APLICACIÓN DE LA METODOLOGÍA ÁGIL: UN INSTRUMENTO PARA MEJORAR LA GESTIÓN DE PROYECTOS DE SOFTWARE EN EL CENTRO DE INVESTIGACIÓN E INNOVACIÓN EN TICS DE LA UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

APLICACIÓN DE LA METODOLOGÍA ÁGIL: UN INSTRUMENTO PARA MEJORAR LA GESTIÓN DE PROYECTOS DE SOFTWARE EN EL CENTRO DE INVESTIGACIÓN E INNOVACIÓN EN TICS DE LA UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

DIRECTOR DE TESIS:

DR. RAMFIS MIGUELENA

ELABORADO POR:

NICHOL D. SÁNCHEZ KIRSCH

TRABAJO DE GRADUACIÓN PARA OPTAR AL TÍTULO DE

MAESTRÍA EN INGENIERÍA DEL SOFTWARE APLICADA

2018

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS Y CUADROS

DEDICATORIA:

Dedico esta investigación primeramente a Dios, quien me ha brindado las fuerzas para seguir adelante, la salud para continuar y perseverancia para vencer los problemas presentados.

A mi madre (QEPD) y padre que durante toda su vida me motivaron siempre a estudiar y seguir adelante sin importar las adversidades. A mi esposo y mis dos (2) hermanas por siempre apoyarme y motivarme a finalizar.

AGRADECIMIENTO:

En primera instancia quiero agradecer a mi asesor Dr. Ramfis Miguelena, quien me orientó durante la elaboración de esta investigación y al equipo de trabajo de la Sección de Fábrica de Software que siempre estuvo dispuesto a apoyarme en el desarrollo de esta investigación.

RESUMEN

La Ingeniería de Software es la rama de la informática dedicada a la gestión de proyectos de Software. Esta consiste en una serie de procesos para la ejecución de actividades de calidad, costo, tiempo y alcance para el desarrollo de un producto de Software. La definición de procesos es útil debido a que proporciona la forma en que deberá efectuarse las actividades; sin embargo, puede influir en que un proyecto, según su complejidad, tenga mayor riesgo a fracasar o no ser finalizado en el tiempo establecido.

Este trabajo de investigación surgió de las necesidades identificadas en la Sección de Fábrica de Software (FS) del Centro de Investigación CIDITIC de la Universidad Tecnológica de Panamá. La Fábrica presentaba un historial de proyectos con incidentes que llevaban a retrasos y sobrecostos en los mismos, por ello se decide realizar un diagnóstico que determine los requerimientos necesarios para el desarrollo y la entrega exitosa de proyectos en la Sección. Lo que da como resultado, la necesidad de contar con una metodología de proyectos de Software.

Existen muchas metodologías para el desarrollo de software, cada una formulada para responder a contextos y complejidades de diversos tipos de proyectos. Por lo tanto, la decisión de cuál metodología es la más adecuada para un equipo de software debe ser analizada a fondo y de manera exhaustiva. Para determinar qué metodología se ajustaría a la FS, se realizó un análisis comparativo considerando un número significativo de metodologías. Se requirió una extensa investigación sobre las diferencias entre el enfoque ágil y tradicional, y la revisión de las metodologías dentro de cada uno de ellos. La cantidad de información sobre cada metodología variaba mucho y en algunos casos, fue necesario descartar aquellas con poca información disponible para estructurar un perfil definido. Finalmente, se redujo a las diez (10) metodologías que mejor se ajustaban a las necesidades de la FS. Estas 10 principales fueron luego evaluadas a través de una encuesta dirigida a expertos, para determinar qué proceso y actividades deberían ser considerados. Los resultados de esta encuesta se usaron como insumos para estructurar y elaborar la propuesta CIDITIC-Scrum. Esta nueva metodología permitirá a la FS tener un mejor control y seguimiento del avance del proyecto, mejorando la calidad, el costo, los tiempos de entrega y, por lo tanto, la satisfacción del cliente.

ABSTRACT

Software engineering is the branch in Computer Science related to Software Project Management; it is a series of processes that outline quality, cost, time and scope activities for developing software. The definition of processes provides the guidelines to improve these activities and increase the success rate of a project.

This research project was first structured after identifying improvement opportunities at the Software Factory Section in the CIDITIC Research Center in the Universidad Tecnologica de Panama. The Factory had a history of delayed and overbudgeted projects, an assesment (also part of this research) allowed identifying that the Factory required a structured software development methodology in order to improve their project performance and success.

There are many methodologies for software development, each formulated to respond to different project contexts and complexities. Therefore, selecting the fittest for a software team must be analyzed throughly and extensively. In order to determine which methodology would adjust to the FS, a comparative analysis of many methodologies was made. It considered an extensive research on differences between agile and traditional, and then reviewal of methodologies within each of these categories. The amount of information on each methodology would varied greatly and, in some cases, the methodology in review was discarded due to little information available for creating a well-structured profile. Ultimately, the analysis was narrowed down to ten (10) methodologies that adjusted the best to the FS needs. This top 10 were then evaluated through a survey aimed at experts, to determine which process and activities should the FS methodology consider. The results of this survey were used as input to structure and create the proposed methodology CIDITIC-Scrum. This new methology will allow FS to have a better control and on follow-up on project progress, improving quality, cost, delivery times and so, customer satisfaction.

ACRÓNIMOS Y ABREVIATURAS

UTP	Universidad Tecnológica de Panamá
CIDITIC	Centro de Investigación, Desarrollo e Innovación en Tecnologías de la Información y las Comunicaciones
DITIC	Dirección General de Tecnología de la Información y Comunicaciones
FS	Fábrica de Software
SP	Software Process
SDM	Metodología de desarrollo de Software
PM	Proyect Management
ASD	Adaptive Software Development
FDD	Feature Driven Development
XP	Extreme Programming
DSDM	Dynamic Systems Development Method
LD	Lean Development
AUP	Agile Unified Process
SM	Scrum Master
PO	Product Owner
BA	Business Analyst

INTRODUCCIÓN

Hoy en día el Software cumple un rol fundamental en nuestra sociedad, ya que existe una gran dependencia a ellos, no solo en el área comercial o científica, sino también en la educación, salud, gobierno, logística y para la sociedad en general. El Software, efectúa las actividades básicas y complejas según las necesidades del cliente o usuario, agilizando los procesos de negocio y la disponibilidad de información. Por esta razón, para la elaboración de un producto de Software se utilizan metodologías de administración de proyectos y de Ingeniería de Software, llevando a una planificación, seguimiento y control del proyecto que incorpora una serie de métodos definidos para el análisis, diseño, implementación y pruebas del software.

Con relación a lo anterior, se ha identificado que en el Centro de Investigación, Desarrollo e Innovación en Tecnologías de la Información y las Comunicaciones – CIDITIC de la Universidad Tecnológica de Panamá (UTP), existe una Sección llamada Fábrica de Software (FS) que elabora productos de Software sin utilizar una metodología de Gestión de Proyectos que facilite el seguimiento y control de los proyectos.

Este trabajo de investigación profundizará en esta problemática, explicando los escenarios e incidentes ocurridos dentro de la Sección de Fábrica de Software y su estado actual respecto a los proyectos, recurso humano y su entorno. Además, en búsqueda de una solución se elaborará una fundamentación teórica, evaluando las metodologías existentes, para posteriormente presentar una propuesta metodológica ágil para el desarrollo y seguimiento de proyectos de Software.

Este trabajo de investigación está compuesto por cinco (5) capítulos esenciales: el Marco conceptual, Marco Teórico, la Metodología de Investigación, Sección de Fábrica de Software de CIDITIC-UTP y el Análisis de los resultados obtenidos.

CAPÍTULO I
MARCO CONCEPTUAL

1.1 INTRODUCCIÓN

El capítulo Marco Conceptual tiene como objetivo exponer los conceptos importantes a conocer de este trabajo de investigación, y describir los puntos principales que lo enmarca. Entre los puntos que serán descritos en este capítulo podemos mencionar: la escogencia del tema, la razón e interés por la que el tema fue seleccionado; el estado actual de las metodologías de proyectos de Software y de la Fábrica de Software del Centro de Investigación e Innovación en TICs de la Universidad Tecnológica de Panamá; el planteamiento del problema identificado; la justificación del trabajo, la hipótesis general, el alcance del proyecto, la delimitación, los objetivos (general y específicos), las limitaciones, las restricciones, aportes de la tesis y la estructura del documento.

A continuación, se presentarán los puntos descritos anteriormente.

1.2 ESCOGENCIA DEL TEMA

La Universidad Tecnológica de Panamá – UTP cuenta hoy en día con una diversa cantidad de Sistemas Informáticos que facilitan las actividades administrativas, financieras, operacionales y de investigación de la institución.

La UTP cuenta con diversos sistemas en marcha; sin embargo, debido a la transformación continua de la institución, existe un gran número de necesidades que requieren el desarrollo de nuevos sistemas a nivel administrativo, de investigación y extensión para su mejor funcionamiento. Actualmente la Dirección General de Tecnología de la Información y Comunicaciones – DITIC es la dirección encargada del desarrollo y mantenimiento de los sistemas administrativos, financieros y operacionales. Y el Centro de Investigación, Desarrollo e Innovación en Tecnologías de la Información y las Comunicaciones – CIDITIC, por medio de una Sección llamada Fábrica de Software – FS, se encarga de desarrollar proyectos de Software para el área de investigación; no obstante, ha sido utilizada desde el año 2013 para el desarrollo de aplicaciones institucionales pequeñas y medianas debido a la necesidad de Software.

Cabe destacar que por ser una sección relativamente nueva no cuenta con una mecánica de trabajo que permita ejecutar los proyectos, aplicando un seguimiento y control de las actividades adecuadamente, lo cual en ocasiones se traduce en sobrecostos y retrasos. Conjuntamente, el número promedio de personas dentro del equipo durante los años 2011 al 2017 ha sido de cuatro (4) colaboradores, lo cual lo categoriza como un grupo pequeño y la razón por la que el equipo se enfoca en proyectos pequeños y medianos; aunque esto no lo excluye de la posibilidad de efectuar proyectos grandes.

Debido a lo expuesto, para que la Sección de Fábrica de Software pueda desarrollar proyectos en el tiempo y costo requerido que satisfagan las necesidades primordiales del cliente, se requiere contar con un procedimiento de trabajo, en este caso una metodología que indique las características necesarias con que debe contar el equipo de desarrollo (perfil, tareas, cantidad mínima del equipo) y la estandarización de los procesos necesarios para completar un proyecto (entradas, procesos y salidas).

Por todo lo anterior, el tema de trabajo de investigación está orientado a la selección de una metodología ágil de gestión de proyectos de Software para proyectos pequeños y medianos, incluyendo el proceso ciclo de vida del proyecto hasta la propuesta de diseño del Software.

1.3 ESTADO ACTUAL

La Ingeniería de Software, está constituida por un conjunto de prácticas necesarias para desarrollar Software de calidad; esta colección de políticas, procesos y procedimientos aplicados dentro de una organización tienen como nombre Metodología de desarrollo de Software – SDM (Barzanallana, 2006). Al hacerse uso de los métodos en el proceso de desarrollo de productos de Software se disminuye y controla la cantidad de errores, la duración de tiempo y aumenta el valor del producto del cliente. Por lo tanto, se puede inferir que las empresas optan por el uso de metodologías para hacer frente al creciente nivel de competencia y exigencias del cliente. Aunque contar con una metodología no garantiza el éxito de

un proyecto, pero sí permite mejorar las probabilidades al establecer una ejecución ordenada de procesos, permitiendo un mayor control y su mejora continua.

El Reporte del Caos del año 2015 elaborado por la organización de asesoramiento de investigación Standish Group, presenta los resultados de un estudio en Estados Unidos sobre el grado de éxito de proyectos de Software. Los resultados publicados indican que de 50,000 proyectos desarrollados: 9,500 (19%) fracasaron, 26,000 (52%) fueron completados, pero con sobrecostos y/o sobretiempo; y sólo 14,500 (29%) fueron completados satisfactoriamente en el tiempo y presupuesto establecido (Standish Group, 2015). Esta información puede ser visualizada en la GRÁFICA 1, mostrada a continuación:

GRÁFICA 1
REPORTE DEL CAOS 2015
PORCENTAJE DE PROYECTOS SEGÚN ESTADO

Fuente: Elaboración propia basada en (Standish Group, 2015)

De igual forma, el reporte indica que los proyectos de menor tamaño tienen un mayor porcentaje de éxito que los proyectos de gran magnitud; y que los proyectos que utilizaron metodologías de tipo ágil fueron más exitosos que aquellos que utilizaron metodologías de tipo tradicional.

Hoy día, la Fábrica de Software de CIDITIC - FS ejecuta proyectos pequeños y medianos utilizando herramientas y técnicas conocidas por el equipo, para llevar a cabo la finalización de los proyectos (diseño de prototipos, elaboración de diagramas UML, diseño y seguimiento de cronograma de actividades, desarrollo, pruebas, entre otros). Desde la creación de la FS no se ha establecido una metodología formal de Proyectos de Software, tampoco cuenta con documentación específica del cómo se debe realizar las etapas de los proyectos; una estructura organizacional, roles del equipo de trabajo; ni controles definidos de: alcance, tiempo y costo. Igualmente, no se cuentan con las revisiones suficientes para evitar

el sobretiempo y que exista una trazabilidad documentada de los requerimientos desde el inicio hasta el final de proyecto. Esta falta de lineamientos trae como consecuencia la sobrecarga de trabajo sobre el personal para finalizar una tarea en particular, el retraso en la fase de desarrollo e implementación, pobre control de cambios de requerimientos y falta de comunicación con los usuarios principales.

1.4 PLANTEAMIENTO DEL PROBLEMA

La Sección de Fábrica de Software – FS de CIDITIC fue creada inicialmente con el objetivo de realizar proyectos de extensión (Sistema de Juego de Azar, acreditación de carreras, inventario, incidencias), seguido fue direccionado a proyectos de investigación (despliegue de resultados de muestreos, recepción y divulgación de proyectos de investigación, congresos, foros y seminarios) y por último a proyectos institucionales (memoria, gestión de solicitudes de apoyo para eventos nacionales e internacionales y automatización de procesos a las unidades de la UTP). Contar con una diversidad de proyectos, ocasionó que la FS tome más tiempo de lo debido para avanzar en el proyecto, ya que el equipo debe capacitarse en el tema: estudiando el proceso actual, la problemática y la búsqueda de la solución según sea el área del proyecto.

El número de proyectos asignados a la FS varían entre cinco (5) a trece (13) proyectos con un personal alrededor de tres (3) a cuatro (4) colaboradores; esto significa que frecuentemente es necesario asignar a una sola persona en uno o más proyectos y el coordinador como analista en todos los proyectos. Al iniciar el proceso de desarrollo del proyecto de Software, se realiza una reunión exploratoria con el cliente para conocer del proyecto. En esta reunión, el cliente asigna a un Interesado encargado de brindar toda la información necesaria para el desarrollo del proyecto. El interesado se encargará de determinar las necesidades, validar los requerimientos establecidos y aceptar el prototipo. Al ser una sola persona que valida toda la información, existe la posibilidad que el producto al ser presentado a los demás interesados, no cumpla en su totalidad con las expectativas del grupo causando que algunos se sientan reacios a utilizar el producto de Software debido

a que consideran que no fueron incluidos en el proceso de desarrollo y por ende, el producto no cumple con lo requerido.

Otro punto fundamental que impacta el desarrollo de productos en FS, está relacionado con el nivel de compromiso del cliente y los interesados con los objetivos del proyecto. En ciertas ocasiones, se ha presentado clientes que desean un producto de Software, pero no participan activamente de las reuniones que requieren de su aprobación para avanzar en el proyecto. Esto trae como consecuencia que el proyecto no progrese o sea descontinuado.

También se dan escenarios donde la rotación de los interesados es muy común durante el desarrollo del proyecto, haciendo que el cliente asigne a un nuevo interesado en repetidas ocasiones llevando a cambios constantes en los requerimientos e impactando negativamente en el tiempo de entrega del producto (falta de claridad en las expectativas del producto causan retrasos en los tiempos de finalización). En ocasiones el proyecto puede permanecer inactivo por largos periodos de tiempo debido a falta de priorización por parte del interesado o el cliente.

Por otro lado, cuando inició la FS, el coordinador inicial de la sección fue trasladado a otra unidad a pocos meses de haber iniciado, lo que causó que muchos proyectos fueran gestionados sin ninguna coordinación o guía. Asimismo, los programadores contratados al inicio de la Fábrica eran estudiantes por culminar su carrera, y se basaban en sus conocimientos académicos para el desarrollo de productos. A lo largo de los años, el equipo construyó una mecánica de trabajo dentro de la sección a través de la aplicación de diversas herramientas y la incorporación de aquellas que conllevaran a los mejores resultados.

La Sección de FS actualmente posee un conjunto de técnicas y herramientas para desarrollar los proyectos de software; pero no cuenta con un procedimiento formal para su ejecución. Además, no posee una documentación que específica el cómo se debe realizar las etapas de los proyectos; la estructura organizacional, los roles del equipo de trabajo; y controles definidos de: alcance, tiempo y costo.

La carencia de procedimientos estructurados conlleva a que los proyectos sean propensos a fracasar o ser desafiantes.

Según el “Reporte de Caos” (Chaos Report) de Standish International Group del año 2003, definen los proyectos en tres (3) clasificaciones según su ejecución: Exitoso (Successful), es completado en su cronograma, presupuesto y con todas las características especificadas; Desafiante (Challenged), es completado y operacional, pero con sobrecostos, cronograma extendido y con pocas características especificadas; y Fracasado (Failed), es cancelado antes de ser completado (Zavala Ruiz , 2004).

Cabe destacar que durante varias administraciones se realizaban solicitudes de desarrollo de Software, sin considerar los lineamientos de trabajos utilizados. Durante el periodo 2013-2018 durante la gestión del Dr. Ramfis Miguelena, se solicita el uso de una metodología que permita un control y seguimiento más preciso del estatus de los proyectos. Por esta razón se selecciona como proyecto de investigación la elaboración de una metodología ágil de proyectos de Software que se adapte a las necesidades principales de la Sección de Fábrica de Software para mitigar incidentes de alcance, tiempo y costo.

Preguntas al problema

Por el planteamiento del problema anterior, se elabora las siguientes interrogantes:

- *¿Influye el uso de una metodología ágil de proyectos de Software en la Sección de Fábrica de Software en cuanto a la gestión de proyecto en los productos de Software desarrollado?*
- *¿Aumentará el nivel de satisfacción del cliente al aplicar una metodología de Proyectos de Software a la Sección de Fábrica de Software?*

1.5 JUSTIFICACIÓN

Es necesario contar con una metodología, que permita documentar y controlar los procesos requeridos para ejecutar adecuadamente el desarrollo de Software. Esta debe promover la comunicación con el cliente, la comunicación en el equipo de

trabajo, la asignación de tareas, la documentación necesaria para validar decisiones y el esfuerzo realizado.

La FS requiere de una metodología de proyectos de Software, compuesta de buenas prácticas existentes de las metodologías de proyectos de software ágil; que permitan desarrollar Software de manera más efectiva: cumpliendo con la satisfacción del cliente con respecto a: alcance, tiempo esperado y presupuesto acordado. Una metodología que permita gestionar los cambios en el producto de Software y promueva la comunicación con el cliente y el equipo de trabajo.

El establecimiento de una metodología base permitirá que la misma pueda ser ajustada y mejorada de acuerdo con las necesidades del FS. De encontrarse algún proceso que no genere los resultados esperados, será posible realizar una retrospectiva con el equipo de trabajo y aplicar otra técnica del mismo ámbito en los proyectos siguientes hasta contar con el método correspondiente.

Esto traerá como beneficio que el equipo de trabajo, siempre se mantenga actualizado con los diferentes métodos existentes permitiendo por medio de la experiencia conocer cuáles son los adecuados según el escenario.

La metodología igualmente servirá como referencia para otros equipos de desarrollo que cuenten con características parecidas a la Sección de Fábrica de Software y podrán acondicionarla según sus necesidades.

Por otro lado, al contar con una metodología de proyectos de Software se podrá ofrecer la posibilidad a los estudiantes la oportunidad de participar en proyectos reales de la institución y así adquirir experiencia profesional; obteniendo nuevas capacidades y aptitudes en el trabajo en equipo, análisis, desarrollo e implementación.

1.6 OBJETIVOS

Para la ejecución del trabajo se define el objetivo general y específicos.

1.6.1 Objetivo General

Aplicar la metodología ágil como instrumento para mejorar la gestión de proyectos de software para la Sección de Fábrica de Software de CIDITIC y mejorar la gestión del alcance, tiempo, costo y el nivel de satisfacción del cliente del proyecto.

1.6.2 Objetivos Específicos

1- Realizar un diagnóstico en la Sección de Fábrica de Software para identificar las oportunidades de mejora y buenas prácticas que puedan ser producidas en los procesos de proyecto.
2- Evaluar el enfoque tradicional y ágil de las metodologías para seleccionar el enfoque adecuado según las necesidades de la Sección de Fábrica de Software.
3- Comparar las metodologías de proyectos de Software en diferentes criterios de gestión de proyecto (alcance, costo, tiempo, riesgos, comunicación del cliente, equipo, documentación, etc.) para identificar los mejores atributos de cada uno.
4- Elaborar y aplicar una encuesta que permita al equipo de Fábrica de Software y expertos de Ingeniería de Software evaluar las metodologías en los criterios comparados y seleccionar las metodologías más convenientes para la Sección de Fábrica de Software.
5- Seleccionar la metodología ágil de proyectos de Software de acuerdo con los resultados de la encuesta, estableciendo las fases del ciclo de vida del proyecto describiendo sus entradas, herramientas, técnicas y salidas.

1.7 HIPÓTESIS GENERAL

De acuerdo con las necesidades para llevar a cabo un mejor desarrollo y manejo de proyectos de Software se señala como hipótesis descriptiva el siguiente enunciado:

El uso de una metodología ágil como un instrumento para la gestión de proyectos de Software mejorará la elaboración del Software con relación a alcance, tiempo, costo y satisfacción del cliente en la FS de CIDITIC de la UTP.

1.8 ALCANCE

El trabajo de investigación, Metodología ágil de proyectos de Software para la Sección de Fábrica de Software de CIDITIC abarca la fase de inicio, planificación, diseño y retrospectiva del proyecto. Cada fase contendrá un conjunto de técnicas y herramientas que permitirán ejecutar las actividades requeridas para avanzar a las fases siguientes. La metodología se enfocará en la definición de los requerimientos, el diseño del prototipo, la retroalimentación y la aprobación por el cliente; y conjuntamente el seguimiento del tiempo y presupuesto del proyecto.

1.9 DELIMITACIÓN

Nuestro trabajo se enmarca en la evaluación y selección de una Metodología ágil de Proyectos de Software para proyectos pequeños y medianos en la Universidad Tecnológica de Panamá, sede Campus Víctor Levi Sasso, en la Sección de Fábrica de Software del Centro de Investigación, Desarrollo e Innovación en Tecnologías de la Información y las Comunicaciones – CIDITIC.

1.10 LIMITACIONES O RESTRICCIONES DE LA INVESTIGACIÓN

1.10.1 Limitaciones

La limitación encontrada durante el desarrollo del trabajo de investigación fue la falta de respuesta en el proceso de encuestas con los Expertos de

Ingeniería de Software para la recopilación de información (encuesta de evaluación de metodologías).

1.10.2 Restricciones

La Metodología propuesta que sería aplicada en la Sección de Fábrica de Software de CIDITIC, no pudo ser ejecutada por motivo que los miembros de la FS fueron trasladados a la Dirección General de Innovación y Tecnología Educativa - DiGITED y a la Dirección General de Tecnología de la Información y Comunicaciones - DITIC.

1.11 APORTACIONES DE LA TESIS

La implementación de una metodología ágil para Proyectos de Software en la Sección de Fábrica de Software de CIDITIC, permitirá un mejor control y seguimiento del alcance, tiempo y presupuesto establecidos, resultando que el cliente esté mayor informado de los avances del proyecto y tenga la potestad de tomar decisiones durante su ejecución. Al contar con más información del proyecto, se facilitará la elaboración de informes de avances para la Dirección que le permitirá una mejor y oportuna toma de decisiones.

La aplicación de metodología contribuirá a mejorar la imagen del Centro de Investigación CIDITIC, ya que promoverá la adopción de buenas prácticas en el desarrollo de aplicaciones, páginas web y aplicaciones móviles para el área de investigación. Adicionalmente, con su respectiva documentación la misma pueda ser actualizada y mejorada, ajustándose mayormente a las necesidades que se presenten a la Sección de Fábrica de Software posteriormente.

Este trabajo de investigación contempla la comparación y análisis de metodologías existentes, cuyos resultados pudiesen ser utilizados como insumos para futuras investigaciones. También se desplegará las experiencias de la Sección de la Fábrica de Software en los proyectos de desarrollo desde su inicio, remarcando los riesgos, vulnerabilidades y oportunidades de mejora.

1.12 ESTRUCTURA DE LA TESIS

El trabajo está conformado bajo la siguiente estructura:

Capítulo I – Marco Conceptual. En el primer capítulo se presenta la escogencia del tema, estado actual, planteamiento del problema, justificación, objetivo del trabajo, hipótesis general, alcance, delimitación del trabajo, limitaciones, restricciones y aportes del trabajo.

Capítulo II – Marco Teórico. Este capítulo describe el estado del arte haciendo referencia a qué es el Software, los tipos y ciclos de vida; los modelos de software existentes, las metodologías de Software, sus enfoques y comparación de las metodologías de proyectos de Software ágiles con respecto a diferentes criterios como: procesos, planificación, gestión de alcance, tiempo, costo, riesgos, pruebas, comunicación del cliente, comunicación entre el equipo de trabajo, documentación y a los tamaños de proyectos que van dirigidos. Por último, se contempla las características que debe contener una Fábrica de Software.

Capítulo III – Metodología de Investigación. En el capítulo número tres (3), se especifica el tipo de investigación, metodología, las variables, el resultado obtenido, su discusión, conclusiones y recomendaciones.

Capítulo IV – Sección de Fábrica de Software de CIDITIC. Se describe los antecedentes de la Sección de Fábrica de Software, la experiencia en los proyectos desarrollados, las oportunidades de mejora en la Sección y los factores de fracaso.

Capítulo V – Resultados de la Investigación. El último capítulo se centra en los resultados de la encuesta elaborada al equipo de Fábrica de Software con relación a las metodologías ágiles existentes en diferentes criterios de gestión de proyecto. Al completarse el análisis de resultados, se expone una propuesta de metodología de proyectos de Software para la Sección de Fábrica de Software, donde se detallará el proceso necesario para el desarrollo de los productos hasta la fase de revisión y retrospectiva.

1.13 CONCLUSIONES

En el marco conceptual desarrollado se presentó un conjunto de elementos que permiten conocer el escenario que será investigado en los próximos capítulos de este trabajo. Mediante el estado actual del escenario y el planteamiento de la problemática encontrada, es posible conocer de manera general qué factores debe enfocarse el trabajo; permitiendo así, definir los objetivos a seguir (general y específicos), delimitar el alcance meta, la hipótesis general a comprobar, especificar las restricciones del trabajo y las aportaciones más significativas de la investigación. Por otra parte, se presentó la estructura del trabajo, detallando la información o actividades que serán desarrolladas en cada sección.

Recolectada toda esta información, la misma será utilizada como base para iniciar este trabajo de investigación.

1.14 REFERENCIA BIBLIOGRÁFICA

Zavala Ruiz, J. M. (2004). ¿Por Qué Fracasan los Proyectos de Software?; Un Enfoque Organizacional. Congreso Nacional de Software Libre 2004.

Barzanallana, R. (30 de 12 de 2006). www.um.es. (Universidad de Murcia) Obtenido de http://www.um.es/docencia/barzana/IAGP/Iagp2.html

Standish Group. (4 de octubre de 2015). www.infoq.com. Obtenido de https://www.infoq.com/articles/standish-chaos-2015

CAPÍTULO II
MARCO TEÓRICO

2.1 INTRODUCCIÓN

Para la selección de una metodología ágil como un instrumento para mejorar la gestión de proyectos en la Sección de FS, es fundamental conocer con claridad ciertos conceptos bases utilizados en la Ingeniería de Software; razón por la en esta sección se definen términos generales como: Software, ciclo de vida del Software, ciclo de vida de desarrollo de Software, proceso de software, proceso de desarrollo de Software, Modelo de proceso de Software, etc. Se identificó que muchos de estos conceptos poseen significados relativamente similares, por lo que se procedió a definir cada término, comparar diferencias y determinar su nivel de relación entre cada uno.

Luego de definir claramente estos conceptos, se profundiza en los modelos de procesos de Software existentes y los enfoques de metodologías, tradicional y ágil, donde se presenta una comparación entre ambos enfoques y se identifica cuál enfoque es el más recomendable para la selección de la Metodología de Proyectos de Software en la Sección de FS. Identificado el enfoque, se seleccionan diez (10) metodologías, las cuales son comparadas en diferentes criterios de gestión de proyectos para determinar sus ventajas y desventajas. Estos criterios de comparación son: alcance, tiempo, costo, riesgos, comunicación del cliente, comunicación del equipo, pruebas y documentación

Y por último se revisan los conceptos y características de una Fábrica de Software.

2.2 SOFTWARE

El Software se conoce como la parte lógica, intangible que administra, controla e integra el hardware de un sistema a través de programas y procesos establecidos. Es definido por la IEEE en 1990 como *"programas y procedimientos de computación, posiblemente asociados con documentación y data relacionada a la operación de un sistema computacional"* (IEEE, 1990)*; l*a ISO amplió esta definición al describirlo como "*todo o una parte de programas, procedimientos, instrucciones y documentación asociada a un sistema de procesamiento de información*" (Prasad

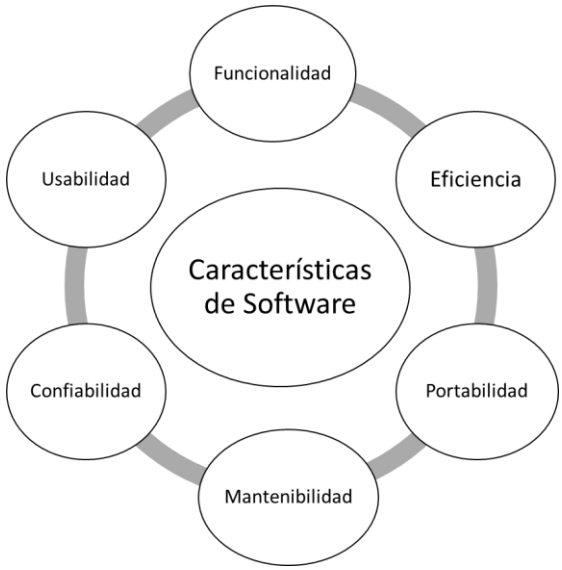
Mahapatra, 2016). Ambas definiciones aun teniendo 20 años de ser establecidas, aún continúan siendo válidas en la actualidad.

