo del ciclo de vida del software. Describen las actividades y resultados que deben producirse durante el desarrollo del producto. El lado izquierdo de la V representa la descomposición de las necesidades, y la creación de las especificaciones del sistema. El lado derecho de la V representa la integración de las piezas y su verificación. V significa «Verificación y validación». Es muy similar al modelo de la cascada clásico ya que es muy rígido y contiene una gran cantidad de iteraciones.

Este modelo contribuyó significativamente al desarrollo del software, demostró ser más efectivo ya que se realizaban pruebas en cada fase, con esto se detectaban los errores a tiempo y se los corregía; el factor negativo que se observó es el costo y tiempo que se invertían antes de liberar el producto.

Simultáneamente con el método anterior aparece el **modelo iterativo,** este modelo tiene la finalidad de minimizar los riesgos en el desarrollo del software poniendo especial énfasis en la fase de especificación de requerimientos.

En el ciclo de vida iterativo, en cada iteración se reproduce el ciclo de vida en cascada a menor escala. Los objetivos de una Iteración se establecen en función de la evaluación de las Iteraciones precedentes. Desde el principio, al final de cada Iteración se le entrega al cliente una versión completa y mejorada del producto. El cliente es quien luego de cada Iteración evalúa el producto y lo corrige o propone mejoras. estas Iteraciones irán refinando el sistema y se repetirán hasta obtener un producto que satisfaga al cliente (Rauterberg, 1992).

El **Modelo de desarrollo incremental** fue propuesto a inicios de la década de los 80 por Harlan Mills, este modelo combina elementos del modelo en cascada con el modelo iterativo.

El modelo incremental consiste en un desarrollo inicial de la arquitectura completa del sistema, seguido de sucesivos incrementos funcionales. Cada incremento tiene su propio ciclo de vida y se basa en el anterior, sin modificar su funcionamiento ni sus interfaces. Una vez entregado un incremento, no se realizan cambios sobre el mismo, sino únicamente corrección de errores. Dado que la arquitectura completa se desarrolla en la etapa inicial, es necesario, al igual que en el modelo en cascada, conocer los requerimientos completos al comienzo del desarrollo (Pressman R. , 2010).

El desarrollo incremental puede identificar partes que pueden desarrollarse desde la especificación hasta el código ejecutable. También conduce a dividir los requisitos en partes adecuadas durante la especificación, lo que permite el desarrollo independiente de los diferentes incrementos y nos permite un buen paralelismo entre el desarrollo y las pruebas (Toshiya, Tadashi, & Takaji, 2011).

El **modelo de desarrollo en espiral** diseñado por Barry Boehem aparece al terminar esta etapa. Este modelo combina las ventajas del modelo en cascada con el modelo evolutivo; el modelo enfatiza el estudio de los riesgos del proyecto, como por ejemplo las especificaciones incompletas. Se prevén, en este modelo, varios ciclos o vueltas de espiral, cada uno de ellos con cuatro etapas: Definición de objetivos, Evaluación y reducción del riesgo, Desarrollo y validación y Planificación del siguiente ciclo. En este modelo, una actividad comienza solo cuando se entienden los objetivos y riesgos involucrados. El desarrollo se incrementa en cada etapa, generando una solución completa. La metodología en espiral ha sido utilizada con éxito en grandes sistemas, pero su complejidad la hace desaconsejable para el desarrollo de sistemas medianos o pequeños (Boehm, 1988).

El modelo de desarrollo espiral es un generador de modelo de proceso impulsado por el riesgo, que se usa para guiar la ingeniería concurrente con participantes múltiples de sistemas intensivos en software. Tiene dos características distintivas principales. La primera es el enfoque cíclico para el crecimiento incremental del grado de definición de un sistema y su implementación, mientras que disminuye su grado de riesgo. La otra es un conjunto de puntos de referencia de anclaje puntual para asegurar el compromiso del participante con soluciones factibles y mutuamente satisfactorias (Pressman R. , 2010).

Este modelo a diferencia de los modelos anteriores que finalizan cuando se entrega el software se ajusta para su utilización durante todo el ciclo de vida del software.

2.2.4 Cuarta fase (1985-2000): Métodos Rápidos y orígenes del desarrollo ágil.

Esta etapa destaca por la definición y conceptualización de los métodos de la ingeniería del software que día a día van tomando la importancia y connotación que merece esta materia en las organizaciones. En el desarrollo del software siempre han existido entornos cambiantes, y para dar respuesta a esto aparecen los llamados **métodos rápidos** o también denominados modelos rápidos o abreviados, los cuales tienen por objetivo reducir el tiempo del ciclo de desarrollo del software. Este método se basó en las ideas de Barry Boehm y Scott Shultz, luego James Martin desarrolló el método RAD (Desarrollo Rápido de Aplicaciones) durante los años 1980 en IBM y finalmente lo formalizó publicando un libro en 1990 (Reece, 2006).

Este método comprende el desarrollo iterativo, la construcción de prototipos de las aplicaciones y el uso de herramientas y utilidades CASE (siglas en inglés de Computer Aided Software Engineering), y por lo general también suele abarcar la usabilidad, utilidad y la rapidez de ejecución (Reece, 2006).

El modelo de desarrollo rápido de aplicaciones es útil para los proyectos cuyos requisitos son razonablemente estables y bien entendidos. Cuando

el cliente exige software para el sistema existente en un corto período de tiempo, incremento por incremento, entonces podemos adaptar el modelo RAD. Internamente, adapta el modelo de cascada, que es un modelo de proceso secuencial lineal. El usuario puede ver el programa de trabajo al final, por lo que debe indicar claramente todos los requisitos en la etapa inicial del proyecto. El modelo de desarrollo rápido de aplicaciones utiliza herramientas automáticas de generación de código y emplea la reutilización de componentes. RAD emplea a más equipos de RAD para desarrollar y cumplir con los horarios (Kumar & Prashanth, 2014).

En 1994 nace el **Método de desarrollo de sistemas dinámicos** también conocido como “DSDM” por sus siglas en inglés “Dynamic Systems Development Method”, desarrollada como un proceso de entrenamiento de negocios en Inglaterra, se estableció para crear una metodología RAD unificada, la cual definiría el marco para desarrollar un proceso de producción de software. El aporte como metodología consiste en entregar sistemas de software en tiempos y presupuestos reales ajustándose a las variaciones y modificaciones de los requisitos durante el proceso de desarrollo del software, enfatiza en la colaboración y compromiso por parte del usuario como aspecto principal para llevar a cabo un proyecto eficiente, de tal forma que la gestión sería más razonable y efectiva (Coleman, 1998).

El método reconoce que los proyectos son limitados por tiempos y recursos, por lo que se dirige al manejo de problemas más frecuentes que ocurren en el desarrollo de los sistemas de información, como es sobrepasar tiempos de entrega de desarrollos, presupuestos, falta de participación del usuario en el proyecto y el apoyo de comisiones por la gerencia (Coleman, 1998).

En 1995 Schwaber y sutherland presentaron una serie de artículos describiendo a **SCRUM** en la cual se formalizó el proceso para la industria de desarrollo de software. Durante los años siguientes Schwaber y Sutherland colaboraron para consolidar los artículos indicados, con sus experiencias y el conjunto de mejores prácticas de la industria que conformarían a lo que actualmente se lo conoce como SCRUM, Para 2001,

Schwaber y Mike Beedle describirían la metodología en el libro Agile Softwre Development with Scrum. Este método definiría un marco de referencia para la gestión de proyectos que se ha usado satisfactoriamente en los 10 años recientes.

Scrum es una metodología ágil y flexible para gestionar el desarrollo de software, apoyándose en una serie de buenas prácticas para trabajar cooperativamente en equipo, y lograr los mejores resultados posibles de un proyecto que su principal objetivo es garantizar el retorno de la inversión (Johnston, 2006).

Scrum es una metodología para trabajar articuladamente y en equipo aplicando una serie de buenas prácticas con la única finalidad de lograr excelentes resultados en los proyectos que se aplique esta metodología.

Scrum adopta plenamente los principios de los métodos ágiles de desarrollo y los incorpora a la gestión de proyectos. Primero y ante todo, abarca la filosofía de que todos los requisitos están inicialmente sin perfeccionar y son poco claros. Teniendo en cuenta que un conjunto de requisitos de productos claros y a largo plazo no se puede obtener desde el enfoque tradicional de recolección de datos, scrum se centra en la mejora de la capacidad del equipo de desarrollo para observar y adaptarse a las nuevas exigencias (Blokehead, 2016).

En 1996, aparece la **Programación Extrema** (“XP” por sus siglas en inglés de eXtreme Programming) es una metodología de desarrollo de Software formulada por Kent Beck, autor del libro inicial sobre el tema en 1999, la programación extrema se diferencia de las metodologías tradicionales principalmente porque se basa en la adaptabilidad de los cambios de los requisitos más que en la previsibilidad de los mismos, su éxito se atribuye porque enfatiza la satisfacción del cliente brindándole el software que necesita en lugar de entregar todo el software que desea en una fecha lejana.

La programación extrema se centra en fomentar las relaciones interpersonales como aspecto principal en el desarrollo del software, fomenta el trabajo en equipo, se preocupa por el enseñanza de los desarrolladores, y apoya la generación de ambientes de trabajo adecuados; se basa en la retroalimentación permanente entre el cliente y el equipo de desarrollo, la comunicación es constante entre los participantes, las soluciones implementadas son simples, existe disponibilidad para enfrentar cambios durante la ejecución. XP es substancialmente apropiado para proyectos con requerimientos poco claros y variables donde existe alto riesgo.

Por su concepción XP puede ser visto como una forma un tanto caótica de trabajo donde se pone énfasis en resolver el problema del presente sin planificar hacia el futuro mediato. Desde esta perspectiva un enfoque de calidad podría verse como una expectativa ambiciosa que no cae dentro de los rasgos de la metodología/práctica (Bertone, Pasini, & Ramon, 2005).

El **Proceso Racional Unificado** o **RUP** (por sus siglas en inglés de Rational Unified Process) es un proceso formal que provee un acercamiento disciplinado para asignar funciones dentro de una organización de desarrollo. La finalidad es garantizar la elaboración de software de gran calidad que satisfaga las exigencias de los usuarios finales sin afectar el cronograma y presupuesto. Fue desarrollado por Rational Software, y está integrado con toda la suite Rational de herramientas. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. Es guiado por casos de uso y centrado en la arquitectura, y utiliza UML como lenguaje de notación.

El Proceso Unificado de Rational es un proceso de ingeniería del software. Facilita la asignación de tareas y responsabilidades en una organización de desarrollo de manera rigurosa y metódica. Su finalidad es garantizar la producción del software de excelente calidad de acuerdo a las especificaciones de los usuarios, con costos y calendarios previsibles (Kruchten, 2001).

2.2.5 Quinta fase (2000 en adelante): Metodologías del proceso de la ingeniería del software.

Ante la constante demanda de desarrollo de software para pequeñas organizaciones, la necesidad de soluciones de bajo costo llevó al crecimiento de métodos más simples y rápidos que ayuden a la gestión de proyectos de software orientando que el producto sea funcional; el uso de prototipos, métodos, procedimientos y normas evoluciona al uso del concepto de metodología del proceso de desarrollo de software, destacando el uso de metodologías ágiles o ligeras. Una de ellas fue XP que simplificó muchas áreas de la ingeniería del software como la forma de levantamiento y recopilación de requerimientos, y las pruebas de confiabilidad. Los productos de software muy grandes aún utilizan metodologías rígidas y muy documentadas; sin embargo, los sistemas más pequeños tienen un enfoque alternativo simple y rápido de gestionar todo el ciclo de desarrollo del software.

Las metodologías ágiles tuvieron tal efecto y crecimiento que en el año 2001 miembros emblemáticos de la comunidad de desarrollo de software se reunieron y adoptaron el nombre de “métodos ágiles”.

El desarrollo ágil de proyectos de software cambió el enfoque en la producción y construcción del software, que evoluciona constantemente con entornos cambiantes y mercados competitivos. La percepción actual es que los procesos rígidos dirigidos por demasiada documentación como por ejemplo CMM e ISO 9000 están perdiendo vigencia. En este contexto es que las metodologías de desarrollo ágil adquieren gran preponderancia ya que contemplan un conjunto de métodos que se basa en el desarrollo iterativo e incremental, reaccionando positivamente ante los entornos cambiantes mediante la colaboración de un equipo de trabajo organizado y multidisciplinario (agilemanifesto.org, 2018).

Los excesivos controles de los procesos en el desarrollo de software en muchos casos no han resultado exitosos debido al cambiante entorno en el que se desarrollan, por lo cual las metodologías ágiles pretenden solucionar

este inconveniente, construyendo soluciones de acuerdo a la necesidad del usuario y garantizando su calidad.

En conclusión, las metodologías ágiles fueron orientadas para equipos de desarrollo pequeños con plazos cortos, requisitos cambiantes y tecnologías en permanente evolución. Las metodologías ágiles en esencia, procuran dar mayor valor a la persona, a la colaboración con el cliente y al desarrollo progresivo del software con iteraciones cortas. Este paradigma está demostrando su eficiencia en proyectos con requerimientos muy volátiles y cuando se exige disminuir rápidamente los tiempos de desarrollo, pero manteniendo la calidad (Sommerville, 2006).

Se puede evidenciar claramente la evolución permanente que ha tenido hasta conformar la concepción de las metodologías del proceso de ingeniería del software, la cual describe el conjunto de herramientas, técnicas, procedimientos y soporte documental para el diseño de sistemas de software.

También se puede definir que “Una metodología de ingeniería de software es un enfoque estructurado para el desarrollo de software cuyo propósito es facilitar la producción de software de alta calidad de forma costeable” (Sommerville, 2006).

2.3 Conceptualización de términos en el proceso de desarrollo de software.

2.3.1 Ingeniería del software

La IEEE, define a la Ingeniería de Software como la aplicación de un enfoque sistemático, cuidadoso y medible al desarrollo y mantenimiento de sistemas de software.

La ingeniería de software es la ciencia del desarrollo de software. El proceso de desarrollo de software se caracteriza por una serie de fases distintas. Primero, las especificaciones detalladas para el software propuesto se crean con la ayuda de quienes utilizarán el software. Luego, se realiza un análisis para determinar los recursos necesarios para el proyecto y los pasos involucrados en su desarrollo. Una vez completado el

análisis, se crea un diseño formal. El diseño generalmente incluye descripciones de los datos necesarios para el sistema y los procesos que deben realizarse en los datos, y algunos cuadros para ayudar a comprender la estructura del sistema propuesto (Whitson, 2013).

En conclusión, la Ingeniería del software es el enfoque sistemático para el desarrollo, operación, mantenimiento del software; definiendo como software los programas, procedimientos, reglas y documentación, así como los datos de operación de un sistema de cómputo. La Ingeniería del Software es una disciplina o área de la Informática que ofrece métodos y técnicas para desarrollar y mantener software de calidad.

2.3.2 Proyectos de software

Un proyecto de software puede definirse como el conjunto de actividades que se desarrollan para alcanzar determinados objetivos. Estas actividades se encuentran interrelacionadas y se desarrollan de manera coordinada. La administración del proyecto involucra planificación, monitoreo y control del personal, procesos y acciones que ocurren conforme el software evoluciona desde un concepto preliminar hasta su despliegue operativo completo (Pressman R. , 2010).

2.3.3 Gestión de proyectos de software

Las instituciones en la actualidad para sobrevivir en el mercado del siglo XXI deben implementar el software como un elemento que permita generar estrategias de diferenciación en sus procesos de negocio.

La gestión exitosa de proyectos de software se basa en dos elementos clave: la planificación y estimación precisas del ciclo de vida del proyecto, y la supervisión y el control del proyecto para llevarlo a cabo con éxito, en términos de tiempo, costo y calidad (Fitzhenry & Gardiner, 1995).

2.3.4 El proceso de desarrollo de software

2.3.4.1 Ciclo de vida del software

Un modelo típico de ciclo de vida del software consta de varias etapas. Comienza con una idea o un concepto para un producto o servicio de software, continúa a través de la ingeniería de sistemas e ingeniería de software, y avanza a la operación, mantenimiento y soporte, y termina en la jubilación (IEEE Computer Society Sponsored by the Software & Systems Engineering Standards Committee, 2011).

Un modelo de ciclo de vida del sistema suele estar segmentado por etapas para facilitar la planificación, el aprovisionamiento, la operación y el soporte del sistema de interés. Estos segmentos proporcionan una progresión ordenada de un sistema a través de puertas de toma de decisiones establecidas para reducir el riesgo y asegurar un progreso satisfactorio. Como se indicó anteriormente, la razón más importante para usar un modelo de ciclo de vida es tomar una decisión según criterios específicos antes de que un sistema pueda avanzar a la siguiente etapa. Un aspecto secundario es que el uso de un modelo de ciclo de vida puede ayudar a una organización a asegurarse de que piensa en su trabajo y sus procesos dentro de un marco más amplio, que puede tener connotaciones comerciales útiles (IEEE Computer Society Sponsored by the Software & Systems Engineering Standards Committee, 2011).

Un modelo de ciclo de vida del software es una visión de las actividades que ocurren durante el desarrollo de software, determina el orden de las etapas involucradas y los criterios de transición asociadas entre estas etapas, describe las fases principales de desarrollo de software, define las fases primarias a ser ejecutadas durante esas fases; ayuda a administrar el progreso del desarrollo, y provee un espacio de trabajo para la definición de un detallado proceso de desarrollo de software. Así, los modelos por una parte suministran una guía para los ingenieros de software con el fin de ordenar las diversas actividades técnicas en el proyecto, por otra parte, suministran un marco para la administración del desarrollo y el

mantenimiento, en el sentido en que permiten estimar recursos, definir puntos de control intermedios, monitorear el avance (ISO, 2008).

Todos los modelos cubren el ciclo de vida del software y tienen implícita o explícitamente las siguientes fases:

- Análisis
- Diseño
- Implementación
- Pruebas
- Mantenimiento

2.3.4.2 Modelos de proceso del software

A partir del concepto de ciclo de vida del software han aparecido varios modelos de proceso de software, a continuación, vamos a citar los más importantes.

a Modelo en cascada

El primer ciclo de vida del software, “Cascada”, fue definido por Winston Royce a fines del 70. Se define como una secuencia de fases donde al final de cada una de ellas se reúne la documentación para garantizar que cumple las especificaciones y los requisitos antes de pasar a la fase siguiente (Pressman R. , 2010).

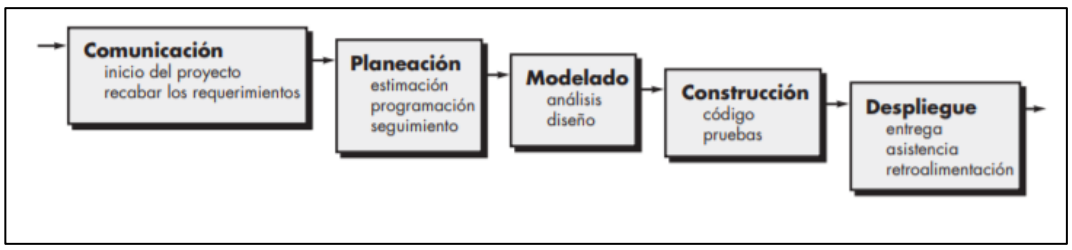

Figura 1: Modelo Cascada
Fuente: (Pressman R. , 2010)

b Modelo en V

El modelo de ciclo de vida en V proviene del principio que establece que los procedimientos utilizados para probar si la aplicación cumple las especificaciones ya deben haberse creado en la fase de diseño (Álvarez, y otros, 2014).

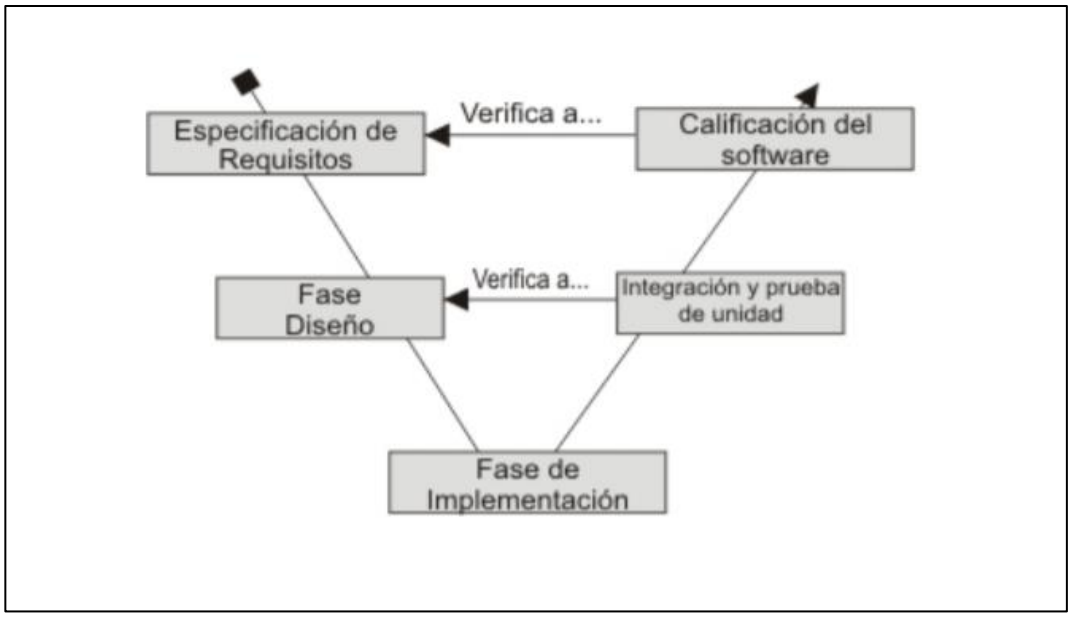

Figura 2: Modelo en V
Fuente: (Álvarez, y otros, 2014)

c Modelo en espiral

El modelo en espiral del proceso del software fue originalmente propuesto por Boehm en 1988. Más que representar el proceso del software como una secuencia de actividades con retrospectiva de una actividad a otra, se representa como una espiral. Cada ciclo en la espiral representa una fase en el proceso del software. Así el ciclo más interno podría referirse a la viabilidad del sistema, el siguiente ciclo a la definición de requerimientos, el siguiente al diseño del sistema, y así sucesivamente (Sommerville, 2006).

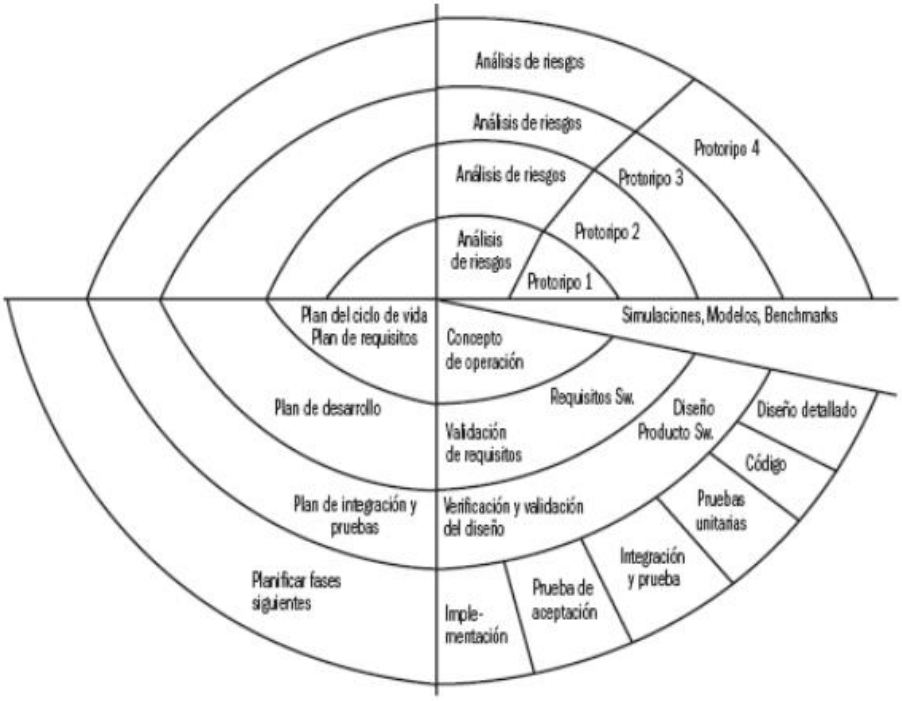

Figura 3: Modelo en Espiral

Fuente: *(Sommerville, 2006)*

d Modelo Incremental

La entrega incremental es un enfoque que combina las ventajas de los modelos en cascada y evolutivo. En un proceso de desarrollo incremental, los clientes identifican, a grandes rasgos, los servicios que proporcionará el sistema. Identifican que servicios son más importantes y cuales menos. Entonces, se definen varios incrementos en donde cada uno proporciona un subconjunto de funcionalidad del sistema. La asignación de servicios a los incrementos depende de la prioridad del servicio con los servicios de prioridad más alta entregados primero (Sommerville, 2006).

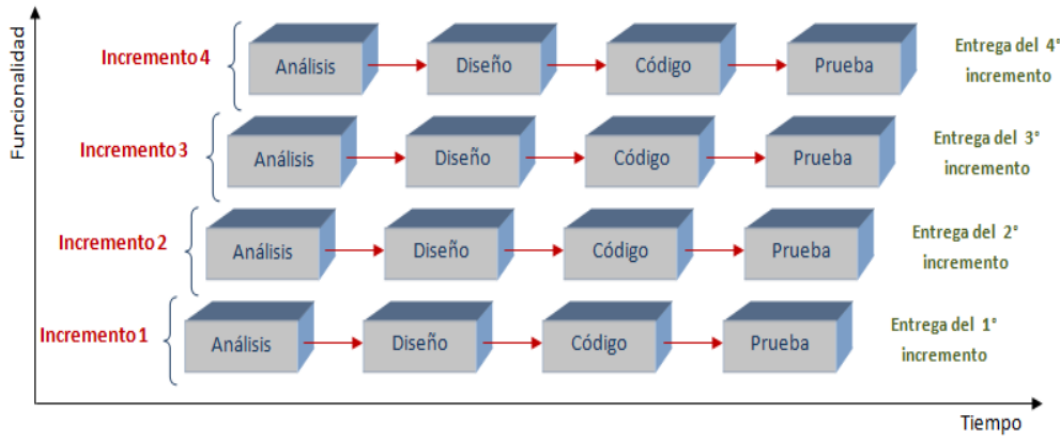

Figura 4: Modelo incremental
Fuente: (Sommerville, 2006)

e Modelo de desarrollo rápido de aplicaciones (DRA)

El desarrollo rápido de aplicaciones o RAD por sus siglas en ingles Rapid Application Development fue propuesto por James Martin en 1980. El método comprende el desarrollo iterativo, la construcción de prototipos y el uso de utilidades CASE. Tradicionalmente, el desarrollo rápido de aplicaciones toma en cuenta aspectos como la usabilidad, utilidad y la rapidez de ejecución (Álvarez, y otros, 2014).

El modelo RAD es lineal y secuencial que da prioridad a un ciclo de desarrollo considerablemente corto.