2.2.1 Características del Software

De acuerdo con la IEEE, adoptado igualmente por la ISO, la característica de un software es "*una cualidad, rasgo o propiedad inherente, posiblemente accidental del software (por ejemplo, su funcionalidad, desempeño, restricciones de diseño)"* (IEEE, 1990). Las características identificadas y su importancia serán notablemente diferentes para un usuario, un desarrollador, un proveedor o un cliente, ya que cada uno de ellos posee una perspectiva y necesidad diferente en el software.

Por lo que se agrupa estas características en las siguientes categorías (Prasad Mahapatra, 2016), tal como presenta la figura 1:

FIGURA 1
CARACTERÍSTICAS DEL SOFTWARE



Fuente: Basada en (Prasad Mahapatra, 2016, pág. 15)

A continuación, se detalla la descripción de cada característica:

- **Funcionalidad:** grado en que el desempeño de un software cumple con su propuesto propósito.
- **Confiabilidad:** capacidad del software de proveer las funcionalidades requeridas bajo condiciones específicas en periodos determinados.
- **Usabilidad:** grado en que un software es fácil de utilizar.
- **Eficiencia:** capacidad del software en utilizar los recursos del sistema en el modo más efectivo y eficiente.
- **Mantenimiento:** facilidad en que el software puede ser modificado para adicionar funcionalidades, mejorar desempeño del sistema o corregir errores.
- **Portabilidad:** facilidad en que desarrolladores de software pueden transferir el software de una plataforma a otra, sin (o con mínimos) cambios.
- **Robustez:** capacidad en que un software puede continuar a pesar de la introducción de entradas inválidas.
- **Integridad:** capacidad en que el Sistema pueda controlar modificaciones o accesos no autorizados al software.

2.2.2 Tipos de Software

Hoy en día, el software es aplicado en numerosas industrias, desde el monitoreo transaccional en tiempo real de las operaciones de un negocio hasta simulaciones para la optimización de uso de recursos en un puerto. El software puede ser utilizado en toda aquella situación donde exista una secuencia lógica de pasos establecidos, iteraciones y decisiones. Es por ello, más que ser clasificado por industrias, se clasifica usualmente por el rango potencial de sus aplicaciones (Punjab Technical University -PTU, 2006):

- **Software de sistemas:** administra y controla operaciones de un sistema computacional. Es un grupo de programas responsable de utilizar recursos informáticos de manera eficiente y efectiva. Por

ejemplo, software operativo es un software de sistemas que controla hardware, administra memoria y tareas múltiples, funcionando como la interface entre programas y aplicación y el computador.

- **Software tiempo-real:** observa, analiza y controla eventos en tiempo real. Generalmente, un sistema de tiempo real garantiza una respuesta a un evento externo en un periodo específico de tiempo.
- **Software de negocios:** utilizado ampliamente en administración y control de actividades empresariales. Este tipo de software brinda las bases para realizar operaciones y tomar decisiones de negocios eficientemente.
- **Software científico y de ingeniería:** ha surgido como una herramienta importante para brindar apoyo en actividades de investigación y desarrollo de tecnología. Este software es diseñado para realizar cálculos precisos de data numérica compleja que son captadas en tiempo real.
- **Software de inteligencia artificial:** Esta clase de software se utiliza para cuando la técnica de resolución de problemas es no-algorítmica en naturaleza. Estos problemas usualmente requieren de técnicas específicas como sistema experto, reconocimiento de patrones y de juego. El rol de este software es brindar cierto grado de inteligencia al hardware mecánico para realizar la tarea deseada con mayor rapidez.
- **Software basado en web:** funciona como interface entre el usuario e internet, incorporando instrucciones para la extracción de data en formato de texto, audio o video.
- **Software de computadores personales:** utilizado para uso oficial y personal en actividades diarias, y ampliamente utilizado en muchas industrias.

El software también puede ser clasificado por el grado en que los clientes están involucrados en el proceso de desarrollo:

- **Software comercial propietario:** no existe un usuario final comprometido. Los usuarios tienen poca o ninguna interacción con el

proveedor durante el desarrollo del software. También puede darse la variación donde luego de que el software es lanzado al mercado, un cliente solicite su personalización para un propósito específico.

- **Software personalizado:** el software es desarrollado para un usuario en específico, con términos establecidos bajo algún tipo de formal de contrato.

2.2.3 Ciclo de Vida del Software

El Ciclo de vida de un Software es el periodo de tiempo desde que un Software es creado hasta cuando el Software es descontinuado; generalmente el ciclo de vida del Software incluye una fase conceptual, fase de requerimientos, de diseño, implementación, pruebas, instalación, entrega, operación y mantenimiento; en ciertos escenarios hasta de retiro (IEEE, 1990). Las fases por la que el ciclo está conformado variarán de acuerdo con la estructura y características de cada empresa desarrolladora, e inclusive por cada proyecto de software a efectuar. A su vez, estas etapas pueden traslaparse o darse de manera simultánea; sin embargo, se considera un proceso como terminado hasta el momento en que sus entradas y salidas sean adecuadamente verificadas y validadas (IEEE, 2008).

Aun cuando estas fases no son específicas, en la FIGURA 2 se muestra un esquema general de las etapas por las cuales un software recorre usualmente durante su vida útil.

FIGURA 2.
CICLO DE VIDA DEL SOFTWARE

Fuente: Elaboración propia basada en (Prasad Mahapatra, 2016, pág. 26)

Observando la figura anterior se detalla cada fase del ciclo de vida del Software:

- **Conceptualización:** Fase donde se describe y se evalúa las necesidades del cliente mediante la documentación. Se establecen las necesidades, reporte de planes de avance, memorándum de iniciación de proyecto, estudio de factibilidad, procedimientos, etc. (IEEE, 1990).
- **Análisis de requerimientos:** Se estudian, analizan y evalúan los requerimientos del usuario. Se trabaja en conjunto con el cliente y usuarios finales para recopilar y especificar las características funcionales y no funcionales del software.
- **Diseño:** Etapa donde el software se diseña en base a los requerimientos establecidos. Se define la arquitectura, componentes, interfaces y las características del Sistema o componente. Este diseño responderá al problema del cliente (IEEE, 1990).
- **Implementación:** Se procede a la implementación de lo diseñado. El trabajo consiste mayormente en codificación junto con tareas de depuración. Usualmente esta fase traslapa con la etapa de prueba.
- **Pruebas:** Se evalúa el software en cualquier cantidad de condiciones para garantizar su desempeño, robustez y confiabilidad. Se garantiza que todos los diferentes módulos se integren con la aplicación y se prueba que las funcionalidades del software cumplan con los requerimientos establecidos.
- **Instalación y entrega:** Consiste en la instalación y paso a producción del software. Se hace entrega del software al cliente y se entrena al personal en su uso y mantenimiento.
- **Documentación:** Realizado durante todo el proceso de desarrollo. Permite contar con información oportuna para el usuario y sirve de soporte para desarrollos futuros y mantenimiento.
- **Mantenimiento:** Etapa donde el desarrollador brinda mantenimiento al sistema del cliente. Esto consiste desde realizar actualizaciones, brindar

mejoras hasta corrección de errores o fallas de emergencia en el sistema.

2.3 MODELOS DE PROCESOS DE SOFTWARE

¿Qué significa el término modelo? Un modelo en términos generales se puede definir como una representación a la realidad, un ideal digno a ser imitado, patrón o guía de acción. Es la representación de una propuesta ideal a seguir que muestra y explica las características de su estructura, sus elementos, mecanismos, procesos y la forma en que se interrelacionan entre ellas (Sesento García, 2008).

Un Proceso de Software, es un grupo de actividades que conllevan a la producción de un producto de software; es el marco de trabajo de las tareas que se requieren para la construcción de un software de Calidad. Entre las actividades principales que componen a un proceso de Software podemos mencionar: la especificación del Software, diseño e implementación del Software, validación del Software y evolución del Software. Los procesos de software pueden ser mejorados por medio de la estandarización, aplicando métodos, técnicas y buenas prácticas a los mismos (Sommerville, 2005).

Por consiguiente, se define como un Modelo de Procesos de Software, la visión y descripción resumida de un proceso de software; los mismos incluyen las actividades, los artefactos y el rol de los participantes involucrados en la elaboración del Software (Sommerville, 2005).

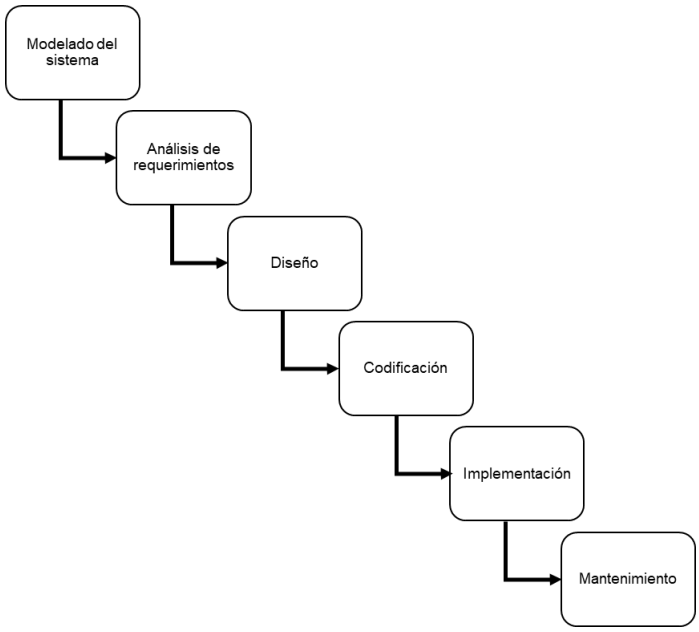
Los Modelos de Procesos de Software generales son:

- Modelo Cascada
- Modelo V
- Modelo Evolutivo
- Modelo de Prototipos
- Modelo Espiral
- Modelo Incremental
- Rapid Application Development Model (RAD)
- Modelo Basado en Componentes (CBSE) (Mohammed Ali Munassar & Govardhan, 2010)

2.3.1 Modelo en Cascada

Introducido primeramente por Royce en 1970. Este primer modelo recorre las fases de manera secuencial y linear, por lo que también se conoce como modelo linear secuencial. Cada fase solamente puede iniciar, después de haberse completado la anterior. Es un modelo lineal sin actividades de retroalimentación. Cada fase es finalizada y el resultado (o salida) de esa etapa se convierte en la entrada de la siguiente. Este modelo propone seis (6) etapas: modelado del sistema, análisis de requerimientos, diseño, codificación, implementación y mantenimiento; tal como muestra la FIGURA 3:

FIGURA 3.
ETAPAS DEL MODELO EN CASCADA



Fuente: Elaborado basado en (Prasad Mahapatra, 2016, pág. 44)

Esta metodología se basa de las ideas de *"definir antes de diseñar"*, *"diseñar antes de codificar",* es decir, para hacer uso de este modelo se debe contar con una clara especificación de requerimientos desde el inicio. Además, el producto en desarrollo solo será visualizado hasta las etapas finales.

A continuación, las ventajas y desventajas encontradas en el modelo Cascada:

TABLA 1
VENTAJAS Y DESVENTAJAS DEL MODELO CASCADA

Modelo Cascada	
Ventajas	**Desventajas**
• Es sencillo de entender y secuencial • Facilita la administración, ya que cada etapa cuenta con fechas de entrega y entregables establecidos • Facilita el control de programaciones, presupuesto y documentación.	• Los requerimientos deberán ser especificados antes que el desarrollo pueda iniciarse. • Los cambios en los requerimientos en las siguientes fases son complejas y poco probables. Significando que una vez el sistema esté en prueba será difícil realizar modificaciones. • El cliente no está involucrado en la etapa de desarrollo y la versión del software estará disponible una vez éste haya sido desarrollado. • No considera administración del riesgo. • Asume que los requerimientos son estables y constantes a lo largo de la duración del proyecto.

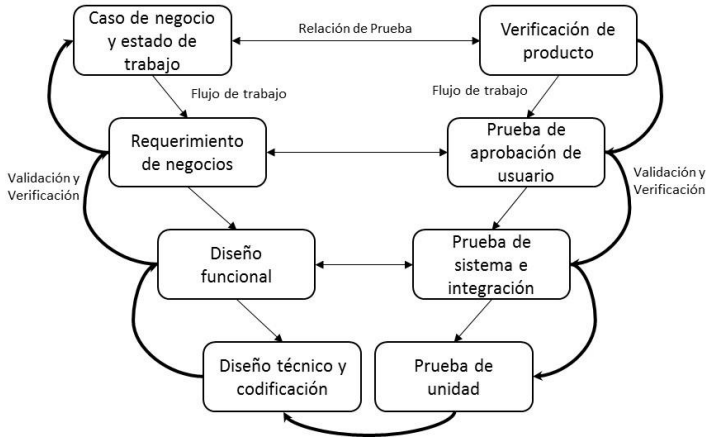
Fuente: Elaboración propia basada en (Prasad Mahapatra, 2016)

2.3.1.1 Modelo en V

Es una variante del Modelo Cascada, este modelo asegura que cada etapa en el desarrollo de software esté asociada con la realización de pruebas. El modelo V se divide en dos (2) secciones: derecha e izquierda. La sección izquierda analiza y determina los requerimientos del software a desarrollar. La sección derecha incluye las actividades de pruebas. Ambas secciones funcionan concurrentemente.

Lo anteriormente mencionado se presenta en la FIGURA 4. Etapas del Modelo V:

FIGURA 4.
ETAPAS DEL MODELO V



Fuente: Basado en (Prasad Mahapatra, 2016)

A continuación, las ventajas y desventajas encontradas en el modelo V:

TABLA 2
VENTAJAS Y DESVENTAJAS DEL MODELO V

Modelo V	
Ventajas	**Desventajas**
• Abarca todas las áreas funcionales.	• Los procesos son fijos durante el proyecto, pero cuando finaliza el

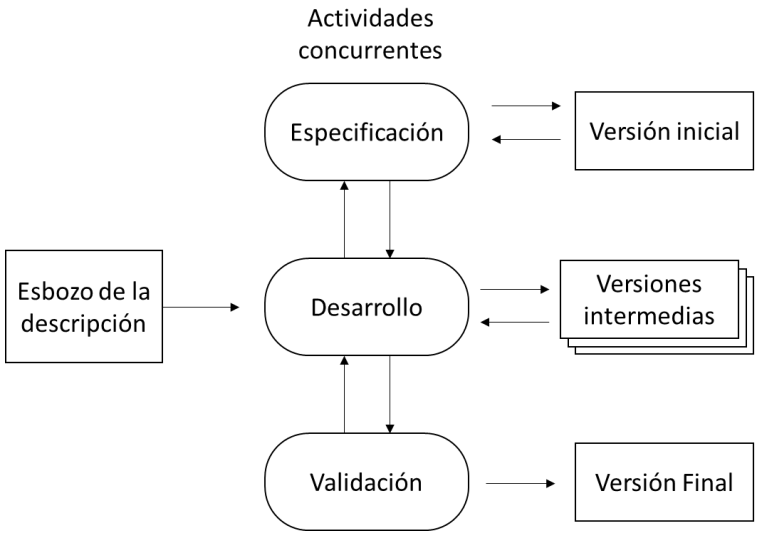
Modelo V	
Ventajas	**Desventajas**
• Enfatiza el significado de las pruebas y su planificación.	proyecto, estos procesos se eliminan. • La realización de pruebas en cada etapa hace este modelo costoso.

Fuente: Elaboración propia basada en (Prasad Mahapatra, 2016)

2.3.2 Modelo Evolutivo

Los modelos evolutivos son iterativos, los mismos se caracterizan por permitir desarrollar versiones más completas del Software (Pressman, 2010). Se enfocan en una implementación inicial, el cual es expuesta al usuario y refinada por medio de versiones de mejora que permite llegar al producto final deseado. El Modelo evolutivo está compuesto por tres (3) actividades que se entrelazan entre ellas: especificación, desarrollo y validación; tal como muestra la FIGURA 5:

FIGURA 5
MODELO EVOLUTIVO



Fuente: Elaboración propia basada en (Pressman, 2010)

Existen dos (2) modelos que se basan en el modelo evolutivo: Modelo de Prototipos y Modelo Espiral.

2.3.2.1 Modelo de Prototipos

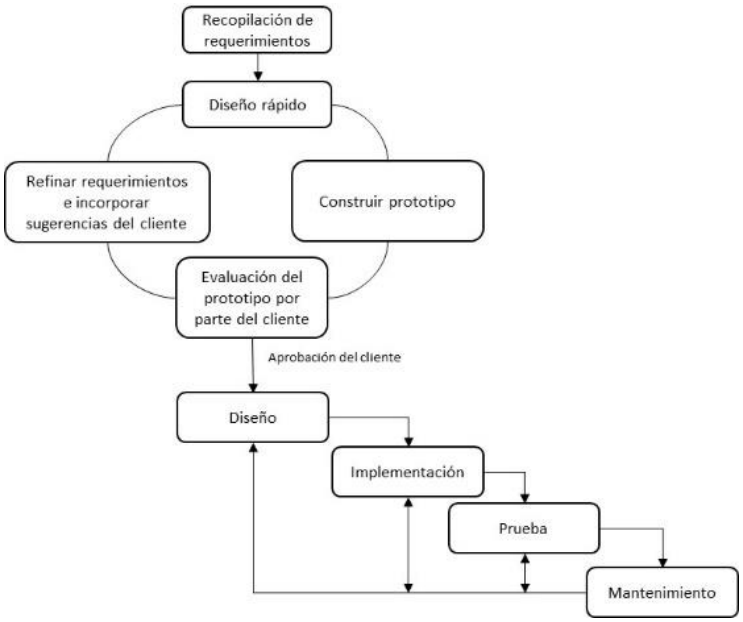
Se basa en la suposición en que es difícil conocer todos los requerimientos de un software al inicio de un proyecto. Brinda flexibilidad en el proceso de desarrollo permitiendo al usuario interactuar con el producto prototipo. Sin embargo, debido a esta flexibilidad, éste podrá ser descartado o modificado durante su diseño hasta cumplir con las expectativas del cliente.

Un prototipo puede prepararse de diferentes formas:

- Crear las interfaces de usuario sin ningún código substancial con el objetivo de que el usuario pueda ver cómo el sistema se verá.
- Abreviando una versión del sistema que efectuará un número limitado de funciones.
- Utilizando componentes de sistemas para demostrar las funciones que estarán incluidos en el sistema en desarrollo.

Este modelo consiste en seis (6) pasos: recopilación y análisis de requerimientos, diseño rápido, construcción de prototipo, evaluación de usuario, refinamiento de prototipo, diseño, implementación, prueba, y mantenimiento**.** Ver FIGURA 6:

FIGURA 6.
ETAPAS DEL MODELO DE PROTOTIPOS



Fuente: Basada en (Prasad Mahapatra, 2016)

A continuación, en la TABLA 3 se despliega las ventajas y desventajas encontradas en el modelo de Prototipos:

TABLA 3.
VENTAJAS Y DESVENTAJAS DEL MODELO DE PROTOTIPOS

Modelo de Prototipos	
Ventajas	**Desventajas**
• Provee el trabajo en proceso al usuario en etapas tempranas permitiendo la mejora en los requerimientos del usuario. • Los programadores adquieren experiencia desarrollando los	• Desarrolla y desecha los prototipos hasta que el prototipo deseado sea realizado. Esto puede llegar a ser costoso y demandar mucho tiempo. • El programador puede perder el enfoque de desarrollo y puede aplicar métodos poco

prototipos, resultando en una mejor implementación de los requerimientos. • Se reduce la ambigüedad en los requerimientos y se mejora la comunicación entre el desarrollador y el usuario. • Existe una gran participación de los usuarios en el proceso de desarrollo. • Permite reducir los riesgos asociados al proyecto.	eficientes o algoritmos inadecuados. • Los prototipos pueden dar falsas expectativas, haciendo pensar al usuario que está terminado. • El diseño e integración de sistema puede ser complejo, debido a que está construido en una serie de capas donde no se considera su integración con los demás componentes desarrollados.

Fuente: Elaborado basándose de (Prasad Mahapatra, 2016)

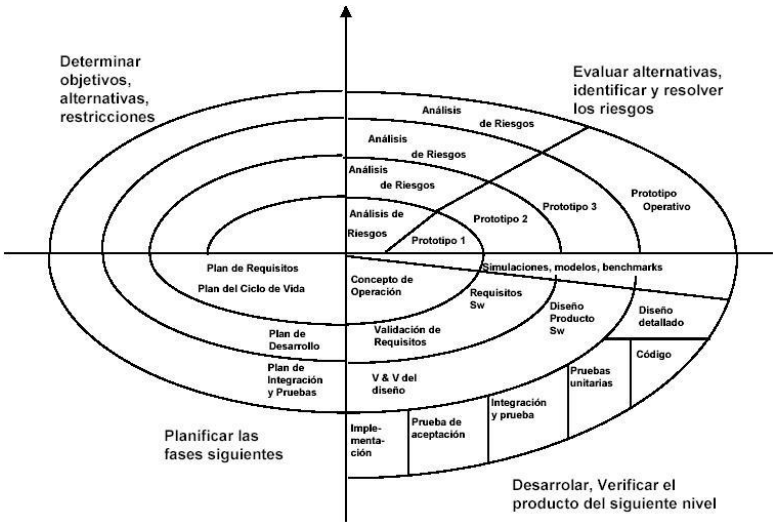
2.3.2.2 Modelo Espiral

Este modelo como su nombre lo señala está organizado en un espiral y el mismo está compuesto por un número de ciclos. El modelo combina los modelos de cascada y de prototipos, lo cual lo hace ideal para proyectos de gran tamaño, complejos y costosos. El modelo considera los problemas de requerimiento en el desarrollo de prototipos y administra el riesgo en cada una de las etapas del ciclo.

Como se observa en el esquema, el modelo realiza las actividades de requerimientos de análisis, diseño, integración y prueba, de manera iterativa hasta completar el software. La dimensión radial representa el costo incurrido en cada una de las etapas recorridas,

y la dimensión angular representan el avance para cada ciclo en la espiral.

FIGURA 7.
ESQUEMA DEL MODELO EN ESPIRAL



Fuente: (Pressman, 2010)

A continuación, las ventajas y desventajas encontradas en el modelo Espiral:

TABLA 4.
VENTAJAS Y DESVENTAJAS DEL MODELO ESPIRAL

Modelo Espiral	
Ventajas	**Desventajas**
• Evita problemas relacionados con el riesgo. • Especifica un mecanismo para actividades de aseguramiento de calidad del software. • Utilizado para proyectos complejos y dinámicos. • La reevaluación de cada etapa, permite realizar	• Evaluación del riesgo del proyecto y su solución no es sencillo. • Dificultoso estimar presupuesto y duración a los inicios del proyecto, ya que parte del análisis no se realiza hasta que parte del software es diseñado.

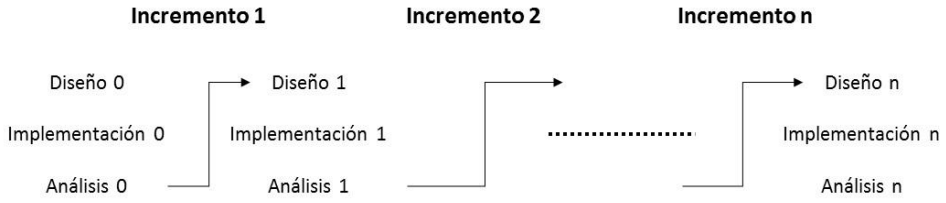
ajustes de tecnología, financieros o por parte del cliente. • Estimación de presupuesto y duración puede validarse a medida que el proyecto avanza.	

Fuente: Elaborado propia basándose en (Prasad Mahapatra, 2016)

2.3.3 Modelo Incremental

Conocido también como modelo iterativo mejorado, éste cuenta con las características de un modelo en cascada iterativo. El mismo está compuesto por fases, donde cada una de las mismas produce un incremento. Los incrementos son pequeños módulos del producto a entregar. Estos incrementos son identificados al inicio del proceso de desarrollo y son realizados a lo largo del proyecto hasta la entrega del software. El proceso iterativo, el cual incluye la entrega de incrementos al cliente, continúa hasta que el software haya sido completamente terminado. El esquema del modelo incremental se presenta en la FIGURA 8.:

FIGURA 8.
ESQUEMA DEL MODELO INCREMENTAL



Fuente: (Prasad Mahapatra, 2016)

La idea principal de este modelo es identificar los requerimientos e iterativamente refinar los mismos hasta que el producto esté finalizado. Se

conoce como “producto cero” el resultado del primer incremento, el cual ayudará a elaborar el plan del siguiente incremento.

De igual forma, como se presenta en el modelo de prototipos, en el modelo incremental también se recibe retroalimentación por parte del cliente. A continuación, las ventajas y desventajas encontradas en el modelo Incremental:

TABLA 5.
VENTAJAS Y DESVENTAJAS DEL MODELO INCREMENTAL

Modelo Incremental	
Ventajas	**Desventajas**
• Evita problemas relacionados con riesgo. • Realiza incrementos a través de refinamientos sucesivos. • Realiza análisis de costo-beneficios antes de aplicar mejoras al software. • Crece incrementalmente orientado hacia la solución con cada iteración. • No es un modelo complejo. • Se cuenta con retroalimentación temprana debido a que la implementación se da para una pequeña porción del software.	• Requiere planificación al nivel gerencial y técnico. • Se torna inválido cuando hay restricciones de tiempo en el proyecto o cuando el usuario no acepta los entregables de cada fase.

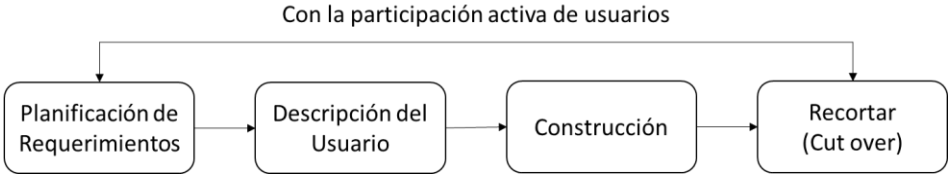
Fuente: (Prasad Mahapatra, 2016)

2.3.4 Rapid Development Model (RAD)

El modelo RAD se basa en el modelo incremental, desarrollado por IBM en los años 80 (Prasad Mahapatra, 2016). El mismo se caracteriza por desarrollar componentes o funciones en paralelo como si fueran proyectos

pequeños. Para la fase de desarrollo se utiliza timeboxing, en donde los productos desarrollados, son entregados e integrados al prototipo en proceso; lo que permite que el cliente pueda constantemente retroalimentar los requerimientos y los productos entregados (ISTQB Certification, s.f.).

FIGURA 9.
MODELO RAPID DEVELOPMENT MODEL



Fuente: (Prasad Mahapatra, 2016)

El proceso del modelo inicia mediante el desarrollo de un prototipo en base a los requerimientos iniciales; completado el prototipo inicial el mismo es presentado al usuario que hace una retroalimentación al mismo, obteniendo como resultado el refinamiento del prototipo. Este proceso sigue hasta que todos los requerimientos identificados estén plasmados en el prototipo. Para este proceso es posible utilizar técnicas de comunicación como sesiones de lluvia de trabajo, QFD (Qualify function deployment), proceso para identificar el nivel de prioridad de los requerimientos de los usuarios, etc. (Prasad Mahapatra, 2016).

A continuación, las ventajas y desventajas encontradas en RAD:

TABLA 6.
VENTAJAS Y DESVENTAJAS DE RAD

Modelo Rapid Development Model	
Ventajas	**Desventajas**
• Reduce el tiempo de desarrollo. • Incrementa la reusabilidad de los componentes. • Rápidas observaciones desde el inicio. • Motiva la retroalimentación del cliente. • La integración desde el inicio resuelve múltiples problemas de integración.	• Depende del rendimiento del equipo y de los miembros. individualmente para identificar los requerimientos de negocio. • Solo los sistemas que pueden ser divididos en módulos pueden utilizar este modelo. • Requiere desarrolladores/ diseñadores de experiencia. • Gran dependencia en las habilidades de modelado. • No es recomendable para proyectos de bajo presupuesto, ya que modelar los procesos y generar código automáticamente es de costo alto.

Fuente: (ISTQB Certification, s.f.)

2.3.5 Modelo basado en Componentes (CBSE)

El Modelo se centra en la reutilización de software diseñado. Estos son modificados según lo necesario y son incorporados al sistema en desarrollo. Esta técnica permite un desarrollo rápido de sistemas, pero se debe poseer una vasta cantidad de componentes de software reutilizables o componentes que son sistemas por sí mismos y brindan funcionalidades específicas (Sommerville, 2005).

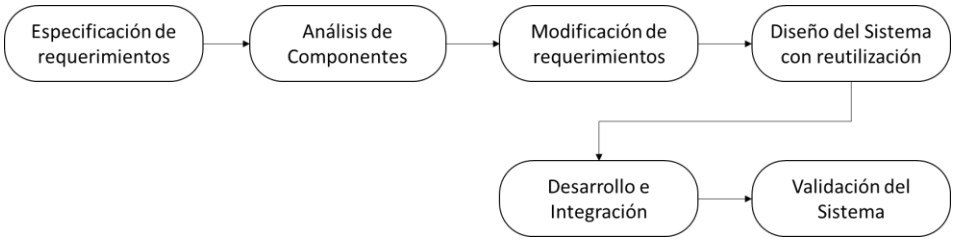
Al momento de seleccionar los componentes a utilizar según los requerimientos identificados, el equipo debe realizarse las siguientes preguntas:

1. *¿Existen Sistemas Comerciales (COTS) disponibles para implementar el requerimiento?*
2. *¿Existen componentes desarrollados internamente reusables para implementar el requerimiento?*
3. *¿Las interfaces son compatibles con los componentes disponibles dentro de la arquitectura que se construirá?*

Los equipos intentan modificar o remover aquellos requerimientos que no pueden ser implementados con los componentes propios o comprados (Verma, 2014).

Este modelo se enfoca en primordialmente en las etapas de diseño, desarrollo e integración. En la siguiente figura se presenta el Modelo basado en componentes:

FIGURA 10.
MODELO BASADO EN COMPONENTES



Fuente: Elaborado basándose de (Sommerville, 2005)

A continuación, las ventajas y desventajas encontradas en el modelo Basado en Componentes:

TABLA 7.
VENTAJAS Y DESVENTAJAS DEL MODELO BASADO EN COMPONENTES

Modelo basado en Componentes	
Ventajas	**Desventajas**
• Se reduce la cantidad de Software a desarrollarse. • Se reduce los costos y los riesgos. • Permite la entrega rápida de Software.	• La evolución del sistema puede estar en riesgo si los componentes están bajo el control de otra organización. • Encontrar los componentes que se adapten a los requerimientos puede ser complejo.

Fuente: (Sommerville, 2005)

2.4 METODOLOGÍA DE DESARROLLO DE SOFTWARE

Una metodología de desarrollo de Software (Software Development Methodology - SDM) es un marco de trabajo utilizado para planificar, estructurar y llevar control de los procesos de desarrollo de un sistema (SEGUE Technologies, 2015).

Otra definición para SDM es "*colección de procedimientos, técnicas, herramientas y documentación que ayudarán a los desarrolladores en sus esfuerzos a implementar un sistema de información*" (Geambasu, Jianu, Jianu, & Gavrila, 2011).

Existe una variedad de metodologías, cada una con características distintas o similares que no necesariamente son adecuadas para todo tipo de proyectos; cada una de ellas se adaptan a ciertos tipos de proyectos, con variaciones técnicas, organizacionales o según el tipo de equipo de trabajo (SEGUE Technologies, 2015).