Este modelo presenta las siguientes fases:

- Modelo de Gestión: que identifica a donde va, quien lo genera, que información se genera y se conduce.
- Modelo de Datos: identifica objetos y relaciones.
- Modelo de Procesos: describe el proceso de negocio.
- Generación de Aplicaciones: reusabilidad de componentes.
- Pruebas y Entregas: prueba de componentes nuevos e interfaces

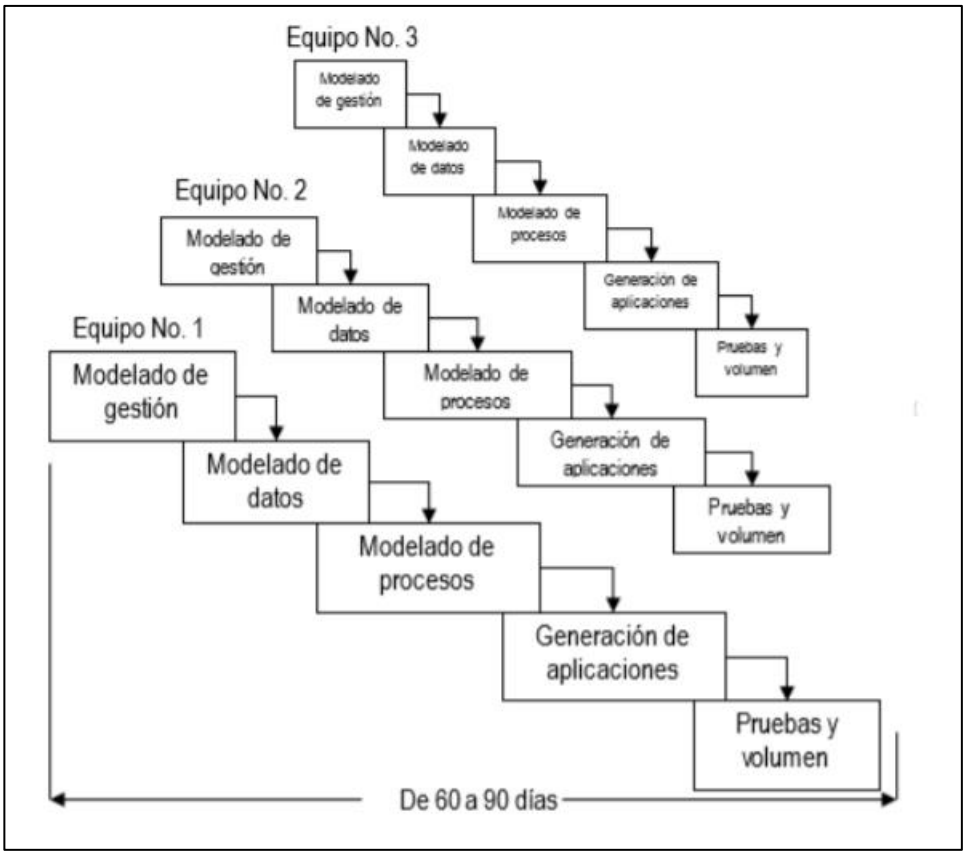

Figura 5: Modelo RAD

Fuente: (Álvarez, y otros, 2014)

2.4 Conceptualización y definición de las metodologías ágiles

Las metodologías ágiles gestionan el desarrollo de software dando prioridad a la colaboración estrecha entre equipos multidisciplinarios. Se caracterizan por enfatizar la comunicación frente a la documentación, por el desarrollo evolutivo y por su flexibilidad.

Las metodologías ágiles se han considerado como las metodologías de programación elegidas para el mundo volátil y de alta velocidad de Internet y el desarrollo de software web. También han sido criticados como otro disfraz por piratería indisciplinada. La realidad depende de la fidelidad a la filosofía ágil con la que se implementan estas metodologías y de la adecuación de la implementación para el entorno de la aplicación (Paulk, 2002).

En el mundo ágil, la gestión de proyectos es menos rígida de lo que estás acostumbrado. Tiene un equipo, como antes, pero en lugar de trabajar durante aproximadamente un año para una gran entrega de software, divide la funcionalidad en partes posibles que puede entregar al cliente de manera continua (el proyecto general aún puede durar un año), pero para cuando llegue, ya habrá entregado varios incrementos del software). Estas pequeñas partes de la funcionalidad (requisitos) se llaman historias de usuario y deben ser definidas por el cliente (Scanlon Thomas, 2011).

La conclusión es que las metodologías ágiles promueven muchas buenas prácticas de ingeniería, aunque algunas prácticas pueden tener una implementación extrema que es controversial y contraproducente fuera de un dominio estrecho (Paulk, 2002).

2.5 Caracterización de las metodologías ágiles

En la actualidad, la utilización de las metodologías ágiles en la industria del software está en auge ya que estas presentan muchas ventajas ante las metodologías pesadas o tradicionales, estas presentan limitaciones de acuerdo al volumen del proyecto y la cantidad de programadores que pueden inmiscuirse. A pesar de esto resultan muy cómodas para el desarrollo de sistemas informáticos en empresas que están despuntando o para el desarrollo de software modular, siempre garantizando la calidad y manteniendo de forma permanente la documentación actualizada.

Después de casi una década de esfuerzos aislados, en febrero del 2001 en Utah-USA, se realizó una reunión con empresarios del mundo del software y despues de una amplia discusión sobre las metodologías agiles, fundamentos y características que deben normar el desarrollo del software de gran calidad, respetando el tiempo acordado y prestando flexibilidad a posibles cambios, se aceptó el término ágil para hacer referencia a nuevos enfoques metodológicos en el desarrollo de software (agilemanifesto.org, 2018).

En este conversatorio se inició “The Agile Alliance”, una institución sin interés de lucro con la finalidad de impulsar y fomentar la discusión de conceptos concernientes con el desarrollo ágil de software y asistir a las organizaciones para que acojan dichas concepciones. Como punto de partida o base fundamental de las metodologías ágiles se redactó y proclamó el manifiesto ágil (agilemanifesto.org, 2018).

2.5.1 Manifiesto ágil

El manifiesto ágil hace énfasis en cuatro valores y doce principios; los valores son consideraciones generales en las cuales se debe enmarcar el desarrollo del software. Los principios hacen referencia a las características que se diferencian entre un proceso ágil y un tradicional, y constituyen las ideas centrales del desarrollo ágil. (agilemanifesto.org, 2018).

2.5.1.1 Valores

Se valora a los individuos y las interacciones sobre los procesos y las herramientas.

La condición para que una metodología sea ágil es entregar software funcionando y útil en pocas semanas. Esto acaba con la incertidumbre, desconfianza, insatisfacción producidas en el cliente debido a las largas esperas para ver resultados concretos. El cliente participa implícitamente validando y probando constantemente el producto durante su ciclo de desarrollo.

Se valora las aplicaciones que funcionan sobre la documentación exhaustiva.

Los profesionales del software reconocen la importancia de producir documentación, de la misma manera entienden que realizar documentación de soporte requiere de tiempo y recursos, en esta línea las metodologías agiles garantizan la trascendencia de la documentación como soporte funcional en un proyecto de software, no obstante se prioriza en documentar la información sumamente precisa, los documentos deben

sintetizar en lo principal y dar preferencia al software funcionando sobre la documentación extensa.

Se valora la colaboración del cliente sobre las negociaciones contractuales.

Habitualmente el cliente es quien solicita e indica qué debe hacer al equipo de desarrollo de software y aspira que el producto sea de acuerdo a las expectativas iniciales y respete los tiempos acordados. Con frecuencia las dos partes asumen posiciones distantes, con ingredientes de rivalidad y prevención al punto de tener que dedicar tiempo valioso a la tarea establecer términos rigurosos de contrato.

La filosofía de las metodologías ágiles busca el beneficio común, del equipo de desarrollo y del cliente. La colaboración del cliente debe ser permanente, desde el inicio hasta la finalización del plan, el trabajo en conjunto con el equipo de desarrollo debe ser continuo.

Se valora la respuesta al cambio sobre el seguimiento de un plan.

En vista a los entornos variables de la tecnología y la mecánica de las organizaciones, un proyecto de software afronta reiteradamente perturbaciones en el transcurso de su ejecución; desde arreglos ligeros en la caracterización del software hasta cambios en las normas, reglamentos, nuevas tendencias tecnológicas, etc. En este sentido las metodologías pesadas, con frecuencia caen en la concepción de mantener el monitoreo y control íntegro de todos los aspectos, desde el inicio hasta el final del proyecto sin la posibilidad de atender a regateos o interrupciones.

Por el contrario, en las metodologías ágiles la programación no debe ser rígida, ya que existen un sinfín de variantes y deben ser adaptables a posibles alteraciones que pueden aparecer (agilemanifesto.org, 2018).

2.5.1.2 Principios

El cliente es de nuestra máxima importancia, por lo tanto se le entrega el producto de acuerdo a sus necesidades de manera anticipada y con valía agregada (agilemanifesto.org, 2018).

Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente (agilemanifesto.org, 2018).

Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible (agilemanifesto.org, 2018).

Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto (agilemanifesto.org, 2018).

Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo (agilemanifesto.org, 2018).

El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es el diálogo cara a cara (agilemanifesto.org, 2018).

El software funcionando es la medida principal de progreso (agilemanifesto.org, 2018).

Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida (agilemanifesto.org, 2018).

La atención permanente a la prestancia técnica y al buen diseño mejora la Agilidad (agilemanifesto.org, 2018).

La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial (agilemanifesto.org, 2018).

Los equipos de desarrollo auto-organizados garantizan siempre mejores arquitecturas, requisitos (agilemanifesto.org, 2018).

A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia (agilemanifesto.org, 2018).

2.5.2 Análisis exploratorio de las metodologías ágiles

2.5.2.1 Scrum

Scrum es una de las metodologías ágiles más populares. Es una metodología de adaptación, iterativa, rápida, flexible y eficaz, diseñada para ofrecer un valor significativo de forma rápida en todo el proyecto. Scrum garantiza transparencia en la comunicación y crea un ambiente de responsabilidad colectiva y de progreso continuo. (SCRUM STUDY targeting success, 2016).

Scrum, es un marco de trabajo para el desarrollo ágil de proyectos, en principio surgido en la industria del software, pero de suficiente sencillez y flexibilidad como para ser aplicado en contextos muy diversos. Dentro de este marco, el proceso de desarrollo de un proyecto se concibe como una sucesión de ciclos cortos de trabajo denominados sprints, obteniendo de cada uno de ellos un producto funcional que va completándose en forma iterativa.

Algunas de las ventajas principales de la utilización de Scrum en un proyecto son:

Adaptabilidad: El control del proceso empírico y el desarrollo iterativo propician a que los proyectos sean parametrizables y abiertos a la agregación de cambios.

Transparencia: Todos los radiadores de información tales como un Tablero de Scrum (del inglés Scrumboard) y una Gráfica del trabajo pendiente del sprint (del inglés Sprint Burndown Chart) se comparten, lo que conduce a un ambiente de trabajo abierto.

Retroalimentación continua: La retroalimentación continua se proporciona a través de los procesos de realizar la reunión diaria de pie, demostración y validación del sprint.

Mejora continua: Los entregables se mejoran progresivamente sprint por sprint a través del proceso de mantenimiento de la lista priorizada de pendientes del producto.

Entrega anticipada de alto valor: El proceso de Creación de la lista priorizada de pendientes del producto asegura que los requisitos de mayor valor del cliente sean los primeros en cumplirse.

Resolución de problemas de forma más rápida: La colaboración y colocación de equipos interfuncionales conducen a la resolución de problemas con mayor rapidez.

Alta velocidad: Un marco de colaboración que le permite a los equipos interfuncionales altamente cualificados alcanzar su potencial y alta velocidad (SCRUM STUDY targeting success, 2016).

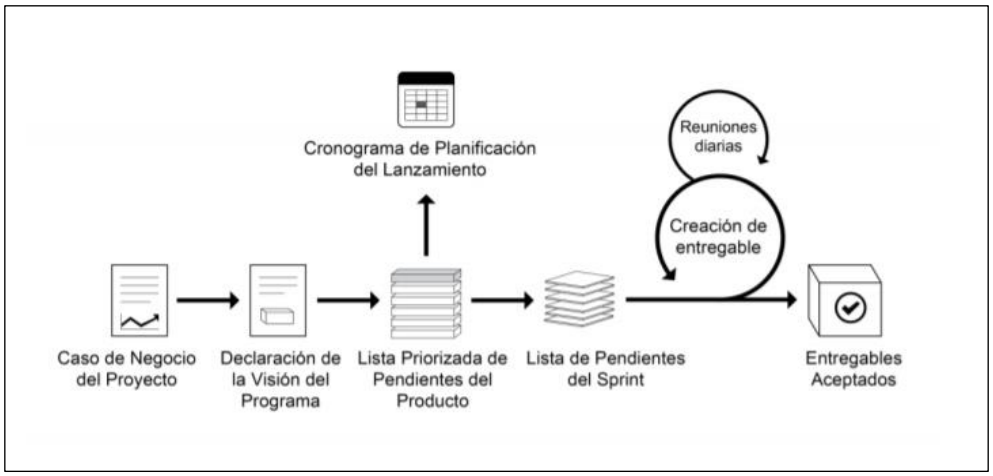

Figura 6: Arquitectura Scrum
Fuente: *(SCRUM STUDY targeting success, 2016)*

Scrum se compone de tres roles, tres reuniones y tres artefactos:

Roles:

- Propietario del Producto (product owner),
- Scrum Master o facilitador,
- Equipo (scrum team).

Reuniones:

- Planificación del Sprint (sprint planning),
- Seguimiento diario del Sprint (daily scrum),
- Revisión del Sprint y Retrospectiva (sprint review/retrospective).

Artefactos:

- Pila del Producto (product backlog),
- Pila del Sprint (sprint backlog),
- Gráfico Burndown (burndown chart).

2.5.2.2 XP (eXtreme Programming)

En el contexto de XP, una metáfora es “una historia que cada quien —clientes, programadores y gerentes— narra, acerca de cómo funciona el sistema” (Beck, 2004).

La programación extrema es un tipo de metodología de desarrollo de software ágil. Esto es capaz de superar las limitaciones del enfoque de desarrollo de software tradicional en términos de implementación, entrega más rápida (pequeñas versiones), fácil adaptación al cambio incluso al final del ciclo de desarrollo y mayor satisfacción del cliente. Algunas de las prácticas de XP son que XP se enfoca en pruebas tempranas y pruebas automatizadas. Las pruebas en XP comienzan muy temprano en las fases de desarrollo. XP mantiene ciclos de desarrollo cortos con diseño y planificación incrementales. Mantiene la participación continua del cliente en cada fase del desarrollo. También implica la integración continua y el despliegue diario (Sharma & Hastecer, 2016).

Es una metodología ligera para cualquier tamaño de software. XP es una metodología de software que se enfoca en la programación de pares, realizando una extensa revisión del código, simplicidad y claridad en el código, estructura de administración plana, realiza un seguimiento frecuente de la comunicación con el cliente y el programador. Pretende mejorar la calidad del software y responder a los cambios en los requisitos del cliente incluso al final del ciclo de desarrollo (Sharma & Hastecer, 2016).

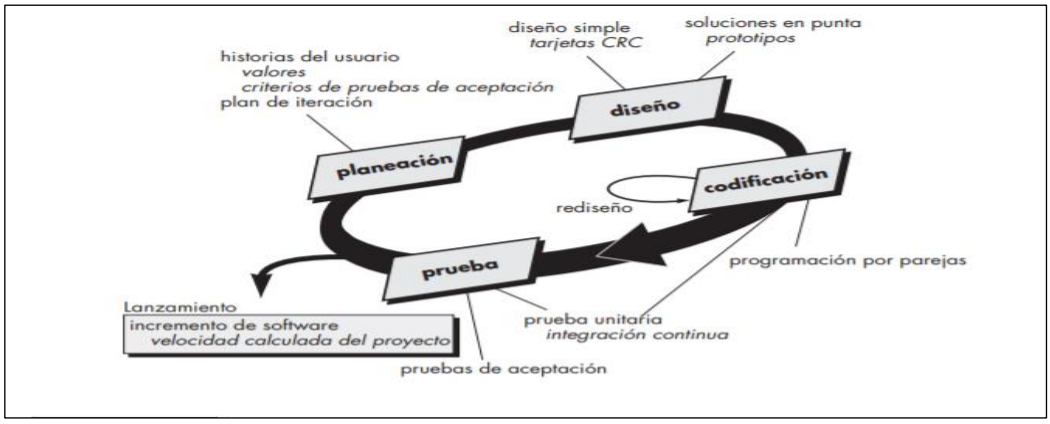

Figura 7: Proceso de la programación extrema
Fuente: (Pressman R. , 2010)

Roles de XP

Rol	Descripción
Programador	El programador escribe las pruebas unitarias y produce el código del sistema.
Cliente	Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio
Responsable de pruebas (Tester)	Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
Responsable de seguimiento (Tracker).	Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones.
Entrenador (Coach).	Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente
Consultor	Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas
Gestor (Big boss)	Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas.

Tabla 1. Role XP
Fuente: (Beck, 2004)

Proceso XP

El ciclo de desarrollo se fundanta en los siguientes pasos:

1. El cliente establece el alcance del valor del producto a desarrollar.

2. El programador realiza una proyección de la demanda de trabajo requerida para la ejecución.

3. El cliente elige qué elaborar, en conveniencia con el orden de importancia y disposición del tiempo.

4. El programador cimienta, edifica y da valor al negocio.

5. Se repite el ciclo desde el paso 1, en las iteraciones el cliente y el programador se instruyen. El trabajo planificado al inicio debe respetarse, para de esta manera garantizar la confiabilidad y entregar en los tiempos previstos. El programador debe garantizar que el producto se revalorice en cada ciclo de repetición (Beck, 2004).

El ciclo de vida de Extreme Program

- Planificación de la Entrega
- Producción
- Exploración
- Mantenimiento
- Iteraciones
- Muerte del Proyecto

2.5.2.3 Kanban

Kanban es "una técnica de gestión de producción basada en un sistema pull (halar) que se fundamentan en la autogestión de los procesos, eliminando la programación centralizada. Se produce y transporta lo que se demanda en los procesos consumidores, manteniendo en rotación sólo aquellas cantidades que garantizan la continuidad del consumo. Cuando se interrumpe el consumo se detiene la producción. Es una herramienta para conseguir la producción Justo a tiempo –JIT–" (Acevedo, Urquiaga, & Gómez, 2001).

El Kanban es un sistema de gestión del trabajo en curso, que sirve principalmente para asegurar una producción continua y sin sobrecargas en el equipo de trabajo. El Kanban es un sistema de gestión donde se produce exactamente aquella cantidad de trabajo que el sistema es capaz de asumir. El Kanban es un sistema de trabajo justo a tiempo, lo que significa que evita sobrantes innecesarios de stock, que en la gestión de proyectos equivale a la inversión innecesaria de tiempo y esfuerzo en lo que no es necesario y evita sobrecargar al equipo (Acevedo, Urquiaga, & Gómez, 2001).

Los principios del método de Kanban son:

- Comience con lo que hace ahora.
- Se acuerda perseguir el cambio incremental y evolutivo.
- Respetar el proceso actual, los roles, las responsabilidades y los cargos.
- Liderazgo en todos los niveles.

Las prácticas que utiliza Kanban son las siguientes:

- Visualizar.
- Limitar el trabajo en curso.
- Dirigir y gestionar el flujo.
- Hacer las Políticas Explícitas del Proceso.
- Utilizar modelos para reconocer oportunidades de mejora.

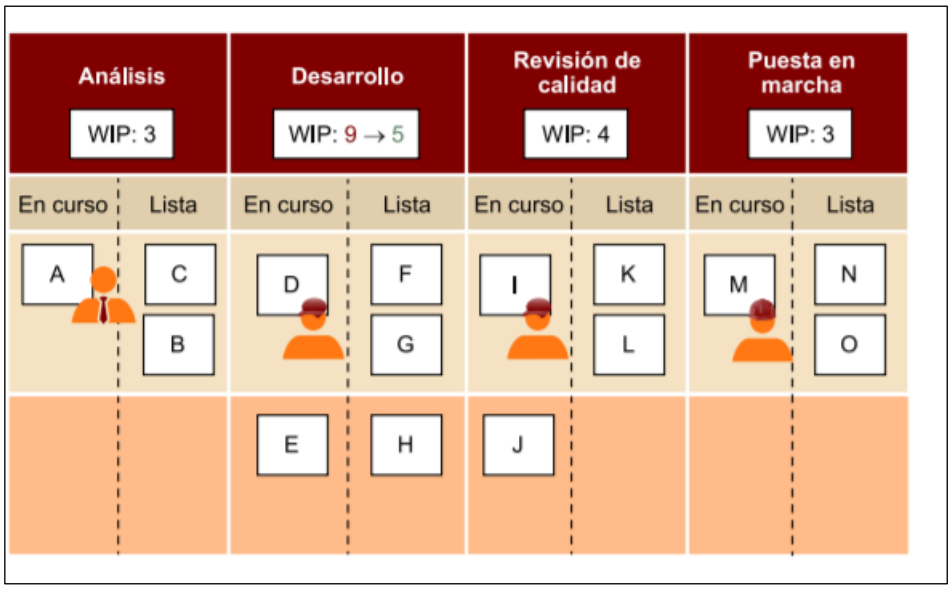

Figura 8: Flujo de trabajo Kanvan
Fuente: (FUOC. Fundación para la Universitat Oberta de Catalunya, 2010)

2.5.2.4 Scrumban

Scrumban es una metodología combinada y derivada de Scrum y Kanban. Algunos equipos sintieron que estas metodologías no calzaban del todo en el trabajo. Scrum es demasiado estricta para muchos entornos que necesitan más agilidad, mientras kanban no se estructura suficientemente. Scrumban intenta proporcionar un punto intermedio, mesclando la estructura del Scrum y la planificación flexible de Kanban para crear una metodología ajustada a entornos en rápida evolución (García Rodríguez, 2015).

De Scrum

- Roles: Cliente, equipo (con los diferentes perfiles que se necesiten).
- Reuniones: reunión diaria.
- Herramientas: pizarra

De Kanban

- Flujo visual
- Hacer lo que sea necesario, cuando sea necesario y solo la cantidad necesaria.
- Limitar la cantidad de trabajo (WIP)
- Optimización del proceso.

Es un modelo de desarrollo especialmente adecuado para proyectos de mantenimiento o proyectos en los que las historias de usuarios varían con frecuencia o en los cuales surjan errores de programación inesperados durante todo el ciclo de desarrollo del producto. Para estos casos, los de la metodología Scrum no son factibles, dado que los errores o imprevistos que surgirán a lo largo de las tareas son difíciles de determinar y, por lo tanto, no es posible estimar el tiempo que conlleva cada historia. Por ello, resulta más beneficioso adoptar flujo de trabajo continuo propio del modelo Kanban.

2.6 Fundamentación filosófica

La presente investigación se enmarca en el paradigma crítico - propositivo, es crítico porque realiza un análisis crítico del problema, y es propositivo porque busca proponer una solución factible al problema.

2.7 Fundamentación legal

Según el Acuerdo 39 de la Contraloría General del Estado y Registro Oficial Suplemento 87 del día 14 de diciembre del 2009 cuya última modificación se realizó el día 6 de febrero del 2014 la cual se encuentra vigente, establecen “las normas de control interno para las entidades, organismos del sector público y de las personas jurídicas de derecho privado que dispongan de recursos públicos”. En el inciso 410 de Tecnologías de la información precisamente en el numeral 07 sobre el “Desarrollo y adquisición de software aplicativo” indica:

La Unidad de Tecnología de Información regulará los procesos de desarrollo y adquisición de software aplicativo con lineamientos, metodologías y procedimientos. Los aspectos a considerar son:

Adopción, mantenimiento y aplicación de políticas públicas y estándares internacionales para: codificación de software, nomenclaturas, interfaz de usuario, interoperabilidad, eficiencia de desempeño de sistemas, escalabilidad, validación contra requerimientos, planes de pruebas unitarias y de integración.

En los procesos de desarrollo, mantenimiento o adquisición de software aplicativo se considerarán: estándares de desarrollo, de documentación y de calidad, el diseño lógico y físico de las aplicaciones, la inclusión apropiada de controles de aplicación diseñados para prevenir, detectar y corregir errores e irregularidades de procesamiento, de modo que éste, sea exacto, completo, oportuno, aprobado y auditable. Se considerarán mecanismos de autorización, integridad de la información, control de acceso, respaldos, diseño e implementación de pistas de auditoría y requerimientos de seguridad. La especificación del diseño considerará las arquitecturas tecnológicas y de información definidas dentro de la organización (CONTRALORIA GENERAL DEL ESTADO, 2011)

La normativa descrita anteriormente fundamenta legalmente la necesidad de implementar metodologías, políticas y controles en el desarrollo del software en cualquier institución pública, en este caso de Yachay Tech.

2.8 Esquema del marco teórico de la investigación

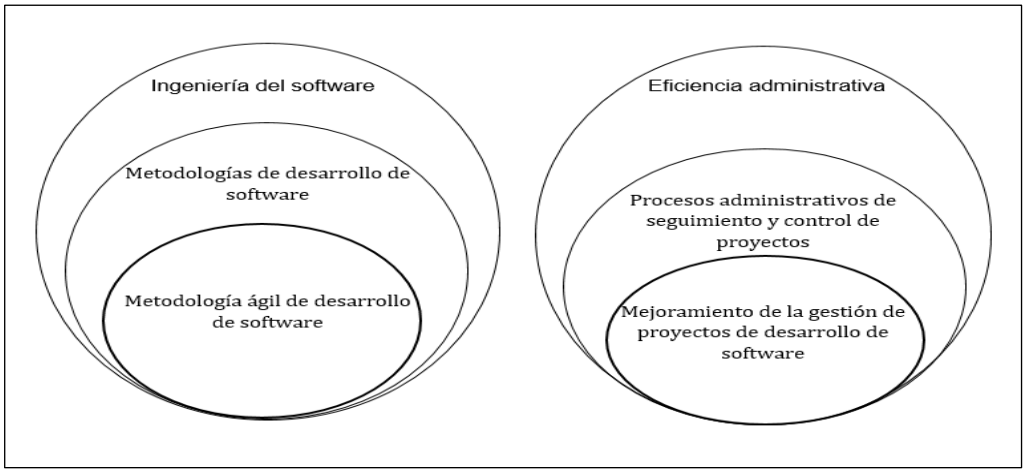

Figura 9. Esquema del marco teórico de la investigación
Fuente*:* El Autor

CAPÍTULO III. METODOLOGÍA DE LA INVESTIGACIÓN

3 Introducción

En este capítulo se realiza la descripción del proceso de la investigación; se definen los distintos componentes del área de estudio, se precisa la metodología a emplear en el caso de estudio. En definitiva, se detalla los pasos a seguir para realizar el estudio.

3.1 Descripción del área de estudio

Esta investigación se desarrollará en la Dirección de Tecnologías de la Información y Comunicación de Universidad Yachay Tech.

3.2 Tipo de Investigación

La investigación tiene un enfoque cualitativo ya que se va a realizar análisis y comparaciones de características que el investigador deberá interpretar de acuerdo a los requerimientos del usuario.

El tipo de investigación será bibliográfica porque utilizará fuentes como libros, documentos, artículos, revistas, etc.; El estudio también será de campo debido a que se buscará información con las áreas involucradas.

3.3 Diseño de la Investigación

3.3.1 Modalidad básica de la investigación

La modalidad básica de la investigación será bibliográfica y de campo.

Investigación Bibliográfica. - Para realizar este estudio es indispensable utilizar todas las referencias bibliográficas relacionadas con el tema a investigarse; de esta revisión y análisis se obtendrán criterios, experiencias y conocimientos que orienten a dar solución al problema.

Investigación de Campo. - Se realiza con la observación directa para identificar de primera mano la actualidad del área a ser investigada; la

utilización de este tipo de investigación permite realizar un diagnóstico acertado de la situación actual, para una subsiguiente solución al problema.

3.3.2 Tipos o niveles de la investigación

Investigación Descriptiva. - La investigación descriptiva se utiliza para detallar y señalar las principales características de los datos recolectados; adicionalmente permite relacionar la variable dependiente e independiente del estudio.

Investigación Exploratoria: Por medio de la investigación exploratoria se facilitará la identificación de oportunidades potenciales para resolver el problema.

3.3.3 Diseño del caso de estudio

Para realizar la presente investigación vamos a utilizar las recomendaciones de Per Runeson (2012). Este tipo de investigación permite el estudio de un objeto o caso, cuyos resultados permanecerán ciertos solo en ese caso específico. Sin embargo, mediante un estudio de caso se podrá obtener una percepción más completa del objeto de estudio, considerándolo como una entidad holística.

El diseño de investigación consiste en relacionar los datos que se van recolectar, la información que se va a analizar con las preguntas iniciales de investigación. Los componentes para el caso de estudio son:

- Preguntas de investigación
- Proposición.
- Unidad de análisis.
- Criterios para interpretación de resultados

Preguntas de investigación

- ¿Qué metodología ágil de desarrollo de software se adaptan a las necesidades de desarrollo de software de la institución?

- ¿En el caso de estudio se aplica todas las fases de la metodología seleccionada?
- ¿Qué factores de éxito se encontraron al implementar la metodología?
- ¿Luego de la implementación de la metodología ágil, mejoró la gestión de los proyectos de software?

Proposición.

La implementación de una metodología de desarrollo ágil, incide en el mejoramiento de la gestión de proyectos de software en la Universidad Yachay Tech.

Unidad de análisis.

Este estudio se llevará a cabo en la Dirección de Tecnologías de la información de la Universidad de Investigación de Tecnología Experimental Yachay.

Criterios para interpretar resultados.

Los resultados pueden ser informados de diferentes maneras. En este caso los resultados permiten una comparación de los factores determinantes que conduzcan a la adopción de una metodología de desarrollo. La comparación puede ser a través de datos cualitativos y cuantitativos.

3.3.4 Conducción del estudio de caso: Preparación para la recolección de datos.

En esta fase se aclara e identifica la información a recolectar, se identifica los objetivos del estudio y las preguntas directrices.

Objetivos de la investigación

- Realizar un análisis comparativo de las metodologías ágiles y determinar cuál de estas se adaptan a las necesidades de desarrollo de software de la institución.

- Aplicar la metodología de desarrollo ágil seleccionada mediante un estudio de caso.
- Determinar los factores críticos de éxito para la adopción de la metodología.

Descripción y actividades a realizar en los objetivos de la investigación

- Análisis y estudio comparativo de metodologías ágiles
- Conclusiones del estudio comparativo.
- Adopción de la metodología de desarrollo ágil seleccionada mediante un estudio de caso.
- Determinación de fases y aspectos de la metodología adoptada.
- Validación y comparación con desarrollos sin metodologías.
- Identificación de factores de éxito del uso de la metodología.

Adicionalmente, las preguntas que siempre deben estar en la mente del investigador son las preguntas directrices.

- ¿Qué metodología ágil de desarrollo de software se adapta a las necesidades de desarrollo de software de la institución?
- ¿En el caso de estudio se aplica todas las fases de la metodología seleccionada?
- ¿Qué factores de éxito se encontraron al implementar la metodología?

3.3.5 Conducción del caso de estudio: Recopilación de datos.

La información se recolecta a través de documentación bibliográfica, observación directa; las entrevistas y encuestas se realizarán al Director de TICs y al equipo de desarrollo de software de la institución. La fuente principal de información saldrá de la revisión sistemática de literatura.

3.3.6 Análisis de datos

En el análisis de datos se debe examinar, categorizar, tabular la información que orienten a alcanzar la proposición del estudio. De este

análisis de información se determinará la metodología de desarrollo ágil a ser implementada, adicionalmente se identificarán los factores de éxito de la implementación.

3.3.7 Resultados y conclusiones.

Para la presentación de los resultados y conclusiones del estudio investigativo se realizará un análisis comparativo de los resultados obtenidos en la investigación con el estado inicial del desarrollo de software en la Dirección de Tics; esto permitirá establecer si los objetivos que se plantearon inicialmente se cumplieron o no; en base a estos aspectos se establecerán las conclusiones finales.

3.4 Población y muestra

La población para la investigación serán los funcionarios del área de TICs de Yachay Tech, estableciendo una confiabilidad de 95% y un error de muestreo de 5%. No se aplica ninguna fórmula para el cálculo de la muestra ya que la población es menor de 100. En el siguiente cuadro se muestra la distribución de la información.

Dirigido a	Número	Porcentaje
Director de TICs	1	20
Analistas de desarrollo	2	40
Pasantes	2	40

Fuente: El Autor
Tabla 2. Distribución de la muestra

3.5 Métodos

Deductivo: Es un método de razonamiento que consiste en tomar conclusiones generales para explicaciones particulares. El método se inicia con el análisis de los postulados, teoremas, leyes y principios, etcétera, de aplicación universal y de comprobada validez, para aplicarlos a soluciones o hechos particulares (Torres, 2006). La investigación que se llevará a cabo tiene un alto componente bibliográfico investigativo por lo cual se tomará

como referencia las conclusiones de otros estudios cuyas particularidades se asemejen a nuestra realidad.

Inductivo: José Cegarra (2004) indica que el método consiste en basarse en enunciados singulares, tales como descripciones de los resultados de observaciones o experiencias para plantear enunciados universales, tales como hipótesis o teorías. Ello es como decir que la naturaleza se comporta siempre igual cuando se dan las mismas circunstancias, lo cual es como admitir que bajo las mismas condiciones experimentales se obtienen los mismos resultados.

3.6 Estrategias Técnicas

Se utilizarán las siguientes técnicas:

- **Encuesta:** La encuesta se utilizará como técnica de investigación para poder realizar recopilación de información que servirá para el análisis e interpretación de estos.
- **Entrevista:** La cual se aplicará a Directores de: Tecnologías, Planificación, y Finanzas.
- **Observación Directa:** Se la realizará mediante visitas a las instalaciones de la Institución en el Departamento de Tecnologías y poder verificar el funcionamiento de los Sistemas.

3.7 Instrumentos

Los instrumentos que se emplearán serán:

- Para la observación se utilizará como instrumento la ficha de observación.
- Celular; como equipo de comunicación.
- Para el caso de la entrevista y la encuesta las preguntas del cuestionario.

CAPÍTULO IV. DESARROLLO DE LA INVESTIGACIÓN

4 Introducción

La implantación de una metodología ágil para la gestión de proyectos de software es parte fundamental para la mejora de la gestión del proceso de desarrollo de software. La gestión de proyectos mediante metodologías ágiles tiene como objetivo dar garantías a las cuatro demandas principales de la industria en la que se ha generado: valor, reducción del tiempo de desarrollo, agilidad y fiabilidad, mejorando la gestión y la satisfacción del cliente (Fuentes, 2015).

Dentro del manifiesto ágil (2018) uno de los aspectos que resalta es la valoración a los individuos y su comunicación, por sobre los procedimientos y las herramientas, también indican que procesos de calidad con personas y relaciones mediocres no darán buenos resultados. Otro aspecto importante del manifiesto es que valora el software que esta funcionando, por sobre la documentación tediosa: la documentación es importante ya que facilita la transmisión del conocimiento, pero su redacción debe limitarse a aquello que aporte valor directo al producto.

Las metodologías ágiles son un grupo de métodos de desarrollo de software fundamentados en el desarrollo iterativo e incremental, en el cual los requerimientos y las soluciones evolucionan a través de la colaboración entre equipos autoorganizados, multifuncionales y clientes. Es un enfoque iterativo de caja de tiempo y alienta una respuesta rápida y flexible a los cambios (Kaur, Jajoo, & Manisha, 2015).

Las metodologías ágiles exigen clientes comprometidos e involucrados en todo el proceso de desarrollo de software, esto garantiza de cierta manera que el producto que se va desarrollando sea validado frecuentemente. Las metodologías ágiles presentan un enfoque totalmente opuesto a las metodologías tradicionales, ofreciendo un enfoque más adecuado para determinados proyectos. Cuando se trata de satisfacer las

necesidades del cliente, las metodologías de desarrollo de software ágil ofrecen muchas ventajas al momento de crear aplicaciones; a través del desarrollo ágil, los programadores y los clientes pueden participar y colaborar fácilmente a lo largo de cada fase del proyecto, este nivel de transparencia ayuda a asegurar que todos los miembros del equipo estén trabajando orientando sus esfuerzos hacia una meta en común. Además, al romper el proceso de desarrollo en etapas hay una mayor capacidad para predecir los costos que van a estar involucrados en la creación de la aplicación. Así como también, las fases de la metodología ágil, conocidas como sprints, permiten a los programadores realizar adaptaciones incluso a medida que se desarrolla el software (agilemanifesto.org, 2018).

4.1 Análisis comparativo de metodologías de desarrollo ágiles

El propósito de este análisis es determinar que metodología se ajusta a las necesidades de desarrollo de la institución; para el análisis se tomará en cuenta la dinámica de la organización y las características propias de las metodologías ágiles. Este análisis se hará de manera progresiva iniciamos observando la orientación de la gestión administrativa de la organización con las metodologías ágiles, posteriormente verificamos al grado de cumplimiento de los principios ágiles y finalmente seleccionamos la metodología que se adapte a la gestión administrativa.

Definición de la escala de importancia.

A continuación se define la escala de importancia que se utilizará para la evaluación tanto de la *orientación de la organización* como del *cumplimiento de principios ágiles*; se ha tomado valores frontera, como *ninguna importancia* y *alta importancia* y valores intermedios de *baja importancia* y *media importancia;* para mejorar la objetividad del análisis a estos se los ha dado un coste de 0 a 3. Las escalas para esta investigación son adaptadas del estudio de Coronado Padilla (2007).

Importancia	valor
Ninguna importancia.	0
Baja importancia.	1
Media importancia.	2
Alta importancia.	3

Tabla 3. Definición escala de importancia para la valoración del enfoque de la organización
Fuente: Propia

4.1.1 Orientación de la organización

En esta etapa se determinará el enfoque de la organización desde el punto de vista de la gestión administrativa frente a los valores ágiles y tradicionales de las metodologías.

Modelo de evaluación del enfoque de la organización frente a las metodologías ágiles y tradicionales.

Para obtener este dato de manera precisa, se comparará cada valor ágil y su relacionamiento con la organización. Se han detallado los valores del manifiesto ágil y se han separado entre orientación ágil y orientación tradicional (Pérez Pérez & González Cabrera, 2012). Estos valores serán evaluados por el Director de TICs según una escala de importancia que se define más adelante.

Orientación ágil		Orientación Tradicional	
Valores ágiles	**Importancia**	**Valores tradicionales**	**Importancia**
Individuo y las interacciones del equipo	*a1*	El proceso y las herramientas	*b1*
Desarrollar software que funciona	*a2*	Conseguir una buena documentación	*b2*
Colaboración con el cliente	*a3*	Negociación contractual	*b3*
Respuesta al cambio	*a4*	Seguimiento de un plan	*b4*

Tabla 4. Definición del modelo de evaluación del enfoque de la organización frente metodologías ágiles y tradicionales.
Fuente: El Autor

En donde variables *a1, a2, a3, a4* representan los valores obtenidos de la valoración de la orientación ágil; y las variables *b1, b2, b3, b4* representan los valores obtenidos de la orientación tradicional.

A continuación se aplica el modelo de evaluación del enfoque de la organización frente a las metodologías ágiles y tradicionales mediante el artefacto de investigación “encuesta” (Ver anexo 3).

Orientación ágil		**Orientación Tradicional**	
Valores ágiles	**Importancia**	**Valores tradicionales**	**Importancia**
Individuo y las interacciones del equipo	2	El proceso y las herramientas	2
Desarrollar software que funciona	3	Conseguir una buena documentación	2
Colaboración con el cliente	3	Negociación contractual	2
Respuesta al cambio	3	Seguimiento de un plan	2
MEDIA	2,75	**MEDIA**	2

Tabla 5. Resultados de la aplicación del modelo de evaluación de las metodologías ágiles y tradicionales.
Fuente: El Autor

Como muestra la tabla 4, la organización presenta un mayor enfoque hacia la orientación ágil lo que corrobora a que la propuesta tenga mayor aceptación por parte de los interesados, sus procesos se enmarcan en las características de las metodologías ágiles; esta conclusión orienta el camino a seguir en esta investigación.

4.1.2 Cumplimiento principios ágiles

En este formulario se valora la relación de los doces principios del desarrollo ágil con la con la gestión organizativa y administrativa de la institución; cada principio ágil extraerá su relación con la organización en el grado que la gerencia estime pertinente (Pérez Pérez & González Cabrera, 2012).

Definición del modelo de evaluación del cumplimiento de los principios ágiles en la gestión de Institución.

Nº	Principios del manifiesto ágil	(Valoración) Director TICs	(Valoración) Jefe de desarrollo	
1	Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.	a1	b1	
2	Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.	a1	b1	
3	Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.	a1	b1	
4	Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.	a1	b1	
5	Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.	a1	b1	
6	El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.	a1	b1	
7	El software funcionando es la medida principal de progreso.	a1	b1	
8	Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.	a1	b1	
9	La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.	a1	b1	
10	La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, son esenciales.	a1	b1	
11	Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.	a1	b1	
12	A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.	a1	b1	
Media		Σ(a1)	Σ(b1)	Σ(Σ (a1)+ Σ (b1))/2

Tabla 6. Definición del modelo de evaluación del cumplimiento de los principios ágiles en la institución.
Fuente: El Autor

Aplicación del modelo de evaluación del cumplimiento de los principios ágiles en la gestión de la Institución.

En la matriz siguiente se evalúa cada uno de los doce principios de las metodologías ágiles; para la evaluación se toma en cuenta su relación con la gestión administrativa de la Universidad, la valoración lo realiza el Director de TICs y el Jefe de desarrollo de proyectos de software; la finalidad de esta evaluación es orientar el estudio analizando el grado de relacionamiento de la institución con los principios ágiles (Ver Anexo 4).

N°	Principios del manifiesto ágil	(Valoración) Director TICs	(Valoración) Jefe de desarrollo	(Media)
1	Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.	2	2	2
2	Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.	2	3	2,5
3	Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.	2	2	2
4	Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.	3	2	2,5
5	Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.	3	3	3
6	El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.	3	3	3
7	El software funcionando es la medida principal de progreso.	3	3	3
8	Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.	2	3	2,5

9	La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.	3	2	2,5
10	La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.	2	3	2,5
11	Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.	3	3	3
12	A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.	3	3	3
Media		31	32	31,5

Tabla 7. Resultados de la evaluación del cumplimiento de los principios ágiles
Fuente: El Autor

Para la interpretación de resultados se toma en cuenta los 12 principios ágiles y considerando que la máxima importancia tiene un valor de 3, en consecuencia el valor frontera que puede alcanzar en torno al cumplimiento de estos principios es de 36. Según el resultado obtenido de 31,5 se deduce que la organización desde la perspectiva de TICs se orienta en un 87,5% al cumplimiento de los principios ágiles. El resultado será más representativo cuantos más miembros de la Institución realicen el cuestionario.

Con esta evaluación se ratifica que la organización tiene un enfoque ágil; por tanto, el siguiente paso es seleccionar la metodología ágil que mejor se adapta a la organización.

4.1.3 Selección de la metodología ágil

A continuación, se seleccionará la metodología de desarrollo, se toma en cuenta las características administrativas de la institución; este estudio lo vamos a realizar utilizando el marco de trabajo de Iacovelli (2008) de evaluación de metodologías ágiles, que se basa en cuatro puntos de vista. Para esto, se realizó un formulario concentrando los puntos de vista: uso, capacidad de agilidad, aplicación, procesos y productos. Se detalló individualmente sus características y propiedades, la valoración y evaluación será realizada por la institución.

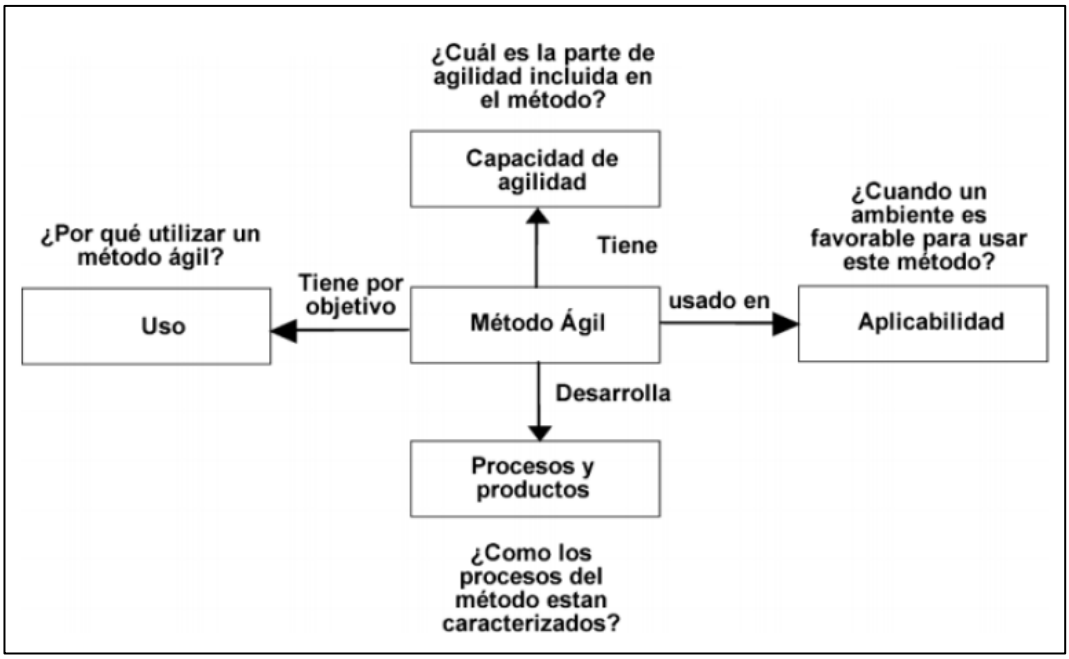

Figura 10. Puntos de vista de las metodologías ágiles (Iacovelli)
Fuente: *(Iacovelli, 2008)*

Uso

Indican las ventajas de la utilización de los métodos ágiles. Las propiedades de este punto de vista valoran y analizan las ventajas que los desarrolladores y los clientes pueden alcanzar si trabajan con las mencionadas metodologías.

Las metodologías ágiles cooperan con una serie de recomendaciones al proceso de desarrollo, aportan normas que dan las pautas necesarias para el trabajo en proyectos con requerimientos muy volátiles, respetando siempre los tiempos de terminación (Iacovelli, 2008).

Las propiedades o atributos son:

- Satisfacción del usuario final.
- Cumplimiento de los requisitos.
- Favorable al offshoring
- Aumento de la productividad.
- El respeto de un nivel de calidad.
- Adaptarse a los entornos turbulentos.
- El respeto de las fechas de entrega.

Capacidad de agilidad

Detalla los atributos y características de agilidad de las metodologías, las propiedades de esta vista incorporan los detalles del concepto de agilidad.

La metodología ágil se ha introducido en el entorno académico como un tema para que los estudiantes de ciencias de la computación aprendan a desarrollar un buen software de manera oportuna para satisfacer los requisitos del cliente. La metodología ágil subraya la comunicación directa y frecuente entre el cliente y el desarrollador a través de sus procesos incrementales. Las soluciones propuestas se desarrollan a través de la comunicación y la colaboración de organizaciones y equipos. Por lo tanto, el éxito clave de la metodología ágil se basa en la comunicación entre los miembros del equipo y la capacidad de adaptarse a los cambios rápidos (Deshinta & Mohana, 2014).

Sus propiedades son:

- Colaboración
- Los requisitos funcionales pueden cambiar
- Nivel de intercambio de conocimientos
- De peso ligero
- Requisito no funcional puede cambiar
- Los recursos humanos pueden cambiar
- Centrado en las personas
- Reactividad
- Refactoring político
- Iteraciones cortas
- Integración de los cambios
- Pruebas de política
- Indicadores de cambio
- Plan de trabajo se puede cambiar

Aplicabilidad

La principal finalidad de este punto de vista es mostrar si el entorno es adecuado para la aplicación de las metodologías ágiles, hace una evaluación de las principales características de un ambiente de trabajo.
Las propiedades son:

- Grado de interacción entre los miembros del equipo
- El grado de interacción con el cliente
- Grado de interacción con los usuarios finales
- Grado de integración de la novedad
- La complejidad del proyecto
- Los riesgos del proyecto
- Tamaño del proyecto
- La organización del equipo
- El tamaño del equipo

Procesos y productos

El punto de vista de los procesos y productos representa cómo se caracterizan los procesos ágiles de la metodología. Los atributos o características describirán a los procesos y los separan en dos aspectos.

El proceso se forma de dos aspectos. El primer aspecto son las tareas propias del desarrollo de software gestionadas por las metodologías ágiles. La segunda indica el grado de adopción de sus normas y reglamentos. (Iacovelli, 2008).
Los atributos son:

Grado de adopción normas y directrices:

- Gestión de proyectos
- Descripción de procesos
- Normas y orientaciones concretas sobre las actividades y productos

Actividades incluidas por el método ágil son:

- Definición de requisitos
- Código
- Pruebas de integración
- Prueba del sistema
- Puesta en marcha del proyecto
- Prueba de aceptación
- Control de calidad
- Sistema de uso
- Pruebas unitarias
- Modelado

Productos de las actividades del método:

- Comentario del código fuente
- Ejecutable
- Pruebas de integración
- Pruebas de sistema
- Modelos de diseño
- Pruebas de aceptación
- Informes de calidad
- Documentación de usuario
- Pruebas unitarias

Para realizar este análisis comparativo se deben llenar inicialmente los formularios que contienen como preguntas los atributos de los 4 puntos de vista de los aspectos de las metodologías ágiles, la respuesta a estos formularios se debe apoyar en las necesidades de la Institución. Las respuestas tienen valores de verdadero o falso para cada una de las preguntas; para este caso la decisión de las respuestas se realizará con apoyo del Director de TICs en base a los intereses de la gestión administrativa de la institución (Ver anexo 5).

Con los mismos atributos del formulario anterior ahora aplicar a las metodologías determinadas para el estudio comparativo, para la evaluación

se debe tomar en cuenta las características de cada una de las metodologías analizadas; las metodologías a utilizarse son Scrum, XP, Kanban y Scrumban que son las más importantes del mercado (Rujana, Romero Franco, Tortosa, Tomasselli, & Pinto, 2016).

A continuación, comparar los valores obtenidos en los dos formularios, en caso de haber coincidencias en las respuestas marcar 1 o caso contrario marcar 0. Finalmente se deben sumar por metodologías los puntos coincidentes, la metodología que mayor puntaje obtenga será la seleccionada.

4.1.3.1 Valoración de los atributos de los puntos de vista de las metodologías ágiles

Uso

Escriba Verdadero (V) o Falso (F) en cada una de las premisas de acuerdo al interés institucional.

Nro.	Premisa	Respuesta
1	Respeto de las fechas de entrega	V
2	Cumplimiento de los requisitos	V
3	Respeto al nivel de calidad	V
4	Satisfacción del usuario final	V
5	Entornos turbulentos	F
6	Favorable al Off shoring	V
7	Aumento de la productividad	V

Tabla 8. Valoración de los atributos del punto de vista *uso*
Fuente: El Autor

Capacidad de agilidad

Ponga la respuesta a los items de acuerdo al interés institucional.

Nro.	Premisa	Respuesta
1	Iteraciones cortas	V
2	Colaboración	V
3	Centrado en las personas	V
4	Refactoring político	F
5	Prueba político	V
6	Integración de los cambios	V
7	De peso ligero	V
8	Los requisitos funcionales pueden cambiar	V

9	Los requisitos no funcionales pueden cambiar	V
10	El plan de trabajo puede cambiar	F
11	Los recursos humanos pueden cambiar	V
12	Cambiar los indicadores	F
13	Reactividad	ITERACIÓN
14	Intercambio de conocimientos (BAJO, ALTO)	BAJO

Tabla 9. Valoración atributos punto de vista *capacidad de agilidad*.
Fuente: El Autor

Aplicación

Escriba Verdadero (V) o Falso (F) en cada una de las premisas de acuerdo al interés institucional.

Nro.	Premisa	Respuesta
1	Tamaño del proyecto (PEQUEÑO, GRANDE)	PEQUEÑO
2	La complejidad del proyecto (BAJA, ALTA)	BAJO
3	Los riesgos del proyecto (BAJO, ALTO)	BAJO
4	El tamaño del equipo (PEQUEÑO, GRANDE)	PEQUEÑO
5	El grado de interacción con el cliente (BAJA, ALTA)	ALTA
6	Grado de interacción con los usuarios finales (BAJA, ALTA)	ALTA
7	Grado de interacción entre los miembros del equipo (BAJA, ALTA)	ALTA
8	Grado de integración de la novedad (BAJA, ALTA)	ALTA
9	La organización del equipo (AUTO-ORGANIZACIÓN, ORGANIZACIÓN JERÁRQUICA)	AUTO

Tabla 10. Valoración atributos del punto de vista *aplicación*
Fuente: El Autor

Procesos y productos

Ponga la respuesta a los ítems de acuerdo al interés institucional.

Nivel de abstracción de las normas y directrices:

Nro.	Premisa	Respuesta
1	Gestión de proyectos	V
2	Descripción de procesos	V
3	Normas y orientaciones concretas sobre las actividades y productos	F

Tabla 11. Valoración del nivel de abstracción de las normas y directrices (Procesos y productos)
Fuente: (Iacovelli, 2008).

Las actividades cubiertas por el método ágil:

Nro.	Premisa	Respuesta
1	Puesta en marcha del proyecto	V
2	Definición de requisitos	V
3	Modelado	F
4	Código	V
5	Pruebas unitarias	F
6	Pruebas de integración	F
7	Prueba del sistema	V
8	Prueba de aceptación	V
9	Control de calidad	F
10	Sistema de uso	V

Tabla 12. Valoración de las actividades cubiertas por el método ágil (Procesos y productos)
Fuente: (Iacovelli, 2008)

Productos de las actividades del método:

Nro.	Premisa	Respuesta
1	Modelos de diseño	F
2	Comentario del código fuente	V
3	Ejecutable	V
4	Pruebas unitarias	F
5	Pruebas de integración	F
6	Pruebas de sistema	V
7	Pruebas de aceptación	V
8	Informes de calidad	F
9	Documentación de usuario	V

Tabla 13. Valoración de los productos de las actividades del método (Procesos y productos)
Fuente: (Iacovelli, 2008)

4.1.3.2 Evaluación de las metodologías ágiles vs los atributos de los cuatro puntos de vista

En el siguiente formulario se evalúan los atributos de los cuatro puntos de vista ágil con las metodologías a realizar el estudio comparativo; para este estudio se han tomado cuatro de las metodologías ágiles más importantes en la actualidad (Rujana, Romero Franco, Tortosa, Tomasselli, & Pinto, 2016). Esta evaluación lo realiza el investigador tomando en cuenta los principios y características de las metodologías a utilizar.