Contar con un SDM permite a un proyecto realizar mejores estimaciones de recurso, entrega de productos más estables, un cliente mejor informado, mejor entendimiento de los requerimientos y tareas a realizar, identificación de riesgos, contar con planes de contingencia, etc. No contar con la misma, frecuentemente conlleva a que los sistemas sean entregados con fecha de atraso, sobrecosto y en

muchos casos no cumple con las necesidades ni expectativas del usuario final, llevándolo a ser un proyecto desafiante o fracasado (SEGUE Technologies, 2015).

2.4.1 Enfoques

Los SDM cuentan con dos (2) categorías o enfoques: tradicionales o peso pesado y ágiles o peso ligero (Geambasu, Jianu, Jianu, & Gavrila, 2011).

2.4.1.1 Tradicional

El enfoque tradicional se caracteriza por tener una alta estructuración y planeación de las actividades del proceso de desarrollo a efectuar desde el inicio del proyecto. Las metodologías de este enfoque requieren una definición clara del proceso de desarrollo de sistemas, se basan en una documentación comprensiva que se traduce en actividades más predictivas y eficientes. Las metodologías tradicionales pueden ser exitosas si son utilizadas para desarrollar sistemas donde los requerimientos son claros desde el inicio.

2.4.1.2 Ágil

Metodologías que poseen un proceso de desarrollo incremental de fácil compresión y adaptación a los cambios, donde existe una cooperación y comunicación constante entre el cliente y los desarrolladores (Geambasu, Jianu, Jianu, & Gavrila, 2011).

En febrero 2001, diecisiete (17) representantes de las metodologías ágiles deciden formar una alianza ágil (Agile Alliance) para promover sus puntos de vistas; generando el "*Manifiesto ágil*" (Agile Alliance, 2001).

El mismo está compuesto por doce (12) principios del desarrollo de Software ágil que serán mencionados en el siguiente listado:

1) Nuestra máxima prioridad es satisfacer al cliente a través de la entrega temprana y continua de Software valioso.

2) Bienvenido requisitos cambiantes, incluso tarde en el desarrollo. Los procesos ágiles aprovechan el cambio para obtener ventajas competitivas del cliente.
3) Entregar software que trabaja con frecuencia desde un par de semanas a un par de meses, con una preferencia a la escala de tiempo más corto.
4) La gente de negocios y desarrolladores deben trabajar juntos todos los días durante todo el proyecto.
5) Construir proyectos en torno a individuos motivados. Darles el ambiente y el apoyo que necesitan y confiar en hacer el trabajo.
6) El método más eficiente y eficaz de transmitir información hacia y dentro de un equipo de desarrollo es la conversación cara a cara.
7) Software que funciona es la principal medida de progreso.
8) Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.
9) La atención continua a la excelencia técnica y el buen diseño mejora la agilidad.
10) Simplicidad – el arte de maximizar la cantidad de trabajo no realizado – es esencial.
11) Las mejores arquitecturas requisitos y diseños emergen de equipos autoorganizados.
12) A intervalos regulares, el equipo reflexiona sobre cómo ser más eficaces, luego afina y ajusta su comportamiento respectivamente (Agile Alliance, 2001).

2.4.1.3 Comparación entre el enfoque tradicional y ágil

Comprendidos los enfoques de las metodologías, se procede a realizar una comparación de características entre ambos enfoques, con el objetivo de identificar cuál es el más conveniente para la Sección de Fábrica de Software de CIDITIC:

TABLA 8.
COMPARACIÓN ENTRE EL ENFOQUE TRADICIONAL Y ÁGIL

Enfoque Tradicional y Ágil		
Criterios	**Tradicional**	**Ágil**
Hipótesis fundamental	Los sistemas son completamente especificados, predecibles y se desarrollan por medio de una planificación detallada	Software de alta calidad y adaptativo, es desarrollado por pequeños equipos que utilizan el principio del mejoramiento continuo del diseño y pruebas basadas en la retroalimentación
Énfasis	Orientado a los procesos	Orientado a las personas
Tamaño de Proyecto	Grandes	Medianos y Pequeños
Tamaño del Equipo	Grande	Pequeño y creativo
Comunicación con el cliente	Baja	Alta
Estilo de Gestión	Es Autocrático, una persona toma las decisiones y los demás miembros lo siguen	Descentralizada, distribuida entre los miembros del equipo
Control de Calidad	Planificación compleja y controlada estrictamente	Control permanente en los requerimientos, diseño y su solución
Medida de Éxito	Entrega de acuerdo con la planificación	Entrega de valor agregado al negocio
Aceptación al cambio	Detiene la inserción de nuevos	Es flexible y se adapta a las

Enfoque Tradicional y Ágil		
Criterios	**Tradicional**	**Ágil**
	requerimientos y es reacio al cambio	necesidades del proyecto
Equipo de Trabajo	Orientados a la planificación, con habilidades adecuadas y acceso a conocimientos externos	Conocimiento avanzado, analítico y colaborativo
Habilidades adicionales requeridas por el equipo de trabajo	Nada en particular	Habilidades interpersonales y básico conocimiento en negocios
Documentación	Mayor documentación es requerida	Menos documentación es requerida
Comunicación entre el equipo de trabajo	Baja	Alta
Pruebas	Después que la codificación es terminada	Cada iteración desarrollada

Fuente: (Stoica, Mircea, & Ghilic-Micu, 2013)

Es posible señalar en el cuadro comparativo anterior, que cada enfoque va dirigido a proyectos con distintas características. Por un lado, el enfoque tradicional se centra en gestionar los proyectos autocráticamente desarrollando las actividades de forma secuencial, con toda la documentación posible y requerida; y el enfoque ágil se concentra en gestionar el proyecto de manera descentralizada, efectuando iteraciones de desarrollo con el apoyo del cliente quien verifica que los requerimientos se ajusten a sus necesidades y la documentación necesaria.

Por otra parte, el Reporte del Caos del 2015 (Chaos Report) desarrollado por la empresa Standish Group, organización de

asesoramiento de investigación enfocada en el rendimiento en los proyectos de Software, indican que en los proyectos donde se utilizaron metodologías ágiles obtuvieron un porcentaje mayor de éxito a aquellos que utilizaron metodologías tradicionales (Standish Group, 2015). En la siguiente tabla (TABLA 9) se muestra los resultados publicados por la empresa:

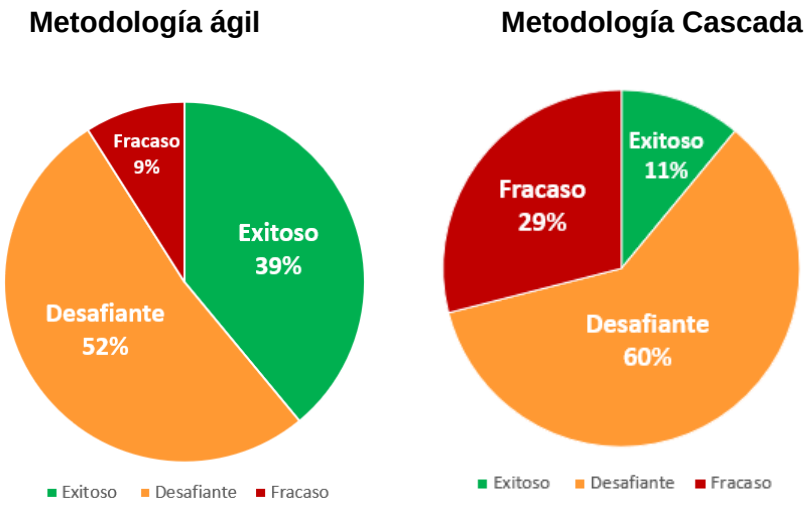
TABLA 9.
PORCENTAJE DE PROYECTOS EXITOSOS, DESAFIANTES Y FRACASADOS POR MÉTODO Y TAMAÑO DE PROYECTO

Tamaño	Método	Exitoso	Desafiante	Fracaso
Proyectos de todos los tamaños	Ágil	39%	52%	9%
	Cascada	11%	60%	29%
Proyectos Grandes	Ágil	18%	59%	23%
	Cascada	3%	55%	42%
Proyectos Medianos	Ágil	27%	62%	11%
	Cascada	7%	68%	25%
Proyectos Pequeños	Ágil	58%	38%	4%
	Cascada	44%	45%	11%

Fuente: Basado en (Standish Group, 2015)

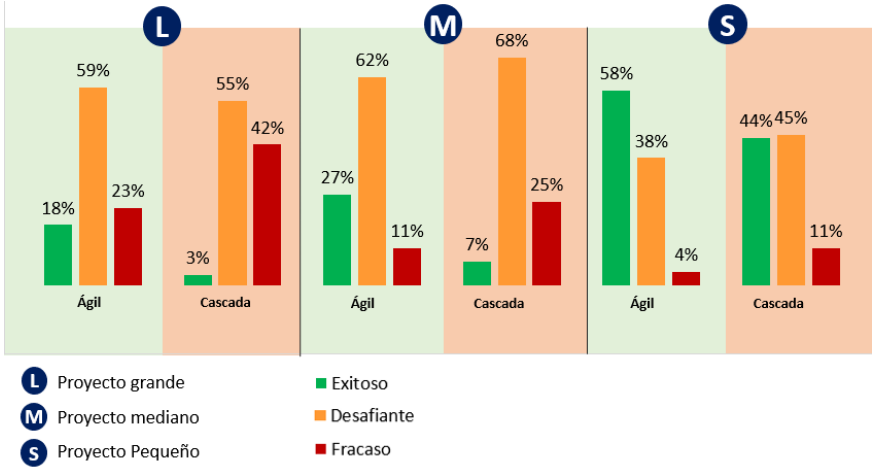
El estudio hace referencia a más de 10,000 proyectos desarrollados a partir del año 2011 hasta el año 2015, clasificándolos por enfoque metodológico (ágil o cascada).

GRÁFICA 2.
PORCENTAJE DE PROYECTOS EXITOSOS, DESAFIANTES Y FRACASADOS UTILIZANDO METODOLOGÍAS ÁGILES Y TRADICIONALES



Fuente: Elaboración propia basada en (Standish Group, 2015)

GRÁFICA 3.
PORCENTAJE DE PROYECTOS EXITOSOS, DESAFIANTES O FRACASADOS SEGÚN EL TAMAÑO DE PROYECTO Y SU MÉTODO



Fuente: Elaboración propia basada en (Standish Group, 2015)

En las gráficas anteriores se puede denotar que los porcentajes de proyectos que se registraron como exitosas en todos los tamaños

del proyecto aplicaron la metodología ágil; los proyectos ágiles tuvieron un 39% de éxito, 52% desafiantes y un 9% fracasó; a diferencia del tradicional con un porcentaje de éxito de 11%, desafiante 60% y fracaso 29%. El enfoque con mayor número de proyectos terminados ya sean exitosos o desafiantes, fue el ágil con un 91%; en comparación con, el enfoque tradicional que presentó un 71% de completados.

2.5 METODOLOGÍAS ÁGILES PARA LA GESTIÓN Y DESARROLLO DE PROYECTOS DE SOFTWARE

Un proyecto cuenta con un conjunto de atributos básicos (alcance, metas, limitaciones y características propias) que debemos conocer para su adecuada ejecución. Los proyectos pueden presentar características similares; pero en sí, cada uno cuenta con sus propias particularidades Por ello, existen múltiples metodologías que pueden ser aplicadas de acuerdo al tipo de proyecto y sus necesidades. Siempre recordando que todas las metodologías siguen el mismo principio “conseguir mejorar el proceso constantemente para poder optimizar los resultados del proyecto” (mainss, 2017).

El uso de una metodología de proyectos brinda la oportunidad de:

- Dividir el proyecto en escenarios más pequeños y manejables.
- Medir el progreso en términos de tiempo, costo y calidad.
- Tomar las correctivas necesarias en caso de presentarse desviaciones en el proyecto.
- Distribuir el recurso (humano y físico) en el proyecto.

Actualmente, existe una gran variedad de metodologías, algunas específicamente para proyectos de Software y otras más generales que también pueden ser aplicados a este campo.

Con el objetivo de conocer a profundidad las metodologías enfocadas a proyectos de Software, se efectuó una búsqueda exhaustiva donde se identificaron en total

diez (10) metodologías ágiles que contaban con suficiente información para la elaboración de un análisis detallado de sus características. Las mismas se encuentran en la siguiente tabla a presentar, donde se despliega su nombre, abreviatura, el tipo de metodología que es (PM: Metodología de Gestión de proyecto y SD: Metodología para el Desarrollo de Software) y su año de introducción:

TABLA 10
METODOLOGÍAS ÁGILES IDENTIFICADAS

No.	Nombre	Abreviatura	Tipo de Metodología	Año
1	Adaptive Software Development	ASD	SD	Introducción en el año 1997. Su versión refinada y extendida fue en 2000.
2	Feature Driven Development	FDD	PM/SD	Introducida en el año 1997. Su versión revisada fue publicada en 2001.
3	Extreme Programming	XP	PM/SD	Desarrollado en 1996. Su versión revisada y refinada fue en el año 2004.
4	Scrum	No posee	PM/SD	Introducida en el año 1995.
5	Kanban	No posee	PM/SD	Introducido el año 2007.
6	Scrumban	No posee	PM/SD	Indefinido.
7	Crystal Transparente	No posee	PM/SD	Introducido como familia de metodo-logías en el año 1998. Los nuevos miembros fueron definidos en el 2001 y 2004.
8	Dynamic Systems Development Method	DSDM	PM/SD	Publicado en 1995.
9	Lean Development	LD	PM/SD	Introducción en el 2001.
10	Agile Unified Process	AUP	SD	Introducido en 2005, revisado en 2006

PM = Proyect Management SD = Software Development

Fuente: Elaboración propia.

2.5.1 Comparación de Metodologías ágiles

Identificadas y seleccionadas las metodologías, se desarrollaron tres (3) cuadros comparativos con distintos criterios a evaluar; el primer cuadro se centra en desplegar la información general de la metodología para su mejor comprensión, esto incluye detalles como: descripción de la metodología, modelo de proceso de software en que se basa, los procesos que lo constituyen y el tamaño de proyecto.

El segundo cuadro compara ciertos criterios pertenecientes a la gestión de proyecto, como: planificación, gestión alcance, gestión de tiempo, costo y riesgos. Y finalmente, el tercer cuadro presenta el resto de los criterios, estos son: comunicación con el equipo, comunicación con el cliente, pruebas y documentación.

A continuación, se muestran los tres (3) cuadros:

- **CUADRO COMPARATIVO 1.** Descripción general de las metodologías ágiles en la página 48.
- **CUADRO COMPARATIVO 2.** Comparación de metodologías por criterios de gestión de proyecto en la página 52.
- **CUADRO COMPARATIVO 3.** Continuación de la Comparación de metodologías por criterios de gestión de proyecto en la página 57.

CUADRO COMPARATIVO 1.
DESCRIPCIÓN GENERAL DE LAS METODOLOGÍAS ÁGILES

Metodología	Descripción	Modelo	Procesos que lo constituyen	Tamaño de Proyecto
Adaptive Software Development (ASD)	Posee un proceso adaptivo a través de la planificación de riesgo, elaboración de evaluaciones, revisión de planes y el desarrollo de procesos de lo aprendido en las iteraciones. Su ciclo de vida está compuesto por: especulación, colaboración aprendizaje. Se centra en el aprendizaje continuo, orientado al cambio, reevaluaciones, aclaración de lo incierto y un alto nivel de cooperación entre los desarrolladores, administrativos y clientes. Características principales: Enfocada a la misión, basada en características, iterativo, time-boxed, dirigido al riesgo y tolerante al cambio.	Iterativo-incremental	-**Especulación** (definición de la misión del proyecto, aclaración de incertidumbre) 1. Inicio de proyecto 2. Planificación de Ciclo Adaptivo -**Colaboración** (Resaltar el trabajo en equipo: generar resultados, compartir conocimiento y tomar decisiones) 3. Ingeniería Concurrente de Componentes -**Aprendizaje** (retrospectivas del proyecto, focus group, revisión por iteración) 4. Evaluación de calidad 5. Revisión Final y Entrega	Para proyectos pequeños, medianos y grandes.
Feature Driven Development (FDD)	Es un proceso de desarrollo de Software adaptativo que se enfatiza en la calidad de cada paso, entrega frecuente de trabajos tangibles, suministrar información significativa y precisa del estado del proyecto. Basado en expresar y desarrollar los requerimientos en términos de pequeñas piezas de funcionalidad de valor agregado para el cliente, llamadas características (features). No es una metodología de ciclo completo; inicia cuando el estudio de factibilidad y planificación del proyecto han sido desarrollados. El caso de negocios ha sido establecido y aprobado. Excluye actividades de post-implementación (verificación y validación del sistema) e implementación y mantenimiento del sistema final.	Iterativo-incremental	1. Desarrollo de Modelo General 2. Construir una lista de características (features) 3. Planificación por característica 4. Diseño por característica 5. Construir por característica	Aplicable tanto para equipos/proyectos pequeños como grandes. Es escalable hasta para grandes equipos debido al concepto de "suficiente diseño inicial".

Metodología	Descripción	Modelo	Procesos que lo constituyen	Tamaño de Proyecto
Extreme Programming (XP)	Se considera como una disciplina de ingeniería de software más que una metodología, pero que incorpora un proceso. Se basa en cinco (5) valores: comunicación (constante con el cliente y el equipo), sencillez (evitar residuo, diseño simple y efectivo), retroalimentación (identificar áreas de mejora), valor (coraje para el cambio) y respeto (trabajo en equipo y su contribución en el proyecto).	Iterativo-incremental	1. Exploración 2. Planificación (Planificación de liberación) 3. Iteraciones a la primera liberación 4. Producción y refinamiento 5. Mantenimiento 6. Muerte	Enfocado principalmente a proyectos pequeños de requerimientos vagos o de rápido cambio.
Scrum	Marco de trabajo comprensivo de desarrollo de software, que enfatiza en la importancia de trabajo en equipo, roles, eventos, artefactos y reglas. Está compuesto por tres (3) pilares: transparencia (procesos y lenguaje claro para todos los participantes), inspección (de los usuarios) y adaptación (al cambio).	Iterativo - Incremental	1. Product backlog 2. Sprint planning 3. Sprint backlog 4. Sprint execution 5. Sprint review 6. Sprint retrospective	Proyectos pequeños, escalable a proyectos grandes
Kanban Development	Tiene como objetivo controlar y administrar el flujo de características (representadas por tarjetas Kanban) permitiendo organizar y visualizar el flujo de trabajo, limitar el trabajo en progreso y detener el inicio de nuevas características e iniciar la finalización de las pendientes. Los valores de Kanban son: Transparencia, balance, colaboración, enfoque en el cliente, fluidez, liderazgo, entendimiento, acuerdo y respeto.	Incremental	1. To Do (características por hacer) 2. Estimated (características estimadas) 3. Work In Progress – WIP (Características en desarrollo) 4. Done (Características completadas)	No determina el tamaño de proyecto
Scrumban	Metodología que combina la estructura de Scrum con el flujo de trabajo de Kanban. El trabajo en desarrollo es limitado por el WIP (Work in progress), no se especifica un tiempo de entrega, la meta es mover las características del tablero ha completado.	Flujo de trabajo continuo, con iteraciones opcionales	1. Metas 2. Lista de características 3. Características aceptadas 4. Desarrollo 5. Pruebas 6. Despliegue Completado	No determina el tamaño de proyecto

Metodología	Descripción	Modelo	Procesos que lo constituyen	Tamaño de Proyecto
Crystal	Basado en el concepto que se requiere mayor rigor en el desarrollo de software a medida que el tamaño del proyecto incrementa. Existen 4 metodologías dentro de Crystal: Crystal transparente (sin color) - aprox. seis (6) – ocho (8) personas en el equipo. Crystal amarilo - aprox. veinte (20) personas Crystal naranja - aprox. cuarenta (40) personas Crystal rojo - aprox. ochenta (80) personas. Crystal transparente toma como ventaja el pequeño tamaño del equipo, promoviendo la comunicación cercana a convertirse en una comunicación osmótica. La metodología se enfoca en tres (3) características: entrega frecuente, mejoramiento reflexivo, comunicación cercana.	Iterativo - Incremental	1. Chartering (Planificación) 2. Entrega cíclica 3. Wrap up (Conclusión)	Crystal transparente está orientado a proyectos pequeños. El número máximo de personas que podrían estar involucradas en un proyecto son considerados como una medida del tamaño del proyecto.
Dynamic Systems Development Method (DSDM)	Marco conceptual y metodología de desarrollo de software. Consiste en nueve (9) principios que dirigen al equipo y crean concepto mental para entregar a tiempo y dentro de presupuesto: 1) La implicación activa de los usuarios es imprescindible 2) Los miembros del equipo tienen la autonomía para tomar decisiones. 3) Enfoque en la entrega frecuente 4) El principal criterio de prioridad, desarrollo y validación de las entregas incrementales es el objetivo y la salud del negocio 5) El desarrollo iterativo o incremental es obligatorio 6) Todos los cambios realizados en el desarrollo son reversibles 7) Los requisitos se establecen a un nivel general 8) Las pruebas forman parte del ciclo de desarrollo 9) Es imprescindible trabajar con espíritu de colaboración con todos los agentes implicados en el sistema que se desarrolla.	Iterativo-Incremental	1. Pre-proyecto 2. Proyecto - propio 2.1 Fases secuenciales 2.1.1 Estudio de factibilidad 2.1.2 Estudio de negocios 2.2. Fases iterativas (Ciclos de Desarrollo 2.2.1 Iteración Modelo Funcional 2.2.2. Iteración Diseño y Desarrollo 3. Post-proyecto	Proyectos pequeños y grandes. Permite la escalabilidad

Metodología	Descripción	Modelo	Procesos que lo constituyen	Tamaño de Proyecto
Lean Development (LD)	Metodología que se enfoca en la rapidez y eficiencia del desarrollo del flujo de trabajo entre los programadores y los clientes. Lean se basa de siete (7) principios fundamentales: 1) Eliminar los desperdicios 2) Amplificar el aprendizaje 3) Decidir lo más tarde posible 4) Entregar tan rápido como sea posible 5) Capacitar al equipo 6) Construir integridad intrínseca 7) Véase todo el conjunto.	Iterativo-incremental	1. Identificar el valor (para el cliente) 2. Representar el Flujo de valor (identificar el conjunto de tareas necesarias y eliminar las que no aporten valor) 3. Crear el flujo (ordenar las actividades con la secuencia adecuada) 4. Facilitar el "pull" (producción por demanda) 5. Búsqueda de la perfección	No determina el tamaño de proyecto
Agile Unified Process	Versión simplificada de Rational Unified Process (RUP). Tiene cuatro (4) fases donde cada una tiene siete (7) flujos de trabajo o disciplinas: Modelado (Entender negocio, identificar problema, identificar solución), implementación, pruebas, despliegue (entrega del sistema), gestión de la configuración (acceso y control de los artefactos), gestión de proyecto y entorno (disponibilidad de recurso).	Iterativo-incremental	1. Concepción 2. Elaboración 3. Construcción 4. Transición	Proyectos grandes o pequeños

Fuente: Elaboración propia en base a: (Agile Alliance) (Bubernak & Schweikert, 2012) (Bubernak & Schweikert, 2012) (Edeki, 2013) (Highsmith, 2002) (Highsmith, 2002) (Joskowicz, 2008) (Pahuja, s.f.) (Poppendieck & Poppendieck, 2003) (Schwaber & Sutherland, 2017) (Scott W. Ambler + Associates, 2014) (Taghavi Dilamani, 2014) (University of Warwick, 2009) (Voigt, 2004) (Wells, 2013)

CUADRO COMPARATIVO 2.
COMPARACIÓN DE METODOLOGÍAS POR CRITERIOS DE GESTIÓN DE PROYECTO

Metodología	Planificación	Gestión de Requerimientos/Alcance	Estimación/Gestión de Tiempo	Estimación/Gestión de Costo	Identificación/Gestión de Riesgos
Adaptive Software Development (ASD)	La fase de Iniciación involucra determinar: la misión, objetivos y comprender las restricciones del proyecto. También, establece la organización, identifica los requerimientos, arma los tamaños iniciales, la estimación del alcance; e identifica los riesgos del proyecto. Además, se establece la cantidad de iteraciones según el tamaño del proyecto y el nivel de incertidumbre.	Para identificar los requerimientos se utilizan sesiones de JAD (Joint Application Design) en un proyecto pequeño o mediano, no toma más de dos (2)- cinco (5) días. En grandes dos (2)- tres (3) semanas. Al identificarse los requerimientos, se agrupan por temas (tal como en Scrum). Se realiza revisión del comportamiento del ciclo a través de talleres de sesiones de focus group con el cliente, donde se presentan los resultados del ciclo efectuado.	Las iteraciones para proyectos pequeños y medianos varían entre dos (2) a ocho (8) semanas.	Durante la fase de iniciación se realiza una planificación inicial donde se establece los recursos requeridos y las iteraciones según los requerimientos establecidos.	En la fase de Iniciación del proyecto, se identifican los riesgos claves del proyecto.
Feature Driven Development (FDD)	Como primera actividad se desarrolla el Modelo general, entre el cliente y el equipo de desarrolladores. El cliente presenta sus necesidades e información del negocio. Mientras que el equipo de desarrollo produce los modelos en base a dicha información.	Realizado el modelo general, se construye la lista de características. Se priorizan según la satisfacción del cliente y midiendo que puedan ser desarrollados en menos de dos (2) semanas. Los requerimientos se agrupan por medio de Sets de Características. Con la lista de requerimientos se planifica su ejecución, creando una secuencia y sus hitos.	Las características se desarrollan e implementan cada dos (2) semanas de lo contrario se subdividiría en características a tamaño más pequeño.	La metodología no se enfoca en la gestión del costo.	La metodología no es una metodología de ciclo completo. Inicia cuando el estudio de factibilidad y planificación del proyecto han sido desarrollados. Además, la metodología excluye las actividades de post-implementación (verificación y validación del sistema) e implementación y mantenimiento del sistema final.

Metodología	Planificación	Gestión de Requerimientos/Alcance	Estimación/Gestión de Tiempo	Estimación/Gestión de Costo	Identificación/Gestión de Riesgos
Extreme Programming (XP)	La planificación está compuesta por: -Planificación en base a las historias de usuario (Release Planning) -Planificación por iteración. En la etapa de planificación se redacta historias de usuarios. Estas historias se utilizan para crear estimados de tiempo para reuniones de planificación de liberación (Release Planning), para luego crear el calendario de liberación (release schedule). Los planes son artefactos temporales que van expirando y se crean nuevos según la necesidad.	Para la identificación de requerimientos se desarrollan historias de usuario (user stories). Los mismos son utilizados para la estimación de tiempo del Release Planning. Se crean pruebas de aceptación a partir de las historias del usuario para verificar que la historia de usuario fue implementada correctamente.	El proyecto es dividido en iteraciones. Las iteraciones son programadas en periodos de uno (1) a tres (3) semanas.	Su meta es reducir el costo de cambios en requerimientos a través de múltiples ciclos cortos de desarrollo, en lugar de uno solo y de mayor duración. Actividades improductivas son eliminadas para reducir costos.	De existir riesgos en las historias de usuario que no permita medir el nivel de complejidad del desarrollo, se aplica pequeños programas de pruebas (spikes), que permite reducir el riesgo para probar o evaluar una solución (son desechados finalizada la evaluación).
Scrum	Para la planificación se lleva a cabo el Sprint Planning, con una duración entre ocho (8) horas a un mes. El Sprint planning determina cuáles elementos del product backlog (lista de requerimientos) serán desarrollados en el sprint; como resultado se genera el Sprint Backlog.	Se desarrolla el product backlog, lista ordenada de todas las necesidades requeridas en el producto. Posee las características, funcionalidades, mejoras y arreglos que el producto a futuro debe contener. Existen Inspecciones para examinar los artefactos y progresos hacia un Sprint Goal para detectar varianzas indeseables. Se realizan adaptaciones para reajustar lo identificado como fuera de los límites de lo	El trabajo es realizado en iteraciones de hasta un mes de calendario llamado sprints. El trabajo en cada sprint debe crear algo de valor tangible al usuario o cliente (incrementos) Se utilizan eventos prescritos para crear regularidad y minimizar la necesidad de reuniones no definidas en scrum. Todos los eventos son fechados y tienen una duración máxima	Reduce desperdicios a través del uso planificado de tiempo.	Los sprints limitan los riesgos a un mes de calendario y costo. Cuando el tiempo de un sprint es demasiado extenso, la definición de lo que se está desarrollando puede cambiar, aumentando la complejidad y así mismo el riesgo. Los sprints permiten predictibilidad asegurando la inspección y adaptación del avance

Metodología	Planificación	Gestión de Requerimientos/Alcance	Estimación/Gestión de Tiempo	Estimación/Gestión de Costo	Identificación/Gestión de Riesgos
		aceptable. Existen cuatro (4) eventos formales de inspección y adaptación: sprint planning, daily scrum, sprint review y sprint retrospective.			hacia una Meta Sprint al menos cada mes.
Kanban Development	En Kanban se llevan a cabo múltiples reuniones en diferentes tiempos del proyecto para planificar y dar seguimiento a las actividades: -Trimestralmente se desarrolla una revisión de las estrategias a proporcionar y si son las adecuadas. -Mensualmente se hace una revisión de operaciones y de riesgos. -Semanalmente se determinan cuáles características se desarrollarán próximamente. -Diariamente el equipo realiza un stand-up meeting para coordinar las tareas del día. -Por cada entrega el equipo realiza una reunión de planificación de entrega.	Limita el trabajo-en-proceso (WIP - Work in Progress) que puede estar "en cola" por cada estado de flujo de trabajo, lo que limita la complejidad, minimiza los residuos, el ciclo de tiempo y permite establecer una velocidad predecible de desarrollo. Kanban tiene una cultura de recursos de holgura, que permite solventar cuellos de botella de ser necesario. Si una persona ha completado su tarea mientras se está realizando una actividad de prueba complicada, puede decidir en apoyar al evaluador a completar esa tarea o tomar otra tarea de la fila (queue) de actividades.	No existe time-boxing. Sin embargo, el trabajo-en-proceso del proyecto es medido a través de dos (2) parámetros: tiempo de tránsito y tiempo de ciclo. Tiempo de tránsito permite conocer cuánto tiempo tomará una actividad una vez recibida del cliente. Tiempo de ciclo es la cantidad de tiempo requerido para completar cierta actividad una vez el desarrollador comenzó a trabajar en ella.	El completar los proyectos en menor tiempo, trabajar iterativamente reduciendo el residuo, involucrar al usuario para que brinde retroalimentación crítica que permite hacer las correcciones pertinentes, priorizar las características a desarrollar permite reducir el costo del proyecto.	Mensualmente se efectúa una revisión de riesgos, con el objetivo de entender y responder a los riesgos de entrega en el servicio brindado.