		METODOLOGÍAS ÁGILES				
			Orientada al desarrollo de Software	Orientada a la gestión de proyectos		
			XP	SCRUM	KANBAN	SCRUMBAN
USO	¿Por qué utilizar un método ágil?	Respeto de las fechas de entrega.	F	V	F	F
		Cumplimiento de los requisitos	V	V	V	V
		Respeto al nivel de calidad	F	F	F	F
		Satisfacción del usuario final	F	V	F	F
		Entornos turbulentos	F	V	V	V
		Favorable al Off shoring	F	V	F	V
		Aumento de la productividad	V	V	V	V
CAPACIDAD DE AGILIDAD	¿Cuáles la parte de agilidad incluida en el método?	Iteraciones cortas	V	V	V	V
		Colaboración	V	V	V	V
		Centrado en las personas	V	V	V	V
		Refactoring político	V	F	F	F
		Prueba político	V	V	F	V
		Integración de los cambios	V	V	V	V
		De peso ligero	V	V	V	V
		Los requisitos funcionales pueden cambiar	V	V	V	V
		Los requisitos no funcionales pueden cambiar	F	F	V	V
		El plan de trabajo puede cambiar	v	F	v	v
		Los recursos humanos pueden cambiar	v	F	v	v
		Cambiar los indicadores	v	F	F	F
		Reactividad (AL COMIENZO DEL PROYECTO, CADA ETAPA, CADA ITERACIÓN)	ITERACIÓN	ITERACIÓN	ITERACIÓN	ITERACIÓN
		Intercambio de conocimientos (BAJO, ALTO)	ALTO	BAJO	BAJO	BAJO
APLICABILIDAD	¿Cuándo un ambiente es favorable para usar este método?	Tamaño del proyecto (PEQUEÑO, GRANDE)	PEQUEÑO	GRANDE/PEQUEÑO	PEQUEÑO	GRANDE/PEQUEÑO
		La complejidad del proyecto (BAJA, ALTA)	BAJA	ALTA	BAJA	ALTA
		Los riesgos del proyecto (BAJO, ALTO)	BAJO	ALTO	BAJO	ALTO
		El tamaño del equipo (PEQUEÑO, GRANDE)	PEQUEÑO	PEQUEÑO	PEQUEÑO	PEQUEÑO
		El grado de interacción con el cliente (BAJA, ALTA)	ALTA	ALTA	BAJA	BAJA
		Grado de interacción con los usuarios finales (BAJA, ALTA)	BAJA	ALTA	BAJA	BAJA

		Grado de interacción entre los miembros del equipo (BAJA, ALTA)	ALTA	ALTA	BAJA	ALTA
		Grado de integración de la novedad (BAJA, ALTA)	ALTA	ALTA	BAJA	ALTA
		La organización del equipo (AUTO-ORGANIZACIÓN, ORGANIZACIÓN JERÁRQUICA)	AUTO - ORGANIZACIÓN	AUTO - ORGANIZACIÓN	AUTO - ORGANIZACIÓN	AUTO - ORGANIZACIÓN
PROCESOS Y PRODUCTOS	¿Cómo están caracterizados los procesos del método?	**Nivel de abstracción de las normas y directrices.**				
		Gestión de proyectos	F	V	F	V
		Descripción de procesos	V	F	F	F
		Normas y orientaciones concretas sobre las actividades y productos	V	F	F	F
		Las actividades cubiertas por el método ágil				
		Puesta en marcha del proyecto	F	F	F	F
		Definición de requisitos	V	V	F	V
		Modelado	V	V	F	V
		Código	V	V	V	V
		Pruebas unitarias	V	V	V	V
		Pruebas de integración	V	V	V	V
		Prueba del sistema	V	V	V	V
		Prueba de aceptación	F	F	F	F
		Control de calidad	F	F	F	F
		Sistema de uso	F	F	F	F
		Productos de las actividades del método ágil				
		Modelos de diseño	F	v	F	v
		Comentario del código fuente	v	v	v	v
		Ejecutable	v	v	v	v
		Pruebas unitarias	v	v	v	v
		Pruebas de integración	v	v	v	v
		Pruebas del sistema	v	F	v	v
		Pruebas de aceptación	F	F	F	F
		Informes de calidad	F	F	F	F
		Documentación de usuario	F	F	F	F

Tabla 14. Resultados de la matriz de evaluación de las metodologías ágiles vs los atributos de los cuatro puntos de vista ágiles de Iacovelli
Fuente: El Autor

4.1.3.3 Formulario de evaluación final

En el siguiente formulario se comparan los valores obtenidos en los dos formularios anteriores, en caso de haber coincidencias en las respuestas marcar 1, caso contrario marcar 0. Finalmente se deben sumar los valores obtenidos por metodologías; en consecuencia, la metodología que mayor puntaje obtenga será la seleccionada para la investigación.

	METODOLOGIAS AGILES				
USO		Orientada al desarrollo de Software	Orientada a la gestión de proyectos		
		XP	SCRUM	KANBAN	SCRUMBAN
	Respeto de las fechas de entrega.	0	1	0	0
	Cumplimiento de los requisitos	1	1	1	1
	Respeto al nivel de calidad	0	0	0	0
	Satisfacción del usuario final	0	1	0	0
	Entornos turbulentos	1	0	0	0
	Favorable al Off shoring	0	1	0	1
	Aumento de la productividad	1	1	1	1
CAPACIDAD DE AGILIDAD	Iteraciones cortas	1	1	1	1
	Colaboración	1	1	1	1
	Centrado en las personas	1	1	1	1
	Refactoring político	0	1	1	1
	Prueba político	1	1	1	1
	Integración de los cambios	1	1	1	1
	De peso ligero	1	1	1	1
	Los requisitos funcionales pueden cambiar	1	1	1	1
	Los requisitos no funcionales pueden cambiar	0	0	1	1
	El plan de trabajo puede cambiar	0	1	0	0
	Los recursos humanos pueden cambiar	1	0	1	1
	Cambiar los indicadores	0	1	1	1
	Reactividad (AL COMIENZO DEL PROYECTO, CADA ETAPA, CADA ITERACION)	1	1	1	1
	Intercambio de conocimientos (BAJO, ALTO)	0	1	1	1
APLICABILIDAD	Tamaño del proyecto (PEQUENO, GRANDE)	0	1	0	1
	La complejidad del proyecto (BAJA, ALTA)	0	1	0	1
	Los riesgos del proyecto (BAJO, ALTO)	1	0	0	1
	El tamaño del equipo (PEQUENO, GRANDE)	1	1	1	1
	El grado de interacción con el cliente (BAJA, ALTA)	1	1	0	0
	Grado de interacción con los usuarios finales (BAJA, ALTA)	0	1	0	0
	Grado de interacción entre los miembros del equipo (BAJA, ALTA)	1	1	0	1
	Grado de integración de la novedad (BAJA, ALTA)	1	1	0	1
	La organización del equipo (AUTO-ORGANIZACION, ORGANIZACION JERARQUICA)	1	1	1	1
PROCESOS Y PRODUCTOS	**Nivel de abstracción de las normas y directrices.**				
	Gestión de proyectos	0	1	0	1
	Descripción de procesos	1	0	0	0
	Normas y orientaciones concretas sobre las actividades y productos	0	1	1	1
	Las actividades cubiertas por el método ágil				
	Puesta en marcha del proyecto	0	0	0	0
	Definición de requisitos	1	1	0	1
	Modelado	0	0	1	0
	Código	1	1	1	1
	Pruebas unitarias	0	0	0	0
	Pruebas de integración	0	0	0	0

	Prueba del sistema	1	1	1	1
	Prueba de aceptación	0	0	0	0
	Control de calidad	1	1	1	1
	Sistema de uso	0	0	0	0
	Productos de las actividades del método ágil				
	Modelos de diseño	1	0	1	0
	Comentario del código fuente	1	1	1	1
	Ejecutable	1	1	1	1
	Pruebas unitarias	0	0	0	0
	Pruebas de integración	0	0	0	0
	Pruebas del sistema	1	0	1	1
	Pruebas de aceptación	0	0	0	0
	Informes de calidad	1	1	1	1
	Documentación de usuario	0	0	0	0
	TOTAL EVALUACIÓN	**28**	**34**	**27**	**33**

Tabla 15. Matriz de valoración de las metodologías ágiles
Fuente: El Autor

4.1.4 Análisis y conclusiones de resultados del estudio comparativo

Metodología	Puntuación	Porcentaje
XP	28	53,85
Scrum	34	**65,38**
Kanban	27	51,92
Scrumban	33	63,46

Tabla 16. Resumen de resultados del estudio comparativo de metodologías ágiles
Fuente: El Autor

Como se puede observar en la *tabla 15* la metodología Scrum tiene el puntaje más alto, por lo tanto, es la metodología ágil que más se ajusta a las necesidades de desarrollo de software de la institución, en consecuencia, y por lo tanto se puede adoptar esta metodología para el presente estudio investigativo.

CAPÍTULO V. ESTUDIO DE CASO

5 Introducción

En este capítulo se va a implantar la metodología ágil seleccionada mediante un estudio de caso; con esto se pretende experimentar el comportamiento y la adaptación de dicha metodología en un proyecto de desarrollo de software. Luego del estudio comparativo de metodologías ágiles se determinó que Scrum se adapta de manera más significativa a las necesidades de desarrollo de software de la institución. Scrum es un marco de trabajo para el desarrollo y el mantenimiento de productos complejos.

Scrum es un conjunto de prácticas enfocadas a aumentar la productividad de la organización, está enfocado a conseguir pequeños incrementos de software completamente funcionales mientras se trabaja en él (SCRUM STUDY targeting success, 2016).

5.1 Definición del estudio de caso

El caso de estudio consistió en desarrollar una solución informática para la elaboración del Plan Operativo Anual (POA); con la finalidad de optimizar el desarrollo de las actividades que se realizan en la planificación anual en las diferentes unidades administrativas de la Universidad.

Este proceso consiste en la elaboración del POA en la fase de solicitud de compras, solicitud avales, solicitud de financiamiento, solicitud de reforma presupuestaria y certificación; el proceso se lo realiza anualmente en la cual los centros de costos de la Universidad realizan solicitudes de compras de bienes y servicios que estén programados en su planificación anual; estas compras o actividades deben estar en concordancia con los objetivos del Plan estratégico institucional y del Plan Nacional de Desarrollo Toda una Vida (Secretaria Nacional de Planificación y Desarrollo, 2017).

5.2 Implementación y desarrollo del estudio de caso

5.2.1 Definición de Scrum

5.2.1.1 Definición del Equipo de trabajo

Rol	Nombre	Función
Propietario del producto (Product Owner)	Analista de planificación y procesos (Dirección de Planificación), área requirente del proyecto.	El propietario del producto es el encargado de garantizar las condiciones para que se desarrolle un producto con valor. El cómo se lleva a cabo esto podría variar ampliamente entre distintas organizaciones, Equipos Scrum e individuos. (SCRUM STUDY targeting success, 2016)
Scrum Master	Analista de desarrollo informático (TICs), funcionario de la dirección de Tecnologías de la información.	El scrum master es responsable de facilitar el proceso de desarrollo, asegurando que el equipo utilice toda la gama de valores, prácticas y reglas ágiles apropiadas. El scrum master lleva a cabo reuniones de coordinación diarias y elimina cualquier impedimento que el equipo encuentre (Bass, 2014).
Development Team	Programador 1 Programador 2 Programador 3	El Equipo de Desarrollo consiste en los profesionales que desempeñan el trabajo de entregar un Incremento de producto "Terminado", que potencialmente se pueda poner en producción, al final de cada Sprint. Solo los miembros del Equipo de desarrollo participan en la creación del Incremento.

Tabla 17. Definición del Equipo de trabajo
Fuente: El Autor

5.2.1.2 Planificación del Sprint Backlog

Se definió que los Sprints tengan una duración de 3 semanas en las cuales la planificación del proyecto se realizó de tal manera que en la primera reunión de definió el Product backlog, y en las demás reuniones se definieron los sprint backlog de las iteraciones.

La reunión de planificación se denomina *sprint planning meeting* a esta concurren el Product Owner, el Scrum Master y el Develoment Team con la finalidad de delimitar, elegir y priorizar la lista Backlog del producto y las

funcionalidades sobre las que se va a trabajar, que serán parte de cada uno de los sprints. (Trigas Gallego & Domingo Troncho, 2014). En esta reunión adicionalmente se define el tiempo estimado de trabajo de cada una de las tareas del sprint.

Product Backlog

El product backlog se utiliza como un contenedor de nuevos requisitos para reemplazar los requisitos anteriores, corregir errores o eliminar características. Los elementos en la parte superior del product backlog se priorizan como elementos más refinados que otros elementos. El product backlog contiene pequeñas cantidades de requisitos denominados "sprint backlog" que se asignan a los miembros del equipo para el desarrollo incremental llamado sprint (Alsalemi & Yeoh, 2015).

Para la implantación del estudio de caso se tomó como referencia las especificaciones del “Product Owner”, en este caso fue la Analista de Procesos de la Dirección de Planificación de la institución; el estudio comprendió el desarrollo de tres Sprints.

La elaboración del Product Backlog se realizó con el Product Owner; en la cual se analizó y definió los requerimientos para el desarrollo del producto. Siguiendo la guía de Scrum se estableció la importancia y prioridad de las historias de usuario, dicha información orientó al momento de elaborar el primer Sprint Backlog.

A continuación, se describen las historias de usuario indicando su respectiva importancia y prioridad.

Identificador (ID) de la historia	Rol	Característica / Funcionalidad	Razón / Resultado	Contexto	Importancia
H-01	Development Team	Estudio y definición de la plataforma tecnológica	Definir y configurar un entorno de desarrollo adecuado que permita desarrollar un producto de calidad	Es importante al iniciar el proyecto estudiar la plataforma tecnológica para analizar su funcionamiento, esto debe estar en concordancia con la disponibilidad de recursos de los sponsors del proyecto.	ALTA
H-02	Analista de planificación	Login de usuario	Pantalla para el acceso al	En esta instancia se gestiona el acceso a	ALTA

			sistema la cual gestione los diferentes tipos de usuarios	los diferentes módulos del sistema; su acceso depende del rol del usuario final.	
H-03	Analista de planificación	Registrar los objetivos operativos de las unidades administrativas	Relacionar los objetivos operativos de las unidades con las estrategias del plan estratégico institucional	Los objetivos de las unidades administrativas deben aportar al cumplimiento de objetivos del plan estratégico de la institución; y estos a su vez estar en concordancia con los objetivos del Plan Nacional del Buen Vivir.	MEDIA
H-04	Analista de planificación	Registrar las actividades o solicitudes de compra de las unidades administrativas	Realizar un consolidado de todas las solicitudes de compras que aporten al cumplimiento de los objetivos de Plan Estratégico Institucional (PEI)	Las solicitudes de compras son las entradas principales del sistema, es la información de partida para la gestión del proceso de solicitud de compras	MEDIA
H-05	Analista de planificación	Realizar la solicitud de aval	Realizar la certificación presupuestaria para la adquisición del bien o servicio.	Para la solicitud de compra de un bien o servicio es necesario el aval del valor presupuestado, verificando previamente la disponibilidad presupuestaria correspondiente tanto dentro de la institución como en los organismos financieros gubernamentales.	MEDIA

Tabla 18. Product Backlog
Fuente: El Autor

Sprint Backlog

En la segunda parte de la reunión se estructura el “Sprint Backlog”, se identifican las tareas de las historias de usuario colocadas en la Product Backlog, estas van a ser desarrolladas durante el primer Sprint llevando el orden de importancia establecida con el Product Owner.

El representante del cliente toma cada historia de usuario y la desglosa de forma que permita identificar de forma más fácil las tareas a llevar a

cabo, el resultado de este desglose es el **"**Sprint Backlog**"** (SCRUM STUDY targeting success, 2016)**.** Para el presente caso de estudio los Sprints tuvieron una duración de 15 días laborales.

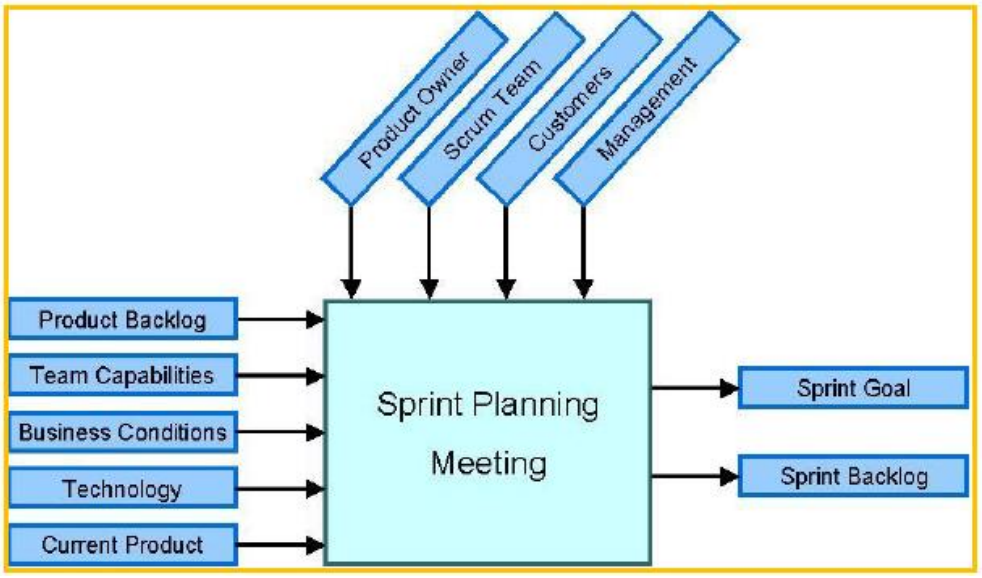

Figura 11. Entradas y salidas de la reunión de planificación
Fuente: (Trigas Gallego & Domingo Troncho, 2014)

Estimación póker

Planning Poker, es una técnica de estimación para proyectos ágiles propuesta en un inicio por Grenning en el año 2002 y popularizada por Cohn, la cual permite hacer una estimación inicial del proyecto rápidamente. Esta estimación se realiza de manera consensuada con base en los requisitos.

El proceso de Planning poker inicia con una reunión entre los miembros del equipo, se plantea una historia de usuario, se analiza y utilizando las cartas cada miembro muestra cuanto cree que sea el valor que debe asignarse. Como número resultante se puede tomar la más alta, la media, la más baja, o cualquier otro método; recuerde que el valor asignado debe ser calculado con un método consensuado por todos los miembros del equipo (Iglesias Solano, 2011).

El equipo garantiza tener listos los elementos de la pila que hayan sido elegidos, siendo esto una clave de Scrum, ya que son ellos mismos

quienes, basándose en su propio análisis y planificación se comprometen a terminar el sprint.

Luego de esto, el equipo estima la cantidad de tiempo que emplearía en cada actividad, empleándose para esto el método Póker. Cada miembro da su estimación con una tarjeta que contiene un número que simbolizaba el número de horas que tardaría en realizar la actividad, siendo luego comparadas con las del resto del equipo y escogiéndose, a partir de estas, la media del tiempo que cada miembro del equipo haya manifestado (Iglesias Solano, 2011). El significado de los números y símbolos utilizados en el póker se muestran a continuación:

- 0: La tarea no tomará tiempo pues ya está realizada.
- ½: Tarea que tomará media hora para su realización.
- ? : El significado que representa esta tarjeta en la reunión es que no se tiene conocimiento de cómo se realizará la actividad o no sabe de qué se está hablando ya que no se maneja el tema.
- α: El significado que representa esta tarjeta en la reunión es que la actividad que se está evaluando tomara un tiempo indefinido, es decir, es incalculable.
- 1, 2, 3, 5, 8, 13, 20, 40, 100: Cada número representa una tarjeta, y representa el número de horas que la actividad tomará en realizarse.

5.2.1.3 Seguimiento y desarrollo del Sprint

Una vez terminada la planeación, se da inicio al desarrollo del Sprint, y con ella una de las características claves de Scrum: la reunión diaria.

Reunión Diaria (Daily Meeting)

Esta es una reunión corta de 15 minutos, siendo dirigida normalmente por el Líder del equipo; se realiza de pie y a ella asiste todo el equipo de trabajo.

La reunión diaria para el seguimiento del sprint es una forma de mantener la auto-organización y auto–control entre los integrantes. En ella se contestan las tres preguntas claves, y si se encontrase algún tipo de

inconveniente se le comunica al Scrum Master que tiene que tomar las medidas al respecto (SCRUM STUDY targeting success, 2016).

Se procede como viene a continuación.

Cada integrante del equipo responde estas tres interrogantes:

1. ¿En qué tarea trabajó ayer?
2. ¿En qué tarea trabajará hoy?
3. Si van a necesitar algo especial o prevén algún impedimento para realizar su trabajo.

Kanban

Con la información de la reunión diaria se actualizan las columnas de estado de avance como:

- *No iniciadas* (TO DO),
- *En Progreso* (IN PROGRESS) y
- *Completadas* (DONE)

Con esta información se puede generar un **Kanban** de estado de las tareas.

Burndown Chart

El *Burndown Chart* genera un gráfico del trabajo pendiente a lo largo del tiempo, muestra la velocidad a la que se está completando las tareas. Ayuda a realizar una proyección indicando si el equipo terminará el trabajo en el tiempo establecido. Burndown chart es una herramienta de visibilidad de proyectos Scrum que muestra el progreso del equipo hacia su objetivo y el trabajo restante para alcanzar tal fin (SCRUM STUDY targeting success, 2016). Diariamente se iba actualizando la matriz con las horas reales empleadas en las diferentes tareas.

5.2.1.4 Reunión de revisión del Sprint

Una vez que terminó el Sprint se hace la revisión del mismo. El Scrum master y el Product Owner revisan los avances o entregables del producto.

Llegado a este punto, se inicia la planeación del siguiente Sprint, se define la fecha y lugar. En este punto el Product Owner podía actualizar la

pila del Product Backlog con cambios o nuevas actividades. El Product Owner y el equipo estaban listos para empezar otro ciclo de Sprint.

Con lo descrito hasta el momento se ha detallado el proceso que se siguió para el desarrollo del primer Sprint y fue repetitivo en el desarrollo de los siguientes, hasta la obtención del producto final.

5.2.1.5 Reunión de Retrospectiva

En esta reunión el equipo habla sobre los aspectos positivos y negativos en el transcurso del desarrollo de primer Sprint. Se evidencia los factores positivos y lo que se puede mejorar, buscando causas y previniéndolas en el próximo Sprint. El Scrum Master es quien dirige esta reunión y se encarga de viabilizar y gestionar las recomendaciones; esta reunión tendrá una duración máxima de tres horas (SCRUM STUDY targeting success, 2016).

5.2.2 Resumen de Sprints.

Nro.	Denominación	Descripción
1	Sprint 0	Estudio y definición de la plataforma tecnológica.
2	Sprint 1	Login de usuarios y registro de objetivos operativos de las unidades operativas.
3	Sprint 2	Registrar las actividades o solicitudes de compra de las unidades administrativas, Realizar la solicitud de aval.

Tabla 19. Resumen de Sprints
Fuente: El Autor

5.2.3 Sprint 0

5.2.3.1 Planificación

Sprint backlog

Con la presencia de todos los involucrados en el proyecto se realizó la reunión de planificación del *Sprint backlog 0,* se desglosó las historias de usuarios en tareas más específicas y determinó las que formarán parte de la primera iteración. En este Sprint básicamente se solventó los requerimientos no funcionales, se definió la arquitectura, analizó y estudió los entornos de desarrollo que cumplan las normativas internas como son

la utilización de software libre, se estudió las tecnologías web más dinámicas que se acoplen al entorno de desarrollo determinado y finalmente se instaló y configuró las herramientas de desarrollo mencionadas. El equipo de desarrollo utilizó método *póker* para la estimación del tiempo de duración.

En la matriz inferior se muestra el sprint backlog para la primera iteración.

SPRINT 0					
Identificador (ID) de la historia	**Característica / Funcionalidad**	**Responsable**	**Actividad N°**	**Tarea**	**Tiempo estimado (Horas)**
H-01	Estudio y definición de la plataforma tecnológica	Programador 1	H-01-1	Investigar IDEs de desarrollo libre	18
		Programador 2	H-01-2	Investigar Tecnología MAVEN	16
		Programador 3	H-01-3	Investigar Frameworks basados en java	17
		Programador 1	H-01-4	Investigar componentes Primeface y Boothfaces	14
		Programador 3	H-01-5	Explorar sistemas gestores de base de datos no privativos	8
		Programador 3	H-01-6	Instalación y configuración del sistema gestor de base de datos	8
		Programador 2	H-01-7	Analizar y definir la arquitectura	13
		Programador 3	H-01-8	Crear la estructura de directorios del proyecto web basado en la arquitectura definida	6
		Programador 2	H-01-9	Definir nomenclatura de términos	6
		Programador 1	H-01-10	Socialización Herramientas a utilizar	12
		Programador 2	H-01-11	Configurar el proyecto con plantilla base	8

Tabla 20. Sprint 0
Fuente: El Autor

5.2.3.2 Seguimiento y desarrollo del Sprint

Reunión diaria

Una vez que el equipo inició el desarrollo del Sprint, las reuniones diarias constituían un espacio para dilucidar posibles atascos en el desarrollo, para el caso del Sprint 0 no hubo mayores incidentes.

DAILY MEETING REGISTER						
Nro.	**Meeting Date**	**Developer**	**What did you do yesterday?**	**What will you do today?**	**Are there impediments in your work?**	**observations**
1	06/03/2018	Programador 1	Investigar IDEs de desarrollo libre	Investigar IDEs de desarrollo libre	none	
2	06/03/2018	Programador 2	Investigar Tecnología MAVEN	Investigar Tecnología MAVEN	none	
3	06/03/2018	Programador 3	Investigar Frameworks basados en java	Investigar Frameworks basados en java	none	
4	07/03/2018	Programador 1	Investigar IDEs de desarrollo libre	Investigar IDEs de desarrollo libre	none	
5	07/03/2018	Programador 2	Investigar Tecnología MAVEN	Investigar Tecnología MAVEN	none	
6	07/03/2018	Programador 3	Investigar Frameworks basados en java	Investigar Frameworks basados en java	none	
7	08/03/2018	Programador 1	Investigar IDEs de desarrollo libre	Investigar IDEs de desarrollo libre	none	
8	08/03/2018	Programador 2	Investigar Tecnología MAVEN	Investigar Tecnología MAVEN	none	
9	08/03/2018	Programador 3	Explorar sistemas gesores de base de datos no privativos	Explorar sistemas gesores de base de datos no privativos	none	

Tabla 21. Registro de Reuniones diarias
Fuente: El Autor.

Mediante la herramienta Sprint Backlog se realizó el seguimiento y control a las tareas; se monitoreó constantemente indicando sus estados conforme al avance, se marcó como *No iniciadas* (TO DO), *En Progreso* (IN PROGRESS) y *Completadas* (DONE).

Identificador (ID) de la historia	**Característica / Funcionalidad**	**Responsable**	**Actividad N°**	**Tarea**	**Tiempo estimado (Horas)**	**Tiempo Real**	**Estado**
H-01	Estudio y definición de la plataforma tecnológica	Programador 1	H-01-1	Investigar IDEs de desarrollo libre	18	16	DONE
		Programador 2	H-01-2	Investigar Tecnología MAVEN	16	10	DONE
		Programador 3	H-01-3	Investigar Frameworks basados en java	17	16	IN PROGRESS
		Programador 1	H-01-4	Investigar componentes Primeface y Boothfaces	14	15	IN PROGRESS
		Programador 3	H-01-5	Explorar sistemas gestores de base de datos no privativos	8	10	IN PROGRESS
		Programador 3	H-01-6	Instalación y configuración del Sistema gestor de base de datos	8	8	IN PROGRESS

		Programador 2	H-01-7	Analizar y definir la arquitectura	13	15	IN PROGRESS
		Programador 3	H-01-8	Crear la estructura de directorios del proyecto web basado en la arquitectura definida	6	8	TO DO
		Programador 2	H-01-9	Definir nomenclatura de términos	6	8	TO DO
		Programador 1	H-01-10	Socialización Herramientas a utilizar	12	7	TO DO
		Programador 2	H-01-11	Configurar el proyecto con plantilla base	8	9	TO DO

Tabla 22. Sprint Backlog 0
Fuente: El Autor

Como parte de seguimiento se puede visualizar el estado de avance de las tareas mediante el tablero Kanban, En la figura inferior se puede observar los cambios de estados de las tareas mientras avanza proyecto.

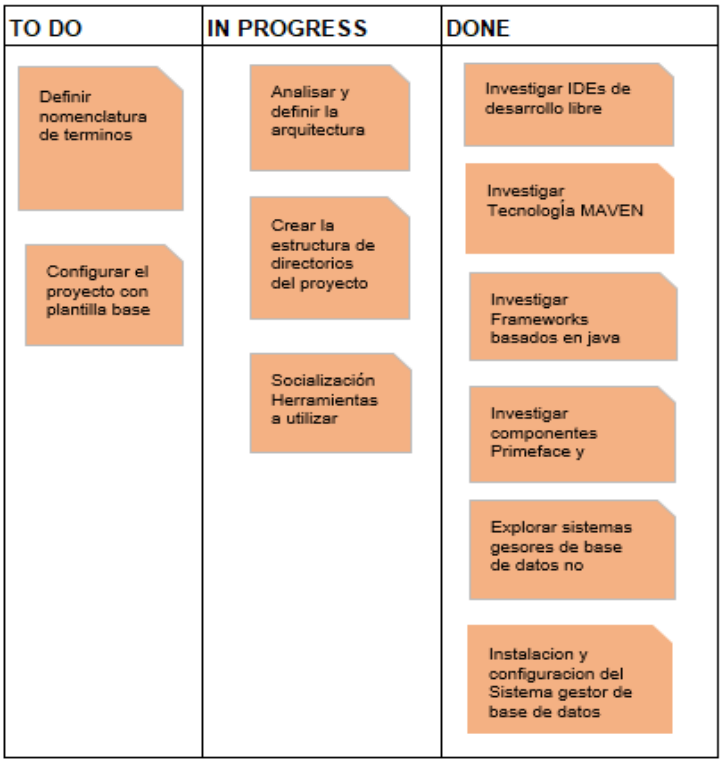

Figura 12. Kanban de estado de avance de tareas.
Fuente: El Autor

El Burndown Chart

Para mejorar el proceso de seguimiento el Burndown chart proporciona una medición día a día de la cantidad de trabajo que resta para terminar el

Sprint, mediante este análisis se debe tomó acciones en caso de observar demora en el desarrollo.

ID Tareas	Estimado(horas)	Dia 15	Dia 14	Dia 13	Dia 12	Dia 11	Dia 10	Dia 9	Dia 8	Dia 7	Dia 6	Dia 5	Dia 4	Dia 3	Dia 2	Dia 1	Total de Horas reales
H-01-1	18	2	2	2	2	2	3	2	1	0	0	0	0	0	0	0	16
H-01-2	16	1	0	0	2	1	3	1	1	0	1	1	2	1	0	0	14
H-01-3	17	0	1	0	3	2	3	1	1	0	2	0	2	1	0	0	16
H-01-4	14	1	1	0	2	2	3	0	0	0	2	0	2	1	1	0	15
H-01-5	8	1	1	0	1	2	2	1	0	0	1	0	1	1	0	0	11
H-01-6	8	0	1	1	1	1	0	0	0	1	1	0	0	0	1	1	8
H-01-7	13	0	0	0	1	1	1	1	2	0	1	0	0	0	0	0	7
H-01-8	6	0	1	1	1	1	3	1	0	0	0	0	0	0	0	1	9
H-01-9	6	0	0	0	1	0	0	2	2	1	0	0	0	0	0	0	6
H-01-10	12	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	4
H-01-11	8	0	0	0	0	1	1	2	2	0	0	0	0	1	1	0	8
Horas reales	126	121	114	110	96	83	63	52	43	41	33	32	25	19	15	12	114
Horas estimadas restantes	126	117,6	109,2	100,8	92,4	84	75,6	67,2	58,8	50,4	42	33,6	25,2	16,8	8,4	0	

Tabla 23. Matriz de control Sprint 0
Fuente: El Autor

En la gráfica inferior se puede observar que la línea roja representa las horas estimadas y la línea azul intensa representa las horas reales empleadas para el trabajo; si la línea azul está por encima de la línea roja significa que existe un leve retraso en el desarrollo de las tareas versus lo planificado inicialmente.

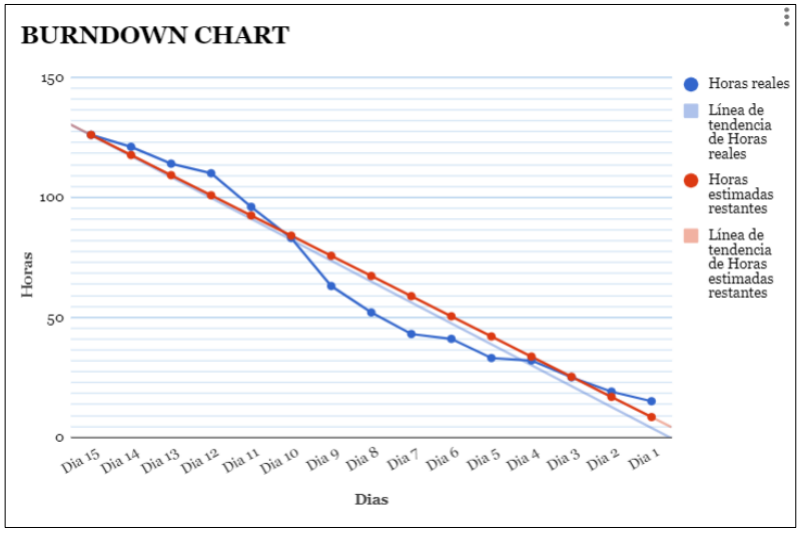

Tabla 24. Gráfica Burndown Chart Sprint 0
Fuente: El Autor

La gráfica muestra que en los primeros días de desarrollo del sprint hubo la tendencia a retrasarse en el cumplimento con relación a lo estimado, a medida que avanza el proyecto la tendencia cambia y se invierte para luego estabilizarse en la línea de trabajo estimado, el equipo también debía ir

analizando nuevos requisitos, separar elementos grandes en otros más pequeños, estimar actividades imprevistas y restimando elementos ya existentes con lo cual se dejaban actividades claras para el siguiente Sprint. La duración del Sprint nunca se prolongó más allá del tiempo establecido, es decir, se terminaron en la fecha asignada.

5.2.3.3 Revisión de Sprint

Una vez que se terminó el Sprint se hizo la revisión y presentación del avance. El Scrum master y el Product Owner revisaron los entregables del producto. En este Sprint se trabajó en los requerimientos no funcionales, que eran básicamente aspectos referentes a la infraestructura y entornos de trabajo.

Incremento del producto potencialmente entregables

En las gráficas siguientes se indican incrementos que se realizaron en el Sprint 0.

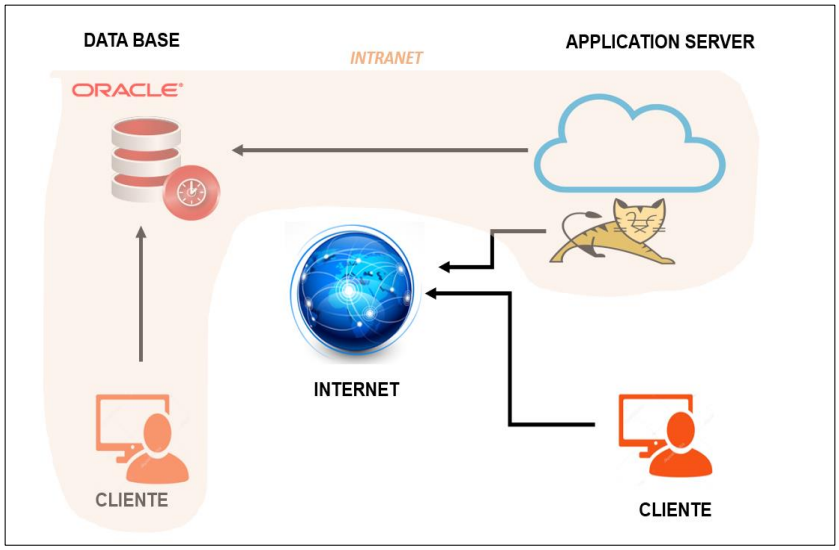

Figura 13. Arquitectura Tecnológica.
Fuente: El Autor

<table>
<tr><td colspan="2">VISTA</td><td>BOOTSFACES
Jquery
HTML5
CSS3
Javascript</td><td>  </td></tr>
<tr><td rowspan="3">CONTROLADOR</td><td rowspan="3">JSF</td><td>MANAGER BEAN</td><td rowspan="3">  </td></tr>
<tr><td>MANAGER</td></tr>
<tr><td>MANAGER DAO</td></tr>
<tr><td rowspan="3">MODELO</td><td rowspan="3">ORM</td><td>JPA</td><td rowspan="3"></td></tr>
<tr><td>ECLIPSE LINK</td></tr>
<tr><td>ENTITIES</td></tr>
</table>

Figura 14. Arquitectura del Software del sistema (Patrón de diseño)
Fuente: El Autor

5.2.3.4 Reunión de Retrospectiva

Una vez finalizado el Sprint se realizó la reunión retrospectiva, el objetivo de esta reunión es analizar cómo está trabajando el equipo, prácticas o herramientas que se están utilizando con la finalidad de mejorar el proceso. Las conclusiones y recomendaciones del primer Sprint se encuentran resumidas a continuación:

REUNIÓN DE RETROSPECTIVA			
POSITIVO	**MEJORABLE**	**SOLUCIONES**	**IDEAS**
Estimaciones cumplidas	Mejorar la visión de los requisitos desde la perspectiva del cliente	Dar más visibilidad de los requerimientos incluyendo a los usuarios finales de los mismos (no solo al Product Owner como jefe)	Planificar la demo para 2 días después del fin de desarrollo de sprint para permitir tiempo de preparación, resolución de posibles problemas de última hora
Requisitos claros para el equipo	Mejorar la toma de requisitos durante el Sprint		
Buena comunicación en Equipo de desarrollo	Entornos de desarrollo y prueba adecuados	Proveer de infraestructura para los entornos requeridos	Conocer infraestructura
			Aportar/plantear posibles desarrollos que puedan interesar a negocio

Tabla 25. Formulario de observaciones Reunión Retrospectiva
Fuente: El Autor

5.2.4 Sprint 1

5.2.4.1 Planificación

Continuando con la ejecución del proyecto en la reunión de planificación del *Sprint Backlog 1,* se desglosó las historias de usuarios en tareas más específicas las que formarán parte del nuevo sprint; en esta iteración se empezó el a atender la especificación de requerimientos funcionales del sistema, se tomaron en cuenta las observaciones y recomendaciones evidenciadas en el Sprint anterior.

En la matriz inferior se muestra la planificación del Sprint Backlog para la segunda iteración.

SPRINT 1					
Identificador (ID) de la historia	**Característica / Funcionalidad**	**Responsable**	**Actividad N°**	**Tarea**	**Tiempo estimado (Horas)**
H-02	Login de usuario	Programador 1	H-02-1	Crear pantalla Login	9
		Programador 2	H-02-2	Consultar a la base de datos la existencia de usuario	7
		Programador 3	H-02-3	Mostrar mensajes de confirmación o error	8
		Programador 1	H-02-4	En caso de logearse correctamente direccionar a la página de inicio	9
		Programador 2	H-02-5	Crear página de inicio que muestre la misión, visión, valores y objetivos institucionales.	12
		Programador 3	H-02-6	Crear un mecanismo de navegación de acuerdo a los roles de usuarios.	13
H-03	Registrar los objetivos operativos de las unidades administrativas	Programador 1	H-03-1	Crear la pantalla objetivos operativos	16
		Programador 2	H-03-2	Registrar los objetivos del plan estratégico institucional de Yachay Tech (PEIYT).	16
		Programador 3	H-03-3	Registrar las estrategias del (PEIYT).	14
		Programador 1	H-03-4	Registrar el objetivo operativo de la unidad administrativa	12
		Programador 2	H-03-5	Registrar el año de gestión	6
		Programador 3	H-03-6	Registrar observaciones	6
		Programador 1	H-03-7	Guardar formulario	16

		Programador 2	H-03-8	Funcionalidad que permita editar los objetivos operativos registrados	16

Tabla 26. Sprint 1
Fuente: El Autor

5.2.4.2 Seguimiento y desarrollo del Sprint

Reunión diaria

En el desarrollo del segundo Sprint, las reuniones se evidenciaban posibles obstáculos en el desarrollo los cuales debían ser gestionados por el Scrum Master. Algunos de los inconvenientes que se presentó fue el acceso a datos de servidores remotos; la gestión del Scrum Master fue muy adecuada para dar solución a estos inconvenientes.

DAILY MEETING REGISTER						
Nro.	**Meeting Date**	**Developer**	**What did you do yesterday?**	**What will you do today?**	**Are there impediments in your work?**	**observations**
1	27/03/2018	Programador 1	Crear pantalla login	Crear pantalla login	none	
2	27/03/2018	Programador 2	Consultar a la base de datos la existencia de usuario	Consultar a la base de datos la existencia de usuario	No se puede acceder al servidor de Base de Datos	Solventar este inconveniente con analista de infraestructura
3	27/03/2018	Programador 3	Mostrar mensajes de confirmación o error	Mostrar mensajes de confirmación o error	none	
4	28/03/2018	Programador 1	En caso de logearse correctamente direccionar a la página de inicio	En caso de logearse correctamente direccionar a la página de inicio	none	
5	28/03/2018	Programador 2	Crear página de inicio que muestre la misión, visión, valores y objetivos institucionales.	Crear página de inicio que muestre la misión, visión, valores y objetivos institucionales.	Solicitar a la dependencia encarga de establecer la misión, visión, valores y objetivos de la Institución.	Solicitar a la Dirección de Planificación y Estrategia esta información
6	28/03/2018	Programador 3	Crear un mecanismo de navegación de acuerdo a los roles de usuarios.	Crear un mecanismo de navegación de acuerdo a los roles de usuarios.	none	

Tabla 27. Registro de Reuniones diarias
Fuente: El Autor

En el Sprint Backlog para esta iteración mostraba la evolución que se observa en la matriz siguiente:

SPRINT 1							
Identificador (ID) de la historia	**Característica / Funcionalidad**	**Responsable**	**Actividad N°**	**Tarea**	**Tiempo estimado (Horas)**	**Tiempo Real**	**Estado**
H-02	Login de usuario	Programador 1	H-02-1	Crear pantalla login	9	6	TO DO
		Programador 2	H-02-2	Consultar a la base de datos la existencia de usuario	7	4	DONE
		Programador 3	H-02-3	Mostrar mensajes de	8	8	DONE

				confirmación o error			
		Programador 1	H-02-4	En caso de logearse correctamente direccionar a la página de inicio	9	11	DONE
		Programador 2	H-02-5	Crear página de inicio que muestre la misión, visión, valores y objetivos institucionales.	12	5	DONE
		Programador 3	H-02-6	Crear un mecanismo de navegación de acuerdo a los roles de usuarios.	13	11	TO DO
H-03	Registrar los objetivos operativos de las unidades administrativas	Programador 1	H-03-1	Crear la pantalla objetivos operativos	16	12	IN PROGRESS
		Programador 2	H-03-2	Registrar los objetivos del plan estratégico institucional de Yachay Tech (PEIYT).	16	12	IN PROGRESS
		Programador 3	H-03-3	Registrar las estrategias del (PEIYT).	14	15	IN PROGRESS
		Programador 1	H-03-4	Registrar el objetivo operativo de la unidad administrativa	12	12	IN PROGRESS
		Programador 2	H-03-5	Registrar el año de gestión	6	9	IN PROGRESS
		Programador 3	H-03-6	Registrar observaciones	6	12	TO DO
		Programador 1	H-03-7	Guardar formulario	16	16	TO DO
		Programador 2	H-03-8	Funcionalidad que permita editar los objetivos operativos registrados	16	9	TO DO

Tabla 28. Seguimiento Sprint Backlog 1
Fuente: El Autor

En el tablero Kanban representado en la figura inferior se puede observar el cumplimiento de las tareas mientras avanza proyecto.

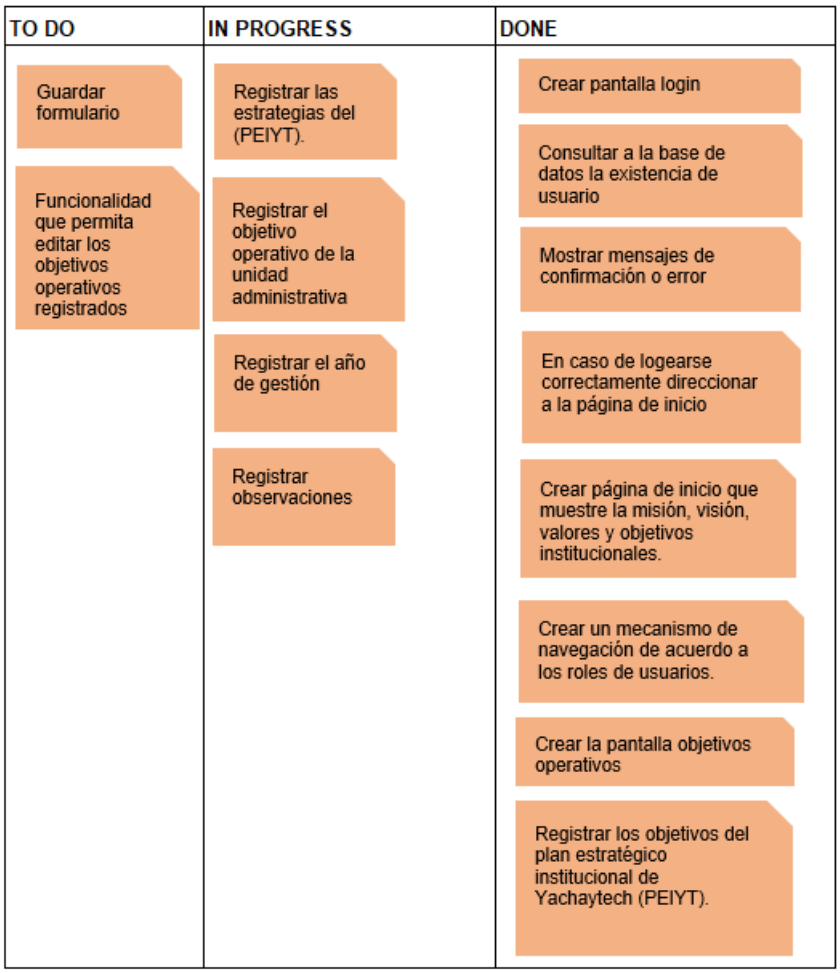

Figura 15. Kanban de estado de avance de tareas.
Fuente: El Autor

El Burndown Chart

La medición de la cantidad de trabajo que restaba para terminar el Sprint, fue evolucionando como se observa en la figura 14 en la cual se evidencia una leve tendencia a retrasarse en trabajo durante todo el Sprint, pero al final de este se puede ver una marcada recuperación del tiempo.

ID Tareas	Estimado(horas)	Dia 15	Dia 14	Dia 13	Dia 12	Dia 11	Dia 10	Dia 9	Dia 8	Dia 7	Dia 6	Dia 5	Dia 4	Dia 3	Dia 2	Dia 1	Total de Horas reales
H-02-1	9	0	0	0	1	0	0	2	2	1	0	0	0	1	1	1	9
H-02-2	7	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	4
H-02-3	8	0	0	0	0	1	1	2	2	0	0	0	0	1	1	1	9
H-02-4	9	1	0	1	0	1	2	0	0	1	2	0	1	1	0	1	11
H-02-5	12	0	0	1	0	0	0	0	0	0	0	1	1	1	2	2	8
H-02-6	13	1	1	0	0	1	1	1	1	0	1	1	2	0	2	2	14
H-03-1	16	0	1	1	1	0	2	0	2	0	1	2	1	0	0	1	12
H-03-2	16	0	1	0	0	0	0	0	2	0	2	2	2	1	1	1	12
H-03-3	14	1	1	0	1	0	1	2	2	1	0	2	1	2	1	0	15
H-03-4	12	0	0	0	2	0	1	0	1	1	1	2	1	2	1	0	12
H-03-5	6	1	1	0	0	1	0	0	1	0	1	0	1	1	2	0	9
H-03-6	6	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	8
H-03-7	16	1	0	1	1	0	2	3	0	1	2	1	2	1	1	0	16
H-03-8	16	1	1	1	1	0	1	1	0	0	0	1	1	2	1	0	11
Horas reales	160	153	146	140	132	127	114	103	90	84	73	60	46	31	16	6	150
Horas estimadas restantes	160	149,33	138,67	128,00	117,33	106,67	96,00	85,33	74,67	64,00	53,33	42,67	32,00	21,33	10,67	0,00	

Tabla 29. Matriz de control Sprint 1
Fuente: El Autor.

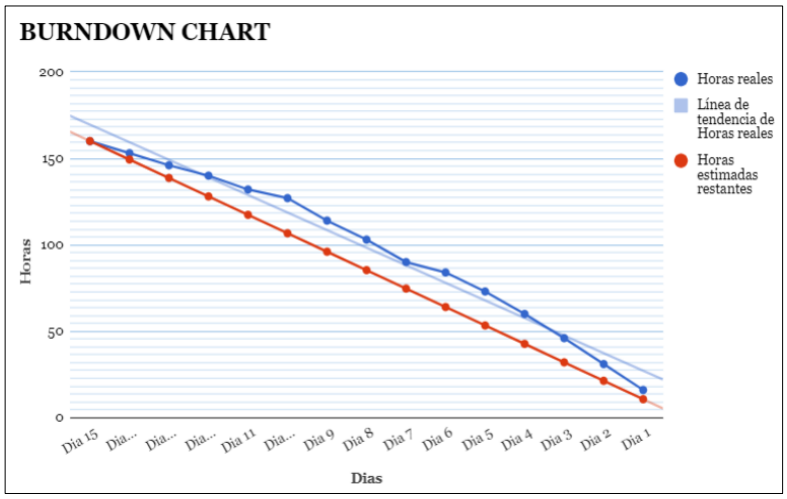

Figura 16. Gráfica Burndown Chart Sprint 1
Fuente: El Autor

La duración del Sprint nunca se prolongó más allá del tiempo establecido, todas las tareas programadas se terminaron en la fecha establecida para la revisión del Sprint.

5.2.4.3 Reunión de revisión de Sprint

Incremento del producto

Una vez que se terminó el Sprint se hizo la revisión y presentación del avance. El Scrum master y el Product Owner revisaron el entregable del producto, los demos y plantillas. Este Sprint desarrollaba los requerimientos funcionales iniciales. Se presentó todas las funcionalidades planificadas en el Sprint, el cliente muestra satisfacción y optimismo.

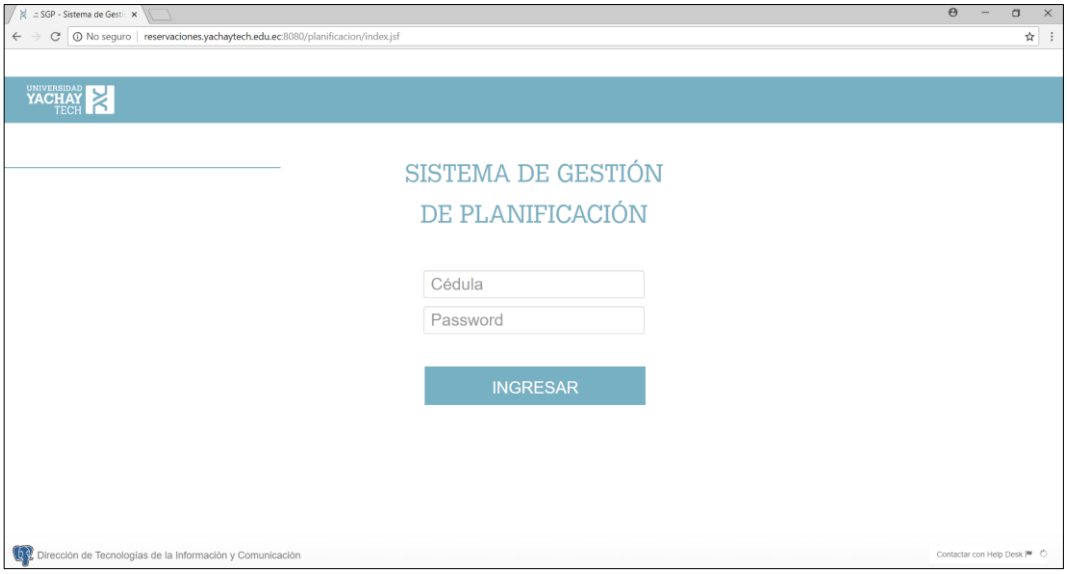

Figura 17. Pantalla de Login
Fuente: El Autor

5.2.4.4 Reunión de Retrospectiva

Las conclusiones y recomendaciones del segundo Sprint se encuentran resumidas a continuación:

REUNIÓN DE RETROSPECTIVA			
POSITIVO	**MEJORABLE**	**SOLUCIONES**	**IDEAS**
Estimaciones cumplidas			
Especificación de requerimientos claros	Tomar en forma detalla los de requisitos durante las reuniones de planificación	Incluir en las reuniones de panificación a los clientes que dan operatividad a los procesos para que ellos desde su experiencia aporten en la toma de requisitos	
Buena comunicación entre el Equipo de desarrollo	Mejorar la comunicación con el cliente para optimizar la toma de requerimientos		
Cantidad de bugs	Reducir la cantidad de bugs	Optimizar el proceso de codificación	
	Mejorar la velocidad de desarrollo.		

Tabla 30. Formulario de observaciones Reunión Retrospectiva
Fuente: El Autor

5.2.5 Sprint 2

5.2.5.1 Planificación

El *Sprint Backlog 2* detalla las tareas de las historias de usuarios 4 y 5, se tomaron en cuenta las observaciones del Sprint anterior.

En la matriz inferior se muestra el Sprint Backlog para la tercera iteración.