Metodología	Planificación	Gestión de Requerimientos/Alcance	Estimación/Gestión de Tiempo	Estimación/Gestión de Costo	Identificación/Gestión de Riesgos
Scrumban	Scrumban no cuentan con reuniones regulares de planificación. Se planea poco a poco apuntando a desarrollar una cantidad por iteración.	Las necesidades son enlistadas, priorizadas y desfragmentadas como tareas y son colocadas en el Scrumban Board. Los miembros del equipo hacen pull (seleccionan y desarrollan hasta terminarlas) a las tareas hasta que la lista se vacíe; procediendo a planificar las demás tareas. Adopta feature-freeze, un tiempo de corte donde los cambios o características adicionales no pueden ser agregadas porque el deadline del proyecto se está acercando.	El proyecto inicialmente no tiene restricciones de tiempo. Sin embargo, la estimación de tiempo de una tarea se torna más aparente a medida que el equipo completa más de ellas. Utiliza las métricas de Kanban de tiempo de tránsito y de ciclo.	Utiliza las métricas de Kanban de tiempo de tránsito y de ciclo, lo que permite conocer el tiempo y costo que toma para desarrollar las tareas y así predecir cuánto tiempo y costo tomará las tareas futuras.	Para evitar los cuellos de botellas en la columna de Work in Progress, se opta por crear columnas detalladas dentro del mismo (Análisis, desarrollo, pruebas, etc.) Se promueve la eliminación de residuos.
Crystal (Transparente)	Se lleva a cabo la Fase de Chartering, donde el equipo establece: el valor del negocio (Business-value), requerimientos, modelo de dominio, plan de tecnología, plan del proyecto, proceso o metodología. Lo mismo puede llevar de unos cuantos días a uno (1) o dos (2) semanas para decidir si se procede con el proyecto o no.	El equipo utiliza el business-value para identificar los casos de uso y los roles claves del sistema. Seguido se realiza un prototipo que presenta las funcionalidades e interacciones del sistema. Para resaltar los conceptos claves se desarrolla el modelo de dominio, lo que permite estimar el tamaño y dificultad del problema a resolver. Y al finalizar el proyecto, a través de un taller, el equipo reflexiona sobre la calidad del producto entregado, la metodología desarrollada y los planes del proyecto.	Una de las características de la metodología es la entrega frecuente. El máximo del tiempo de entrega es de cuatro (4) meses, utilizando la técnica de timeboxing. Después que el trabajo es desarrollado y entregado, el equipo mide la velocidad y la tasa de éxito.	La metodología realiza en la fase de chartering un estudio de factibilidad para conocer si el proyecto es viable con respecto al costo y ganancias del producto a desarrollar.	Se realiza un análisis de factibilidad preliminar conducido como mecanismo de mitigación de riesgo.

Metodología	Planificación	Gestión de Requerimientos/Alcance	Estimación/Gestión de Tiempo	Estimación/Gestión de Costo	Identificación/Gestión de Riesgos
Dynamic Systems Development Method (DSDM)	Como primera actividad se desarrolla la fase de Pre-proyecto donde se determina si el proyecto se debe desarrollar. Seguido se elabora un estudio de factibilidad y en base a este se efectúa el estudio del negocio (comprensión de requerimientos).	Se enfoca en la entrega frecuente. Todos los cambios durante el desarrollo son reversibles. Utilización de iteraciones cortas , asegurando que se pueda regresar a estados de desarrollo anteriores, cualquier camino equivocado puede ser corregido de manera segura. Para la priorización de los requerimientos se utiliza las reglas MsSCoW: -Must have (Características requeridas del sistema, si no las posee no funciona) -Should have (Características que brindan valor al sistema, pero que son omitidas por tiempo) -Could have (Características que son funcionales pero que pueden ser agregadas posteriormente) -Want to have (Características dirigidas a un pequeño grupo de personas y da poco valor).	Para gestionar el tiempo se utiliza time-boxing, sets de iteraciones no más de seis (6) semanas de desarrollo, donde cada uno se deberá presentar los entregables planificados.	Un estudio de negocios y de factibilidad son realizados al inicio del proyecto, reduciendo grandemente la aparición de sorpresas financieras, especialmente en las etapas finales del proyecto.	Se realizan análisis de riesgos en los estudios de factibilidad y de negocios. Se enfoca en la entrega frecuente, lo que permite descubrir los errores temprano y fácilmente son reversibles.

Metodología	Planificación	Gestión de Requerimientos/Alcance	Estimación/Gestión de Tiempo	Estimación/Gestión de Costo	Identificación/Gestión de Riesgos
Lean Development (LD)	La planificación de la iteración tiene como objetivo la implementación de un conjunto de características por iteración.	Se modela el flujo de valor del negocio, mediante el registro de las actividades que se llevan a cabo y el tiempo que toman en el proceso y la espera de las siguientes tareas para eliminar desperdicios. Se identifican y desarrollan las características de alta prioridad primer, dejando las de baja prioridad para el final. Si los requerimientos que son complejos y no pueden desarrollarse en una sola iteración son reducidos a características más pequeñas.	Las iteraciones en Lean son cortas. Donde al finalizar cada una de ella se debe generar una característica funcional. No se indica la cantidad exacta.	Mediante la identificación de residuos y la priorización de características se reduce el costo del producto.	Se diseña las interfaces y luego los componentes, permitiendo identificar los riesgos mayores al inicio del proyecto cuando todavía existe tiempo y presupuesto.
Agile Unified Process (AUP)	En la Fase de Incepción se define el alcance, riesgos, costos, cronograma, un estudio de factibilidad y la preparación del ambiente de trabajo.	En la fase de Incepción se definen los requerimientos sin considerar su implementación, se identifican los interesados del proyecto (stakeholders), comprensión del problema del cliente. Brinda las guías para establecer áreas de trabajo seguras para cada desarrollador, aislándolo de otras áreas de trabajo y controlando cambios en artefactos de software (modelos, códigos, documentos, etc.).	En la etapa de Incepción, se estima el costo y tiempo del proyecto.	En la etapa de Incepción, se estima el costo y tiempo del proyecto.	En la etapa de Incepción, se definen los riesgos, se determina la factibilidad y alcance del proyecto. En la etapa de elaboración, se comprenden y eliminan los riesgos de alta prioridad en el proyecto.

Fuente: Elaboración propia basándose en: (Agile Alliance) (Bubernak & Schweikert, 2012) (Bubernak & Schweikert, 2012) (Edeki, 2013) (Highsmith, 2002) (Highsmith, 2002) (Joskowicz, 2008) (Pahuja, s.f.) (Poppendieck & Poppendieck, 2003) (Schwaber & Sutherland, 2017) (Scott W. Ambler + Associates, 2014) (Taghavi Dilamani, 2014) (University of Warwick, 2009) (Voigt, 2004) (Wells, 2013)

CUADRO COMPARATIVO 3.
COMPARACIÓN DE METODOLOGÍAS POR CRITERIOS DE GESTIÓN DE PROYECTO

Metodología	Pruebas	Trabajo en equipo (Comunicación, colaboración)	Nivel de participación del cliente en el proyecto	Documentación de Proyecto
Adaptive Software Development (ASD)	Existe una integración y pruebas constantes en cada iteración. Para la revisión de la calidad técnica, se realiza periódicamente la programación en par y la revisión de toda la arquitectura, llevada a cabo semanalmente o al final de la iteración.	Promueve la colaboración: apoyo en problemas técnicos, requerimientos de negocio, rápida toma de decisiones. Debe ser un equipo creativo y con iniciativa. En proyectos pequeños (Menos de diez (10) personas) cuando el equipo se encuentra cerca físicamente, se promueve la colaboración informal por medio de mensajes de texto, apuntes en el tablero, ya que es considerado efectivo. En proyectos grandes (100 desarrolladores) se requiere prácticas adicionales, como lo son las herramientas de colaboración y un gestor de proyecto. Al final de cada iteración se realiza una retrospectiva del equipo, cuáles actividades hay que realizar más o cuáles menos. Promoviendo la mejora continua del equipo.	La comunicación del cliente es constante: - En el proceso de asignación, el cliente elige en la priorización de características, utilizando las estimaciones elaboradas por el equipo de trabajo. -Finalizada la iteración, los clientes brindan retroalimentación del software resultante de cada iteración por medio de sesiones de Focus Group.	Retroalimentación y solicitud de cambios por parte del cliente son cuidadosamente documentados de modo que puedan ser considerados en siguientes iteraciones.
Feature Driven Development (FDD)	Dentro del proceso de Diseño y construcción por característica se realizan pruebas e integración (Dentro de las 2 semanas de desarrollo por iteración).	El equipo de modelado consistirá en varios desarrolladores expertos (jefes de programación) y uno o más expertos del dominio (que podrá ser el cliente, analista de negocios o usuarios). Este equipo es guiado por un experto en Modelado (Jefe de Arquitectura). En las siguientes fases, continuarán involucrados los jefes de programación. En conjunto con un Gerente de Proyecto y un Gerente de Desarrollo (supervisa jefes de programación). El equipo de desarrolladores construye la lista de características. En el equipo por cada set de características hay un	Cliente está involucrado en la etapa del desarrollo del modelo general del caso de estudio aprobado. Se define el alcance y contexto del proyecto de desarrollo de software.	En la etapa de Diseño por característica, se desarrolla diagramas de secuencia (que muestran cómo los objetos deben interactuar en la corrida de modo que se implemente cada característica). También se refina el Modelo Objeto (diagramas de clase) para que esté alineado a los diagramas de secuencia elaborados. Se redactan prólogos de clase y método para los elementos del modelo. Se crea un

Metodología	Pruebas	Trabajo en equipo (Comunicación, colaboración)	Nivel de participación del cliente en el proyecto	Documentación de Proyecto
		programador líder, que administra la característica por desarrollar en ciclos de dos (2) semanas.		Paquete de Diseño que cuenta con: diagramas de secuencia, refinamiento del modelo objeto, prólogos, y notas de las alternativas, limitaciones y supuestos de diseño.
Extreme Programming (XP)	Primeramente, se escriben las pruebas que el producto debe superar; seguido el desarrollo es el mínimo necesario para completar las pruebas definidas anteriormente. Todo código deberá pasar por pruebas unitarias por el programador	Enfatiza en el trabajo en equipo, siendo todos del mismo nivel: gerentes, clientes, desarrolladores. Implementa equipos altamente productivos, donde se autoorganizan y resuelven el problema lo más eficiente posible. Se mantiene comunicación fluida con el equipo por medio de: pruebas unitarias, programación en pares y estimación de tareas. Se hace uso de CRC (Use class, responsibilities & collaboration) para diseñar el sistema con todo el equipo de trabajo. Se llevan a cabo Stand up meetings diarias donde se discuten problemas, soluciones y promueve el enfoque del equipo. Los equipos son de cuatro (4) a ocho (8) personas, números pares para que pueda realizarse la programación en pares. Pueden contar con un coach (entrenador) programador con experiencia que oriente al equipo en mantener las prácticas XP.	Un cliente o gerente de proyecto debe estar asignado al equipo tiempo completo, para que exista una conexión diaria de la empresa para que se produzca el mejor producto posible. Las historias de usuarios son escritas por los clientes como las necesidades que necesitan que hagan por ellos. Los desarrolladores liberan pequeñas entregas frecuentes, que son mostradas al cliente para su retroalimentación.	Redacción de las historias de usuario en la etapa de planificación.

Metodología	Pruebas	Trabajo en equipo (Comunicación, colaboración)	Nivel de participación del cliente en el proyecto	Documentación de Proyecto
Scrum	Los equipos multifuncionales de Scrum deben elaborar las pruebas dentro de cada sprint.	El equipo está compuesto por: un product owner (responsable de maximizar el valor del producto y gestionar el product backlog), equipo de desarrollo y un Scrum master (responsable de promover a Scrum tal como se define en la guía). Miembros individuales en el equipo de desarrollo pueden contar con habilidades especializadas y áreas de enfoque, pero la responsabilidad pertenece al equipo como un todo. Los equipos van entre tres (3) a nueve(9) personas. Los equipos se autoorganizan y son multifuncionales. El modelo del equipo está diseñado para ser flexible, creativo y productivo. Realizan actividades como el Sprint Planning, el product backlog, daily scrum (reuniones diarias para discutir las actividades del día o en busca de solución de problemas), sprint review (retroalimentación con el cliente) y sprint retrospective (ajustes que deben realizarse en los siguientes sprints) se elabora con todo el equipo.	En la actividad de Sprint Review, se realiza una reunión para inspeccionar y adaptar el producto que está siendo desarrollado. En esta evaluación, participan el equipo scrum, actores principales, patrocinadores, clientes y miembros de otros equipos de interés.	A través de una actividad llamada grooming, la visión es dividida en un set de características que son recolectadas en una lista de prioridades llamada product backlog. Sprint backlog es un set de ítems de product backlog seleccionados para un sprint, incluye un plan para la entrega del incremento de producto y realizar el sprint goal. El Incremento es la suma de todos los ítems de product backlog completados en un sprint y el valor de los incrementos en los sprint anteriores.
Kanban Development	Las pruebas se realizan dentro de la iteración.	Kanban promueve la colaboración para mejorar el trabajo en equipo; el liderazgo necesario para que exista una mejora continua y entrega de valor; respeto, el entendimiento y consideración con los demás. Dentro del equipo de trabajo existe un Gerente de Solicitud de Servicio y Gerente de Entrega de Servicio.	Incluye dentro del flujo de tareas secciones de pruebas y evaluación con el cliente. Uno de los valores de Kanban es el enfoque al cliente.	Visualización del flujo de trabajo, a través del Kanban Board permite identificar dónde están los cuellos de botella. El Kanban Board permite de manera intuitiva representar al proyecto y monitorear el estatus actual.

Metodología	Pruebas	Trabajo en equipo (Comunicación, colaboración)	Nivel de participación del cliente en el proyecto	Documentación de Proyecto
Scrumban	Las pruebas se realizan dentro de la iteración.	Los equipos se auto-organizan y son multifuncionales. Cada uno cuenta con un rol de trabajo. Cada miembro del equipo puede elegir qué tarea va a trabajar. Se realizan reuniones diarias de scrum, retrospectivas para la mejora y aprendizaje de los errores.	No se define en la metodología.	Utiliza un Scrumban Board para mantener visibilidad del trabajo. La misma puede ser extendida a más columnas para ser más específico en qué estado se encuentra la tarea.
Crystal (Transparente)	La comunicación osmótica permite la retroalimentación entre el equipo, resultando que se detecten los errores y sean modificados lo más pronto posible. Se hace uso de pruebas automatizadas (Se puede iniciar buscando unit test framework). Se realizan integraciones constantes, de ser posible diariamente, permitiendo encontrar los errores temprano.	La metodología promueve la comunicación osmótica: todo el equipo se encuentra en la misma habitación y todos escuchan la información de todos; si una persona del equipo realiza una pregunta, los demás miembros del equipo contribuyen a la discusión. Los equipos son de tres (3) a ocho (8) personas. Se promueve el mejoramiento reflexivo En la fase de Chartering (Planificación), se crea un equipo base. Este equipo base requerirá un Patrocinador Ejecutivo, que actúa como un experto de dominio. Un Líder Diseñador que actúa como gerente de proyecto, coordinador, experto técnico y entrenador. Un Usuario Embajador, que actúa como experto en el uso del sistema. Y analistas de sistemas, programadores diseñadores, expertos de negocios, entre otros.	Dentro de cada iteración se realizan reuniones frecuentes con los usuarios expertos, lo que proporciona seguridad tanto a los clientes como a los desarrolladores para seguir con la iteración. El ciclo de entrega cuenta con una fase de entrega a usuarios reales, donde el sistema integrado es entregado a un número pequeño de usuarios y se utiliza retroalimentación para mejorar el sistema y revisar los planes y/o requerimientos.	Creación de un estudio preliminar de factibilidad en la fase de chartering (planificación) y el desarrollo de un plan inicial. -Revisión y actualización de los planes de proyectos de acuerdo con las experiencias ganadas en los ciclos de entrega realizados.

Metodología	Pruebas	Trabajo en equipo (Comunicación, colaboración)	Nivel de participación del cliente en el proyecto	Documentación de Proyecto
Lean Development (LD)	Las pruebas se realizan seguido de la codificación para evitar que se acumulen defectos. Se elaboran pruebas automatizadas.	Los equipos son pequeños, pero deben contar con la experiencia necesaria; tienen la libertad, apoyo y habilidades para cumplir con sus metas. Unos de los principios de Lean es eliminar desperdicios, si una persona está involucrada en varios proyectos teniendo que cambiar de tareas, el tiempo de cambio se vuelve desperdicio. Recomiendan que una persona esté involucrada en un proyecto a la vez. Se utilizan Information Radiators (Radiadores de información). El mismo permite visualizar la información del proyecto: qué está pasando, qué necesita ser realizado, qué problemas existen y qué progresos se han hecho.	El cliente en conjunto con el equipo brinda retroalimentación para realizar las correcciones correspondientes a los productos generados por cada iteración. La retroalimentación puede hacer más compleja un proyecto, pero de igual forma brinda un valor considerable al mismo.	La documentación se mantiene al mínimo, en la metodología se incita a ejecutar, probar, escribir, mostrar al cliente antes de escribir documentos complejos.
Dynamic Systems Development Method (DSDM)	Se realizan las pruebas tempranas dentro del proceso de desarrollo. Se elaboran documentos de pruebas y son revisadas por todo el equipo.	Dentro del equipo existen múltiples roles: gestor de proyectos, líder técnico de proyecto, desarrolladores (programador, diseñador o analista), tester, usuarios embajadores (user embassador: usuarios tiempo completo en el proyecto) y usuarios asesores (advisor users: usuarios medio tiempo en el proyecto). Se requieren de usuarios y desarrolladores, ambos entrenados en la metodología para su aplicación efectiva. La metodología promueve a que los equipos puedan tomar decisiones para que no exista un retraso en el proyecto (Requerimientos en práctica, las funcionalidades que deben ser incluidas en un incremento, priorización de requerimientos y características, detalles de soluciones técnicas). Se promueve la colaboración entre el equipo	Uno de los principios de la metodología es "la participación activa del usuario es imperativa", donde se trabaja con un pequeño grupo de usuarios continuamente. Su naturaleza iterativa permite a representantes de negocios ver la solución evolucionar, a través de retroalimentación y solicitud de cambios a través del desarrollo de la solución. Dentro de la metodología se llevan a cabo varios tipos de talleres: -Definición de requerimientos (Recolección de información) -Información de los Beneficios del Negocio -Operación del Sistema técnico	El prototipado es considerado el modelo principal. Dentro de la metodología se posee: -Prototipo de Negocio -Prototipo de Usabilidad -Prototipo de Rendimiento/Capacidad -Prototipo de Capacidad/Técnica También se llevan a cabo: -Modelo de Datos -Requerimientos

Metodología	Pruebas	Trabajo en equipo (Comunicación, colaboración)	Nivel de participación del cliente en el proyecto	Documentación de Proyecto
		técnico y del negocio para que exista una atmósfera de confianza y honestidad, donde se pueda adquirir retroalimentación de los productos.	-Aceptación del Plan de Pruebas -Diseño del Sistema	
Agile Unified Process (AUP)	Cuenta con un flujo de trabajo de Prueba consiste en la evaluación objetiva, esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, verificando que se cumplan los requerimientos.	En la fase de inicio, el equipo de desarrollo identifica los riesgos del proyecto. Durante la fase de elaboración, el analista recolecta los requerimientos, mientras el equipo de desarrollo crea mock-ups para el usuario para su revisión y retroalimentación. En la Fase de Construcción, cada iteración que se concluye es un entregable y se realiza pruebas de usuario. Y en la transición, el equipo se enfoca en mover el producto a producción.	En la etapa de construcción, se liberan versiones tempranas del proyecto para obtener retroalimentación del usuario. En la etapa de transición se entrega el producto; pero aun así es posible realizar modificaciones al mismo.	Crea modelos y documentos que son lo suficientemente buenos como para limitar su cantidad: -Modelo de Caso de Uso -Diagrama de caso de uso -Historias de usuario -Modelo de Flujo de Trabajo

Fuente: Elaboración propia basándose en: (Agile Alliance) (Bubernak & Schweikert, 2012) (Bubernak & Schweikert, 2012) (Edeki, 2013) (Highsmith, 2002) (Highsmith, 2002) (Joskowicz, 2008) (Pahuja, s.f.) (Poppendieck & Poppendieck, 2003) (Schwaber & Sutherland, 2017) (Scott W. Ambler + Associates, 2014) (Taghavi Dilamani, 2014) (University of Warwick, 2009) (Voigt, 2004) (Wells, 2013)

Realizada la comparación de los cuadros comparativos se concluye lo siguiente:

- **CUADRO COMPARATIVO 1. DESCRIPCIÓN GENERAL DE LAS METODOLOGÍAS ÁGILES**

  Se identifica que cinco (5) de las metodologías cuentan con valores o principios a seguir: Extreme Programming, Scrum, Kanban, Dynamic System Development Method y Lean; basándose principalmente en la comunicación con el cliente, comunicación y colaboración en el equipo de trabajo, la adaptación al cambio y la retroalimentación. Las Metodologías Extreme Programming y Lean se enfocan adicionalmente en la reducción de residuos o desperdicios al momento de ejecutar el proyecto.

  Con respecto al tamaño del proyecto, cuatro (4) metodologías: Adaptive Software Development, Feature Driven Development, Dynamic Systems Development Method y Agile Unified Process son funcionales para proyectos pequeños o grandes; mientras que tres (3) de las metodologías se enfocan en proyectos pequeños: Extreme Programming, Scrum y Crystal Transparente; con diferencia que Scrum es escalable, al igual que Feature Driven Development y Dynamic Systems Development Method. Por último, ocho (8) de las metodologías cuentan con procesos de desarrollo de Software son iterativo-incremental; y dos (2), Kanban y Scrumban son incremental y de flujo de trabajo continuo respectivamente.

- **CUADRO COMPARATIVO 2. COMPARACIÓN DE METODOLOGÍAS POR CRITERIOS DE GESTIÓN DE PROYECTO**

  Cada metodología cuenta con su forma de trabajo, algunas enfocándose mayormente en la planificación desde el inicio (identificando la misión, objetivos, factibilidad, valor del negocio, etc.); entre estos se encuentran: ASD, Crystal Clear, DSDM y AUP. Otros se centran en la identificación de requerimientos como tarea prioritaria, tal como: FDD, Scrum, XP, Kanban y Lean. La metodología con mayor

duración de una iteración es Crystal, donde una iteración puede ser de cuatro (4) meses como máximo, seguido ASD con máximo de dos (2) meses, DSDM con seis (6) semanas, Scrum con cuatro (4) semanas, XP con tres (3) semanas y FDD con dos (2) semanas. Las metodologías Kanban, Scrumban, Lean no indican cantidad de tiempo.

Entre otras características que se distinguen entre las metodologías, es la priorización de requerimientos y la aplicación de técnicas que se aplican para la misma: DSDM (Reglas de priorización), Scrum, Scrumban (enlista, prioriza y fragmenta) y Lean. Con respecto a la gestión del tiempo, la metodología: Kanban mide el tiempo por medio de "tiempo de tránsito y tiempo de ciclo", Scrumban aplica la técnica Feature-Freeze impidiendo que se agreguen más requerimientos durante el desarrollo para lograr la meta deseada en el tiempo definido; XP estima y controla el tiempo por medio de los user stories y DSDM aplica timeboxing.

Otro criterio importante por mencionar es la gestión de riesgos: ASD, Lean, DSDM, Crystal, AUP reducen los riesgos desde la fase de inicio por medio de un estudio de factibilidad. Kanban mensualmente inspecciona los riesgos del proyecto, Scrum limita las iteraciones a un mes para reducir los riesgos; y XP, según el nivel de complejidad del artefacto a desarrollar, utiliza "Spikes" para evaluar y probar la solución a bajo costo.

- **CUADRO COMPARATIVO 3. COMPARACIÓN DE METODOLOGÍAS POR CRITERIOS DE GESTIÓN DE PROYECTO.**

  Este cuadro se basaba en cuatro (4) criterios: pruebas, trabajo en equipo, nivel de participación del cliente en el proyecto y documentación de proyecto. Todas las metodologías aplican pruebas ya sea a inicio o durante la iteración; entre ellas existen dos (2) metodologías que aplican pruebas automatizadas Crystal y Lean; y pruebas de integración constante ASD y Crystal.

  El número de integrantes de un equipo de trabajo en las metodologías señaladas tienen como máximo diez (10) personas. Aquellas

metodologías que promueven la colaboración entre los miembros del equipo son: Crystal (Comunicación Osmótica), DSDM, ASD, XP, Kanban. Scrum y Scrumban requieren de equipos multifuncionales. Y Lean tiene la particularidad de contar con radiadores de información (Information Radiators) para asegurar que todo el equipo tenga a mano información completa del proyecto.

Las metodologías ágiles reconocen que es de gran importancia involucrar al cliente en los procesos de desarrollo, ya sea con su retroalimentación como realiza: XP, Kanban, Scrum, ASD, Crystal, DSDM, Lean y AUP. En el caso de XP, el cliente asignado es tiempo completo en el proyecto y escribe las historias de usuario; en ASD, el cliente prioriza los requerimientos; en FDD, el cliente se involucra en el desarrollo del modelo general y el alcance; en Scrum se realiza un Sprint Review para que el cliente conozca los avances del producto y las adaptaciones necesarias a realizar.

En referencia a la documentación del proyecto, cada metodología utiliza diferentes elementos para el seguimiento del proyecto: XP, Scrum y AUP utilizan User Stories para los requerimientos; Kanban y Scrumban utilizan un board (tablero) para especificar los requerimientos a desarrollar y el seguimiento de los mismos. FDD hace un paquete de diseño que contempla un diagrama de secuencia y modelo de objetos. Crystal documenta el plan de proyecto, los casos de uso y factibilidad del proyecto; DSDM elabora el prototipo, modelo de datos y los requerimientos; AUP se basa en el Modelo de caso de uso, historias de usuario, modelo de flujo de trabajo; ASD documenta los cambios; y en Lean, la documentación es mínima.

Estas comparaciones permiten reconocer que cada metodología cuenta con sus características particulares. Sin embargo, todas están fundamentados bajo el manifiesto ágil, dando relevancia a la aplicación de iteraciones cortas, comunicación con el cliente, generación de resultados, colaboración en el equipo de trabajo y documentación mínima.

2.6 FÁBRICA DE SOFTWARE

El término Fábrica de Software surge a finales de los años 60 con la propuesta de adoptar métodos de Fábricas tradicionales y organizaciones de Software por R.W. Bemer de General Electric y Dr. M.D. Mcllroy de la AT&T (ambos considerados autores del concepto). Bemer se enfocaba en la idea de un "Sistema automatizado de producción de Software" mediante el uso de herramientas estandarizadas, interfaces de computadoras y base de datos históricas que serían utilizadas para manejar las finanzas y la administración. Por otro lado, Mcllroy enfatizaba en la reutilización de componentes (piezas de Software o rutinas parametrizadas que funcionaran como bloques de construcción) para la edificación de nuevos programas. Mcllroy indicaba que se requería de tiempo y capital para tener un inventario amplio de componentes, pero que a largo plazo los programadores les sería beneficioso el desarrollo de nuevos programas por medio de componentes existentes con diseños más eficientes (Cusumano, 1988).

Más adelante, se crearon empresas que adoptaron el término de Fábrica de Software. La primera empresa en adoptar el término fue la empresa Hitachi, seguido estaban: Toshiba, NEC y Fujitsu. Estas empresas se caracterizaban por realizar ciertas actividades específicas: enfoque en determinados segmentos de mercados con procesos y organizaciones optimizadas para esos segmentos; aplicación de procesos de análisis y control (productividad, costo, tiempo, calidad, etc.); estandarización de los procesos; soporte de herramientas centralizado; equipos de trabajos orientados a las herramientas y a los métodos; inversión en entrenamiento para los trabajadores; implementación de procesos de mejoras; reutilización de código; y gestión de integración.

La tecnología se ha convertido en un elemento indispensable para la administración de procesos de negocios en las organizaciones. La empresa CA Technologies, clasificada como una de las corporaciones independientes más grandes del Software en el mundo, lanzó una campaña global "The Modern Software Factory" en mayo 2017 (Computer Asocciates International Inc., 2017), donde señala que el núcleo de la ventaja competitiva para las organizaciones está en la Tecnología. Indican que el desafío es crear mejores aplicaciones, más rápidas y seguras; y la

obtención de información a base de los datos recolectados. Para CA Technologies, una Fábrica de Software tendrá éxito si aplica metodologías ágiles, adopta una cultura de DevOps y brindar a su cliente la seguridad que su información está a salvo (Berkes, Digitally Remastered: Building Software into Your Business DNA, 2016).

Según la empresa consultora francesa Octo Technology, una Fábrica de Software es un conjunto de herramientas enfocadas para el desarrollo y la automatización del proceso de construcción de Software, desde la etapa de construcción hasta la entrega de la producción (Robert, Canuel, Benazza, & Descossy , 2012). Los mismos indican que una Fábrica de Software debe contar con características como:

- Contar con una recopilación de código fuente que funcione como referencia y más adelante con un manejador de código fuente.
- Ejecución de pruebas automatizadas.
- Medición del alcance de las pruebas al código fuente y su calidad.
- Respeto a los estándares de desarrollo.
- Verificación del buen uso de patrones de diseño.
- Cumplimiento de las reglas de arquitectura.

Esta empresa define el concepto Fábrica de Software 2.0, donde al igual que CAT Technologies, propone el uso de DevOps para mejorar la colaboración entre el equipo de desarrollo y de operación (Robert, Canuel, Benazza, & Descossy , 2012).

2.7 CONCLUSIONES

La conceptualización de los términos relacionados a los ciclos de vida del Software, la comparación de enfoques (tradicional y ágil), metodologías (10 en total), y la definición de las características de una Fábrica de Software brindan a esta investigación una base teórica nutrida para el desarrollo de una propuesta alineada a las tendencias actuales en el campo de Ingeniería de Software y las TICs.

2.8 REFERENCIA BIBLIOGRÁFICA

Agile Alliance. (2001). www.agilealliance.org. Obtenido de https://www.agilealliance.org/agile101/the-agile-manifesto/

Agile Alliance. (s.f.). www.agilealliance.org. Obtenido de https://www.agilealliance.org/glossary/kanban/#q=~(filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'kanban))~searchTerm~'~sort~false~sortDirection~'asc~page~1)

Berkes, O. (2016). https://www.ca.com/. Obtenido de https://www.ca.com/content/dam/ca/us/files/ebook/digitally-remastered-building-software-into-your-business-dna.pdf

Bubernak, C., & Schweikert, M. (23 de marzo de 2012). cs.colorado.edu. Obtenido de https://www.cs.colorado.edu/~kena/classes/5828/s12/presentation-materials/schweikertmarcbubernakchris.pdf

Computer Asocciates International Inc. (2017). https://www.ca.com/. Obtenido de https://www.ca.com/us/company/newsroom/press-releases/2017/ca-technologies-launches-new-global-marketing-campaign-the-modern-software-factory.html

Cusumano, M. A. (july de 1988). dspace.mit.edu. Obtenido de https://dspace.mit.edu/bitstream/handle/1721.1/2204/SWP-2036-19215282.pdf?sequence=1

Edeki, C. (septiembre de 2013). Agile Unified Process. International Journal of Computer Science and Mobile Applications, 13-17 . Obtenido de http://erytheia.nied.unicamp.br:8081/interhad/courses/roberto-pereira/ci163-projeto-de-software-ufpr-1/agenda/auppaper.pdf

Geambasu, C., Jianu, I., Jianu, I., & Gavrila, A. (2011). Influence Factors for the choice of a Software Development Methodology. Accounting and Management Information Systems.

Highsmith, J. A. (2002). Agile Software Development Ecosystems.

IEEE. (28 de September de 1990). https://www.jyu.fi/en. Obtenido de http://www.mit.jyu.fi/ope/kurssit/TIES462/Materiaalit/IEEE_SoftwareEngGlossary.pdf

IEEE. (01 de 02 de 2008). disi.unal.edu.co. Obtenido de http://disi.unal.edu.co/dacursci/sistemasycomputacion/docs/SystemsEng/ISO_IEC_12270_2008.pdf

ISTQB Certification. (s.f.). istqbexamcertification. Obtenido de istqbexamcertification: http://istqbexamcertification.com/istqb-certification-the-definitive-guide/

Joskowicz, J. (10 de febrero de 2008).

mainss. (7 de febrero de 2017). mainss.com. Obtenido de https://mainss.com/las-4-metodologias-de-gestion-de-proyectos-mas-utilizadas/

Mohammed Ali Munassar, N., & Govardhan, A. (2010). A Comparison Between Five Models Of Software Engineering. IJCSI International Journal of Computer Science Issues.

Pahuja, S. (s.f.). agilealliance.org. Obtenido de https://www./what-is-scrumban/

Poppendieck, M., & Poppendieck, T. (2003). Obtenido de Lean Software Development: An Agile Toolkit

Prasad Mahapatra, R. (2016). Obtenido de https://books.google.com/books?isbn=9382609687

Pressman, R. S. (2010). Ingeniería del Software (Séptima ed.). McGraw-Hill.

Punjab Technical University -PTU. (2006). Sbsceducation.org. Obtenido de http://sbsceducation.org/wp-content/uploads/2014/06/BCA-303.pdf

Robert, M., Canuel, V., Benazza, A., & Descossy , C. (29 de mayo de 2012). https://blog.octo.com/. Obtenido de https://blog.octo.com/en/toward-a-better-software-factory/

Schwaber, K., & Sutherland, J. (2017). scrumguides.org. Obtenido de http://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100

Scott W. Ambler + Associates. (2014). ambysoft.com. Obtenido de http://www.ambysoft.com/unifiedprocess/agileUP.html

SEGUE Technologies. (7 de 08 de 2015). seguetech.com. Obtenido de https://www.seguetech.com/benefits-adhering-software-development-methodology-concepts/

Sesento García, L. (septiembre de 2008). eumed.net Enciclopedia Virtual. Obtenido de http://www.eumed.net/tesis-doctorales/2012/lsg/concepto_modelo.html

Sommerville, I. (2005). Ingeniería del Software (Séptima ed.). Pearson Addison Wesley.

Standish Group. (4 de octubre de 2015). www.infoq.com. Obtenido de https://www.infoq.com/articles/standish-chaos-2015

Stoica, M., Mircea, M., & Ghilic-Micu, B. (2013). Software Development: Agile vs. Traditional. Informatica Economica.

Taghavi Dilamani, M. (2014). http://ispe-usa.com. International Conference on Advanced and Agile Manufacturing.

University of Warwick. (2009). Feature Driven Development & Empirical Modelling. Obtenido de https://warwick.ac.uk/fac/sci/dcs/research/em/publications/web-em/04/featurelist.pdf

Verma, I. (2014). ijcscn. Obtenido de http://www.ijcscn.com/Documents/Volumes/vol4issue3/ijcscn2014040305.pdf

Voigt, B. J. (2004). Dynamic System.

Wells, D. (8 de october de 2013). extremeprogramming.org. Obtenido de http://www.extremeprogramming.org/rules.html

CAPÍTULO III
METODOLOGÍA DE LA INVESTIGACIÓN

3.1 INTRODUCCIÓN

Este capítulo describe la metodología utilizada para la ejecución de este trabajo. En los siguientes puntos se determina el tipo de la investigación, el diseño de la investigación, la población y muestra; las variables conceptuales y operacionales, la instrumentación, su procedimientos, resultados, discusiones y recomendaciones.

3.2 TIPO DE INVESTIGACIÓN

El trabajo de investigación fue identificado en primera instancia como tipo descriptivo, debido a que la investigación considera dos aspectos importantes: la observación y documentación del estado actual de las FS, y una revisión literaria exhaustiva para el desarrollo de un marco teórico (Hernández Sampieri, Fernández Collado, & Baptista Lucio, 2010).

Igualmente es considerada cuasiexperimental, ya que se realizó un planteamiento de un problema, una posible solución y una hipótesis propuesta para ser comprobada; pero no se contó con un grupo experimental y control para la aplicación de la metodología propuesta (UJA, 2015). Además, este trabajo de investigación lleva todos los requisitos de una investigación experimental (título, introducción, justificación, objetivos, método, resultados, discusión, bibliografía, anexos) (Arias, 2012).

Lo fundamental de este estudio descriptivo y cuasiexperimental es haber definido las variables y haber aplicado el método de investigación. Después de estos puntos relevantes el estudio sigue las mismas condiciones que un estudio experimental. Por lo que podemos concluir que este tipo de investigación bien llevado nos permitirá obtener una visión general a resolver y aplicar un estudio de caso para demostrar la hipótesis a responder.

3.3 DISEÑO DE INVESTIGACIÓN

Para responder al problema y preguntas planteadas del trabajo de investigación, se llevó a cabo un diseño de campo no experimental, se recolectaron datos directamente de la Sección de FS y no se manipularon las condiciones existentes (Arias, 2012). Este proceso inició con la elaboración y el uso de un instrumento de evaluación para valorar un número de metodologías ágiles por medio de una encuesta contestada por el equipo de la Sección de Fábrica de Software y expertos de Ingeniería de Software. Luego de obtener los resultados de esta evaluación, se elaboró una propuesta metodológica para la gestión de proyectos en la Sección de Fábrica de Software. Este procedimiento será detallado en el punto 3.7 en la página 80.

3.4 POBLACIÓN Y MUESTRA

La población delimitada es el equipo de trabajo de la Sección de Fábrica de Software que pertenece al Departamento de Extensión del Centro de Investigación, Desarrollo e Innovación en Tecnologías de la Información y las Comunicaciones – CIDITIC, de la Universidad Tecnológica de Panamá, Sede Campus Levi Sasso. También, se contempló a expertos de Ingeniería de Software en la Universidad Tecnológica de Panamá, de la Facultad de Ingeniería de Sistemas Computacionales – FISC, particularmente docentes de pregrado de Ingeniería de Software o de Maestría en el área de Ingeniería de Software.

TABLA 11.
POBLACIÓN Y MUESTRA

Sección de Fábrica de Software	(1) Coordinador de la Sección (3) Analistas/Programadores (2) Programadores
Docentes de Ingeniería de Software de la Facultad de Ingeniería en Sistemas Computacionales de la UTP	(7) Expertos de Ingeniería de Software

Fuente: Elaboración propia

3.5 DEFINICIÓN DE VARIABLES

En este trabajo de investigación las **variables dependientes** identificadas son: alcance, costo y tiempo y satisfacción del cliente (sobre el producto de Software elaborado). La **variable independiente** es la metodología ágil de gestión de proyectos de Software. A continuación, se definen las variables:

3.5.1 Variable Independiente

Metodología Ágil de Gestión de Proyectos de Software		
Concepto	Operacional	Medición
Marco de trabajo con la funcionalidad de planificar, estructurar y controlar los procesos de desarrollo de Software considerando criterios de calidad en cada una de las fases elaboradas.	Nivel de funcionalidad de la metodología con respecto a sus procesos, resultados y plantillas.	Instrumento de evaluación de procesos de la metodología (ver Anexo E).

3.5.2 Variables Dependientes

1. Alcance del Proyecto		
Concepto	Operacional	Medición
El alcance del proyecto es el trabajo requerido para generar el producto entregable del proyecto. El alcance se define a menudo por una estructura de desglose del trabajo y los cambios deben tener lugar solo a través de procedimientos formales de control de cambios.	Número de requisitos que deben ser desarrollados para determinar que un proyecto fue finalizado correctamente.	Número total de requisitos identificados por iteración o por plazo de tiempo.
2. Tiempo del proyecto		
Concepto	Operacional	Medición
Es un conjunto de procesos necesarios para gestionar que un proyecto finalice adecuadamente; entre los procesos incluidos en la gestión de tiempo se encuentran: definición de actividades, secuencia de actividades, estimación de los	Cantidad de tiempo estimado y ejecutado para el desarrollo de un proyecto.	Número acumulado de horas invertidas en el proyecto por un

recursos necesarios, la duración, elaborar el horario y control.		periodo o iteración.
3. Costo del Proyecto		
Concepto	**Operacional**	**Medición**
Indicador clave de rendimiento para los proyectos. Los procesos se centran en la planificación, estimación, presupuestación, financiamiento y administración de costos para que el proyecto pueda completarse dentro del presupuesto aprobado.	Presupuesto total invertido para la ejecución y culminación del proyecto.	Monto de flujo de dinero acumulado según el total de horas invertidas en el proyecto por un periodo o iteración.
4. Satisfacción del Cliente		
Concepto	**Operacional**	**Medición**
Como una de las medidas del desempeño del sistema de gestión de la calidad, la organización debe realizar el seguimiento de la información relativa a la percepción del cliente con respecto al cumplimiento de sus requisitos por parte de la organización (NORMA ISO 9001, 2008).	Nivel de satisfacción existente del cliente por medio de la experiencia obtenida durante la ejecución de un proyecto.	Instrumento de medición de satisfacción del cliente durante la ejecución del proyecto (ver anexo F).

3.6 INSTRUMENTACIÓN

Para este trabajo se han utilizado dos (2) técnicas para la recolección de información primaria: la observación y la encuesta.

3.6.1 Fundamentación del marco Teórico

Para conocer sobre la situación actual de la Sección de Fábrica de Software con respecto al desarrollo de proyectos de Software, se observó y se recolectó los informes o reportes existentes en la sección, como: informe de costos anuales, informe de avances y de cierre de proyectos, cronogramas planificados, entre otros; para conocer qué problemáticas se presentaron durante su ejecución y sus causas. Además, el investigador como colaborador dentro de la sección, dispuso de sus experiencias propias dentro de los proyectos de software como: programador, analista y coordinador (4.1 Estado Actual – Sección de Fábrica de Software, página 85).

De igual forma, se realizó una búsqueda literaria con respecto a las metodologías: modelos de procesos de software existentes, enfoque tradicional y ágil; descripción de las metodologías ágiles existentes y características que debe poseer las Fábricas de Software (Ver Capítulo II Marco Teórico, página 16).

3.6.2 Evaluación de Metodologías ágiles

Se diseñó una evaluación de metodologías ágiles con el objetivo de seleccionar las metodologías más convenientes para la FS. Se utilizó como referencia las diez (10) seleccionadas en el marco teórico.

Durante la revisión literaria, se identificaron los elementos claves para medir y monitorear el desempeño de un proyecto. Estos elementos fueron base para la formulación de los criterios. El investigador estableció 14 criterios que se agruparon en seis (6) secciones de la siguiente manera:

TABLA 12.
ESTRUCTURA GENERAL DE LA EVALUACIÓN DE METODOLOGÍAS ÁGILES

No	Secciones	# de Criterios	%
1	Planificación	**4**	**30**
2	Gestión de alcance	**3**	**25**
3	Gestión de tiempo	**3**	**25**
4	Gestión de costo	**1**	**10**
5	Gestión de riesgos	**2**	**10**
6	Pregunta general	**1**	**0**
	Total	**14**	**100**

Fuente: Elaboración propia.

A cada criterio se le asignó un porcentaje de acuerdo con su grado de importancia. Este porcentaje fue asignado bajo el criterio del investigador. En el Anexo A: TABLA 28 en la página 164, se despliega los criterios detallados por sección, con el porcentaje asignado. Este porcentaje fue utilizado por el investigador para cuantificar los resultados de las evaluaciones.

Se le indicó a los encuestados que debiesen calificar qué tanto el criterio cumplía con las características de las metodologías enlistadas. La calificación consistió en el uso de la escala de valoración de Likert (Murillo Torrecilla, 2006). La TABLA 13. presenta esta escala de valoración:

TABLA 13.
ESCALA DE VALORACIÓN

Escala	Valor Numérico
Excelente	2.0 puntos
Bueno	1.0 puntos
Poco	0.5 puntos
Carente	0 puntos

Fuente: elaboración propia

Como muestra la tabla: Excelente es valorado con el puntaje máximo de dos (2.0) y Carente con el puntaje mínimo de cero (0) puntos. El

cuestionario fue aplicado al equipo de la FS y a expertos del área de Ingeniería de la Universidad. Los resultados de estas evaluaciones arrojaron las metodologías con mayor puntaje, indicando que estas fueron las más adecuadas para la FS.

Finalizada la evaluación de metodologías ágiles, se procede al desarrollo de la propuesta de metodología para la Sección de Fábrica de Software (Capítulo 5, punto 5.4. Propuesta de Metodología ágil de Proyectos de Software para la FS). Completa la metodología se elaboran dos (2) instrumentos para evaluar su funcionalidad y el nivel de satisfacción del cliente.

3.6.3 Evaluación de Metodología propuesta

Con el propósito de medir si la metodología propuesta tiene alguna influencia en los procesos por ejecutarse, se diseña un instrumento de medición que permita conocer si los procesos indicados en la metodología son los más indicados o funcionales para el equipo que lo aplica en un proyecto. Este instrumento debe ser completado al final de cada iteración dentro del proyecto; y el mismo evaluará criterios como: la comprensión de los procesos descritos, el uso de las plantillas, la funcionalidad de los procesos, las funciones de cada miembro del equipo y la utilidad de las salidas o resultados de los procesos (Ver anexo E. Evaluación de Metodología propuesta, página 188).

3.6.4 Evaluación de Nivel de Satisfacción del Cliente

En conjunto con el instrumento anterior, para conocer cómo influye la metodología propuesta con respecto al cliente, se diseña la Evaluación de Nivel de Satisfacción del Cliente, enfocado en medir el nivel de Satisfacción del Cliente en la ejecución de un proyecto. Esta evaluación mide: el nivel de inclusión del cliente en el proyecto con respecto a la identificación de necesidades, la comunicación de los avances, la consideración de las retroalimentaciones dadas por el cliente y usuarios por parte del equipo desarrollador, la satisfacción en general del cliente con relación a la forma que se está gestionando el proyecto y su conformidad en los resultados

(artefactos, informes, módulos del Software). Para mayor detalle de la evaluación ver anexo F. Evaluación de Nivel de Satisfacción del Cliente en la página 189.

3.7 PROCEDIMIENTO

En el siguiente contenido se procede a enlistar las etapas ejecutadas en este trabajo de investigación:

- **Etapa 1**. Selección del tema según las necesidades encontradas en el Centro de Investigación CIDITIC.
- **Etapa 2.** Revisión bibliográfica de libros, white papers, documentos web, entre otros; para la identificación y análisis de modelos de desarrollo de Software, enfoques de metodologías, metodologías existentes y características que debe poseer una Fábrica de Software.
- **Etapa 3.** Documentación de la situación actual de la Sección de Fábrica de Software – FS mediante la revisión de informes de avances, costos, minutas, entre otros. haciendo énfasis en los proyectos de mayor impacto para el Centro de Investigación CIDITIC desarrollados por la Sección de FS y la identificación de incidentes ocurridos en cada uno de ellos.
- **Etapa 4.** Elaboración de un cuadro comparativo sobre diez (10) metodologías ágiles, considerando criterios de Gestión de proyectos de Software y las necesidades de la Sección de Fábrica de Software: planificación, alcance, tiempo, costo, riesgos, pruebas, comunicación con el cliente, comunicación en el equipo de trabajo y documentación.
- **Etapa 5.** Diseño de un instrumento de evaluación de Metodologías (encuesta) considerando los aspectos claves identificados en el cuadro comparativo. La evaluación cuenta con seis (6) secciones: planificación, alcance, tiempo, costo, riesgos, y pregunta general.
- **Etapa 6**. Aplicación de la encuesta a todo equipo de la Sección de Fábrica de Software y a expertos en el área de Ingeniería de Software de la Universidad. Se procede a la recolección de los resultados y se identifican las metodologías con mayor valoración.

- **Etapa 7**. Análisis de la metodología base a utilizar y elaboración de la propuesta metodológica para la Sección de Fábrica de Software.

- **Etapa 8**. Diseño de dos (2) instrumentos de evaluación de la metodología: Evaluación de Metodología propuesta y Evaluación de Nivel de Satisfacción del Cliente; con el objetivo de comprobar la funcionalidad de la metodología con el equipo de la Sección y el nivel de satisfacción del cliente.
- **Etapa 9**. Documentación de análisis y resultados obtenidos a lo largo del proceso de investigación.
- **Etapa 10**. Presentación de consideraciones, recomendaciones y trabajos futuros.

3.8 RESULTADOS Y DISCUSIONES

El Marco Metodológico presenta el tipo y proceso de investigación necesario para responder a la hipótesis general planteada. Este trabajo de investigación consideró el diseño de tres tipos de evaluación: identificación de la metodología a proponer, análisis de los resultados luego de la implementación de la metodología propuesta, y una evaluación final sobre cómo el uso de esta metodología propuesta causó algún impacto en el nivel de satisfacción del cliente o usuario. Cabe mencionar, que las evaluaciones relacionadas a la implementación de la metodología no pudieron ser ejecutados debido al traslado del personal de la Sección de FS a otros departamentos de la institución.

3.9 CONCLUSIONES

El marco metodológico detalla las etapas ejecutadas en este trabajo de investigación, al igual que el tipo, diseño utilizado, los métodos de recolección y de análisis de datos.

El capítulo V presenta los resultados obtenidos en este trabajo de investigación, y los análisis que llevaron a la propuesta de la metodología ágil para la Sección de FS.

3.10 REFERENCIA BIBLIOGRÁFICA

Arias, F. G. (2012). Proyecto de Investigación - Introducción a la metodología científica. Caracas, Venezuela: Episteme.

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, M. d. (2010). Metodología de la Investigación. México D.F: McGRAW-HILL / INTERAMERICANA EDITORES, S.A. DE C.V.

Murillo Torrecilla, F. J. (2006). uam.es. Obtenido de https://www.uam.es/personal_pdi/stmaria/jmurillo/Met_Inves_Avan/Materiales/Apuntes%20Instrumentos.pdf

UJA. (30 de octubre de 2015). ujaen.es. Obtenido de http://www.ujaen.es/investiga/tics_tfg/estu_cuasi.html

CAPITULO IV
SECCIÓN DE FÁBRICA DE SOFTWARE DE CIDITIC-UTP

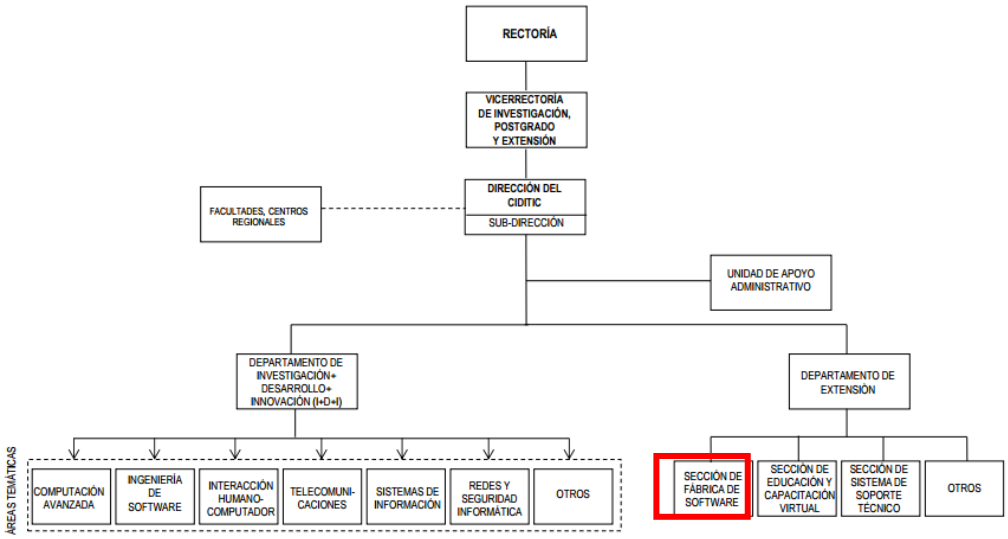
4.1 ESTADO ACTUAL – SECCIÓN DE FÁBRICA DE SOFTWARE

La Universidad Tecnológica de Panamá - UTP a partir del 25 de mayo de 2010, crea el Centro de Investigación, Desarrollo e Innovación en Tecnologías de la Información y las Comunicaciones – CIDITIC. El Centro de Investigación CIDITIC fue establecido con el objetivo de generar investigaciones científicas, desarrollar tecnología, innovar en el área de las TIC's, crear un mayor acercamiento entre la investigación y la docencia, y así, proyectar sus resultados al país y a la comunidad internacional.

CIDITIC actualmente cuenta con dos (2) Departamentos que lo constituyen: el Departamento de Investigación+Desarrollo+Innovación (I+D+i) y el Departamento de Extensión.

El Dentro del Departamento de Extensión se encuentra la Sección de Fábrica de Software (CIDITIC, 2017); ver FIGURA 11:

FIGURA 11.
ORGANIGRAMA DEL CENTRO DE INVESTIGACIÓN, DESARROLLO E INNOVACIÓN EN TECNOLOGÍA DE LA INFORMACIÓN Y LAS COMUNICACIONES - CIDITIC



Fuente: http://www.ciditic.utp.ac.pa/organigrama

La Sección de FS fue creada en conjunto con el Centro de Investigación CIDITIC y fue coordinada inicialmente por el Ingeniero de Sistemas Eddy Mojica bajo la dirección del Licenciado Jeremías Herrera. La Sección de FS tenía como objetivo

principal, el desarrollo de proyectos de Software para empresas externas e instituciones de gobierno para garantizar su sostenibilidad. Durante su gestión, se contaba con dos (2) ingenieros de Sistemas y cinco (5) programadores; y se iniciaron con proyectos como:

- Mejoras al Sistema Bibliotecario.
- Automatización de Procesos de Fiscalización de los Programas y Planes de estudios universitarios.
- Juego de Telebingo.
- Sistema de Inventario interno.

La Sección de FS desarrollaba productos de Software aplicando un Marco de Trabajo (Framework de desarrollo) llamado Core, desarrollado por el mismo equipo de trabajo con el lenguaje de programación web JSF (Java Server Faces), en el servidor web, GlassFish. El Core era una estructura tecnológica, compuesta de módulos determinados que permitían el desarrollo a menor tiempo de proyectos de Sistemas de Información. Este marco de trabajo fue utilizado durante la gestión del Ing. Eddy Mojica hasta la ejecución del proyecto de automatización de Procesos de Fiscalización de programas y planes. Se descontinuó su uso debido a que el Framework era complejo de mantener y presentaba dificultades para trabajar sesiones a nivel web, un factor requerido para los proyectos en ese momento.

A partir del año 2013, inició la gestión de un nuevo Director del Centro de Investigación, Dr. Ramfis Miguelena y por traslado administrativo del Ing. Mojica, se asignó como coordinador de la Sección de FS, el Ingeniero Kexy Rodríguez, quien tenía la responsabilidad de monitorear y dar seguimiento a los proyectos. Desde ese año, la FS fue reorientada a ofrecer apoyo en proyectos de Software tanto a la Institución (Centros de Investigación, Facultades, Unidades Administrativas, etc.), como también a entes externos. Durante la administración del nuevo Director, se iniciaron desarrollos de proyectos como:

- Sistema de Información de Muestreo de Aguas.
- Apoyo a Congresos de investigación.

- Sistema de Incidencias para una plataforma educativa institucional Moodle.
- Sistema de Educación Continua de CIDITIC.
- Continuación de los proyectos adquiridos en la gestión anterior.

Durante este periodo, el equipo se redujo de siete (7) a cuatro (4) integrantes: un (1) Ingeniero de Sistemas Computacionales y tres (3) programadores.

A partir del año 2014 hasta la actualidad, por licencia de estudio del Ing. Kexy en un periodo de tiempo, se asignó como Coordinadora de la Sección a la Lcda. Nichol Sánchez. La Sección de FS durante ese periodo estaba conformada por tres (3) colaboradores (el coordinador y dos (2) desarrolladores). Los proyectos administrados durante este tiempo fueron:

- La finalización del proyecto de Sistema de Gestión de Inventario.
- Desarrollo y lanzamiento del Sistema de Información de Muestreo de Aguas para el Centro de Investigación CIHH.
- Dos (2) congresos de investigación.
- Un (1) Foro para el Centro de Producción e Investigaciones Agroindustriales (CEPIA).
- El inicio y desarrollo de la automatización de procesos de un Laboratorio de Ensayo de Materiales para el Centro Experimental de Ingeniería (CEI).

A pesar del reducido tamaño de personal, el equipo logró cumplir con la mayoría de los proyectos según su prioridad; pero el tiempo y el costo se vieron afectados por la cantidad del personal. A partir de ese momento, el Director de CIDITIC indicó que era necesario contar con un marco de trabajo para el desarrollo de Software que permitiese el seguimiento y medición del avance de los proyectos. Esta tarea fue asignada al Coordinador de la Sección. Durante el transcurso del año 2015 se reincorpora un Ingeniero en Sistemas y se contrata un estudiante de investigación para apoyar a la Sección de FS, aumentando el personal a cinco (5) colaboradores.

A finales del año 2016 e inicio del 2017 se integran dos (2) colaboradores más: un (1) programador tiempo completo y un (1) asistente estudiantil de investigación.

Meses después se integró un (1) programador adicional, siendo un total de ocho (8) colaboradores a finales de año 2017.

Durante este periodo se culminaron proyectos como: la automatización de procesos del Laboratorio de Ensayo de Materiales para el CEI, elaboración de un Concurso de Programación de Videojuegos, Sistema de la Jornada de Iniciación Científica, apoyo a dos (2) congresos nacionales y uno (1) Internacional.

Seguidamente a partir del 5 de diciembre de 2017, por cambio de estamento de investigación del Coordinador, se asigna como nuevo coordinador de la Sección de FS al Licdo. Francisco Marchena, contando con un equipo de tres (3) programadores; siendo un equipo de cuatro (4) colaboradores, (los cuatro (4) restantes fueron asignados a un 100% en investigación).

A inicios de marzo 2018, tres (3) miembros de la Sección de FS son trasladados a otros Departamentos como DITIC (Dirección General de Tecnología de la Información y Comunicaciones) y DIGITEC (Dirección General de Innovación y Tecnología Educativa) dentro de la Institución; trayendo como consecuencia que la Sección de FS deje de funcionar temporalmente. Actualmente, el Director está en espera de aprobación por parte de las autoridades de la universidad, la contratación de un equipo de cuatro (4) personas para dar continuación a proyectos externos e institucionales.

4.2 EXPERIENCIAS EN ESCENARIOS DE DESARROLLO

La ejecución de los proyectos en la Sección de FS inició a partir del año 2012. Desde ese año, la Sección ha participado en diferentes tipos de proyectos: páginas Web para distintos eventos (congresos, foros y jornadas de investigación), implementación de plataformas de recepción de artículos para congresos o revistas de investigación; y sistemas de información con objetivos variados: gestión de incidentes, gestión de inventario, gestión de educación continua, apoyo para participación de eventos nacionales e internacionales, automatización de flujos de trabajo, entre otros. En la TABLA 14. se presenta la cantidad total de proyectos que se han desarrollado desde el año 2012 hasta el 2017, clasificados por tipo:

TABLA 14.
CANTIDAD DE PROYECTOS POR TIPO
AÑO 2012 - 2017

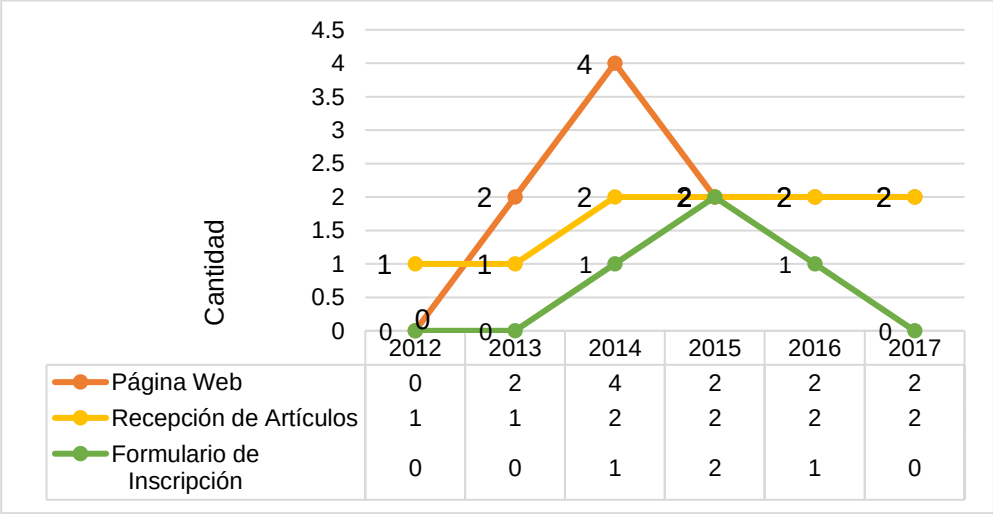
Tipo de Proyectos	Cantidad
Página Web	12
Sistemas de Información	11
Sistema de gestión de Conferencias	10
Formulario de Inscripción	04
Total	37

Fuente: Elaboración propia basándose en información de CIDITIC

En la gráfica anterior, se observa que el tipo de proyectos que más se ha desarrollado en la Sección son las páginas Web, seguido por los sistemas de Información, Plataformas de Recepción de Artículos y Formularios de Inscripción.