SPRINT 2					
Identificador (ID) de la historia	**Característica / Funcionalidad**	**Responsable**	**Actividad N°**	**Tarea**	**Tiempo estimado (Horas)**
H-04	Registrar las actividades o solicitudes de compra de las unidades administrativas	Programador 1	H-04-1	Registrar la descripción de la actividad	6
		Programador 2	H-04-2	Registrar el presupuesto	6
		Programador 3	H-04-3	Registrar la categoría	4
		Programador 1	H-04-4	Registrar el objetivo operativo	4
		Programador 2	H-04-5	Registrar el año fiscal	6
		Programador 3	H-04-6	Registrar tipo de gasto si es corriente o inversión	4
		Programador 1	H-04-7	Registrar el programa de ejecución presupuestaria	4
		Programador 2	H-04-8	Registrar el monto en el cuatrimestre de ejecución	4
		Programador 3	H-04-9	Registrar si el aval es anual o plurianual	4
		Programador 1	H-04-10	Guardar formulario	12
		Programador 2	H-04-11	Funcionalidad para editar la actividad	12
		Programador 3	H-04-12	Funcionalidad para eliminar la actividad	12
		Programador 1	H-04-13	Funcionalidad para ver las actividades ingresadas	8
H-05	Realizar la solicitud de aval	Programador 1	H-05-01	Funcionalidad para asignar el financiamiento a la actividad desde el usuario Planificación	16
		Programador 2	H-05-02	Funcionalidad para ver las actividades financiadas desde el usuario Planificación	12
		Programador 3	H-05-03	Funcionalidad para ver las actividades rechazadas desde el usuario Planificación	12

		Programador 1	H-05-04	Funcionalidad para registrar aval	20
		Programador 2	H-05-05	Funcionalidad para ver avales registrados	12
		Programador 3	H-05-06	Funcionalidad para guardar aval desde unidad	18
		Programador 1	H-05-07	Funcionalidad para solicitar aval desde unidad	20
		Programador 2	H-05-08	Funcionalidad para cargar anexos desde pantalla solicitar aval	20
		Programador 3	H-05-09	Funcionalidad para firmar electrónicamente desde pantalla solicitar aval	20
		Programador 1	H-05-10	Funcionalidad para cancelar aval desde unidad	18
		Programador 2	H-05-11	Generar en el memorándum de solicitud de aval en formato pdf.	24
		Programador 3	H-05-12	Diseñar plantilla para gestión de avales y reformas desde el usuario PLANIFICACIÓN	12
		Programador 1	H-05-13	Funcionalidad para devolver aval a unidad	16
		Programador 2	H-05-14	Funcionalidad para guardar y modificar aval	16
		Programador 3	H-05-15	Funcionalidad para aprobar solicitud de aval	18
		Programador 1	H-05-16	Funcionalidad para ver avales aprobados	12

Tabla 31. Sprint 2
Fuente: El Autor

5.2.5.2 Seguimiento y desarrollo del Sprint

Reunión diaria

En las reuniones diarias del tercer Sprint, el equipo evidencia mayores dificultades para desarrollar el producto; esto se debe sustancialmente a la complejidad de la lógica del negocio. Los posibles obstáculos en el

desarrollo son gestionados y solventados por el Scrum Master en conjunto con el Product Owner.

DAILY MEETING REGISTER						
Nro.	**Meeting Date**	**Developer**	**What did you do yesterday?**	**What will you do today?**	**Are there impediments in your work?**	**observations**
1	20/03/2018	Programador 1	Registrar la descripción de la actividad	Registrar la descripción de la actividad	none	
2	20/03/2018	Programador 2	Registrar el presupuesto	Registrar el presupuesto	none	
3	20/03/2018	Programador 3	Registrar la categoría	Registrar la categoría	none	
4	21/03/2018	Programador 1	Registrar el objetivo operativo	Registrar el objetivo operativo	none	
5	21/03/2018	Programador 2	Registrar el año fiscal	Registrar el año fiscal	none	
6	21/03/2018	Programador 3	Registrar tipo de gasto si es corriente o inversión	Registrar tipo de gasto si es corriente o inversión	none	
7	22/03/2018	Programador 1	Registrar el programa de ejecución presupuestaria	Registrar el programa de ejecución presupuestaria	none	
8	22/03/2018	Programador 2	Registrar el monto en el cuatrimestre de ejecución	Registrar el monto en el cuatrimestre de ejecución	none	
9	22/03/2018	Programador 3	Registrar si el aval es anual o plurianual	Registrar si el aval es anual o plurianual	none	
10	23/03/2018	Programador 1	Guardar formulario	Guardar formulario	none	
11	23/03/2018	Programador 2	Funcionalidad para editar la actividad	Funcionalidad para editar la actividad	none	
12	23/03/2018	Programador 3	Funcionalidad para eliminar la actividad	Funcionalidad para eliminar la actividad	none	
13	26/03/2018	Programador 1	Funcionalidad para ver las actividades ingresadas	Funcionalidad para ver las actividades ingresadas	none	
14	26/03/2018	Programador 2	Funcionalidad para asignar el financiamiento a la actividad desde el usuario Planificación	Funcionalidad para asignar el financiamiento a la actividad desde el usuario Planificación	El proceso está poco claro	
15	26/03/2018	Programador 3	Funcionalidad para ver las actividades financiadas desde el usuario Planificación	Funcionalidad para ver las actividades financiadas desde el usuario Planificación	Se desconoce las características para la funcionalidades del usuario PLANIFICACIÓN	
16	27/03/2018	Programador 1	Funcionalidad para ver las actividades rechazadas desde el usuario Planificación	Funcionalidad para ver las actividades rechazadas desde el usuario Planificación	none	
17	27/03/2018	Programador 2	Funcionalidad para registrar aval	Funcionalidad para registrar aval	Desconocimiento de usuarios involucrados en este proceso	
18	27/03/2018	Programador 3	Funcionalidad para ver avales registrados	Funcionalidad para ver avales registrados		
19	28/03/2018	Programador 2	Funcionalidad para guardar aval desde unidad	Funcionalidad para guardar aval desde unidad	Desconocimiento de montos para los autorizadores de gasto	

Tabla 32. Registro de reunión diaria
Fuente: El Autor

En la tabla inferior se puede observar el Sprint Backlog de seguimiento para la segunda iteración.

SPRINT 2							
Identificador (ID) de la historia	**Característica / Funcionalidad**	**Responsable**	**Actividad N°**	**Tarea**	**Tiempo estimado (Horas)**	**Tiempo Real**	**Estado**
H-04	Registrar las actividades o solicitudes de compra de las unidades administrativas	Programador 1	H-04-1	Registrar la descripción de la actividad	6	6	DONE
		Programador 2	H-04-2	Registrar el presupuesto	6	6	DONE
		Programador 3	H-04-3	Registrar la categoría	4	6	DONE
		Programador 1	H-04-4	Registrar el objetivo operativo	4	5	DONE
		Programador 2	H-04-5	Registrar el año fiscal	6	7	DONE
		Programador 3	H-04-6	Registrar tipo de gasto si es corriente o inversión	4	4	IN PROGRESS
		Programador 1	H-04-7	Registrar el programa de ejecución presupuestaria	4	5	IN PROGRESS
		Programador 2	H-04-8	Registrar el monto en el cuatrimestre de ejecución	4	6	DONE
		Programador 3	H-04-9	Registrar si el aval es anual o plurianual	4	4	IN PROGRESS
		Programador 1	H-04-10	Guardar formulario	12	12	IN PROGRESS
		Programador 2	H-04-11	Funcionalidad para editar la actividad	12		IN PROGRESS
		Programador 3	H-04-12	Funcionalidad para eliminar la actividad	12		IN PROGRESS
		Programador 1	H-04-13	Funcionalidad para ver las actividades ingresadas	8		IN PROGRESS
H-05	Realizar la solicitud de aval	Programador 1	H-05-01	Funcionalidad para asignar el financiamiento a la actividad desde el usuario Planificación	16		IN PROGRESS
		Programador 2	H-05-02	Funcionalidad para ver las actividades financiadas desde el usuario Planificación	12		IN PROGRESS
		Programador 3	H-05-03	Funcionalidad para ver las actividades rechazadas desde el usuario Planificación	12		IN PROGRESS
		Programador 1	H-05-04	Funcionalidad para registrar aval	20		TO DO
		Programador 2	H-05-05	Funcionalidad para ver avales registrados	12		TO DO
		Programador 3	H-05-06	Funcionalidad para guardar aval desde unidad	18		TO DO
		Programador 1	H-05-07	Funcionalidad para solicitar aval desde unidad	20		TO DO
		Programador 2	H-05-08	Funcionalidad para cargar anexos desde pantalla solicitar aval	20		TO DO
		Programador 3	H-05-09	Funcionalidad para firmar electrónicamente desde pantalla solicitar aval	20		TO DO
		Programador 1	H-05-10	Funcionalidad para cancelar aval desde unidad	18		TO DO
		Programador 2	H-05-11	Generar en el memorándum de solicitud de aval en formato pdf.	24		TO DO
		Programador 3	H-05-12	Diseñar plantilla para gestión de avales y reformas desde el usuario PLANIFICACIÓN	12		TO DO
		Programador 1	H-05-13	Funcionalidad para devolver aval a unidad	16		TO DO

		Programador 2	H-05-14	Funcionalidad para guardar y modificar aval	16		TO DO
		Programador 3	H-05-15	Funcionalidad para aprobar solicitud de aval	18		TO DO
		Programador 1	H-05-16	Funcionalidad para ver avales aprobados	12		TO DO

Tabla 33. Seguimiento Sprint Backlog 2
Fuente: El Autor

En el Kanban representado en la figura inferior se muestra el estado de las tareas mientras avanza el Sprint.

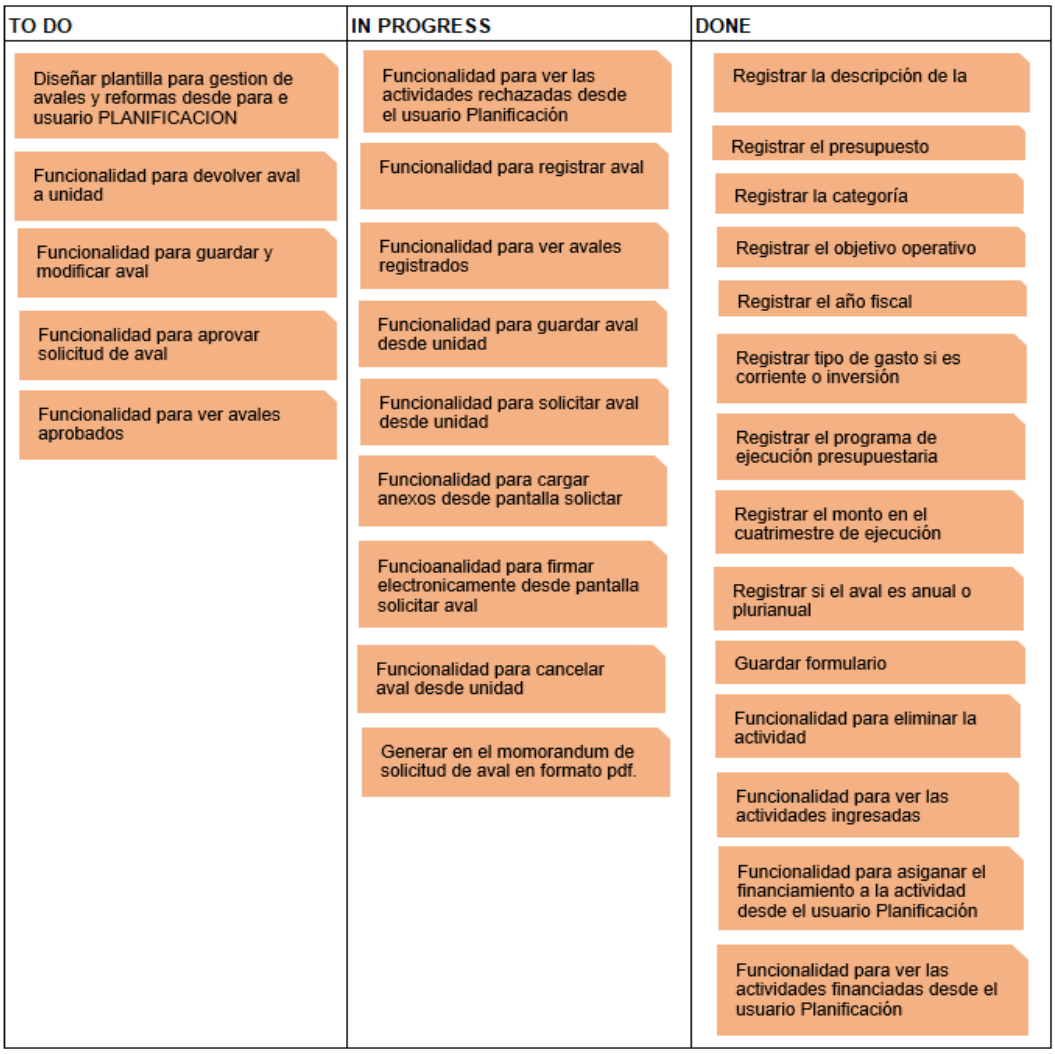

Figura 18. Kanban de estado de avance de las tareas
Fuente: El Autor

El Burndown Chart

En la gráfica *Burndown chart* se observa al inicio una marcada tendencia a retrasarse en el cumplimiento de las tareas asignadas, pero con la gestión

adecuada del Scrum Master y demás involucrados en el proyecto se puede ver la recuperación del tiempo al final del Sprint.

ID Tareas	Estimado(horas)	Dia 15	Dia 14	Dia 13	Dia 12	Dia 11	Dia 10	Dia 9	Dia 8	Dia 7	Dia 6	Dia 5	Dia 4	Dia 3	Dia 2	Dia 1	Total de Horas reales
H-04-1	6	1	1	0	1	0	0	0	0	1	0	1	0	1	0	0	6
H-04-2	6	0	0	1	0	0	1	0	0	1	0	1	0	1	0	0	5
H-04-3	4	1	1	0	0	0	0	0	0	1	0	1	1	0	0	0	5
H-04-4	4	0	1	0	1	1	1	0	0	1	1	1	0	1	0	0	8
H-04-5	6	0	2	1	1	2	1	0	0	0	1	0	1	0	0	0	9
H-04-6	4	0	1	0	1	1	0	0	0	2	1	1	0	0	0	0	7
H-04-7	4	0	1	0	1	2	0	0	0	0	1	1	0	0	0	0	6
H-04-8	4	1	1	1	1	1	0	0	1	0	0	1	1	0	0	0	8
H-04-9	4	2	1	0	1	0	0	0	0	1	0	0	1	0	0	0	6
H-04-10	12	0	0	1	2	2	0	1	0	0	1	0	0	2	2	0	11
H-04-11	12	2	0	0	1	1	0	0	0	2	0	0	1	2	1	1	11
H-04-12	12	1	0	0	0	0	1	0	1	1	1	0	1	2	2	2	12
H-04-13	8	1	0	1	0	0	1	0	0	1	1	0	2	1	1	0	9
H-05-01	16	1	0	0	1	0	1	0	1	0	0	0	2	2	2	3	13
H-05-02	12	0	0	0	0	0	1	0	1	0	2	0	2	3	1	2	12
H-05-03	12	2	0	0	0	1	1	0	0	0	2	0	2	2	0	3	13
H-05-04	20	0	0	1	0	1	1	2	0	0	2	0	2	3	3	3	18
H-05-05	12	2	0	0	1	1	1	0	1	0	2	0	2	1	1	0	12
H-05-06	18	1	0	1	1	0	0	3	0	2	2	1	1	3	1	2	18
H-05-07	20	2	0	0	2	1	1	1	1	2	1	1	1	3	2	1	19
H-05-08	20	1	0	1	2	2	0	1	0	2	1	2	1	3	2	1	19
H-05-09	20	1	0	0	2	1	2	1	0	1	1	2	2	2	2	2	19
H-05-10	18	1	0	1	1	0	1	1	1	1	1	2	1	2	2	3	18
H-05-11	24	0	0	2	1	1	2	0	0	0	1	2	1	3	3	3	19
H-05-12	12	0	0	2	1	0	1	0	0	0	0	1	1	2	3	3	14
H-05-13	16	0	0	2	0	1	1	0	0	3	1	2	1	2	3	3	19
H-05-14	16	0	0	0	0	0	0	0	1	1	1	2	1	2	0	2	10
H-05-15	18	0	0	2	0	0	0	0	1	0	1	2	2		3	1	12
H-05-16	12	0	0	0	2	0	1	0	1	1	1	2	1	2	3	0	14
Horas reales	352	332	323	306	282	263	244	234	224	200	174	148	117	72	35	0	352
Horas estimadas restantes	352	328,53	305,07	281,60	258,13	234,67	211,20	187,73	164,27	140,80	117,33	93,87	70,40	46,93	23,47	0,00	

Tabla 34. Matriz de control Sprint 2.
Fuente: El Autor

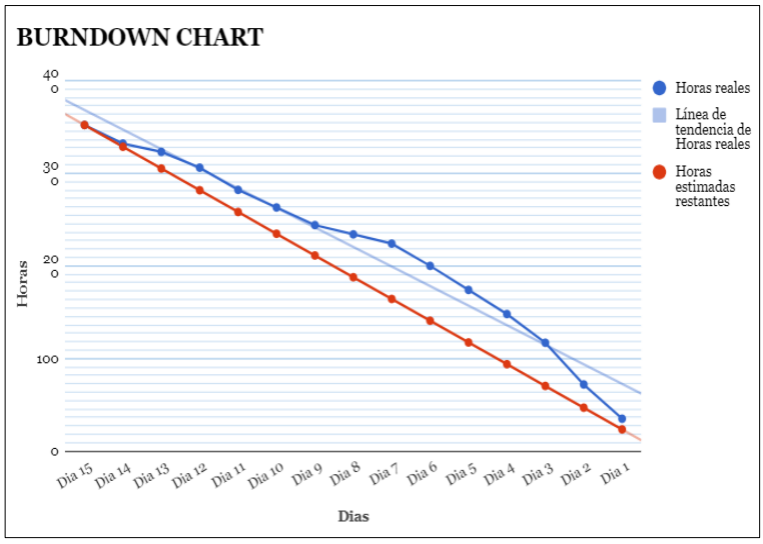

Figura 19. Burndown Chart Sprint 2
Fuente: El Autor

5.2.5.3 Reunión de revisión del Sprint

Incremento del producto

El Scrum master y el Product Owner revisaron los nuevos avances del producto. En este Sprint se desarrolló funcionalidades complejas por lo que la revisión se la realizó de manera minuciosa, al final el cliente validó y aceptó de manera favorable esta iteración.

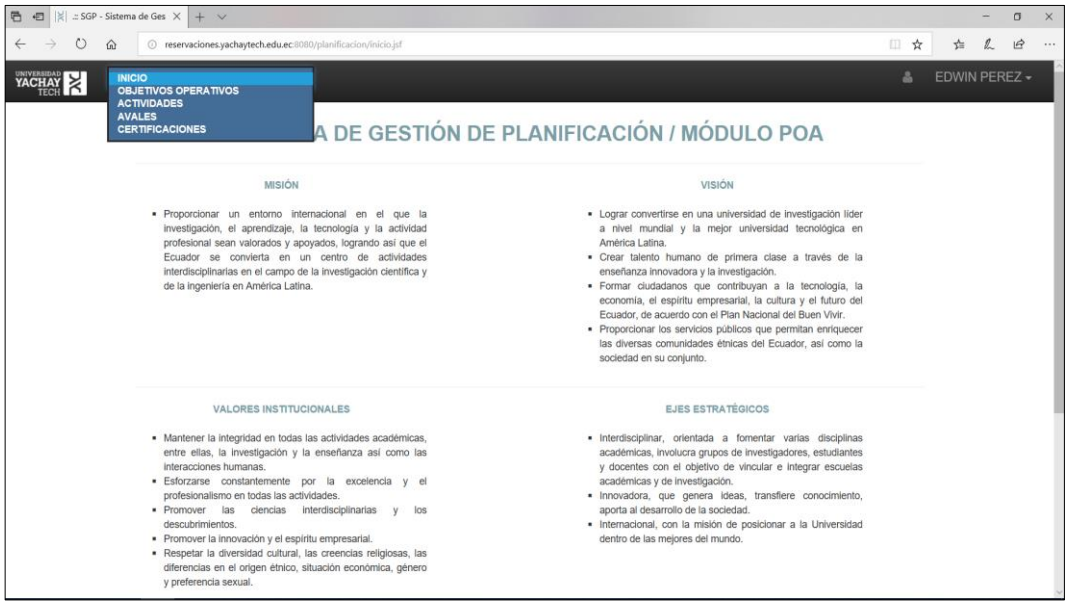

Figura 20. Plantilla estándar del Proyecto.
Fuente: El Autor

5.2.5.4 Reunión de Retrospectiva

Las conclusiones y recomendaciones del tercer Sprint se encuentran resumidas a continuación:

REUNIÓN DE RETROSPECTIVA			
POSITIVO	**MEJORABLE**	**SOLUCIONES**	**IDEAS**
Estimaciones cumplidas			
Funcionalidades de usuarios involucrados en el proceso	Establecer funcionalidades para el usuario PLANIFICACIÓN		
	Desconocimiento de montos para los autorizadores de gasto.	Establecer montos para los autorizadores de gasto	
	Intranet presenta fallas de conectividad.	Revisión de conectividad en los entornos de desarrollo y producción	

Tabla 35. Reunión de retrospectiva
Fuente: El Autor

CAPÍTULO VI. PRESENTACIÓN DE RESULTADOS

6 Introducción.

En este capítulo se analizan los factores críticos de éxito que se han observado durante el desarrollo del estudio de caso en el cual se implantó la metodología Scrum para mejorar la gestión de los proyectos de software en la dirección de Tecnologías de la Información de la Universidad Yachay Tech.

La medición de los factores de éxito de la implementación de la metodología se extrajo mediante el instrumento de investigación *encuesta* que se aplicó a los integrantes de la Dirección de Tecnologías de la Información (Ver ***Tabla 2.*** *Distribución de la muestra*).

6.1 Factores críticos de éxito de la implementación

Para identificar los factores de éxito lo realizamos en base a la caracterización de la investigación empírica realizada por Misra, Kumar & Kumar (2009) sobre factores que influyen en la adopción de metodologías ágiles. Los autores realizaron un estudio en diferentes empresas alrededor del mundo para validar las hipótesis sobre aquellos factores más influyentes en el éxito de implementaciones de proyectos de desarrollo de software con metodologías ágiles. En este caso se recolectó información sobre la experiencia obtenida en el estudio de caso y se realizó un proceso de validación sobre los factores de éxito planteados en la investigación.

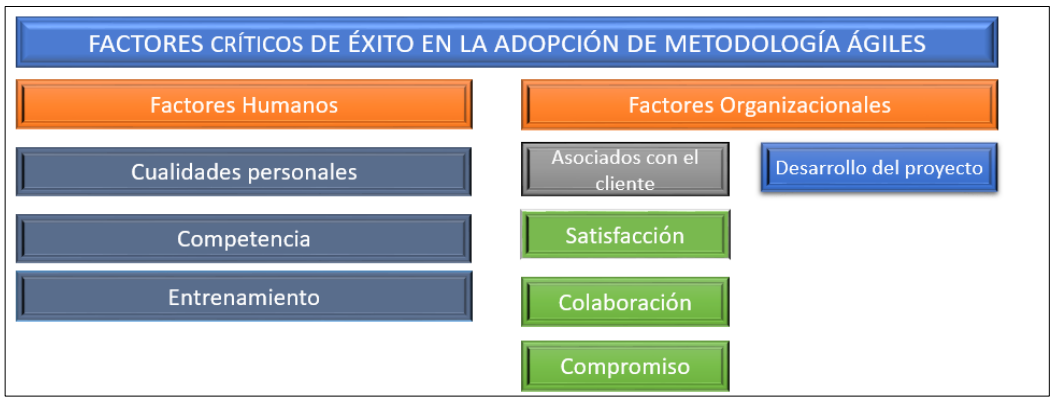

Figura 21. Factores críticos de éxito planteados por Misra, Kumar & Kumar *(2009)*
Fuente: (Misra, Kumar, & Kumar, 2009)

6.1.1 Definición de escalas de medición.

La medición de los factores de éxito se utilizó el instrumento de investigación *Encuesta*, para más objetividad en la evaluación las preguntas fueron cerradas, estas recogieron las consideraciones establecidas por Misra y Kumar.

Los criterios de evaluación de estos factores se basaron en las escalas de Likert. (García Sanchéz, Aguilera, & Castillo Rosas, 2011); se asignó un peso a cada ítem, en cada pregunta el entrevistado marcó con una (x) la repuesta deseada y se sumó los valores por cada opción de respuesta; el resultado representa el porcentaje de acuerdo, satisfacción y conformidad.

En la ***Tabla 36*** se describe la especificación de escalas de medición para la evaluación de factores de éxito.

Ítems de Evaluación(Factores humanos)	Ítems de Evaluación(Factores asociados al cliente)	Ítems de Evaluación(Nivel de adopción de la metodología Scrum)	Peso
Totalmente de acuerdo	Muy satisfactorio	Alto	5
De acuerdo	Satisfactorio	Medio	4
Ni de acuerdo ni en desacuerdo	Medianamente satisfactorio	Bajo	3
En desacuerdo	Poco satisfactorio	No se usa	2
Totalmente en desacuerdo	Nada satisfactorio	No sabe	1

Tabla 36. Especificación de las escalas de medición
Fuente: (García Sanchéz, Aguilera, & Castillo Rosas, 2011)

6.2 Resultados de la encuesta

6.2.1 Factores Humanos

A nivel personal

Misra, Kumar & Kumar (2009) manifiestan que la experiencia en agilismo y competencias laborales de los integrantes del grupo es importante, pero un aspecto más determinante es el factor humano, es decir sus cualidades y valores; todos los individuos tienen personalidades diferentes, comparten valores como honestidad, trabajo en equipo, interés en aprender y espíritu emprendedor, que en conjunto potencializan al grupo y ayudan con su

desempeño; las preguntas de las encuestas se realizaron en base a las consideraciones del estudio mencionado.

En la **Tabla 37** se muestra los resultados obtenidos después de la tabulación de las encuestas (formato de encuesta Anexo 6) aplicadas a la población sujeta al estudio.

Factores Humanos	Evaluación				
	Totalmente de acuerdo	De acuerdo	Ni de acuerdo ni en desacuerdo	En desacuerdo	Totalmente en desacuerdo
¿Considera que su experiencia previa en metodologías ágiles influyó positivamente en la implementación del estudio de caso?	15	4	3	0	0
¿Su experiencia previa con el agilismo en esta implementación fue favorable?	10	8	3	0	0
¿Considera que los valores como la honestidad, trabajo en equipo, sentido de responsabilidad, interés en aprender y espíritu emprendedor contribuyeron positivamente en la implementación de la metodología?	10	8	0	2	0
¿Considera que su adaptación a las prácticas Scrum establecidas por la metodología fueron positivas?	15	8	0	0	0

Tabla 37. Matriz de tabulación de resultados de la encuesta *Factores Humanos.*
Fuente: (Misra, Kumar, & Kumar, 2009).

Los entrevistados coinciden en que la experiencia previa es importante, así como también la actitud, el compromiso, las ganas de involucrarse con el agilismo y aplicarlos día a día en el trascurso de la ejecución del proyecto. El comportamiento personal de cada miembro del equipo es un factor importante al momento de conformar equipos de trabajo, de su interacción y comodidad con el resto de integrantes también dependerá su rendimiento profesional; en este sentido la evaluación arrojó las siguientes consideraciones:

Figura 22. Experiencia previa en metodologías ágiles
Fuente: El Autor

Figura 23. Experiencia con el agilismo
Fuente: El Autor

Los entrevistados indican estar *totalmente de acuerdo* que la experiencia previa influye positivamente en la implantación, la evalúan con un 68,2%; con respecto a la opción *de acuerdo* la asignan un 18,2%. El 13,6% es valoración a que la *experiencia es indiferente* al momento de la implementación.

Con respecto al agilismo como experiencia personal positiva en la implantación de la metodología se calificó con un 47,6% a lo opción *Totalmente de acuerdo*; el valor para la opción *de acuerdo* fue de 38,1%; la calificación a la opción neutral fue 14,3%.