Con respecto a la duración de los proyectos, a continuación, se despliega la cantidad de páginas web, plataformas de Recepción de artículos y los formularios de inscripción categorizados por año:

GRÁFICA 4
PÁGINA WEB, RECEPCIÓN DE ARTÍCULOS Y FORMULARIO DE INSCRIPCIÓN POR AÑO 2012 - 2017



	2012	2013	2014	2015	2016	2017
Página Web	0	2	4	2	2	2
Recepción de Artículos	1	1	2	2	2	2
Formulario de Inscripción	0	0	1	2	1	0

Fuente: Elaboración propia basándose de información suministrada por CIDITIC

Estos tipos de proyectos cuentan con una duración no más de un año, a diferencia de los Sistemas de Información que duraron más de un año:

TABLA 15.
CRONOLOGÍA DE EJECUCIÓN DE LOS SISTEMAS DE INFORMACIÓN

Sistemas de Información	Duración (Años)											
	2012		2013		2014		2015		2016		2017	
Telebingo												Cerrado
Gestión y Fiscalización de los Programas y Planes de Estudios												Sin determinar
Gestión de acervos digitales para la biblioteca				Culminado								
Administración de encuestas online				Culminado								
Gestión de Inventario								Culminado				
Gestión de Incidentes								Culminado				
Gestión de Muestreos de la calidad de las aguas								Culminado				
Objetos Digitales de Aprendizaje								Culminado				
Gestión de Apoyo a la Participación de eventos Nacionales e Internacionales											Detenido	
Sistema de Educación Continua												En ejecución
Gestión de Procesos de Laboratorio de Ensayo de Materiales											Culminado	

En ejecución | Posible atraso | Atrasado | Completado | Detenido/Cerrado

Fuente: Elaboración propia – Información suministrada por CIDITIC

Para tener una mejor comprensión de cómo se han ejecutado los Sistemas de Información en la Sección de Fábrica de Software, presentamos la TABLA 15., donde se visualizan 11 proyectos en total. Cada fila despliega un número de estados que describe la vida de cada proyecto, permitiendo identificar si fue completado, se encuentra en ejecución, tiene riesgos de retraso o presenta retrasos. Para representar el estado del proyecto se utilizó la Clasificación RAG[1] (Red, Amber, Green Status) estándar que permite visualizar el estado del proyecto a medida que avanza. Cuando un proyecto es de color verde significa que hay poco riesgo, está bajo lo planeado según tiempo, presupuesto y objetivos. Cuando un proyecto es amarillo, indica que sigue en ejecución, pero existen riesgos (tiempo, presupuesto u objetivos) que deben ser controlados. El color rojo simboliza que existe un problema y debe ejecutarse un plan de contingencia para

[1] http://pmtips.net/blog-new/what-does-rag-status-mean

ajustar los términos costo, tiempo y objetivos. El color azul es utilizado para indicar que un proyecto fue finalizado. Por último, fue agregado a la lista por el autor de este trabajo, el color gris, con el motivo de poder identificar aquellos proyectos que fueron cerrados, detenidos o descontinuados.

Los años con mayor ejecución de proyectos de Sistemas de Información son los años 2013, 2014 y 2015. Durante ese periodo solo se encontraban tres (3) personas en la Sección, dos (2) programadores y el coordinador, lo que puede corroborar por qué los proyectos demoraron más de un año. El contar con una sola persona en un proyecto de Software o contar dos (2) personas en varios proyectos, impedía que el equipo ejecutara correctamente. A finales del año 2016 se adicionó un programador que permitió una repartición de actividades un poco más balanceada.

4.3 IDENTIFICACIÓN DE LAS INCIDENCIAS EN LOS PROYECTOS

El análisis del desempeño de proyectos permite identificar incidencias y oportunidades de mejora, con el objetivo de aumentar las probabilidades de éxito de futuros proyectos. Para identificar estas oportunidades en la FS, se seleccionaron los proyectos de mayor impacto y relevancia para la Sección. Los proyectos que analizaremos serán cinco (5) en total: dos (2) proyectos externos: Sistema de Telebingo y Sistema de Fiscalización de los programas y Planes de Estudios; dos (2) proyectos internos de CIDITIC: Sistema de Inventario de CIDITIC, Sistema de Registro de Incidencias de Moodle; y por último, un (1) proyecto de extensión: Sistema de Procesos del Laboratorio de Ensayo de Materiales.

4.3.1 Sistemas de Telebingo

Proyecto que inició su ejecución en el año 2012. El proyecto Telebingo consistió en el desarrollo de un Sistema de Juego de Bingo que debe generar y controlar los cartones de juego, sacar aleatoriamente los números en el juego, sacar los ganadores según el tipo de alineación y generar un reporte del juego.

TABLA 16.
CRONOLOGÍA DE SISTEMA DE TELEBINGO

Sistemas de Información	Duración (Años)											
	2012		2013		2014		2015		2016		2017	
Telebingo												Cerrado

Fuente: Elaboración propia – Información suministrada por CIDITIC

Como muestra la TABLA 16., el proyecto fue desarrollado por seis (6) años hasta su cierre a finales del año 2017, y cincuenta por ciento (50%) de su ciclo de vida presentó retrasos en su ejecución. Entre las causas de retrasos se pueden mencionar:

1) **Convenio ambiguo entre las partes**

   Al iniciar el proyecto se elaboró un convenio marco y un memorándum de entendimiento. Los mismos son los contratos que describen las funciones que cada parte debe efectuar. Al elaborarse, el nivel de especificación fue mínima, existían puntos con ambigüedades, ya que el personal de desarrollo de Software no se especializa en la elaboración de documentos legales. Además, no se consideró información en caso de que se presentasen situaciones alternas al proyecto, trayendo así, la necesidad de actualizar los documentos para eliminar las ambigüedades y agregar detalladamente las responsabilidades y soluciones a situaciones posibles.

2) **Comunicación entre el cliente y el analista**

   A pesar de haber existido una comunicación al inicio del desarrollo del proyecto y aprobación de los primeros módulos desarrollados, a medida que avanzaba el cronograma, el cliente perdía comunicación con el equipo de desarrollo de la FS de CIDITIC; el cliente durante las reuniones indicaba que entregaría cierto recurso (información, equipo, contacto) y esta no era entregada. Asignaba un personal para laborar con el equipo y el mismo no entregaba la información según se requería y su comunicación igualmente era poca.

   Al no contar con la información requerida en el momento y comunicación con el cliente por largos periodos de tiempo, otros

proyectos establecidos por la dirección fueron tomando prioridad y el proyecto fue detenido.

3) **Comunicación entre la alta Dirección, nivel medio y nivel operativo**

La poca comunicación entre los niveles trajo como consecuencia que el objetivo y la visión del proyecto no fueran transmitida correctamente entre todos los niveles; al no comunicarse correctamente, al cambiar los administradores de alta dirección y del nivel medio, el nivel operativo no contaba con la información suficiente para avanzar según los objetivos inicialmente establecidos, ni comunicar a las nuevas direcciones sobre los avances del proyecto.

4) **Rotación de Personal**

Durante la ejecución del proyecto hubo una rotación de personal significativa. El encargado de la Sección de FS fue trasladado a otro punto de la institución; y al presentarse ese escenario el resto de los integrantes de la sección iniciaron la búsqueda de nuevas oportunidades de trabajo o estudio.

4.3.2 Sistema de Fiscalización de los programas y Planes de Estudios

El proyecto trata sobre la automatización de los procesos de fiscalización y evaluación de los programas de estudios de las universidades particulares en Panamá. Este proyecto nació el mismo año que el proyecto de Telebingo, con el mismo administrador de la Sección y éste no ha sido finalizado. Tal como muestra la TABLA 17., el proyecto inicia en el año 2012 y actualmente tiene una duración total de seis (6) años.

TABLA 17.
CRONOLOGÍA DE SISTEMA DE FISCALIZACIÓN DE LOS PROGRAMAS Y PLANES DE ESTUDIO

Sistemas de Información	Duración (Años)											
	2012		2013		2014		2015		2016		2017	
Gestión y Fiscalización de los Programas y Planes de Estudios												Sin determinar

Fuente: Elaboración propia – Información suministrada por CIDITIC

Este proyecto presenta características similares al proyecto anterior:

1) **Definición del Alcance con el Cliente.**

En la fase de Análisis del proyecto no se definió entre las partes el alcance del sistema. El cliente solicitaba una sección que señalaban como indispensable, pero el equipo de desarrollo no lo consideraban como parte del alcance. Por ello, la importancia de definir cuáles son todos los elementos que contempla el proyecto y que ambas partes elaboren un documento que lo defina.

2) **Comunicación intermitente entre el cliente y el analista**

El nivel de compromiso del cliente es una característica fundamental para la ejecución del proyecto. Durante la ejecución del proyecto, el cliente no entregaba la información y se perdía el enlace de comunicación. No contar con la información requerida en el desarrollo del proyecto, no permitía el avance planeado del proyecto.

3) **Procesos por automatizar no aprobados**

El proyecto inició a desarrollarse con base a procesos que no estaban aprobados en el momento por la empresa cliente. El cliente indicó que iniciaran con el desarrollo del sistema con los procesos otorgados; y al estar definidos oficialmente se realizarían los cambios pertinentes en el sistema. Situación que puede afectar significativamente el avance del proyecto.

4) **Recurso irreal, planificación irreal**

Del presupuesto original, el cliente aceptó pagar muy por debajo del presupuesto real (24% del presupuesto estimado). Al comprometerse el presupuesto, el proyecto se ve significativamente afectado en la calidad del producto.

4.3.3 Sistema de Inventario para CIDITIC

El sistema de Gestión de Inventario tiene como objetivo administrar la información de inventario actual del Centro CIDITIC, llevando registro de la información del personal, los artículos, la relación del personal con su inventario, traslados de inventario entre el personal y la generación de

Informe de inventario. El proyecto inicia en el año 2013 y tomó tres (3) años para su finalización. Ver TABLA 18.:

TABLA 18.
CRONOLOGÍA DEL SISTEMA DE GESTIÓN DE INVENTARIO

Sistemas de Información	Duración (Años)											
	2012		2013		2014		2015		2016		2017	
Gestión de Inventario								Culminado				

Fuente: Elaboración propia – Información suministrada por CIDITIC

Durante el proyecto ocurrieron las siguientes situaciones:

1) **Documentación del Proyecto**

   Al iniciarse el proyecto el programador encargado en el momento no realizó el análisis correspondiente de los requerimientos e inició directamente la codificación. Tiempo después el programador fue trasladado a otro departamento y no existía documentación alguna del proyecto.

2) **Capacitación del Personal**

   Al trasladarse el programador inicial, el proyecto fue asignado a otro programador. El mismo no recibió una documentación o capacitación para continuar correctamente el proyecto. El personal establecido a un proyecto debe contar con las capacidades necesarias para el proyecto y además debe recibir una capacitación o documentación para que no existan ambigüedades al momento de desarrollar el proyecto.

3) **Fechas de entregas infinitas**

   Al no contar con una documentación, un personal capacitado y planeación, el personal asignado (1 persona), cambiaba la fecha de entrega, y no avanzaba en el desarrollo del Proyecto. Como alternativa en el momento después de un (1) año sin avance, se agrega otro programador y se realiza nuevamente análisis, diseño, desarrollo e implementación del proyecto. Luego de este cambio, el proyecto se desarrolló en un periodo en seis (6) meses.

4) Cierre de Proyecto

Al completarse la fase de desarrollo, pruebas e implementación, se contactó al cliente y se presentó el sistema. Este sugirió ciertas modificaciones que fueron tomadas en cuenta. Se realizaron las modificaciones, la capacitación al cliente y se entregó el sistema para iniciar su uso de forma verbal. El sistema no fue utilizado por un periodo mayor de 6 meses por falta de personal por parte del cliente, dando la percepción que el sistema no estaba terminado, debido a que no se finalizó formalmente.

4.3.4 Sistema de Registro de Incidencia de Moodle

Es un Sistema para gestionar las incidencias ocurridas en la Plataforma de Educación Moodle para los estudiantes, docentes y administradores de Moodle. Registra los tipos de incidencias, el flujo de trabajo de los incidentes y la generación de reportes. El proyecto al igual que el Sistema de Inventario inicia en el año 2013 y toma 3 años para su implementación, ver TABLA 19:

TABLA 19.
CRONOLOGÍA DEL SISTEMA DE REGISTRO DE INCIDENCIAS DE MOODLE

Sistemas de Información	Duración (Años)											
	2012		2013		2014		2015		2016		2017	
Gestión de Incidentes								Culminado				

Fuente: Elaboración propia – Información suministrada por CIDITIC

Entre las debilidades encontradas en el proyecto, podemos mencionar:

1) Programar por encima del alcance

En el periodo de desarrollo del proyecto el programador agregaba elementos que realizaban más de lo que solicitaba el alcance y como resultado la fecha de entrega se extendía.

2) Poco Personal asignado al proyecto

Durante la vida del proyecto se tenían asignados dos (2) personas al proyecto, el coordinador y el programador. Durante el desarrollo,

pruebas e implementación, un solo programador era el encargado de realizar todas las actividades requeridas para finalizar el proyecto.

3) **Extenso periodo de cierre de proyecto**

Finalizado el desarrollo del proyecto, el sistema se mantuvo en un largo periodo de pruebas por parte de los usuarios, sin respuesta definida de cuáles eran las modificaciones por realizar. Luego de finalizado el periodo de pruebas, el usuario aun solicitaba modificaciones a requerimientos ya definidos con anterioridad.

4.3.5 Sistema de Gestión de procesos de Laboratorio de Ensayo de Materiales

Es un Sistema que automatiza los procesos para el Laboratorio de Ensayo de Materiales, está compuesto por cinco (5) procesos principales: Solicitud de Cotización, Solicitud de Trabajo, Orden de trabajo, seguimiento y revisión de Informe, Reportes para Dirección y Administración del Sistema. El proyecto fue lanzado a finales del 2017.

TABLA 20.
CRONOLOGÍA DE SISTEMA DE GESTIÓN DE PROCESOS DE LABORATORIO DE ENSAYO DE MATERIALES

Sistemas de Información	Duración (Años)											
	2012		2013		2014		2015		2016		2017	
Gestión de Procesos de Laboratorio de Ensayo de Materiales											Culminado	

Fuente: Elaboración propia – Información suministrada por CIDITIC

El proyecto inició el año 2015 mediante la recolección de requerimientos y modelación de los procesos del laboratorio, ver TABLA 20. Algunas dificultades presentadas en la ejecución del proyecto:

1) **Cambio constante del cliente contacto**

Siendo el laboratorio un sitio con mucha carga de trabajo durante todo el año, el tener un contacto estable que nos permitiera reunir la información que se requería, podía llegar a ser complejo. Durante la vida del proyecto se han asignado cuatro (4) cliente contactos que han ido rotando durante la ejecución del proyecto.

2) **Repartición de tareas**

Se asignaron dos (2) programadores para el desarrollo del primer módulo. Durante la ejecución del proyecto, la tarea asignada entre los mismos programadores no era equitativa, un programador desarrollaba la mayoría de las funciones de mayor complejidad y el otro desarrollaba las menores. Como consecuencia, un programador se atrasaba por la carga de trabajo y el otro terminaba mucho antes.

3) **Ejecución de otros proyectos paralelamente**

Por motivos de prioridad de otros proyectos en la Sección de Fábrica de Software, ocurrieron múltiples interrupciones en el proyecto, lo que extendía la fecha de entrega. Además, el trabajar en múltiples proyectos no permitía que el personal se enfocara correctamente en el cumplimiento de los objetivos.

4.4 RESUMEN DE INCIDENCIAS IDENTIFICADAS

A continuación, presentamos un resumen de las incidencias identificadas en cada uno de los proyectos:

TABLA 21.
INCIDENCIAS IDENTIFICADAS EN LOS PROYECTOS DE SOFTWARE

Proyecto	Incidencias
Sistema de Telebingo	Acuerdo entre las partes
	Comunicación entre el cliente y el analista
	Comunicación entre la alta dirección, nivel medio y nivel operativo
	Rotación del personal
Sistema de Fiscalización de los programas y Planes de Estudio	Poca definición del alcance con el cliente
	Comunicación intermitente entre el cliente y el analista
	Procesos por automatizar no aprobados
	Planificación y recurso irreal
Sistema de Inventario para CIDITIC	Documentación del proyecto
	Capacitación del personal
	Fechas de entregas indeterminadas
	Cierre de proyecto
Sistema de Registro de incidencias de Moodle	Programar por encima del alcance
	Poco personal asignado al proyecto
	Extenso periodo de cierre de proyecto
Automatización de procesos de Laboratorio de ensayo de Materiales	Cambio constante del cliente contacto
	Repartición de Tareas no equitativas
	Ejecución de otros proyectos paralelamente

Fuente: Elaboración propia.

A continuación, se clasificarán estas incidencias en los cuatro (4) factores comunes de fracaso.

4.4.1 Factores de fracaso

Los autores Lehtinen, Mäntylä, Vanhanen, Itkonen, & Lassenius reunieron un listado de las causas comunes por lo que un proyecto de Software fracasa. Este listado a su vez es una recopilación de varios autores como: McLeod, MacDonell y Verner.

FIGURA 12
FACTORES COMUNES POR LO QUE UN PROYECTO FRACASA

Fuente: (Lehtinen, Mäntylä, Vanhanen, Itkonen, & Lassenius, 2014)

Los cuatro (4) factores identificados tal como muestra la figura anterior son: personal, métodos, tareas y entorno:

1. **Personal:** Existen múltiples causas en que las personas pueden influir en el éxito de un proyecto (interacción, habilidades y motivación). Esto incluye circunstancias tales como muchas personas involucradas en un proyecto, la falta de comunicación o conflictos entre los involucrados, la falta de premiación al equipo y la falta de experiencia en el personal puede causar el sobrepaso de los recursos o fracaso del proyecto (Lehtinen, Mäntylä, Vanhanen, Itkonen, & Lassenius, 2014).
2. **Métodos**: Las prácticas utilizadas por los miembros del proyecto son también causas comunes de fracaso. La manera en que se llevan a cabo las

actividades permite conocer la calidad del producto que se generará. Los involucrados que pueden afectar la calidad del producto son: los desarrolladores, usuarios, gerencia, agentes externos y equipo del proyecto (Lehtinen, Mäntylä, Vanhanen, Itkonen, & Lassenius, 2014).

3. **Tareas:** La ejecución correcta de los procesos influye en la calidad de los artefactos generados. Un artefacto mal elaborado puede causar que un proyecto fracase. Entre las actividades identificadas como fundamentales en un proyecto se encuentran: la identificación de los requerimientos, la gestión del proyecto, la participación del usuario, las capacitaciones y la gestión del cambio (Lehtinen, Mäntylä, Vanhanen, Itkonen, & Lassenius, 2014).
4. **Entorno:** Se refiere a las condiciones ambientales y propiedades organizacionales que tienen impacto en el software a producir. Un ambiente caótico es una causa común para que un proyecto de Software se sobrepase en recursos o fracase (Lehtinen, Mäntylä, Vanhanen, Itkonen, & Lassenius, 2014).

Para identificar cuáles son las causas principales por la que los proyectos en la Sección de FS no fueron finalizados según lo planificado, se seleccionaron los 5 sistemas de información especificados en el punto anterior y se clasificaron las incidencias ocurridas en cada desarrollo, utilizando los cuatro (4) factores de fracaso (Personal, tareas, métodos y entorno). A continuación, en el siguiente cuadro se presentan los resultados de este análisis:

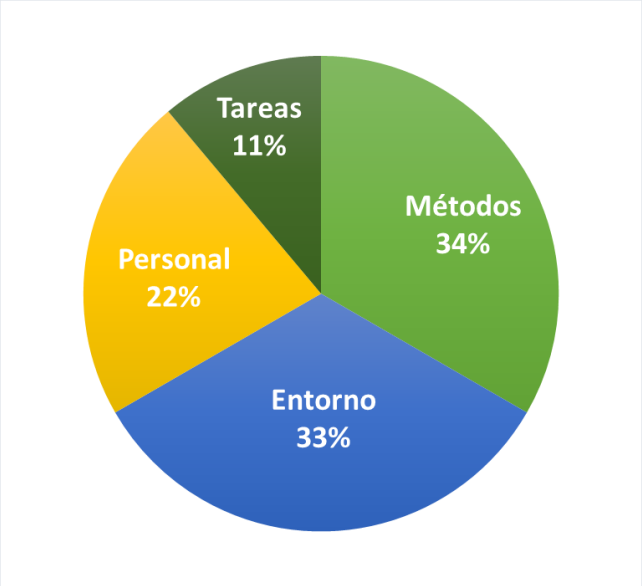
TABLA 22
CATEGORIZACIÓN DE PROYECTOS DE SISTEMAS DE INFORMACIÓN POR FACTORES DE FRACASO

Proyectos de Software						
Categorías de Causas	**Telebingo**	**Gestión y Fiscalización de los Programas y Planes de Estudios**	**Gestión de Inventario**	**Gestión de Incidentes**	**Gestión de Procesos de Laboratorio de Ensayo de Materiales**	**TOTAL**
Personal	1.Comunicación en la organización 2.Rotación de personal		2. Capacitación del personal	1. Programar por encima del alcance		**4**
Tareas		1.Planificación y recurso irreal.	1. Documentación del proyecto.			**2**
Métodos		2. Poca definición del alcance con el cliente.	3. Fechas de entrega indeterminadas. 4. Cierre de Proyecto indefinido.	3. Extenso periodo de cierre de proyecto	2. Repartición de tareas no equitativas. 3. Ejecución de otros proyectos paralelamente	**6**
Entorno	3.Acuerdos entre las partes. 4. Comunicación entre el cliente y el analista.	3. Comunicación intermitente entre el cliente y el analista. 4. Procesos por automatizar no aprobados		2.Poco personal asignado al proyecto.	1. Cambio constante del cliente contacto	**6**

Fuente: Elaboración propia

La Tabla 20 muestra las incidencias categorizadas bajo los cuatro (4) factores de fracaso. La siguiente gráfica resume los resultados por factor:

GRÁFICA 5
CATEGORIZACIÓN DE INCIDENCIAS POR FACTORES DE FRACASO



Fuente: Elaboración propia

Se observa que la mayoría de las incidencias responden a factores de Métodos (33%) y Entorno (33%), seguido de Personal (22%) y Tareas (12%). Esto indica que al menos 45% de las incidencias responden a problemáticas relacionadas a los componentes en una metodología de software (Métodos + Tareas), y reitera la necesidad de establecer una metodología que subsane estas condiciones y mejore los procesos para la finalización adecuada de proyectos en la FS.

4.5 CONCLUSIONES

El cuarto capítulo se enfocó en recolectar los antecedentes de la sección de Fábrica de Software, detallando los proyectos desarrollados, sus características y su ejecución a lo largo de los años.

Se destaca que los proyectos de mayor relevancia para la FS son los sistemas de información por su nivel de complejidad durante su ejecución, y se describe el perfil de estos proyectos para identificar las problemáticas (incidencias) más comunes en su desarrollo.

Estas incidencias fueron luego documentadas a detalle y clasificadas por factor de fracaso. Este análisis concluyó que los factores relacionados al tema de metodologías (tareas y métodos) son los de mayor recurrencia en la FS, y reiteró la necesidad de una propuesta de metodología de desarrollo adecuada que subsane las condiciones actuales. El siguiente capítulo presenta y describe esta propuesta.

4.6 REFERENCIA BIBLIOGRÁFICA

CIDITIC. (2017). ciditic.utp.ac.pa. Obtenido de http://www.ciditic.utp.ac.pa/

Lehtinen, T. O., Mäntylä, M. V., Vanhanen, J., Itkonen, J., & Lassenius, C. (2014). Perceived Causes of Software Project Failures - An Analysis of their relationships. Information and Software Technology, 3-9.

CAPÍTULO V
RESULTADOS DE LA INVESTIGACIÓN

5.1 INTRODUCCIÓN

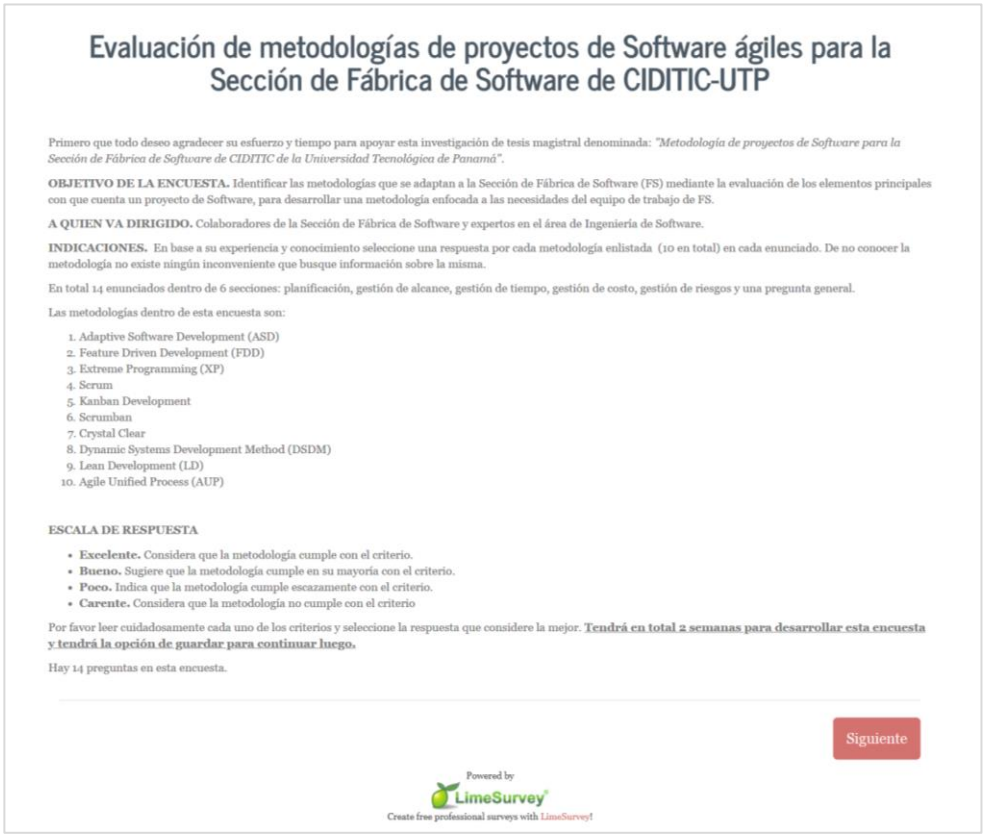
Este capítulo está compuesto de tres (3) partes fundamentales: los resultados de evaluación de las metodologías ágiles, la propuesta de metodología para la gestión de Proyectos de Software en la Sección de FS; y la sección de resultados finales, discusiones, recomendaciones y trabajos futuros.

Se presenta el proceso de evaluación de las metodologías, su aplicación y resultados obtenidos. Adicionalmente, se explica la propuesta de metodología, su formulación, especificaciones y características (principios y valores de la metodología, roles en el equipo, perfil de cada rol, descripción de procesos definiendo las entradas, técnicas, herramientas y salidas; y diseños de las plantillas necesarias para su ejecución).

5.2 APLICACIÓN DE LA EVALUACIÓN DE METODOLOGÍAS ÁGILES

Para aplicar la evaluación de metodologías ágiles se utilizó la herramienta de encuestas Limesurvey, Software Open Source (LimeSurvey, 2018). El Software ofrecía la opción de alojamiento personal (servidor propio) o uso de alojamiento de la empresa. En este caso se optó por utilizar el alojamiento de la empresa, y se obtuvo como enlace (url): metodologiasoftware.limequery.com/. Se introdujeron las preguntas y opciones a responder, resultando en la siguiente página:

FIGURA 13.
INTRODUCCIÓN DE LA ENCUESTA



Evaluación de metodologías de proyectos de Software ágiles para la Sección de Fábrica de Software de CIDITIC-UTP

Primero que todo deseo agradecer su esfuerzo y tiempo para apoyar esta investigación de tesis magistral denominada: *"Metodología de proyectos de Software para la Sección de Fábrica de Software de CIDITIC de la Universidad Teenológica de Panamá".*

OBJETIVO DE LA ENCUESTA. Identificar las metodologías que se adaptan a la Sección de Fábrica de Software (FS) mediante la evaluación de los elementos principales con que cuenta un proyecto de Software, para desarrollar una metodología enfocada a las necesidades del equipo de trabajo de FS.

A QUIEN VA DIRIGIDO. Colaboradores de la Sección de Fábrica de Software y expertos en el área de Ingeniería de Software.

INDICACIONES. En base a su experiencia y conocimiento seleccione una respuesta por cada metodología enlistada (10 en total) en cada enunciado. De no conocer la metodología no existe ningún inconveniente que busque información sobre la misma.

En total 14 enunciados dentro de 6 secciones: planificación, gestión de alcance, gestión de tiempo, gestión de costo, gestión de riesgos y una pregunta general.

Las metodologías dentro de esta encuesta son:

1. Adaptive Software Development (ASD)
2. Feature Driven Development (FDD)
3. Extreme Programming (XP)
4. Scrum
5. Kanban Development
6. Scrumban
7. Crystal Clear
8. Dynamic Systems Development Method (DSDM)
9. Lean Development (LD)
10. Agile Unified Process (AUP)

ESCALA DE RESPUESTA

- **Excelente.** Considera que la metodología cumple con el criterio.
- **Bueno.** Sugiere que la metodología cumple en su mayoría con el criterio.
- **Poco.** Indica que la metodología cumple escazamente con el criterio.
- **Carente.** Considera que la metodología no cumple con el criterio

Por favor leer cuidadosamente cada uno de los criterios y seleccione la respuesta que considere la mejor. **Tendrá en total 2 semanas para desarrollar esta encuesta y tendrá la opción de guardar para continuar luego.**

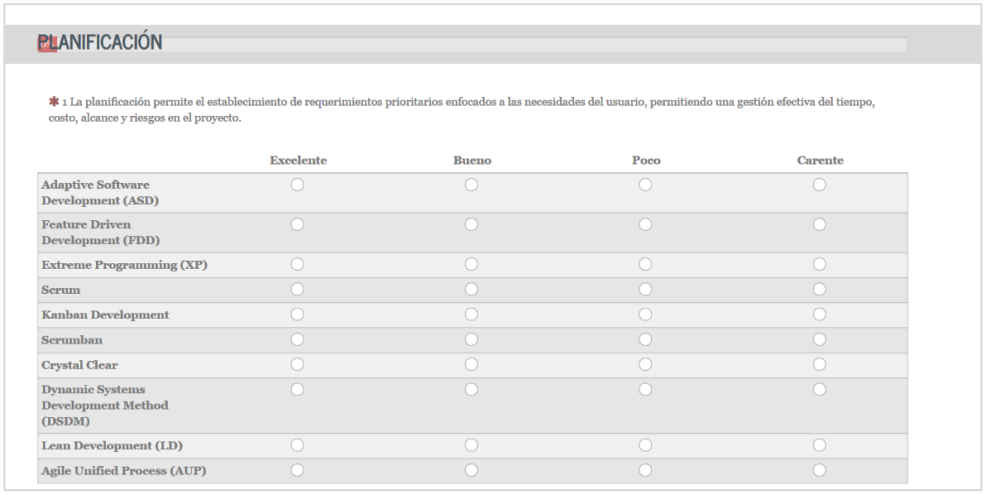
Hay 14 preguntas en esta encuesta.

Siguiente

Powered by LimeSurvey
Create free professional surveys with LimeSurvey!

Fuente: elaboración propia - (LimeSurvey, 2018)

FIGURA 14.
EJEMPLO DE PREGUNTA Y RESPUESTA DE LA SECCIÓN DE PLANIFICACIÓN



PLANIFICACIÓN

* 1 La planificación permite el establecimiento de requerimientos prioritarios enfocados a las necesidades del usuario, permitiendo una gestión efectiva del tiempo, costo, alcance y riesgos en el proyecto.

	Excelente	Bueno	Poco	Carente
Adaptive Software Development (ASD)	○	○	○	○
Feature Driven Development (FDD)	○	○	○	○
Extreme Programming (XP)	○	○	○	○
Scrum	○	○	○	○
Kanban Development	○	○	○	○
Scrumban	○	○	○	○
Crystal Clear	○	○	○	○
Dynamic Systems Development Method (DSDM)	○	○	○	○
Lean Development (LD)	○	○	○	○
Agile Unified Process (AUP)	○	○	○	○

Fuente: elaboración propia - (LimeSurvey, 2018)

Los participantes recibían un correo electrónico de invitación para acceder a la encuesta. Al ingresar, la pantalla inicial mostraba la introducción (ver FIGURA 13), desplegando: el título, el objetivo, a quién va dirigido, las indicaciones, escala de respuestas y el tiempo de desarrollo de la encuesta. La encuesta era de acceso privado y contaba con la opción de guardar y continuar luego. Seguido de la introducción, se presentaban seis (6) secciones de preguntas (ver detalle de secciones y preguntas en el Anexo A. Evaluación de Metodologías por medio de criterios de gestión de proyectos), las preguntas se visualizaban tal como muestra la FIGURA 14. en la página anterior.

5.2.1 Involucrados encuestados

Los participantes de la encuesta se eran dos (2) tipos: equipo de trabajo de la FS y docentes expertos de Ingeniería de Software. Ver cantidad de encuestados en TABLA 23.:

TABLA 23.
NÚMERO DE PARTICIPANTES PARA LA EVALUACIÓN DE METODOLOGÍAS

Encuestados	Cantidad
Experto de Ingeniería de Software	**7**
Miembro de la Sección de la Fábrica de Software	**6**
Total	**13**

Fuente: elaboración propia

La encuesta en línea se encontró activa por un (1) mes y el número de respuestas de la evaluación fue el siguiente:

TABLA 24.
NÚMERO DE EVALUACIONES RECIBIDAS

Evaluaciones recibidas	Cantidad
Completas	**8**
Incompletas	**2**
Sin respuesta	**3**
Total	**13**

Fuente: elaboración propia

Como muestra las tablas anteriores, de trece (13) evaluaciones enviadas se recibieron ocho (8) completas, dos (2) incompletas y tres (3) sin

respuesta. Se utilizaron las evaluaciones completas como insumos para la selección de la metodología.

5.3 RESULTADOS DE LA EVALUACIÓN

Al adquirirse las evaluaciones completadas, se procedió a cuantificar los resultados por pregunta basado en la escala de Likert definida en la TABLA 13 de la página 79 en el del Capítulo III. De estos resultados, se seleccionaron aquellas metodologías que obtuvieron el mayor puntaje en cada criterio (los puntajes finales de todas las metodologías en cada criterio pueden visualizarse en el Anexo C. Resultados con base a la escala propuesta).

Seguido se calculó cuánto equivale el puntaje según la ponderación del criterio definido (ver ponderaciones por criterio en Anexo A. Evaluación de Metodologías por medio de criterios de gestión de proyectos). Dando como resultado que la metodología de mayor ponderación es la que mejor se ajusta a las características del criterio y por ello, son las más adecuadas para la propuesta. A continuación, el resultado de este análisis:

TABLA 25.
METODOLOGÍAS CON MAYOR PUNTAJE POR CRITERIO

Criterio	Sección	Enfoque	Metodología	Puntaje Total	%
1	Planificación	Establecimiento de requerimientos prioritarios	Scrum	**12.5**	**7.8%**
2	Planificación	Participación del equipo en la planificación	DSDM	**12.5**	**7.8%**
3	Planificación	Documentación clara y compartida	Kanban	**11**	**3.4%**
4	Planificación	Retroalimentación de actividades ejecutadas	Scrum	**14.5**	**4.5%**

Criterio	Sección	Enfoque	Metodología	Puntaje Total	%
5	Alcance	Priorización de requerimientos para maximizar valor del negocio	Scrum	**14**	**7.9%**
6	Alcance	Refinamiento de requerimientos para aumentar la consistencia	DSDM	**13**	**5.7%**
7	Alcance	El cliente participa activamente en el diseño del producto	ASD	**14**	**7.9%**
8	Tiempo	Colaboración del equipo de trabajo para estimar el tiempo	Scrum	**13**	**7.3%**
9	Tiempo	Colaboración del equipo para determinar las iteraciones y tareas	Crystal Clear	**13**	**5.7%**
10	Tiempo	Los clientes participan en las iteraciones retroalimentando	DSDM	**12**	**6.8%**
11	Costo	Colaboración del equipo de trabajo para estimar el costo	Crystal Clear	**13**	**8.1%**
12	Riesgos	Colaboración del equipo de trabajo para la identificación de riesgos	DSDM	**12**	**3.8%**
13	Riesgos	Elaboración oportuna de pruebas para asegurar la calidad	ASD	**14**	**4.4%**
			Total de evaluación	**169/208**	**81/100**

Fuente: Elaboración propia

La TABLA 26 presenta que las metodologías de mayor interés para los encuestados es Scrum con un 28% obteniendo el mayor puntaje en los criterios de planificación, alcance y tiempo. Seguido por DSDM con un 24% en los criterios de planificación, alcance, tiempo y riesgos; ASD con un 14% en los criterios de alcance y riesgos; Crystal Clear con 12% en tiempo y costo; y Kanban con un 3% en planificación.

TABLA 26
RESUMEN DE RESULTADOS

No.	Metodología	Criterios con mayor porcentaje	Total de Porcentaje
1	Scrum	2 (Planificación) 1 (Alcance) 1 (Tiempo)	28%
2	DSDM	1 (Planificación) 1 (Alcance) 1 (Tiempo) 1 (Riesgos)	24%
3	ASD	1 (Alcance) 1 (Riesgos)	14%
4	Crystal Clear	(1) Tiempo (1) Costo	12%
5	Kanban	(1) Planificación	3%

Fuente: Elaboración propia

Para la propuesta se utilizará SCRUM como metodología base, complementada con DSDM, ASD, Crystal Clear y Kanban (considerando aquellas características donde estas metodologías presentaron mayor porcentaje).

Al realizar la sumatoria de todos los porcentajes de cada criterio con su metodología correspondiente, se obtiene un resultado de 81%/100% (Tal como muestra la TABLA 25.). Esto quiere decir que las metodologías resultantes pueden ser calificadas como válidas y buenas para la propuesta. Se puede corroborar esta información utilizando como referencia la escala de puntuación de la Universidad Tecnológica de Panamá que expresa la TABLA 27 (Universidad Tecnológica de Panamá, 2015), donde el puntaje de 81% calificaría como bueno:

TABLA 27

SISTEMA DE CALIFICACIONES DE LA UTP

Calificación	Significado	Rango numérico
A	Sobresaliente	91 – 100
B	Bueno	81 – 90
C	Regular	71 – 80
D	Mínima de Promoción	61 – 70
F	Fracaso	Menos de 61

Fuente: Sistema de Calificaciones de la UTP del Capítulo VI, Régimen Académico, Sección D, artículo 177 (Universidad Tecnológica de Panamá, 2015)

5.4 PROPUESTA DE METODOLOGÍA ÁGIL DE PROYECTOS DE SOFTWARE PARA LA FS

Cada etapa de este trabajo de investigación contribuyó a la definición de una metodología ajustada y adaptada para las necesidades de la sección de FS. Se consideraron las 5 metodologías resultantes en la evaluación: Scrum, Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Crystal Clear y Kanban. Por recomendaciones del Centro de Investigación, se tomó la decisión de llamar la metodología “CIDITIC-Scrum". En la siguiente sección, se procede a la presentación de la propuesta.

5.4.1 Introducción

Con el objetivo de mejorar la gestión de proyectos en relación con el alcance, el tiempo, costo y la satisfacción del cliente en la Sección de FS de CIDITIC-UTP, se recomienda la metodología Scrum complementada con principios fundamentales del pensamiento ágil de DSDM, valores de Crystal Clear, y técnicas de ASD y Kanban.

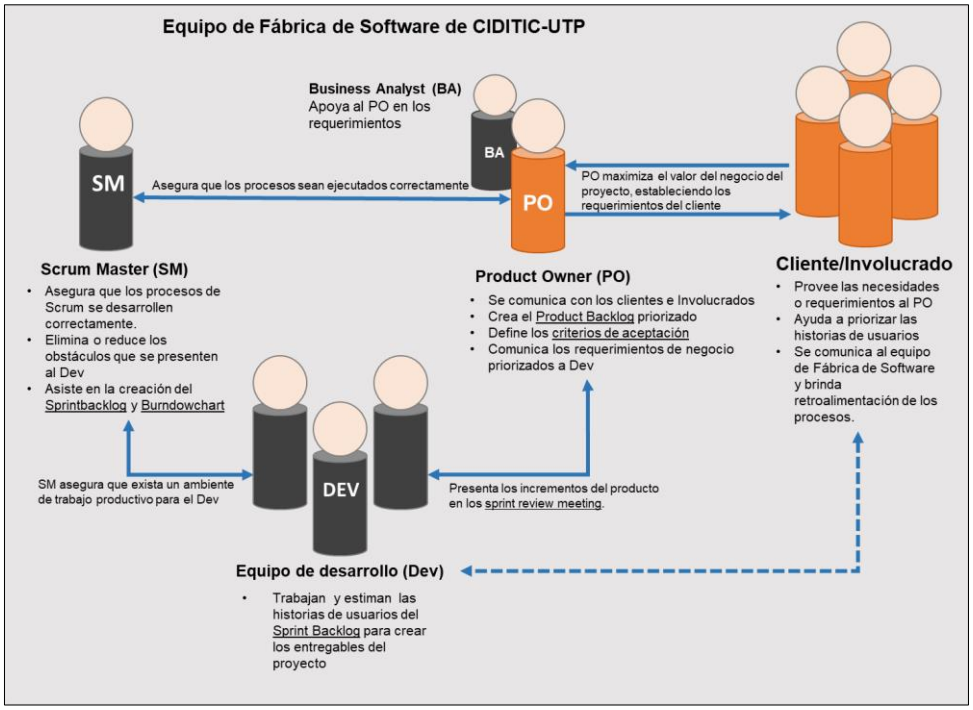
CIDITIC-Scrum se centra en las actividades prioritarias a desarrollar, como base inicial para el equipo de trabajo. A medida que el equipo acumule experiencia en el uso de la metodología podrá agregar técnicas, herramientas y procesos que mejoren su calidad.

Esta propuesta se compone de tres (3) partes fundamentales: la definición de la estructura del equipo de trabajo (estructura organizacional a utilizarse y perfil requerido de cada miembro del equipo); los principios y valores ágiles a seguir por el equipo de trabajo; y el ciclo de vida de la Metodología exponiendo las fases de la metodología (inicio, planificación, ejecución, revisión y retrospectiva) definiendo: las entradas, técnicas, herramientas, salidas e incluyendo el diseño de las plantillas a utilizar.

5.4.2 Estructura del Equipo de Trabajo

Para la ejecución correcta de proyectos de Software se requiere contar como mínimo los siguientes roles de trabajo. Cada uno contará con funciones específicas y necesarias de ejecución:

FIGURA 15.
ROLES DE TRABAJO DENTRO DE LA METODOLOGÍA CIDITIC-Scrum



Fuente: elaboración propia, basado en (SCRUMstudy, 2017)

Tal como muestra la FIGURA 15.

, existen dos (2) tipos de personal clasificados por colores: los grises que representan al equipo de la Sección de Fábrica de Software (Scrum Master, Equipo de desarrollo y el

analista del negocio) y los naranjas que representan el lado del Cliente (Product Owner, Clientes e involucrados). Se recomienda un equipo de entre tres (3) a nueve (9) miembros:

- (1) Scrum Master
- (4) programadores (Equipo de desarrollo)
- (1) Analista de Negocios

5.4.2.1 Roles de trabajo

- **Product Owner (PO).** Es la persona responsable de maximizar el valor del negocio del proyecto, establece los requerimientos del cliente y mantiene la justificación del proyecto; se considera como la voz del cliente. El Product Owner debe entender y apoyar las necesidades e intereses de todos los involucrados; además de entender las necesidades y trabajos del equipo de desarrollo.

  - **Perfil del Product Owner recomendado**

    Se recomienda que el colaborador asignado como Product Owner cuente o adquiera las siguientes características actitudinales:

    - Conocimiento de Scrum y sus procesos
    - Habilidades de comunicación
    - Dominar el tema del negocio
    - Habilidad para manejar la incertidumbre
    - Habilidad de negociación
    - Sea accesible, proactivo y decisivo
    - Pragmático y orientado a metas

- **Scrum Master (SM).** Facilitador que asegura el equipo de desarrollo se encuentre en un ambiente favorable para completar exitosamente el producto del proyecto; guiando, facilitando y enseñando las prácticas de Scrum a todos los involucrados en el proyecto; se encarga de remover los impedimentos para el equipo

de desarrollo y se asegura que los procesos de Scrum sean seguidos. El Scrum Master se considera el “líder de servicio” ya que modera y facilita las interacciones en el equipo como el entrenador y motivador.

- **Perfil del Scrum Master**

  Se recomienda que el colaborador que sea asignado para ser el Scrum Master cuente o adquiera el siguiente perfil actitudinal:
  - Conocimientos intermedios o experto de Scrum
  - Líder de servicio, moderador
  - Solucionador de problemas
  - Accesible, motivador, mentor
  - Habilidades de coordinación
  - Introspectivo

- **Equipo de desarrollo (Dev).** Equipo de personas responsables de entender los requerimientos del negocio especificados por el Product Owner, estimando historias de usuarios y finalmente creando los entregables del proyecto.
  - **Perfil del Equipo de desarrollo**

    Los colaboradores que sean asignados a este rol se recomiendan que cuenten o adquieran las siguientes características actitudinales:
    - Conocimiento de Scrum
    - Colaborativo, autoorganizado
    - Altamente motivado y proactivo
    - Expertos técnicos
    - Perspectiva multifuncional
    - Independiente, responsable e intuitivo
    - Orientado a metas
    - Introspectivo

- **Analista de Negocios (Business Analyst – BA).** Rol adicional que ofrece una perspectiva agregada en el equipo de Scrum. El mismo facilita la comunicación entre el equipo de desarrollo y el Product Owner; verifica que las necesidades del negocio se estén evaluando correctamente. Además, evalúa los requerimientos técnicos. Entre sus responsabilidades se encuentra:
  - Resolver ambigüedades entre el equipo de técnico y el equipo de negocio.
  - Gestión de información relacionado a los requerimientos del negocio.
  - Apoya al Product Owner en detallar las historias de usuario del Backlog del Producto.
  - Asegura que las implicaciones comerciales de las decisiones cotidianas se consideren adecuadamente. (Craddock, Richards, Tudor, Roberts, & Godwin, 2012)

  El analista de negocios apoyará al Product Owner en la ejecución de las actividades que le corresponde.

- **Cliente/Involucrados.** Término colectivo que incluye a los clientes, usuarios, patrocinadores; los cuales sostienen interacción frecuente con el equipo de Scrum, facilitando la información y creación de los entregables del proyecto.

5.4.3 Principios de la Metodología

Los principios de la metodología CIDITIC-Scrum a seguir por el equipo están basados en la metodología de Scrum y DSDM (2.5.1Comparación de Metodologías ágiles, ver página 44); en total son nueve (9) principios a seguir:

1. **Control empírico de procesos**: Principio enfatizado en la filosofía central de Scrum basada en las tres (3) ideas principales de transparencia (fácil y transparente flujo de información en toda la organización, creando una cultura abierta de trabajo), inspección (uso

de Kanban Board, retroalimentación frecuente e inspección final para validar el producto) y adaptación (mejora del trabajo por parte del equipo de Scrum y los involucrado por aplicar la transparencia e inspección) (SCRUMstudy, 2017).

2. **Priorización basada en el valor del negocio.** Ofrecer el máximo del valor del negocio durante toda la vida del proyecto (SCRUMstudy, 2017).
3. **Desarrollo Iterativo:** Combinar el desarrollo iterativo, las demostraciones frecuentes y la revisión exhaustiva para motivar la retroalimentación oportuna del cliente. Y aceptar los cambios como parte del proceso evolutivo permitiendo al equipo buscar la mejor solución (Agile Business Consortium, 2014).
4. **Time-Boxing:** El tiempo es un factor importante para el éxito de un proyecto. Es considerado una restricción de limitación en la metodología, facilitando gestionar eficazmente la planificación y ejecución de proyectos.

   Entre los elementos de Time-Boxed se incluyen Sprints, Reunión diaria de Scrums, Reuniones de Planificación de Sprint y Reuniones de Revisión de Sprint (SCRUMstudy, 2017).
5. **Autoorganización:** Los equipos autoorganizados entregan un mayor valor significativo, debido al trabajo, al compromiso y la propiedad compartida de lo alcanzado; en un entorno innovador y creativo que propicia el crecimiento (SCRUMstudy, 2017).
6. **Colaboración:** El principio se centra en tres (3) dimensiones relacionadas al trabajo colaborativo: la conciencia, la articulación y la apropiación. Este principio se centra en las tres (3) dimensiones fundamentales relacionadas con el trabajo colaborativo: la conciencia (los miembros del equipo deben conocer el trabajo de los demás), la articulación (los miembros del equipo deben dividir el trabajo en unidades, dividir las unidades entre los miembros y finalizado reintegrar las unidades resultantes) y la apropiación (adaptar la tecnología a la propia situación) (SCRUMstudy, 2017).

7. **Comuníquese de forma continua y clara:** La poca comunicación es una de las causas principales de fracaso de los proyectos. La metodología promueve la interacción humana por medio de reuniones diarias de Scrum, talleres con el cliente y la creación de prototipos para mejorar la comunicación (Agile Business Consortium, 2014).
8. **Nunca comprometer la calidad:** Desde el inicio se acuerda el nivel de calidad esperado con el cliente y todo el trabajo debe ir enfocado a ese nivel, ni más ni menos; permitiendo entregar el producto mínimo viable que cumpla con los criterios de aceptación acordados (Agile Business Consortium, 2014).
9. **Construya incrementalmente desde bases firmes:** Establecer bases firmes para el proyecto antes de comprometerse con un desarrollo significativo; se debe comprender primeramente del alcance del problema a resolver y la solución propuesta sin demasiado detalle (Agile Business Consortium, 2014).

5.4.4 Propiedades de la Metodología

La metodología cuenta con tres (3) propiedades o valores que debe tener el equipo para gestionar más eficiente el desarrollo del proyecto de Software, y deben estar presentes en todos los proyectos (Cockburn , 2004):

1. **Entregas frecuentes:** El primer valor que debemos tomar en cuenta es la entrega de código en ejecución y probado a los usuarios finales en pocos meses (El periodo recomendado es dos (2) meses y no más de cuatros (4) meses). Ventajas que presenta esta propiedad:
    - Los patrocinadores o clientes reciben retroalimentación en la tasa de progreso del equipo.
    - Los usuarios descubren si las solicitudes iniciales son realmente lo que necesitaban y las retroalimentaciones son utilizadas para para el proceso de desarrollo.

- El equipo de Scrum se mantiene enfocado y deshace dudas.
- El equipo de desarrollo depura los procesos en elaboración e implementación y obtiene un impulso moral a través de los logros.

2. **Crecimiento reflexivo:** El equipo debe enlistar qué está funcionando y qué no lo está, discutir qué podría funcionar mejor y hacer esos cambios en la siguiente iteración; es decir *reflejar y mejorar*. Esta actividad puede realizarse una (1) hora semanalmente.
3. **Comunicación Osmótica**: La información fluye dentro del espacio donde se encuentra el equipo de Scrum, los miembros escuchan información relevante de los demás del equipo creando una ósmosis, se comparte la misma oficina; de tal forma que, si algún integrante hace una pregunta, los de más de la oficina pueden estar o no al tanto; contribuyendo o continuando con sus tareas. En un ambiente osmótico las preguntas y respuestas fluyen naturalmente y hay retroalimentaciones más enriquecedoras (Cockburn , 2004).

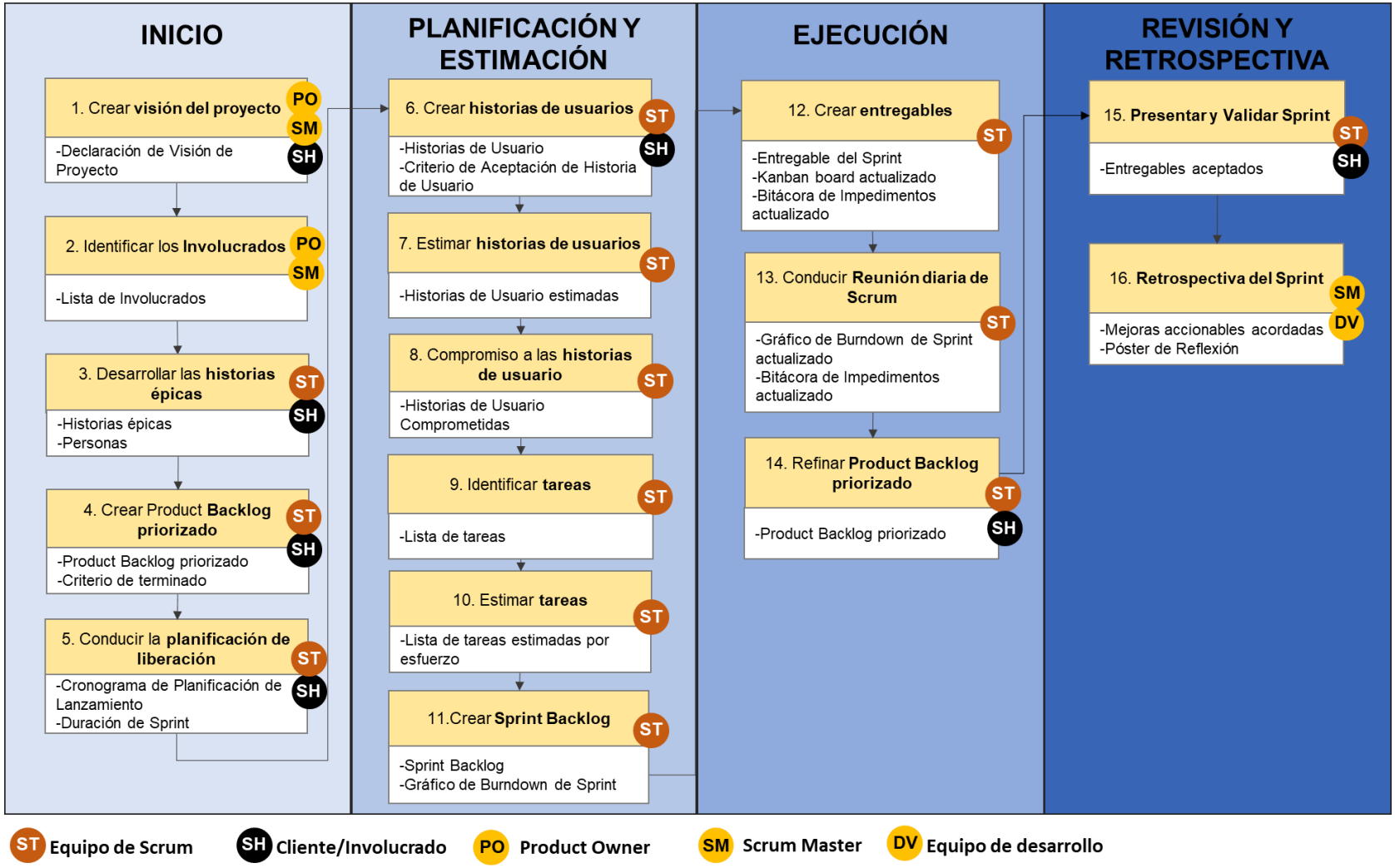
5.4.5 Ciclo de Vida de la Metodología

El ciclo de vida de la metodología propuesta está basado en la Metodología Scrum, con un total de dieciséis (16) procesos fraccionadas en las cuatro (4) fases:

1. Inicio
2. Planificación y estimación
3. Ejecución
4. Revisión y retrospectiva (SCRUMstudy, 2017).

A continuación, se presenta proceso de la metodología, mostrando las fases y sus procesos. El diagrama profundiza en la función de proceso, las entradas, herramientas, técnicas y salidas correspondientes.

FIGURA 16.
PROCESOS DE LA METODOLOGÍA CIDITIC-Scrum



Fuente: Elaboración propia basándose en (SCRUMstudy, 2017)

FASE I. Fase de Inicio

La fase de Inicio tiene como objetivo principal crear la base inicial del proyecto, se identifican a los actores y las necesidades; luego se categorizan en historias épicas. Esta fase se compone de cinco (5) procesos fundamentales:

FIGURA 17.
PROCESOS DE LA FASE DE INICIO

Fuente: elaboración propia basándose de (SCRUMstudy, 2017)

1. **Crear Visión del Proyecto.** El Caso de Negocio y el modelado de negocio del proyecto elaborados antes del inicio el proyecto, son utilizados como insumos para generar la Declaración de Visión del Proyecto. Este elemento permitirá que el equipo mantenga el enfoque durante todo el proyecto.

Entradas
1- *Caso de Negocio del proyecto*. Documento estructurado o declaración verbal que expresa la iniciación del proyecto. Incluye ocasionalmente información esencial de los antecedentes, propósito del negocio, análisis FODA, listado de los riesgos identificados y estimaciones (esfuerzo, tiempo y costo).
2- Modelado de Negocio Diagramas elaborados colaborativamente que definen el flujo del problema o la posible solución del negocio; facilitan la comunicación ya que son visuales, mejoran el entendimiento entre todas las partes y limita el alcance del problema o solución a resolver.
Herramientas y Técnicas
1- *Reunión de Visión del Proyecto*. Sesión llevada a cabo con los involucrados, el Product Owner, Analista de Negocios y el Scrum Master para identificar el contexto del negocio, los requerimientos y las expectativas de los involucrados; y así elaborar la Declaración de Visión del Proyecto. Se debe determinar la misión, objetivos y las restricciones del proyecto.
2- *Joint Application Design (JAD)*. Técnica de recolección de requerimientos que acelera el proceso "*crear visión del proyecto*", permitiendo que los involucrados del negocio o técnicos y los tomadores de decisiones lleguen a un consenso en el alcance, objetivos y otras

especificaciones del proyecto. Las sesiones pueden ser más de una y pueden ejecutarse entre dos (2) a cinco (5) días.
3- *Talleres de facilitación* Talleres que promueven el trabajo colaborativo y la toma de decisiones en equipo en cortos periodos de tiempo, útiles para: identificación de requerimientos, priorización, análisis de riesgos, etc.; idealmente son facilitados por alguien externo al proyecto que cree un ambiente participativo. El proceso del taller debe incluir: objetivo, identificación de los participantes, agenda, logística y distribuir cualquier información o lectura antes del taller. *4- Prueba del Concepto (Prototipo)* Prototipo que demuestra y verifica si la idea a desarrollarse es viable financiera y técnicamente; ayuda a entender los requerimientos y evaluar las decisiones de diseño a inicios del proyecto.
Salidas
1- Declaración de Visión del Proyecto. Explica las necesidades del negocio que desea cumplir, pero no el cómo. El mismo no es tan específico y debe ser flexible ya que la mayoría de la información puede llegar a ser especulaciones. Se debe enfocar en los problemas del negocio **(Anexo D. Plantillas de la Metodología, A. Declaración de Visión de Proyecto).**

Fuente: (SCRUMstudy, 2017)

2. **Identificar los involucrados de proyecto.** Se identifican los involucrados del proyecto por medio del Product Owner, Analista de Negocios y el Scrum Master utilizando la técnica de criterio de selección.

Entradas
1- Declaración de Visión del Proyecto.
Herramientas y Técnicas
1- *Criterio de Selección.* La identificación de involucrados relevantes es crucial para el éxito de cualquier proyecto, ya que ellos influyen durante toda la vida de este. Tomar en cuenta que un involucrado puede ser: los clientes, usuarios, patrocinadores. Deberán ofrecer la información y facilitar la creación de productos del proyecto.
Salidas
1- Involucrados identificados Incluye clientes, usuarios, patrocinadores que constantemente tendrán comunicación con el equipo de Scrum, trabajando colaborativamente y retroalimentando durante todo el proceso del proyecto **(Anexo D. Plantillas de la Metodología, B. Lista de Involucrados).**

Fuente: (SCRUMstudy, 2017)

3. **Desarrollar las historias épicas.** En este proceso se utiliza la Declaración de Visión de Proyecto y la reunión con grupos de usuarios para identificar las historias épicas iniciales.

Entradas
1- *Declaración de Visión del Proyecto.* 2- *Equipo de Scrum (Product Owner, Analista de Negocios, Scrum Master y equipo de desarrollo)* 3- *Contratos aplicables* El contrato define el alcance del trabajo y términos específicos; puede ser para todo o una porción del proyecto. Existe el contrato de entrega incremental, que incluye puntos de inspección a intervalos determinados, permitiendo al cliente tomar decisiones sobre la elaboración del producto durante el proyecto, como: la aceptación del desarrollo del producto, detener el desarrollo o la solicitud de modificaciones al producto.
Herramientas y Técnicas
1- Reunión con grupo de usuarios. Implica a todos los involucrados relevantes (usuarios y clientes del producto) que proporcionarán información valiosa de sus expectativas, facilitando la generación de los criterios de aceptación del producto y la elaboración los *Epics o historias épicas* (gran cuerpo de trabajo que contiene un conjunto historias de usuarios). *2- Taller de Historia de Usuarios.* El Scrum Master es el encargado de facilitar esta sesión para ejecutar el proceso de desarrollo de historias épicas. Implica todo el equipo de Scrum y en ocasiones es deseable incluir a ciertos involucrados. Mediante este taller el Product Owner prioriza los requerimientos y permite a todo el equipo de Scrum contar con la misma perspectiva sobre los criterios de aceptación (Consenso en el equipo).

3- Prototipado Modelo experimental de una idea que es mostrada al equipo o a los involucrados del proyecto con el objetivo de adquirir retroalimentación en los requerimientos identificados. Existe varios tipos de prototipado: bocetos y prototipos de papel (recomendado cuando se está iniciando la idea); bocetos y prototipo digital (prototipo interactivo digital sin programación); y prototipo nativo (prototipo interactivo con programación) (Google for Entrepreneurs, 2016).
Salidas
1- Historias épicas. Son historias de usuario de alto nivel elaboradas con descripciones amplias a temprana etapa del proyecto. Cuando las mismas forman parte del Product Backlog priorizado, estas son desfragmentadas en pequeñas historias de usuario (descripciones pequeñas y simples, fáciles de implementar en un Sprint) **(Anexo D. Plantillas de la Metodología, C. Historias épicas).** *2- Personas.* Personajes de ficción altamente detallados que representan a la mayoría de los usuarios. Creados para identificar las necesidades de los usuarios meta y ayudar al Product Owner y al equipo entender el proyecto (usuarios, requerimientos y metas) **(Anexo D. Plantillas de la Metodología, D. Personas).**

Fuente: (SCRUMstudy, 2017)

4. **Crear Product Backlog priorizado.** En este proceso las historias épicas son refinadas y priorizadas por el equipo de scrum con el objetivo de crear el Product Backlog para el proyecto. De igual forma se establece el criterio de terminado (Done).