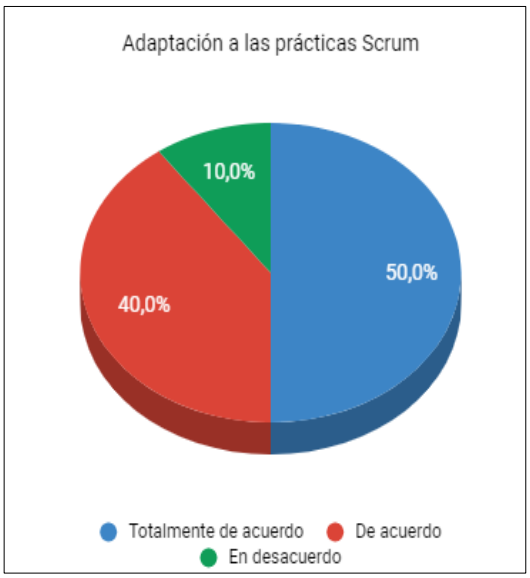

Figura 24. Adaptación a las Prácticas Scrum
Fuente: El Autor

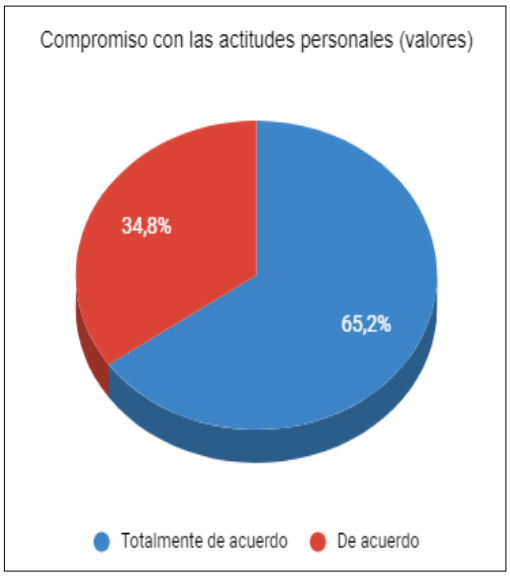

Figura 25. Compromiso con los valores personales.
Fuente: El Autor

Respecto a adaptación a las prácticas Scrum la opción *Totalmente de acuerdo* obtiene una significativa valoración del 50%; la opción *De acuerdo* se valora con el 40%; También se obtiene una calificación del 10% a la opción *En desacuerdo*.

Respecto al compromiso con los valores personales la evaluación que el equipo determinó fue muy significativa, en los ítems positivos su valoración fue de 65,2 y 34,8 por ciento respectivamente.

6.2.2 Factores Organizacionales

Factores asociados con el cliente

Los factores asociados con el cliente según Misra, Kumar & Kumar (2009) son satisfacción del cliente, colaboración del cliente, y compromiso del cliente.

La satisfacción del cliente es la prioridad dentro del proceso de desarrollo y esto se logra a través de la entrega de incrementos de software tempranos y continuos (agilemanifesto.org, 2018). Los clientes deben estar motivados, comprometidos, dispuestos y activos no solo con ellos mismos sino también con el equipo de desarrollo, para esto se necesita una relación de confianza mutua.

La encuesta (formato encuesta Anexo 7) estuvo orientada a determinar el grado de satisfacción del cliente en torno al proceso de desarrollo.

Matriz de resultados de la evaluación

Factores asociados con el cliente	Evaluación				
	Muy satisfactorio	Satisfactorio	Medianamente satisfactorio	Poco satisfactorio	Nada satisfactorio
Qué grado de satisfacción merece la comunicación con el cliente?	15	8	0	0	0
Qué grado de satisfacción merecen los procesos de negociación con el cliente?	15	4	3	0	0
Cuál es el grado de satisfacción del cliente respecto a la colaboración en el proceso de desarrollo del software?	5	12	2	0	0
Qué nivel de satisfacción percibió el cliente ante la utilización de las	15	8	0	0	0

metodologías ágiles en el desarrollo del estudio de caso?					
Qué grado de satisfacción mostró el cliente cada vez que se presentaba los incrementos del software en cada Sprint?	15	4	3	0	0

Tabla 38. Matriz de tabulación de resultados de la encuesta *Factores asociados con el cliente*.
Fuente: *(Misra, Kumar, & Kumar, 2009)*.

Los resultados de las encuestas las representamos gráficamente a continuación.

Figura 26. Comunicación con el cliente
Fuente: El Autor

Figura 27. Proceso de negociación
Fuente: El Autor

Como se observa en la figura 27 la comunicación del equipo de desarrollo con el cliente es muy buena, se valoró como *Muy satisfactorio* en un 65,2% y *satisfactorio* en un 34,8%.

Con respecto a la negociación con el cliente los resultados son muy favorables, se valoró como Muy satisfactorio en 68,2%, como satisfactorio en un 18,2% y tan solo con un 13,6% como *medianamente satisfactoria.*

Figura 28. Colaboración de cliente
Fuente: El Autor

Figura 29. Utilización de metodologías ágiles
Fuente: El Autor

La colaboración del cliente con el equipo de desarrollo es muy buena, se valoró como *Muy satisfactorio* en un 63,2% y *satisfactorio* en un 26,3%.

Con respecto al nivel de satisfacción percibido por el cliente ante la utilización de metodologías ágiles en el desarrollo, los resultados son muy favorables, se valoró como *Muy satisfactorio* con un 65,2%, como *Satisfactorio* con un 34,8.

Figura 30. Entregas Incrementales del software
Fuente: El autor

Otro aspecto importante que valora el cliente son las entregas de forma incremental del software, valoró como *Muy satisfactorio* en un 68,2% y *Satisfactorio* en 18,2%.

Factores asociados con el desarrollo del proyecto.

Aun cuando las metodologías ágiles son incrementales y se ajustan a los cambios, debe existir mecanismos de seguimiento y control que sirvan como guía durante el desarrollo del proyecto, que es ejecutado y monitoreado por el mismo equipo.

De la evidencia práctica observada en el estudio de caso se han extraído algunos factores asociados al desarrollo del proyecto que están en concordancia con los factores que propone Misra y Kumar (Misra, Kumar, & Kumar, 2009).

Para la evaluación (formato encuesta Anexo 8) de los factores asociados con el desarrollo del proyecto utilizamos de las especificaciones de la ***Tabla 36***.

Factores asociados con el desarrollo del proyecto.	**Evaluación**				
	Totalmente de acuerdo	De acuerdo	Ni de acuerdo ni en desacuerdo	En desacuerdo	Totalmente en desacuerdo
Las tareas planificadas para cada sprint se cumplen en el tiempo acordado para la revisión	15	8	0	0	0
El equipo prioriza el desarrollo a medida, obteniendo la funcionalidad indicada con el cliente.	10	8	3	0	0
Durante el desarrollo del proyecto cuando ocurrieron incidentes inesperados, el Scrum Master los solvento de manera adecuada.	20	4	0	0	0
Durante el desarrollo del proyecto se dio prioridad Software funcionando sobre documentación extensiva.	15	8	0	0	0
El seguimiento constante de las actividades hace que se puedan tomar acciones correctivas de forma rápida para encausar los problemas.	15	8	0	0	0

Tabla 39. Matriz de tabulación de resultado de la encuesta *Factores asociados al desarrollo del proyecto*

Fuente: *(Misra, Kumar, & Kumar, 2009)*.

Los resultados obtenidos de las encuestas se interpretan gráficamente a continuación.

Figura 31. Cumplimiento de tareas en el Sprint
Fuente: El Autor

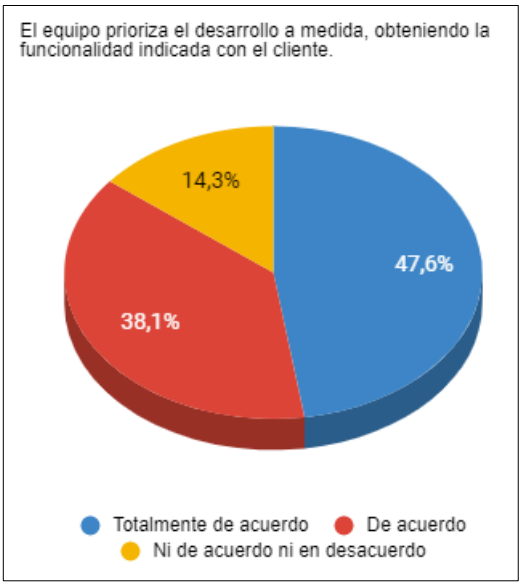

Figura 32. Desarrollo a gusto del cliente
Fuente: El Autor

El cumplimiento de las tareas planificadas para cada Sprint fue altamente efectivo, el equipo valoró estar *Totalmente de acuerdo* en un 65,2% y de *acuerdo* en un 34,8%.

El equipo manifiesta que el proyecto se desarrolló al gusto del cliente; sus integrantes mencionan estar *totalmente de acuerdo* con esta afirmación en un 47.6% y *de acuerdo* en un 38,1%.

Figura 33. Gestión de incidentes
Fuente: El Autor.

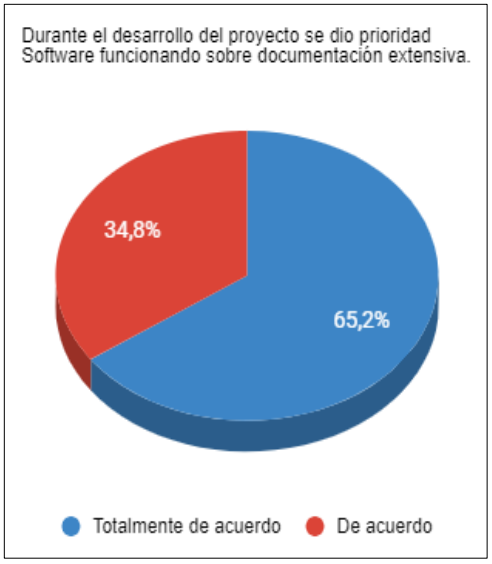

Figura 34. Software funcionando sobre documentación
Fuente: El Autor

Durante el desarrollo del proyecto se puso mucho énfasis en la gestión de incidentes, de hecho, se valoró estar *totalmente de acuerdo* en un 83,3% y de *acuerdo* en un 16,7%.

El Equipo de desarrollo trabajó en función de obtener software funcionando, sobre documentación excesiva; se evaluó estar *totalmente de acuerdo* en un 65,2% y *de acuerdo* en 34,8%.

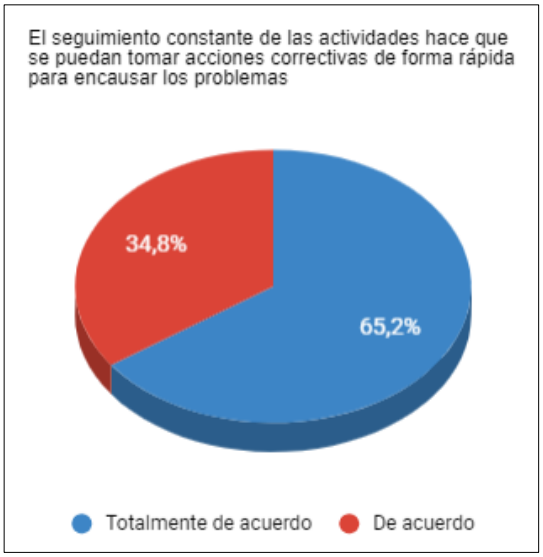

Figura 35. Seguimiento permanente, correcciones rápidas
Fuente: El Autor

Un factor muy importante dentro del proceso de desarrollo fue realizar seguimiento permanente para tomar correcciones a tiempo, se valoró estar *totalmente de acuerdo* en un 65,2% y *de acuerdo* en un 34,8%. Esto demuestra gran apertura a la utilización de metodologías ágiles.

6.2.3 Otros factores

Satisfacción general del equipo

La satisfacción del equipo y el cliente en términos generales, fue positiva. Las percepciones de satisfacción fueron recogidas al finalizar cada sprint (formato encuesta Anexo 9), en la cual los participantes del proyecto evaluaron los resultados presentados en cada reunión de revisión de sprint; otorgando un valor comprendido entre 0 y 10.

La ***Figura 36*** muestra gráficamente el grado de satisfacción a lo largo del proceso de desarrollo.

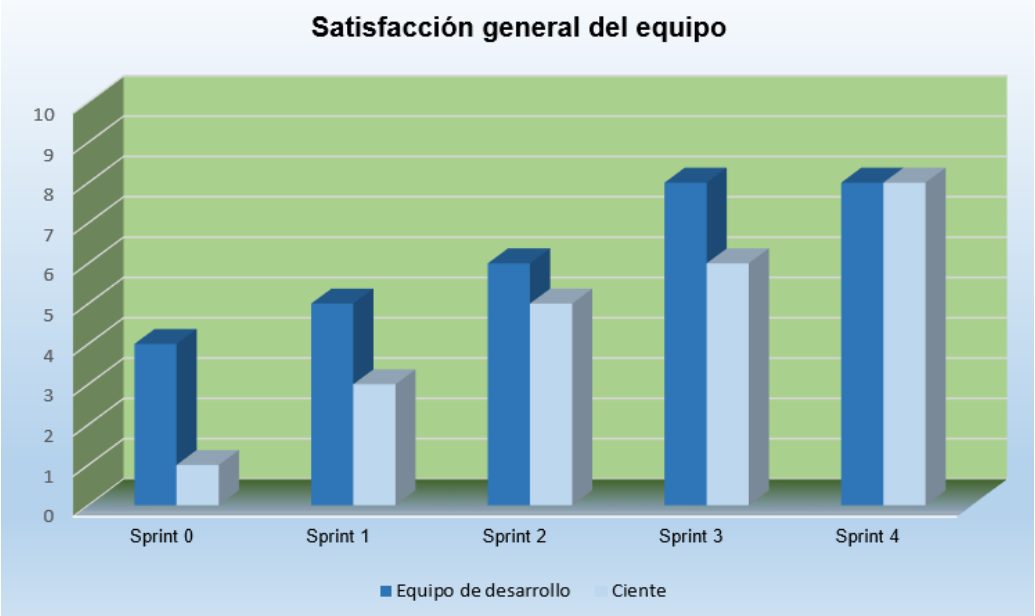

Figura 36. Satisfacción general del equipo.
Fuente: El Autor

Como se observar en el gráfico, la satisfacción del cliente incrementó a medida que avanzaba el desarrollo del proyecto, debido principalmente al cumplimiento de las actividades planificadas para cada sprint. No obstante, aunque dicha satisfacción refleja claramente el agrado del cliente con la calidad del producto obtenido, el cliente siempre se reserva el derecho a la duda por alguna eventualidad que pudiere ocurrir en el proyecto por más que se concluya cubriendo el 100% de los objetivos marcados en un inicio, influye evidentemente, en que su satisfacción no llegué a alcanzar el valor máximo. Por otro, el grado de satisfacción del equipo de desarrollo es sostenido y aumenta gradualmente en el transcurso del proyecto; dicho aumento del nivel de satisfacción se debe al acoplamiento y experiencia que este iba adquiriendo durante el proyecto.

Nivel de adopción de prácticas Scrum por parte del equipo de desarrollo.

Para realizar este análisis se seleccionaron las principales prácticas ágiles, con base en Agile Allience (2018) , las cuales se muestran en la tabla 36.

Para esta evaluación utilizamos de las especificaciones del nivel de adopción de la metodología Scrum indicada de la ***Tabla 36***.

Practicas ágiles	Descripción
Entrega incremental	Cada producto resultante es usable y cada versión construida sobre la versión anterior adiciona funcionalidad visible al usuario.
Iteraciones	Reuniones diarias con todo el equipo: cada miembro describe brevemente cualquier contribución completada y cualquier obstáculo que encuentre. Usualmente se usan las tres preguntas y se realiza en frente del taskboard.
Daily Stand-up Meetings	Reuniones diarias con todo el equipo: cada miembro describe brevemente cualquier contribución completada y cualquier obstáculo que encuentre. Usualmente se usan las tres preguntas y se realiza en frente del taskboard.
Burndown Chart	Gráfico de avance relacionado con la cantidad de trabajo restante y el tiempo transcurrido desde el inicio del proyecto o sprint.
Definition of Done	Definición de Hecho. El equipo está de acuerdo con los criterios que deben ser cumplidos antes que un incremento de producto es considerado hecho.
Definition of Ready	Definición de Listo. El equipo hace visible y explicito que las historias de usuario cumplen los criterios para ser aceptadas en una iteración.
Reunión de planeación	Ceremonia de *Scrum* que se realiza al inicio de un sprint para definir que se hará en el sprint y como se realizará.
Planning Poker	Método de estimación basado en Wideband Delphi en donde se estima basado en la analogía, el juicio experto y la desagregación.
Backlog	Pila de producto que contiene las características priorizadas del producto a ser desarrollado.
Reunión Retrospectiva	Ceremonia de *Scrum* que permite la mejora del proceso, se realiza después de cada sprint y genera planes de acción a ejecutar en el próximo sprint.

Tabla 40. Practicas ágiles asociadas a Scrum
Fuente: (Agile Alliance, 2018)

La evaluación (formato de encuesta Anexo 10) muestra un gran nivel aceptación, en la **figura 37** se observa que las prácticas con un alto nivel de adopción son: La *Entrega incremental, Iteraciones, Daily Stand-up*

Meetings, Burndown Chart, Definition of Done, Reunión de planeación, Backlog con un 75%.

Mientras tanto se muestran con un nivel de adopción del 50% las siguientes prácticas: *Definition of ready, Planning póker, Reunión Retrospectiva.*

Por lo observado se puede concluir que todos los integrantes del Equipo han adoptado las practicas asociadas a Scrum en al menos un 50%, lo que indica en términos generales la buena aceptación a la implantación de las metodologías de desarrollo ágil.

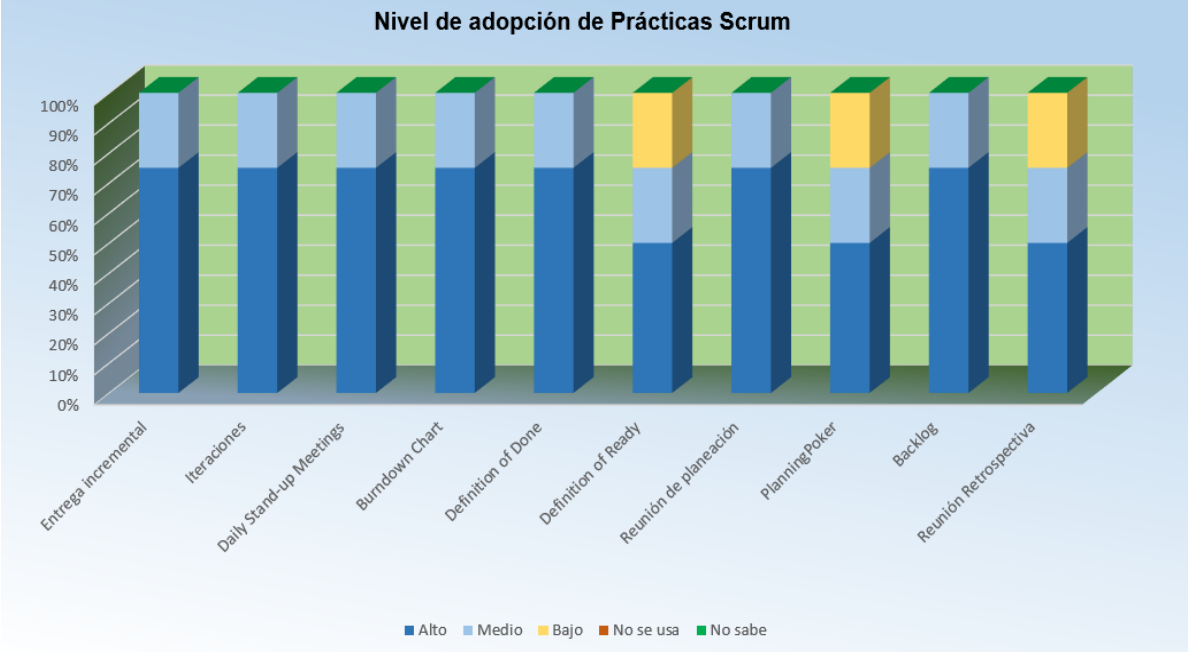

Figura 37. Resultados nivel de adopción de prácticas Scrum
Fuente: El Autor

6.2.4 Resumen de resultados

Factores críticos de éxito	Resumen y valoración
Factores humanos	Los resultados de la encuesta indican que la experiencia, la actitud y el compromiso son factores que se observó en el equipo al momento de la implementación. La evaluación obtenida tomando en cuenta los valores positivos de las encuestas (*totalmente de acuerdo y de acuerdo*) da una valoración positiva del 90,5%.
Factores Organizacionales	Los factores organizacionales son analizados desde la perspectiva del cliente y del desarrollo

	del proyecto; los resultados indican que cuando los clientes están motivados, comprometidos y activos no solo con ellos mismos sino también con el equipo, el trabajo se realiza de excelente forma. La gestión de incidentes y la utilización de prácticas ágiles fueron muy favorable; en general se muestra un grado de satisfacción del 94,8%.
Otros factores	La satisfacción del equipo y el cliente en términos generales, fue positiva; así como también el nivel de adopción de las prácticas Scrum por parte del equipo de desarrollo. Este factor se evalúa con un 69,85% de conformidad.

Tabla 41. Resumen de resultados
Fuente: El Autor

Análisis de resultados

Para el análisis de resultados se tomó como punto de partida y línea base, la encuesta realizada en el planteamiento del problema de la investigación *(**Planteamiento del problema)**,* en el cual se evidenció y fundamentó las múltiples deficiencias en el desarrollo del software; se comparó los resultados obtenidos en el levantamiento de la información inicial y los obtenidos luego del proceso de investigación. Se concluyó que luego de la implementación de una metodología ágil, el desarrollo de software en la Universidad Yachay Tech mejoró positivamente la gestión en un 47,7% lo cual indica que se ha logrado cumplir con el objetivo principal planteado en el presente estudio.

De acuerdo a los resultados obtenidos, ha sido posible observar como un aspecto principal la satisfacción del cliente, esta ha ido subiendo a lo largo del desarrollo por el aumento incremental en la consecución de objetivos en la evolución del producto. Se ha evidenciado que uno de los aspectos que más positivamente repercute en la satisfacción del cliente es

la continua aportación de valor a su negocio, al permitir de forma progresiva ir aprovechando de nuevas funcionalidades del producto.

Factores críticos de éxito	Estado inicial	Estado final	Mejoras
Factores humanos	30%	90,5%.	60,5%
Factores Organizacionales	44,53%	94,8%	50,27%
Otros factores	37,53%	69,85%	32,32%
Promedio total			47,7%.

Tabla 42. Análisis comparativo del mejoramiento en la gestión de los proyectos de software
Fuente: El autor

CAPÍTULO VII. CONCLUSIONES Y RECOMENDACIONES

7 Introducción

En este capítulo se presentan las conclusiones a las que se ha llegado tras la realización del presente estudio de investigación. Primeramente, se indican las conclusiones a las que se ha llegado después de la finalización del mismo respecto a los objetivos que han guiado la investigación. Finalmente, se indican algunas recomendaciones o líneas de estudio de la investigación a futuro.

En este trabajo de Tesis se realizó un estudio investigativo para la Implantación de una metodología ágil para el mejoramiento de la gestión de los proyectos de software en la Universidad Yachay Tech.

Para el cumplimiento del objetivo se realizó inicialmente un estudio comparativo de metodologías ágiles; con este estudio se determinó la metodología que más se adapta a las necesidades de desarrollo de la institución. Para corroborar el estudio comparativo se realizó un caso de estudio en el cual se determinó varios factores de éxito de esta implementación.

7.1 Conclusiones

Conclusiones particulares

- Mediante la elaboración del marco teórico se pudo establecer una línea base conceptual de las metodologías ágiles y sus antecedentes históricos para la realización de este proyecto de investigación.
- Se realizó un estudio comparativo de metodologías ágiles donde se evidenció que los procesos administrativos de la institución son muy variables y flexibles por lo que es necesario gestionar los proyectos de desarrollo de software con una metodología ágil.
- Se determinó que la metodología ágil Scrum es la que más se ajusta a las necesidades de desarrollo de software, en consecuencia, se

adoptó esta metodología para la implementación mediante un estudio de caso.

Factor Humano

Se concluye que desde el punto de vista del factor humano se influyó positivamente en la adopción de las metodologías de desarrollo, aunque no son determinantes, en el estudio se demuestra que la más de la mitad del equipo reacciona positivamente en el proceso de implantación de Scrum. La evaluación obtenida tomando en cuenta los valores positivos de las encuestas (*totalmente de acuerdo y de acuerdo*) da una valoración positiva del 90,5%.

Factores organizacionales

La satisfacción del cliente ha ido incrementando a lo largo del desarrollo por el aumento incremental en la consecución de objetivos en la evolución del producto.

Los resultados indican que cuando los clientes están motivados, comprometidos, dispuestos y activos no solo con ellos mismos sino también con el equipo de desarrollo; en este estudio se muestra un grado de satisfacción del 94,8%

Otros factores

La satisfacción del equipo y el cliente en términos generales, fue positiva; así como también el nivel de adopción de las prácticas Scrum por parte del equipo de desarrollo. Este factor se evalúa con un 69,85% de conformidad.

Conclusión general

Luego de finalizar el estudio de investigación se concluyó que la implementación de una metodología ágil para el mejoramiento de la gestión de los proyectos de desarrollo de software en la Universidad Yachay Tech mejoró positivamente la gestión en un 47,7% lo que muestra un mejoramiento sustancial. Por otra parte, se refleja aumento de la

productividad lo cual tiene influencia directa en que el cliente se muestre satisfecho y muestre confianza hacia el equipo de desarrollo.

7.2 Recomendaciones

- La adopción de Scrum exige la capacitación de las personas en los niveles que estén involucrados directamente en el desarrollo de los proyectos en la organización.
- Se recomienda hacer la adopción de manera incremental con equipos empoderados y comprometidos buscando que los miembros del equipo no sean todos junior y con equipos co-localizados.
- Es importante sensibilizar al Product Owner externo sobre Scrum y sus ventajas, mostrando resultados.
- Involucrar y capacitar al cliente desde el principio y todo el proyecto para que tome conciencia del papel que juega dentro del equipo.
- La reunión retrospectiva al final de cada entrega crea lazos de confianza.
- No personalizar Scrum a los requerimientos de la organización.
- El cambio de cultura y actitud es importante para llegar a ser un equipo más auto-organizado y disciplinado.

REFERENCIAS BIBLIOGRÁFICAS

Acevedo, J., Urquiaga, A., & Gómez, M. (2001). Gestión de la Cadena de suministro. *Centro de estudio de Tecnología de Avanzada (CETA) y Laboratorio de Logística y Gestión de la Producción (Logespro)*.

Agile Allience. (12 de Julio de 2018). *Agile Allience*. Recuperado el Julio de 2018, de https://www.agilealliance.org/agile101/subway-map-to-agile-practices/

agilemanifesto.org. (5 de Mayo de 2018). *Manifiesto por el desarrollo de software ágil*. Recuperado el 15 de Diciembre de 2017, de http://agilemanifesto.org/iso/es/manifesto.html

Alsalemi , A. M., & Yeoh, E. T. (2015). A survey on product backlog change management and requirement traceability in agile (Scrum). *2015 9th Malaysian Software Engineering Conference (MySEC)* (págs. 189-194). Kuala Lumpur: IEEE. doi:10.1109/MySEC.2015.7475219

Álvarez, F. J., Hurtado Alegría, J. A., Mondragón Arellano, M., Muñoz Arteaga, J., Velázquez Amador, C. E., & Hernández Bieliukas, Y. C. (2014). *GESTIÓN DE PROYECTOS DE SOFTWARE*.

Anaya, R. (2006). Una visión de la enseñanza de la Ingenieria de Software como apoyo al mejoramiento de las empresas de software. *REVISTA Universidad EAFIT*, 61.

Bass, J. M. (2014). Scrum Master Activities: Process Tailoring in Large Enterprise Projects. *2014 IEEE 9th International Conference on Global Software Engineering* (págs. 6-15). Shanghai: IEEE. doi:10.1109/ICGSE.2014.24

Beck, K. (2004). *Extreme Programming Explained: Embrace Change* . Addison-Wesley.

Bertone, R., Pasini, A., & Ramon, H. (2005). Programación extrema y calidad - Estudio de compatibilidad XP – CMM. *XI Congreso Argentino de Ciencias de la Computación.* Buenos Aires.

Blokehead, T. (2016). *Scrum - ¡Guía definitiva de prácticas ágiles esenciales de Scrum!* Babelcube Inc.

Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. *Computer (IEEE)*, págs. 61 - 72 .

Cegarra Sánchez, J. (2004). *METODOLOGÍA DE LA INVESTIGACIÓN CIENTÍFICA Y TECNOLÓGICA.* Madrid: Ediciones Díaz de Santos.

Coleman, G. (1998). A quality software process for rapid application development. *Software Quality Journal*, 107–122. Recuperado el 12 de 11 de 2017, de http://www.itu.dk/people/katten/speciale/RAD_a_quality_software_process.pdf

CONTRALORIA GENERAL DEL ESTADO. (2011). NORMAS DE CONTROL INTERNO PARA LAS ENTIDADES, ORGANISMOS DEL SECTOR PÚBLICO Y PERSONAS JURÍDICAS DE DERECHO PRIVADO QUE DISPONGAN DE RECURSO PÚBLICOS.

Coronado Padilla, J. (2007). Escalas de Medición. *Paradigmas, 2*, 104-125.

Davis, A. (1993). *Software requirements: objects, functions, and states.* Englewood Cliffs, N.J.: PTR Prentice Hall.

Deshinta, A. D., & Mohana, M. (2014). The agility of agile methodology for teaching and learning activities. *2014 8th. Malaysian Software Engineering Conference (MySEC)* (págs. 255-259). Langkawi: IEEE.

Díaz Novás, J., Gallego Machado, B., & Calles Calviño, A. (2011). Bases y aplicación del método hipotético-deductivo en el diagnóstico. *Revista Cubana de Medicina General Integral*, 378-387. Recuperado el 29 de Octubre de 2018, de http://search.ebscohost.com/login.aspx?direct=true&db=asn&AN=77483051&lang=es&site=eds-live&scope=site

Everett, G., & McLeod, R. (2007). *The Software Development Life Cycle - Wiley-IEEE Press books.* TEXAS: Wiley-IEEE Press. Obtenido de https://ieeexplore.ieee.org/document/5989501

Fitzhenry, P., & Gardiner, G. (1995). Enhanced software project management by application of metrics and cost estimation techniques. *IEE Colloquium on Project Management for Software Engineers* (págs. 1-4). Londres: IEEE Xplore. doi:10.1049/ic:19951542

Fuentes, J. R. (2015). *Desarrollo de Software Ágil: Extremme Programming y Scrum. 2ª Edición.* Vigo: IT Campus Academy.

Garcés Guayta, L. R., & Egas, L. M. (2013). EVOLUCION DE LAS METODOLOGIAS DE DESARROLLO DE LA INGENIERIA DE SOFTWARE EN EL PROCESO LA INGENIERIA DEL SOFTWARE. *REVISTA CIENTÍFICA Y TECNOLÓGICA UPSE,* 3.

García Rodríguez, M. J. (2015). Estudio comparativo entre metodologias ágiles y tradicionales para la gestión de proyectos de software. *TRABAJO FIN DE MASTER*. España.

García Sanchéz, J., Aguilera, J. R., & Castillo Rosas, A. (Enero de 2011). Guía técnica para la construcción de escalas de actitud. *Odiseo, Revista electrónica de pedagogía*(8). Recuperado el 3 de Septiembre de 2018, de https://www.odiseo.com.mx/2011/8-16/pdf/garcia-aguilera-castillo-guia-construccion-escalas-actitud.pdf

Garzás, J. (2013). *Gestión ágil de Proyectos de Software.* Recuperado el 5 de Diciembre de 2017, de http://www.inf.utfsm.cl/~guerra/publicaciones/Gestion de Proyectos de Software.pdf

Iacovelli, A. (2008). *Framework for Agile Methos Classification, Workshop on Model Driven Informat.* Systems Eng.: Enterprise.

IEEE 1074. (1997). *IEEE Standard for Developing Software Life Cycle Processes.*

IEEE Computer Society Sponsored by the Software & Systems Engineering Standards Committee. (2011). IEEE Guide–Adoption of ISO/IEC TR 24748-1:2010 Systems and Software Engineering–Life Cycle Management–Part 1: Guide for Life Cycle Management. *IEEE Std 24748-1-2011*, 1-96. doi:10.1109 / IEEESTD.2011.5871657

Iglesias Solano, A. M. (2011). Story Points y su Importancia en la Estimación de Proyectos Bajo el Enfoque Ágil. *Investigación y Desarrollo en TIC, 2*(1), 50-58.

ISO. (2008). ISO/IEC 12207 Information Technology / Software Life Cycle Processes.

Kaur, K., Jajoo, A., & Manisha. (27 de Febrero de 2015). Applying Agile Methodologies in Industry Projects: Benefits and Challenges. *IEEE Xplore*, 832-836. doi:10.1109/ICCUBEA.2015.166

Kruchten, P. (2001). *The Rational Unified Process An Introduction* (Vol. III). Vancouver: Addison-Wesley.

Kumar, B. P., & Prashanth, Y. (2014). Improving the Rapid Application Development process model. *2014 Conference on IT in Business, Industry and Government (CSIBIG)* (págs. 1-3). Indore: IEEE. doi:10.1109 / CSIBIG.2014.7056962

Lynch, R., & Cross, K. (1995). *Measure up!: Yardsticks for continuous improvement.* Cambridge, Cambridge: MA (USA): Black Bussiness.

Macluskey. (Marzo de 2009). *Historia de un Viejo Informático. La Programación Estructurada.* Recuperado el 8 de Noviembre de 2017, de El Cedazo: https://eltamiz.com/elcedazo/2009/03/16/historia-de-un-viejo-informatico-la-programacion-estructurada/

Misra, S., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *The Journal of Systems and Software*, 82, 1869-1890.

Muñoz, M., Mejía, M., Calvo Manzano, J., & Sant Feliu, T. (2013). Involvement of Stakeholders in Software Processes Improvement to Reduce Change Resistance. *EuroSpi*, 202.

Pantoja, L., Collazos, C., & Penichet, V. (2013). ENTORNO COLABORATIVO DE APOYO A LA MEJORA DE PROCESOS DE SOFTWARE EN PEQUEÑAS ORGANIZACIONES. *revistas.unal.edu.co*, 1-2.

Paulk, M. C. (2002). Agile Methodologies and Process Discipline. *The Journal of Defense Software Engineering*, 15-16. Obtenido de http://repository.cmu.edu/isr/3

Peña, A. (2006). *Ingeniería de Software : Una Guía para Crear Sistemas de Información.* Mexico: INSTITUTO POLITÉCNICO NACIONAL.

Pérez Pérez, M. J., & González Cabrera, F. J. (2012). *Guá Comparativa de Metodologías Ágiles.* Valladolid.

Pressman, R. (2010). *Ingeniería del software: Un enfoque práctico* (Tercera Edición ed.). New York: The McGraw-Hill Companies, Inc.

proyectosagiles.org. (2016). *https://proyectosagiles.org*. Obtenido de https://proyectosagiles.org/que-es-scrum/

Rauterberg, M. (1992). An iterative Cyclic Software Process Model. *Proceedings Fourth International Conference on Software Engineering and Knowledge Engineering.* Capri: IEEE. doi:10.1109/SEKE.1992.227899

Reece, J. (2006). *Prototyping and Rapid Application Development (RAD) RAD - Background What is RAD RAD - Goals RAD - Quality RAD - Properties RAD - Cost.*

Revista Tecnológica ESPOL – RTE. (2014). Aumento de la productividad en la gestión de proyectos, utilizando una metodología ágil aplicada en una fábrica de software en la ciudad de Guayaquil. *Revista Tecnológica ESPOL – RTE*, 1-36.

Royce, W. W. (1970). *Managing the development of large software systems.*

Rujana, M., Romero Franco, N., Tortosa, N., Tomasselli, G., & Pinto, N. (2016). Análisis sobre adopción de metodologías ágiles en los equipos de desarrollo en pymes del NEA. *FRRE - Producción de Investigación - Grupo GICS - Difusión Científica - Trabajos Presentados en Congresos* , 648.

Runeson, P. (2012). *Experimentation in Software Engineering* (Primera ed.). Berlin: Springer-Verlag Berlín Heidelberg.

Scanlon Thomas, E. (2011). *Breaking the Addiction to Process: An Introduction to Agile Project Management.* United Kingdom: IT Governance Ltd.

SCRUM STUDY targeting success. (2016). *CUERPO DE CONOCIMIENTO DE SCRUM (GUÍA SBOK™).* Phoenix, Arizona, USA: SCRUMstudy™.

Secretaria Nacional de Planificación y Desarrollo. (2017). Plan Nacional de Desarrollo Toda una Vida. Quito, Ecuador.

Sharma, P., & Hastecer, N. (2016). Analysis of linear sequential and extreme programming development methodology for a gaming application. *2016 International Conference on Communication and Signal Processing*

(ICCSP) (págs. 1916-1920). Melmaruvathur: IEEE. doi:10.1109/ICCSP.2016.7754505

Sommerville, I. (2006). *Ingenieria del Software* (Séptima Edición ed.). Madrid: Pearson Education. S.A.

Sutherland, J., & Schwaber, K. (2013). *La Guía de Scrum.*

Torres, C. (2006). *Metodología de la investigación: para administración, economía, humanidades y ciencias sociales.* Pearson Educación.

Toshiya, F., Tadashi, D., & Takaji, F. (2011). Towards quantitative software reliability assessment in incremental development processes. *2011 33rd International Conference on Software Engineering (ICSE)* (págs. 41-50). Honolulu: IEEE. doi:10.1145 / 1985793.1985800

Trigas Gallego, M., & Domingo Troncho, A. (2014). *Gestion de proyectos informàticos - Metodologia SCRUM.*

Whitson, G. M. (2013). *Ingeniería de software.* Enciclopedia Salem Press de la ciencia. Obtenido de http://search.ebscohost.com/login.aspx?direct=true&db=ers&AN=89250579&lang=es&site=eds-live&scope=site

ANEXOS

Anexo 1:

Fundamentación de la problemática actual del desarrollo del software en Yachay Tech.

Investigación: Implementación de una metodología de desarrollo ágil para el mejoramiento de la gestión de los proyectos de software en Yachay Tech.

Entrevistador: Lincon Chamorro Fecha: 05/02/2018

Esta entrevista pretende fundamentar y evidenciar la problemática en el desarrollo de software en la dirección de Tics de Yachay Tech.

El cuestionario contiene preguntas básicas en torno a las actividades de desarrollo de software. Evalúe de acuerdo a lo que usted piensa con una valoración de 1 a 100 puntos.

Encuesta "Problemas de desarrollo del software"

Preguntas	Valoración sobre 100%
Valore el grado de compromiso frente los siguientes principios básicos para el desarrollo de software: Trabajo en equipo, colaboración con el cliente, capacidad de negociación, entorno de negocio cambiante.	30
¿En qué porcentaje se ha atendido la demanda de construcción de software a medida hasta la actualidad?	50
Valore el porcentaje de cumplimiento de las fechas planificadas para la entrega de proyectos de software.	30
Valore el grado de comunicación con el cliente durante todas las etapas de desarrollo del desarrollo del software.	35
Valore esta afirmación: Los requerimientos suelen ser muy claros, lo cual facilita entender la lógica del negocio y a su vez se obtiene como resultado final un software que se ajusta a las necesidades del cliente.	58
Valore el compromiso y la participación de la alta gerencia de la institución en los proyectos de desarrollo de software.	38

Ing. Edwin Pérez
Director de Tics Yachay Tech

Firma:

Matriz de tabulación de resultados de la encuesta “problemas de desarrollo del software”.

Problemas detectados	Evaluación sobre(100)
Valore el grado de compromiso frente los siguientes principios básicos para el desarrollo de software: Trabajo en equipo, colaboración con el cliente, capacidad de negociación, entorno de negocio cambiante.	30
	40
	30
	30
	20
¿En qué porcentaje se ha atendido la demanda de construcción de software a medida hasta la actualidad?	50
	60
	40
	60
	50
Valore el porcentaje de cumplimiento de las fechas planificadas para la entrega de proyectos.	30
	40
	35
	45
	25
Valore el grado de comunicación con el cliente durante todas las etapas de desarrollo del desarrollo del software.	35
	45
	30
	40
	50
Valore esta afirmación: Los requerimientos suelen ser muy claros, lo cual facilita entender la lógica del negocio y a su vez se obtiene como resultado final un software que se ajusta a las necesidades del cliente.	58
	65
	55
	50
	55
Valore el compromiso y la participación de la alta gerencia de la institución en los proyectos de desarrollo de software.	38
	45
	50
	55
	50

Anexo 2:

Operacionalización de las variables.

Variable independiente: Metodologías de desarrollo ágil

Conceptualización	Dimensiones	Indicadores	Ítems	Técnicas e instrumentos
Las metodologías de desarrollo de software son un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información (CENTER FOR MEDICARE & MEDICAID SERVICES, 2005).	Paradigmas Entornos de trabajo	Conoce los paradigmas de ingeniería del software. Si o No La utilización metodologías de desarrollo ágil presenta ventajas y desventajas	¿El conocimiento de paradigmas de ingeniería del software permite la selección adecuada de uno de estos? ¿Qué ventajas y desventajas representa la utilización de metodologías de desarrollo ágil?	Encuesta-Cuestionario Encuesta – cuestionario

Variable dependiente: Mejoramiento de la gestión de proyectos de software.

Conceptualización	Dimensiones	Indicadores	Ítems	Técnicas e instrumentos
La gestión proyectos de software se basa en dos elementos clave: la planificación y estimación precisas del ciclo de vida del proyecto, y la supervisión y el control del proyecto para llevarlo a cabo con éxito, en términos de tiempo, costo y calidad	Actividades Tareas no planificadas Tareas en las fases del ciclo de desarrollo.	Porcentaje de cumplimiento de actividades planificadas Porcentaje de cumplimiento de tareas no planificadas Porcentaje de las tareas en las fases del ciclo de desarrollo.	¿Cuál es el porcentaje de actividades planificadas? ¿Cuál es el porcentaje de cumplimiento de tareas no planificadas? ¿Cuál es el porcentaje de avance en las	Encuesta – cuestionario Encuesta – cuestionario Encuesta – cuestionario

			fases del ciclo de desarrollo?	

Anexo 3:

Aplicación del modelo de evaluación del enfoque de la organización frente a las metodologías ágiles y tradicionales.

Investigación: Implementación de una metodología de desarrollo ágil para el mejoramiento de la gestión de los proyectos de software en Yachay Tech.

Entrevistador: Lincon Chamorro Fecha: 05/02/2018

Esta entrevista pretende analizar el enfoque de la organización; para esto se considera el comportamiento desde su gestión administrativa y de procesos frente a los valores ágiles y tradicionales de las metodologías de desarrollo de software. Mediante esta encuesta queremos conocer la visión desde la Dirección de Tics de la Universidad Yachay Tech.

El cuestionario tiene dos secciones las cuales describen los valores agiles y tradicionales de las metodologías, Evalúe de acuerdo a lo que usted piensa. La escala de valoración se especifica en la tabla inferior.

Importancia	valor
Ninguna importancia.	0
Baja importancia.	1
Media importancia.	2
Alta importancia.	3

A. Orientación ágil

Por favor evalúe cada uno de los valores agiles en la escala indicada.

Valores ágiles	Importancia
El individuo y las interacciones del equipo	2
Desarrollar software que funciona	3
Colaboración con el cliente	3
Respuesta al cambio	3

B. Orientación tradicional

Valores Tradicionales	Importancia
E proceso y las herramientas	2
Conseguir una buena documentación	2
Negociación contractual	2
Seguimiento de un plan	2

Ing. Edwin Pérez
Director de Tics Yachay Tech

Firma:

Anexo 4:
Aplicación del modelo de evaluación del cumplimiento de los principios ágiles en la gestión de Institución.

Investigación: Implementación de una metodología de desarrollo ágil para el mejoramiento de la gestión de los proyectos de software en Yachay Tech.

Entrevistador: Lincon Chamorro Fecha: 05/02/2018

Esta encuesta pretende analizar el cumplimiento de los principios ágiles de la organización; para esto se considera el comportamiento desde la gestión administrativa y de procesos frente a los principios ágiles de las metodologías de desarrollo de software. Mediante esta encuesta queremos conocer la visión desde la Dirección de Tics de la Universidad Yachay Tech.

El cuestionario describe los valores ágiles de las metodologías, Evalúe de acuerdo a lo que usted piensa. La escala de valoración se especifica en la tabla inferior.

Importancia	valor
Ninguna importancia.	0
Baja importancia.	1
Media importancia.	2
Alta importancia.	3

CUMPLIMIENTO DE LOS PRINCIPIOS ÁGILES

A. Cumplimiento principios ágiles

Por favor evalúe cada uno de los principios ágiles en la escala indicada.

N°	Principios ágiles	Importancia
1	Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.	2
2	Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.	2
3	Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.	2
4	Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.	3
5	Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.	3
6	El método más eficiente y efectivo de comunicar información al quipo de desarrollo y entre	3

1

N°	Principios ágiles	Importancia
	sus miembros es la conversación cara a cara.	
7	El software funcionando es la medida principal de progreso.	3
8	Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.	2
9	La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.	3
10	La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.	2
11	Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.	3
12	A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.	3

Ing. Edwin Pérez
Director de Tics.

Firma:

Anexo 5

Valoración de los 4 puntos de vista de las metodologías ágiles.

Investigación: Implementación de una metodología de desarrollo ágil para el mejoramiento de la gestión de los proyectos de software en Yachay Tech.

Entrevistador: Lincon Chamorro Fecha: 05/02/2018

Las preguntas contienen los atributos de los 4 puntos de vista de las metodologías ágiles, la respuesta a estos formularios se debe apoyar en las necesidades de la Institución.

Las respuestas tienen valores de verdadero o falso para cada una de las preguntas; para este caso la decisión de las respuestas se realizará en base a los intereses de la gestión administrativa de la institución.

Valoración de los atributos de los cuatro puntos de vista de las metodologías ágiles.

USO

Nro.	Premisa	Respuesta
1	Respeto de las fechas de entrega	V
2	Cumplimiento de los requisitos	V
3	Respeto al nivel de calidad	V
4	Satisfacción del usuario final	V
5	Entornos turbulentos	F
6	Favorable al Off shoring	V
7	Aumento de la productividad	V

CAPACIDAD DE AGILIDAD

Nro.	Premisa	Respuesta
1	Iteraciones cortas	V
2	Colaboración	V
3	Centrado en las personas	V
4	Refactoring político	F
5	Prueba político	V
6	Integración de los cambios	V
7	De peso ligero	V
8	Los requisitos funcionales pueden cambiar	V
9	Los requisitos no funcionales pueden cambiar	V
10	El plan de trabajo puede cambiar	F
11	Los recursos humanos pueden cambiar	V
12	Cambiar los indicadores	F
13	Reactividad (AL COMIENZO DEL PROYECTO, CADA ETAPA, CADA ITERACIÓN)	ITERACIÓN
14	Intercambio de conocimientos (BAJO, ALTO)	BAJO

APLICACIÓN

Nro.	Premisa	Respuesta
1	Tamaño del proyecto (PEQUEÑO, GRANDE)	PEQUEÑO

1

2	La complejidad del proyecto (BAJA, ALTA)	BAJO
3	Los riesgos del proyecto (BAJO, ALTO)	BAJO
4	El tamaño del equipo (PEQUEÑO, GRANDE)	PEQUEÑO
5	El grado de interacción con el cliente (BAJA, ALTA)	ALTA
6	Grado de interacción con los usuarios finales (BAJA, ALTA)	ALTA
7	Grado de interacción entre los miembros del equipo (BAJA, ALTA)	ALTA
8	Grado de integración de la novedad (BAJA, ALTA)	ALTA
9	La organización del equipo (AUTO-ORGANIZACIÓN, ORGANIZACIÓN JERÁRQUICA)	AUTO

PROCESOS Y PRODUCTOS

Nivel de abstracción de las normas y directrices:

Nro.	Premisa	Respuesta
1	Gestión de proyectos	V
2	Descripción de procesos	V
3	Normas y orientaciones concretas sobre las actividades y productos	F

Las actividades cubiertas por el método ágil:

Nro.	Premisa	Respuesta
1	Puesta en marcha del proyecto	V
2	Definición de requisitos	V
3	Modelado	F
4	Código	V
5	Pruebas unitarias	F
6	Pruebas de integración	F
7	Prueba del sistema	V
8	Prueba de aceptación	V
9	Control de calidad	F
10	Sistema de uso	V

Productos de las actividades del método:

Nro.	Premisa	Respuesta
1	Modelos de diseño	F
2	Comentario del código fuente	V
3	Ejecutable	V
4	Pruebas unitarias	F
5	Pruebas de integración	F
6	Pruebas de sistema	V
7	Pruebas de aceptación	V
8	Informes de calidad	F
9	Documentación de usuario	V

Ing. Edwin Pérez
Director de Tics Yachay Tech

Firma:

2

Anexo 6

Encuesta factores críticos de éxito (factores humanos).

Investigación: Implementación de una metodología de desarrollo ágil para el mejoramiento de la gestión de los proyectos de software en Yachay Tech.

Entrevistador: Lincon Chamorro Fecha: 09/02/2018

Evalué los siguientes factores críticos de éxito durante el desarrollo del estudio de caso en el cual se implantó la metodología Scrum. Las preguntas describen los factores humanos en la implantación.

Marque con una (X) la valoración que usted considere necesaria. La escala de valoración se especifica en la tabla inferior.

Ítems	valor
Totalmente de acuerdo	5
De acuerdo	4
Ni de acuerdo ni en desacuerdo	3
En desacuerdo	2
Totalmente en desacuerdo	1

Preguntas	Valoración				
	5	4	3	2	1
¿Considera que su experiencia previa en metodologías ágiles influyó positivamente en la implementación del estudio de caso?	x				
¿Su experiencia previa con el agilismo en esta implementación fue favorable?		x			
¿Considera que los valores como la honestidad, integridad, trabajo en equipo, sentido de responsabilidad, interés en aprender y espíritu emprendedor contribuyeron positivamente en la implementación de la metodología?		x			
¿Considera que su adaptación a las prácticas Scrum establecidas por la metodología fueron positivas?		x			

Guido Chávez
Asistente de desarrollo de software.

Firma:

Anexo 7

Encuesta factores críticos de éxito (factores asociados con el cliente).

Investigación: Implementación de una metodología de desarrollo ágil para el mejoramiento de la gestión de los proyectos de software en Yachaytech.

Entrevistador: Lincon Chamorro Fecha: 09/02/2018

Evalué los siguientes factores críticos de éxito durante el desarrollo del estudio de caso en el cual se implantó la metodología Scrum. Las preguntas describen los factores asociados con el cliente.

Marque con una (X) la valoración que usted considere necesaria. La escala de valoración se especifica en la tabla inferior.

Ítems	valor
Muy satisfactorio	5
Satisfactorio	4
Medianamente satisfactorio	3
Poco satisfactorio	2
Nada satisfactorio	1

Preguntas	Valoración				
	5	4	3	2	1
Qué grado de satisfacción merece la comunicación con el cliente?	x				
Qué grado de satisfacción merecen los procesos de negociación con el cliente?	x				
Cuál es el grado de satisfacción del cliente respecto a la colaboración en el proceso de desarrollo del software?	x				
Qué nivel de satisfacción percibió el cliente ante la utilización de las metodologías ágiles en el desarrollo del estudio de caso?		x			
Qué grado de satisfacción mostró el cliente cada vez que se presentaba los incrementos del software en cada Sprint?	x				

Guido Chávez
Asistente de desarrollo de software.

Firma:

1

Anexo 8

Encuesta factores críticos de éxito (factores asociados con el desarrollo del proyecto).

Investigación: Implementación de una metodología de desarrollo ágil para el mejoramiento de la gestión de los proyectos de software en Yachay Tech.

Entrevistador: Lincon Chamorro Fecha: 09/08/2018

Evalué los siguientes factores críticos de éxito durante el desarrollo del estudio de caso en el cual se implantó la metodología Scrum. Las preguntas describen los factores asociados con el desarrollo del proyecto.

Marque con una (X) la valoración que usted considere necesaria. La escala de valoración se especifica en la tabla inferior.

Ítems	valor
Totalmente de acuerdo	5
De acuerdo	4
Ni de acuerdo ni en desacuerdo	3
En desacuerdo	2
Totalmente en desacuerdo	1

Preguntas	Valoración				
	5	4	3	2	1
Las tareas planificadas para cada sprint se cumplen en el tiempo acordado para la revisión.	x				
El equipo prioriza el desarrollo a medida, obteniendo la funcionalidad indicada con el cliente.		x			
Durante el desarrollo del proyecto cuando ocurrieron incidentes inesperados, el Scrum Master los solvento de manera adecuada.		x			
Durante el desarrollo del proyecto se dio prioridad Software funcionando sobre documentación extensiva.	x				
El seguimiento constante de las actividades hace que se puedan tomar acciones correctivas de forma rápida para encausar los problemas.	x				

Guido Chávez
Asistente de desarrollo de software.

Firma:

1

Anexo 9

Encuesta grado de satisfacción general del cliente y equipo de desarrollo.

Investigación: Implementación de una metodología de desarrollo ágil para el mejoramiento de la gestión de los proyectos de software en Yachay Tech.

Entrevistador: Lincon Chamorro Fecha: 09/08/2018

Evalúe el grado de satisfacción durante el desarrollo del estudio de caso en el cual se implantó la metodología Scrum.

Pregunta	Valor (sobre 100)
Valore en términos generales el grado de satisfacción en el desarrollo del presente Sprint; la escala de evaluación es de 1 a 10.	30

Guido Chávez
Asistente de desarrollo de software

Firma:

1

Anexo 10

Encuesta grado de adopción de las practicas Scrum.

Investigación: Implementación de una metodología de desarrollo ágil para el mejoramiento de la gestión de los proyectos de software en Yachay Tech.

Entrevistador: Lincon Chamorro Fecha: 09/08/2018

Evalué el grado de adopción de las prácticas Scrum durante el desarrollo del estudio de caso en el cual se implantó la metodología.

Marque con una (X) la valoración que usted considere necesaria. La escala de valoración se especifica en la tabla inferior.

Items	valor
Alto	5
Medio	4
Bajo	3
No se usa	2
No sabe	1

Prácticas Scrum	Valoración				
	5	4	3	2	1
Entrega incremental	x				
Iteraciones		x			
Daily Stand-up Meetings	x				
Burndown Chart	x				
Definition of Done	x				
Definition of Ready		x			
Reunión de planeación	x				
Planning Poker	x				
Backlog		x			
Reunión Retrospectiva	x				

Guido Chávez
Asistente de desarrollo de s

Firma:

1

Matriz de resultados de la encuesta de adopción de las prácticas Scrum.

Prácticas Scrum	Ítems de Evaluación				
	Alto	Medio	Bajo	No se usa	No sabe
Entrega incremental	15	4	0	0	0
Iteraciones	10	4	3	0	0
Daily Stand-up Meetings	15	4	0	0	0
Burndown Chart	15	4	0	0	0
Definition of Done	15	4	0	0	0
Definition of Ready	10	4	3	0	0
Reunión de planeación	10	4	3	0	0
PlanningPoker	10	4	3	0	0
Backlog	15	4	0	0	0
Reunión Retrospectiva	10	4	3	0	0