Entradas
1- *Equipo de Scrum* 2- *Historias Épicas* 3- *Personas*
Herramientas y Técnicas
1- Métodos de priorización de Historias de Usuarios (DSDM) ▪ *Esquema de priorización MoSCoW.* El nombre se deriva de las primeras letras de las frases: Must have - M (Debe tener esta característica, no es negociable), Should have - S (debería tener esta característica, debería incluirse de ser posible), Could have - C (podría tener esta característica, es menos crítico) y Wont have - W (no tendrá; característica con menor valor y menos crítico; puede considerarse para el futuro). El product Owner y el equipo clasifican las historias de usuario con las cuatro (4) frases; las historias de usuario que queden en Must have formarán parte del Producto Mínimo Víable. ▪ *Método de 100 puntos.* El cliente cuenta con 100 puntos para votar en las historias de usuarios y asignará un mayor peso a las historias de mayor prioridad e importancia. Cada grupo de miembros asigna los puntos a varios casos de usuario, seguido de una votación final para calcular y determinar los puntos asignados a cada historia de usuario.
Salidas
1- Product Backlog priorizado Lista priorizada de requerimientos de negocio y del proyecto, escritas como historias épicas, las cuales son desarrolladas por el Product Owner y el Analista de Negocio. El Product Backlog

priorizado se basa en tres (3) factores importantes: el valor (asegurar la entrega de las características con mayor valor del negocio primero), riesgo o incertidumbre (a mayor incertidumbre, mayor riesgo en el proyecto; los productos priorizados en el Product Backlog con alto riesgo se categorizan de alta prioridad y del mismo modo, deben contar con un plan de contingencia) y dependencias (requerimientos funcionales frecuentemente dependen de otros requerimientos funcionales o no funcionales; esto impacta el orden de prioridad en el Product Backlog) **(Anexo D. Plantillas de la Metodología, E. Product Backlog)**

2- *Criterio de terminado (Done).* Es el conjunto de reglas que son aplicadas a todas las historias de usuarios; estas deben ser claras debido a que evita la ambigüedad y ayuda al equipo seguir las normas de calidad obligatorias. Se considera una historia de usuario terminada cuando es mostrada y aprobada por el Product Owner que juzga por medio de los criterios de terminado y los criterios de aceptación.

Fuente: (SCRUMstudy, 2017)

5. **Realizar la planificación del lanzamiento.** El equipo de Scrum revisa las Historias de Usuario en el Product Backlog priorizado para desarrollar el Cronograma de Planificación de Lanzamiento (calendario de implementación por fases) y es compartido con los involucrados. También se establece la duración del Sprint.

Entradas
1- *Equipo de Scrum* 2- *Involucrados* 3- *Declaración de Visión del Proyecto* 4- *Product Backlog priorizado* 5- *Criterio de terminado*
Herramientas y Técnicas
1- Sesiones de planificación de lanzamiento. Las sesiones tienen como objetivo desarrollar el Cronograma de Planificación de Lanzamiento (definiendo los conjuntos de funcionalidades usables o productos que serán entregados al cliente); además de permitir al equipo de Scrum visualizar de forma general los lanzamientos y el calendario de entregas en desarrollo, verificando que estén alineados a las expectativas del cliente e interesados. El Cronograma de Planificación de Lanzamiento producido de las sesiones no necesariamente debe ser detallado desde el inicio, el mismo puede ser actualizado continuamente, en cuanto exista información relevante a incluir. *2- Métodos de priorización de lanzamiento.* Los métodos de priorización pueden ser definidos por la organización o puede referirse a los métodos Esquema de priorización MoSCoW y *Método de 100 puntos (Ver en Proceso* 4. *Crear Product Backlog priorizado).*
Salidas
1- Cronograma de Planificación de Lanzamiento. Define los productos que deben ser suministrados al cliente, junto con los intervalos planificados y fechas de entregas. No

necesariamente al finalizar un sprint debe haber un entregable para el cliente, la entrega puede ser planeada después de la ejecución de un conjunto de sprints, tomando en cuenta que ofrezca un suficiente valor del negocio al cliente.

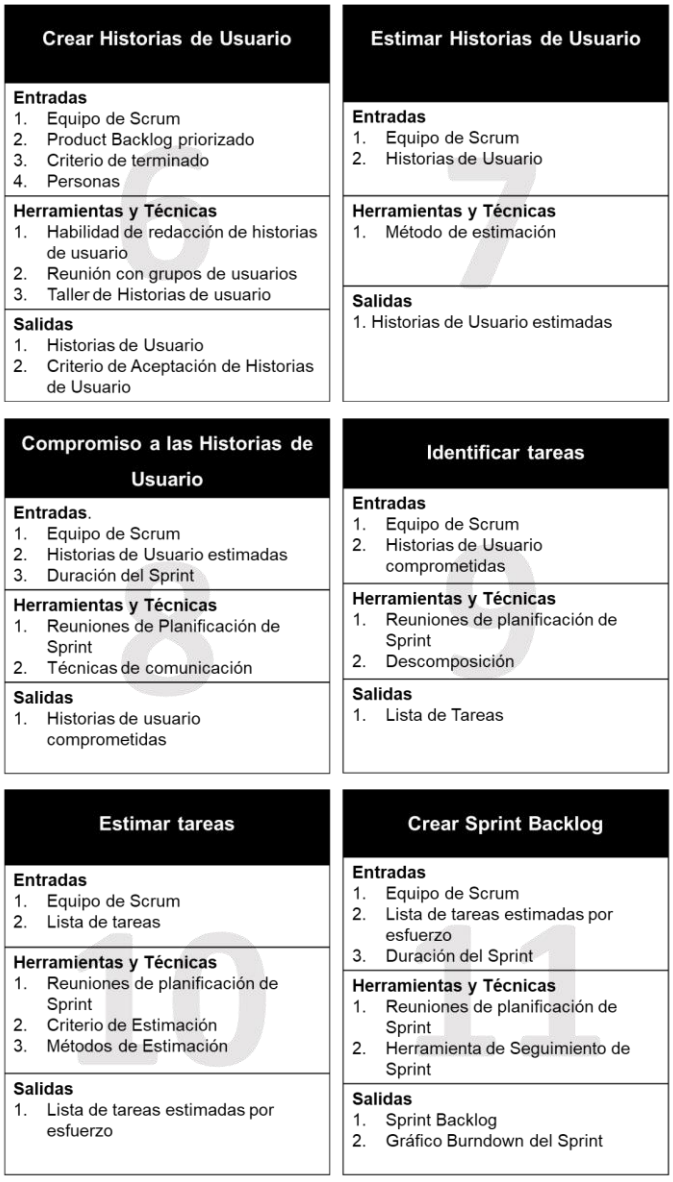
2- *Duración del Sprint.*
El equipo de Scrum basado en los requerimientos y el cronograma de planificación de lanzamiento decide la duración del sprint del proyecto (con frecuencia permanece el mismo durante todo el proyecto, aunque puede variar al inicio del proyecto hasta que el equipo considere la duración adecuada). El sprint puede durar de 1 a 6 semanas dependiendo de la estabilidad de los requerimientos, se recomienda 4 semanas aplicando la técnica de Time Boxing (duración máxima de tiempo para elaborar una o un conjunto de tareas).

Fuente: (SCRUMstudy, 2017)

FASE II. Planificación y Estimación

La siguiente fase está constituida por procesos de planificación y estimación de tareas. Dentro de la fase se encuentran seis (6) procesos: crear Historias de Usuario, estimar Historias de Usuario, compromiso a las Historias de Usuario, identificar tareas, estimar tareas y crear el Sprint Backlog.

FIGURA 18.
PROCESOS DE LA FASE DE PLANIFICACIÓN Y ESTIMACIÓN



Fuente: elaboración propia basándose de (SCRUMstudy, 2017)

6. **Crear Historias de Usuario.** Proceso donde el Product Owner crea las historias de usuario y su criterio de aceptación en conjunto con el Analista de Negocios; garantizando que se representen claramente y sean comprendidos por todos los involucrados. Igualmente, se pueden desarrollar talleres con el equipo de desarrollo para crear las historias de usuario. Todas las historias de usuarios creadas son agregados al Product Backlog priorizado.

Entradas
1- *Equipo de Scrum* 2- *Product Backlog priorizado* 3- *Criterio de terminado (Done)* 4- *Personas*
Herramientas y Técnicas
1- Habilidad de redacción de historias de usuarios. El Product Owner por medio de la interacción con los involucrados, el conocimiento del negocio, la habilidad y aporte del equipo, desarrolla las Historias de Usuario que formarán parte del Product Backlog priorizado inicial del proyecto; recibe apoyo por parte del Analista de Negocios. El objetivo de este proceso es elaborar y refinar las Historias de Usuario que posteriormente puedan ser estimadas y comprometidas por el equipo de desarrollo. Se pueden elaborar talleres de redacción de Historias de Usuario si es necesario. *2- Reunión con grupo de usuarios.* Involucra a todos los involucrados relevantes: usuarios y clientes del producto; los cuales proporcionan información valiosa de sus expectativas, facilitando generar los criterios de aceptación del producto y elaborar los épicos. *3- Taller de Historia de Usuarios.* El Scrum Master es el encargado de facilitar esta sesión para ejecutar el proceso de desarrollo de historias épicas. Implica todo el equipo de Scrum y en ocasiones es deseable incluir a

ciertos involucrados. Mediante este taller el Product Owner prioriza los requerimientos y permite a todo el equipo de Scrum contar con la misma perspectiva con los criterios de aceptación (Consenso en el equipo).

Salidas

1- *Historias de Usuario.*

Artefacto con estructura específica y simple que documenta los requisitos y funcionalidades deseadas por los usuarios; contestando tres (3) preguntas fundamentales de los requerimientos: Quién, Qué y Por qué. Las Historias de Usuario que son muy grandes para ser elaboradas en un solo sprint son consideradas épicas, las cuales son descompuestas en Historias de Usuarios más pequeñas al momento de ejecutarse. Todos los cambios realizados en los requerimientos son agregados al Product Backlog (repriorización, nuevo, actualización, refinación y eliminación) **(Anexo D. Plantillas de la Metodología, E. Product Backlog priorizado).**

Formato de Historia de Usuario

Yo como ***<rol/persona>****, quisiera poder* ***<requerimiento>*** *para así* ***<beneficio>****.*

2- *Criterio de Aceptación de Historias de usuario.*

Todas las Historias de Usuarios están asociados a un criterio de aceptación que proporciona la objetividad requerida para determinar si está terminado o no un requerimiento durante el Sprint Review (reuniones con todo el equipo y los involucrados para aceptar o desaprobar los entregables terminados en el sprint finalizado); además, brinda claridad al equipo en las expectativas esperadas y disminuye las ambigüedades. El Product Owner define y comunica los criterios de aceptación al equipo de desarrollo.

En los Sprint Review, el Product Owner utiliza los criterios de aceptación para decidir si la historia de usuario fue completada satisfactoriamente y el Scrum Master se asegura que el Product Owner no cambie los criterios de aceptación de una historia de usuario comprometida en un sprint en ejecución **(Anexo D. Plantillas de la Metodología, E. Product Backlog priorizado).**

Fuente: (SCRUMstudy, 2017)

7. **Estimar Historias de Usuario.** En este proceso el Product Owner aclara las Historias de Usuario con el objetivo que el Scrum Master y el equipo desarrollo puedan estimar el esfuerzo requerido para desarrollar las funcionalidades de cada historia de usuario.

Entradas
1- *Equipo de Scrum* 2- *Historias de Usuario*
Herramientas y Técnicas
1- Métodos de estimación. Existe un gran número de técnicas para estimar las historias de usuarios: – Wideband Delphi Técnica de estimación basada en grupo para determinar cuánto trabajo y tiempo se requiere para ser completado. Los miembros del equipo individualmente proveen una estimación por historia de usuario anónimamente y las mismas son trazadas en un gráfico; seguido el equipo discute los factores que influyen en las estimaciones y proceden al segundo periodo de estimación; este proceso se repite hasta que las estimaciones individuales son cercanas unas a otras y se llega a un consenso final. – Affinity Estimation Técnica utilizada para estimar rápidamente un gran número de historias de usuarios. Se escriben las historias de usuario en un papel pequeño (sticky notes) y cada miembro del equipo coloca un subconjunto de historias de usuario del product backlog priorizado en orden de tamaño pequeño a grande en un tablero o pared; esta primera colocación es realizada en silencio. Colocados todas las historias de usuarios, todo el equipo revisa y mueve historias de usuario según corresponda. La segunda parte del ejercicio se discute lo presentado y finalmente el

Product Owner indica las categorías en la pared (puede ser pequeño, mediano y grande o pueden enumerarse utilizando valores de puntos de historias para indicar el tamaño relativo); seguido el equipo categoriza las historias de usuario. Este método es transparente, visible para todos y fácil de llevar.

- Otros métodos: Fist of Five y Planning Poker.

Salidas

1- Historias de Usuario estimadas

Las historias que pasaron por un proceso de estimación por medio del uso de métodos son considerados Historias de Usuarios Estimados. Este proceso asigna un valor evaluando el tamaño de las historias de usuario considerando el nivel de riesgo, el esfuerzo requerido y el nivel de complejidad. Esta asignación es desarrollada por el equipo de desarrollo que al evaluar una de las historias de usuario dentro del product backlog priorizado puede tomar como referencia la primera para evaluar las demás historias de usuario **(Anexo D. Plantillas de la Metodología, E. Product Backlog priorizado).**

Fuente: (SCRUMstudy, 2017)

8. **Compromiso a las Historias de Usuario.** El equipo de desarrollo se compromete en entregar las Historias de Usuario aprobadas por el Product Owner en un Sprint.

Entradas
1- *Equipo de Scrum* 2- *Historias de Usuarios estimadas* 3- *Duración del Sprint*
Herramientas y Técnicas
1- Reuniones de Planificación de Sprint El equipo de desarrollo planifica el trabajo que efectuará en el sprint mediante la revisión del Product Backlog priorizado con las Historias de Usuario estimadas, en presencia del Product Owner en caso de que se requiera una aclaración o conocer las prioridades. La reunión tiene como límite de tiempo (time-boxed) dos (2) horas por semana de la duración del Sprint. Como resultado se obtiene el compromiso del equipo de desarrollo en entregar el subconjunto de historias de usuario del Product Backlog priorizado. *2- Técnicas de comunicación* Scrum incita a la comunicación informal (interacción cara a cara). Cuando sea necesaria la distribución del equipo de desarrollo, el Scrum Master se encarga de brindar los métodos necesarios para mantener la comunicación efectiva para que el equipo pueda autoorganizarse, colaborar y trabajar.
Salidas
1- Historias de Usuario comprometidos El equipo de desarrollo se compromete a elaborar el subconjunto de Historias de Usuario que ellos creen que pueden completar en el próximo sprint basado en su velocidad. Se debe recordar que las Historias de Usuario siempre deben ser seleccionados considerando las prioridades definidas por el Product Owner **(Anexo D. Plantillas de la Metodología, E. Product Backlog priorizado).**

Fuente: (SCRUMstudy, 2017)

9. **Identificar tareas.** Proceso enfocado en desfragmentar las Historias de Usuario comprometidas en tareas específicas y agruparlas en una lista de tareas.

Entradas
1- *Equipo de Scrum* 2- *Historias de usuarios comprometidos*
Herramientas y Técnicas
1- Reuniones de planificación de Sprint. El equipo de desarrollo se reúne para planificar el trabajo a ejecutar, revisando las Historias de Usuario comprometidas en el sprint, identificando las actividades o tareas requeridas para implementar los entregables necesarios para cumplir con las Historias de Usuario y su criterio de aceptación. Durante estas reuniones el Product Owner se encuentra presenta para aclarar dudas y apoyar en la toma de decisiones del diseño. 2- Descomposición El equipo de desarrollo descompone las tareas de alto nivel de las Historias de Usuario a tareas más detalladas. El Product Backlog priorizado debe estar lo suficientemente descompuesto a tal nivel que provee al equipo de desarrollo la información adecuada para crear los entregables de las tareas mencionadas en la lista de tareas.
Salidas
1- Lista de Tareas. Lista comprensible que posee todas las tareas que el equipo de desarrollo está comprometido a desarrollar en el sprint actual. Posee la descripción de cada tarea con el estimado elaborado en el proceso de identificación de tareas. La lista de tareas debe incluir cualquier prueba o esfuerzo de integración, de forma que el incremento del producto del Sprint pueda ser exitosamente integrado a los entregables anteriores de otros Sprints **(Anexo D. Plantillas de la Metodología, F. Lista de Tareas).**

Fuente: (SCRUMstudy, 2017)

10. **Estimar tareas.** En este proceso el equipo de Scrum estima los esfuerzos requeridos para cumplir con cada tarea de la lista de tareas.

Entradas
1- *Equipo de Scrum* 2- *Lista de Tareas*
Herramientas y Técnicas
1- Reuniones de planificación de Sprint. En la reunión el equipo de desarrollo se enfoca en estimar el esfuerzo y recurso requerido para terminar una tarea o un subconjunto de tareas en el Sprint, haciendo uso de la lista de tareas. El uso de reuniones trae como beneficio que el equipo comparta la misma perspectiva de las Historias de Usuario y requerimientos, para así estimar confiablemente el esfuerzo requerido. Es recomendable incluir también a involucrados claves en esta actividad. Esto asegurará que ambas partes comprendan claramente el nivel de esfuerzo y recursos que requerirá el proyecto. *2- Criterio de Estimación.* Este criterio se puede expresar de variadas formas, dos (2) elementos comunes son: puntos de historia (story points) y tiempo idea. Los puntos de historia representan el esfuerzo relativo o comparativo de completar las tareas; y el tiempo idea describe el número de horas que los miembros del equipo se encuentran trabajando exclusivamente en el desarrollo del proyecto, sin incluir el tiempo en otras actividades que no son parte del proyecto. *3- Métodos de Estimación.* Los mismos métodos utilizados para estimar Historias de Usuario puede ser aplicados para las tareas igualmente.

Salidas
1- Lista de tareas estimadas por esfuerzo. Lista de tareas asociada con las Historias de Usuarios comprometidas incluidas en el Sprint. La exactitud dependerá de las habilidades del equipo y la estimación del esfuerzo dependerá del criterio de estimación seleccionado por el equipo. Esta lista será utilizada por el equipo de desarrollo para crear el sprint backlog y el Gráfico de burndown del sprint; además la lista permitirá conocer si el equipo debe reducir la cantidad de compromiso o puede adquirir más compromiso en los sprint siguientes **(Anexo D. Plantillas de la Metodología, F. Lista de Tareas).**

Fuente: (SCRUMstudy, 2017)

11. **Crear Sprint Backlog.** El equipo de Scrum lleva a cabo reuniones de planificación de Sprints dando como resultado el Sprint Backlog que posee todas las tareas necesarias para completar el Sprint.

Entradas

1- *Equipo de Scrum*

2- *Lista de tareas estimadas por esfuerzo*

3- *Duración del Sprint*

Herramientas y Técnicas

1- Reuniones de planificación de Sprint

Durante estas reuniones el equipo de desarrollo por medio de las Historias de Usuario comprometidas en un Sprint y las tareas identificadas y estimadas; cada miembro usa la lista de esfuerzo estimado para seleccionar las tareas que planea trabajar en el Sprint, tomando en cuenta sus habilidades y experiencias. Además, con todos los artefactos se generan el Sprint Backlog y el Gráfico de burndown del sprint.

2- Herramienta de Seguimiento de Sprint – Kanban Board

Para dar seguimiento al progreso en un Sprint se hace uso del Kanban Board, conocido igualmente como tablero de tareas o flujo de proceso. El Kanban Board está dividido en cinco (5) secciones: Por hacer (To do), en ejecución (WIP - Work in progress) dividido por las subsecciones: análisis, desarrollo, pruebas; y terminado (done).

Por hacer	En ejecución (WIP)			Terminado
	Análisis	Desarrollo	Pruebas	

Haciendo uso de sticky notes se representan las tareas o Historias de Usuario, colocándolas en la categoría correspondiente. A medida que se avance en las tareas, irán pasando a la siguiente categoría (To do, Work in progress y Complete work). Al aplicar el Kanban board se debe limitar el

trabajo en ejecución, mediante la limitación de un número de tareas por subsecciones, se inculca terminar las tareas pendientes antes de iniciar nuevas ((BTI360, 2011).
Salidas
1- Sprint Backlog Lista de tareas que serán ejecutadas por el equipo de desarrollo en el Sprint. El mismo puede ser representado por un Kanban Board aportando una constante visibilidad del estado de las Historias de Usuario. En el Sprint Backlog de igual forma se despliega los riesgos asociados a las tareas; y su mitigación debe ser incluido como tareas dentro del Sprint Backlog. Al finalizarse la elaboración del Sprint Backlog no se debe agregar más Historias de Usuario; y de requerirse los mismos deben ser agregados al futuro Sprint. 2- Gráfico de burndown del sprint Gráfico que representa la cantidad de trabajo faltante en el sprint en ejecución. El Gráfico de burndown del sprint se actualiza cada día que el trabajo es completado y permite observar el progreso que realiza el equipo de desarrollo y facilita de detección de estimaciones incorrectas. De mostrarse en el gráfico que el equipo de desarrollo no está en camino de finalizar las tareas dentro del Sprint en el tiempo determinado, el Scrum Master deberá identificar los impedimentos para completar correctamente o tratar de eliminarlos. Cabe destacar que existe el Gráfico Burnup del Sprint que despliega el trabajo completado en cada sprint, a diferencia del Gráfico de burndown del sprint que presenta la cantidad de trabajo faltante.

Fuente: (SCRUMstudy, 2017)

FASE III. Ejecución

La fase de Ejecución se enfoca en el desarrollo de las tareas y actividades para crear el producto del proyecto. Dentro de sus procesos se encuentran: crear entregables, conducir reuniones diarias de Scrum y refinar el Product Backlog priorizado. En la siguiente figura se presentan los procesos que constituyen la fase de Ejecución:

FIGURA 19.
PROCESOS DE LA FASE DE EJECUCIÓN

Crear entregables
Entradas 1. Equipo de Scrum 2. Sprint Backlog 3. Kanban board 4. Bitácora de impedimentos
Herramientas y Técnicas 1. Habilidad del Equipo
Salidas 1. Entregables del Sprint 2. Kanban board actualizado 3. Bitácora de impedimentos actualizado

Conducir Reunión diaria de Scrum
Entradas 1. Equipo de desarrollo 2. Scrum Master 3. Gráfica de Burndown de Sprint 4. Bitácora de Impedimentos
Herramientas y Técnicas 1. Reunión diaria de Scrum 2. Tres preguntas diarias
Salidas 1. Gráfica de Burndown de Sprint 2. Bitácora de impedimentos actualizado

Refinar Product Backlog priorizado
Entradas 1. Equipo de Scrum 2. Product Backlog priorizado
Herramientas y Técnicas 1. Reunión de revisión del Product Backlog priorizado
Salidas 1. Product Backlog priorizado actualizado

Fuente: elaboración propia basándose de (SCRUMstudy, 2017)

12. Crear entregables

El equipo de desarrollo trabaja en las tareas desplegadas en el Sprint Backlog para generar los entregables; se hace uso del Kanban Board para dar seguimiento del trabajo y las actividades que están siendo ejecutadas. De haber impedimentos en el proceso los mismos son agregados en el log o bitácora de impedimentos.

Entradas
1- *Equipo de Scrum* 2- *Sprint Backlog* 3- *Kanban Board* 4- *Bitácora de impedimentos*
Herramientas y Técnicas
1- Habilidad del equipo Se refiere al conjunto de habilidades que posee los miembros del equipo de desarrollo para entender las Historias de Usuario y tareas en el Sprint Backlog con el fin de producir los entregables finales. Según su juicio y experiencia aplican los aspectos técnicos y de gestión de proyecto para elaborar los entregables.
Salidas
1- Entregables del Sprint Al finalizar cada Sprint se debe obtener como resultado un incremento o un entregable completado (debe contener todas las características y funcionalidades definidas en las historias de usuario incluidas en el Sprint y debió se probado exitosamente). *2- Kanban Board actualizado* El Kanban Board es actualizado regularmente por el equipo de desarrollo a medida que se completan las tareas. Al finalizarse el sprint el mismo es reiniciado para crear otro Kanban Board con el sprint siguiente.

3- *Bitácora de impedimentos actualizado*

Los obstáculos que reducen la productividad de equipo de desarrollo deben ser identificados, resueltos y removidos para que el equipo pueda trabajar de manera efectiva.

Los impedimentos pueden ser internos (flujo de proceso ineficiente, poca comunicación) o externos (problemas de licencia de software, documentación innecesaria de requerimientos); estos deben ser registrados formalmente por el Scrum Master en la bitácora de impedimentos y deben ser discutidos en la Reunión diaria de Scrum y Sprint Review **(Anexo D. Plantillas de la Metodología, G. Bitácora de Impedimentos).**

Fuente: (SCRUMstudy, 2017)

13. Conducir reunión diaria de Scrum

Todos los días se lleva a cabo una reunión altamente enfocada y con tiempo exacto (time-boxed), donde el equipo de desarrollo comunica a los demás miembros su progreso e impedimentos.

Entradas
1- *Equipo de desarrollo* 2- *Scrum Master* 3- *Gráfico de burndown del sprint* 4- *Bitácora de Impedimentos*
Herramientas y Técnicas
1- Reunión diaria de Scrum (Daily Standup Meeting) Reunión corta diaria donde el equipo de Scrum se reúne en un tiempo exacto de 15 minutos (time-boxed) para reportar el progreso en el sprint actual y planificar las actividades del día. Se espera que todos los miembros del equipo se encuentren, aunque la misma no es cancelada o retrasada de no atender uno (1) o dos (2) miembros del equipo. 2- *Tres* (3) *preguntas diarias* En el Reunión diaria de Scrum facilitado por el Scrum Master, los miembros del equipo de desarrollo responden tres (3) preguntas específicas: – ¿Qué he hecho desde la última reunión? – ¿Qué planeo hacer antes de la próxima reunión? – ¿Qué impedimentos estoy enfrentando actualmente? Se recomienda que las dos (2) primeras preguntas sean respondidas por el equipo de desarrollo de forma cuantificable, en vez de respuestas largas cualitativas. Los equipos pueden organizar reuniones adicionales después del Reunión diaria de Scrum para discutir temas adicionales.
Salidas
1- Gráfico de burndown del sprint actualizado **2-** Bitácora de impedimentos actualizado **(Anexo D. Plantillas de la Metodología, G. Bitácora de Impedimentos).**

Fuente: (SCRUMstudy, 2017)

14. Refinar Product Backlog priorizado

El Product Backlog priorizado es continuamente actualizado mediante la reunión de revisión del Product Backlog priorizado, donde cualquier cambio o actualización es discutido e incorporado según lo apropiado.

Entradas
1- *Equipo de Scrum* 2- *Product Backlog priorizado*
Herramientas y Técnicas
1- Reunión de revisión del Product backlog priorizado El Product Owner puede tener múltiples y separadas reuniones con involucrados relevantes, Scrum Master y el equipo de desarrollo, para sí asegurar tener la suficiente información para actualizar el Product Backlog priorizado. Esta reunión además de verificar que las Historias de usuario, los criterios de aceptación sean claros y entendibles para todos los miembros, igualmente se asegura que se remuevan las Historias de Usuario irrelevantes y se agreguen las solicitudes de cambios al Product Backlog priorizado.
Salidas
1- Product Backlog priorizado actualizado El Product Backlog puede ser actualizado con nuevas Historias de Usuario, solicitudes de cambio, identificación de nuevos riesgos, actualización de Historias de Usuario o re-priorización de Historias de usuario existentes **(Anexo D. Plantillas de la Metodología, E. Product Backlog priorizado).**

Fuente: (SCRUMstudy, 2017)

FASE IV. Revisión y Retrospectiva

La fase de Revisión y Retrospectiva se orienta a la revisión de los entregables elaborados, revisión del trabajo realizado y la identificación de las mejoras requeridas en la práctica.

FIGURA 20.
PROCESOS DE LA FASE DE EJECUCIÓN

Presentar y Validar Sprint	Retrospectiva del Sprint
Entradas 1. Equipo de Scrum 2. Entregables del Sprint 3. Sprint Backlog 4. Criterio de Terminado 5. Criterio de Aceptación de Historias de Usuario	**Entradas** 1. Equipo de desarrollo 2. Scrum Master 3. Resultados del proceso Presentar y Validar Sprint
Herramientas y Técnicas 1. Reunión de Sprint Review	**Herramientas y Técnicas** 1. Reunión de Retrospectiva del Sprint
Salidas 1. Entregables aceptados	**Salidas** 1. Acuerdo de mejoras 2. Póster de Reflexión

Fuente: elaboración propia basándose de (SCRUMstudy, 2017)

15. Presentar y validar Sprint

Proceso en el cual se presentan los entregables al Product Owner y a los involucrados relevantes en la reunión de Sprint Review; su propósito es asegurar la aprobación y aceptación del producto por parte del Product Owner.

Entradas
1- *Equipo de Scrum* 2- *Entregables del Sprint* 3- *Sprint Backlog* 4- *Criterio de Terminado* 5- *Criterio de Aceptación de Historias de Usuario*
Herramientas y Técnicas
1- *Reunión de Sprint Review* En esta reunión participa todo el equipo de Scrum y los involucrados relevantes del proyecto con el objetivo de discutir la aceptación o rechazo de los entregables en base a los criterios de aceptación de las Historias de Usuario elaborado anteriormente. Estas reuniones se desarrollan al final de cada Sprint, donde el equipo de Scrum presenta los resultados logrados en el Sprint (nuevas funcionalidades y productos creados); procediendo el Product Owner y los involucrados relevantes inspeccionar los resultados y determinar si es requerido algún cambio que debe ser considerado en los siguientes Sprints.
Salidas
1- Entregables aceptados Los entregables que cumplen con los criterios de aceptación son aprobados por el Product Owner y agregados a una lista de entregables aceptados, el cual es actualizado al final de cada Reunión de Sprint Review. Si un entregable no cumpliera con el criterio de aceptación, será continuado en los siguientes Sprint (situación poco deseable ya que el objetivo de cada Sprint es cumplir con los criterios de aceptación de los entregables) **(Anexo D. Plantillas de la Metodología, H. Registro de Entregables aceptados).**

Fuente: (SCRUMstudy, 2017)

16. Retrospectiva del Sprint

El proceso tiene como objetivo la discusión de las lecciones aprendidas durante la ejecución del Sprint, participan el Scrum Master y el equipo de desarrollo. La información recopilada es documentada como lecciones aprendidas que podrán ser consideradas en los Sprints siguientes; además en ocasiones los resultados son registrados como acuerdo de mejoras de acción o como actualización de recomendaciones de la guía de Scrum.

Entradas
1- *Equipo de desarrollo* 2- *Scrum Master* 3- *Resultados del proceso Presentar y Validar Sprint*
Herramientas y Técnicas
1- *Reunión de Retrospectiva del Sprint* Elemento final y fundamental de Scrum en relación con "inspeccionar y adaptar". La reunión es facilitada y moderada por el Scrum Master, todos los miembros del equipo de Scrum atienden a la reunión, se recomienda la presencia del Product Owner pero no es obligatorio. Durante la reunión, un miembro del equipo documenta la discusión y la propuesta de trabajo futuro; el ambiente debe ser abierto y relajado para que los miembros sientan confianza y tranquilidad en participar y discutir qué ocurrió correctamente y qué no. Los objetivos principales de la reunión son: • Elementos que el equipo debe seguir haciendo: buenas prácticas • Elementos que el equipo debe iniciar hacer: proceso de mejora • Elementos que el equipo debe dejar de hacer: problemas de procesos y cuellos de botella.

Salidas
1- Acuerdo de mejoras de acción Lista de elementos de acción que el equipo crea para abordar los problemas encontrados y así mejorar los procesos para aumentar el rendimiento de trabajo en los siguientes Sprints **(***Póster de Reflexión.* Póster que funciona como radiador de información, mostrando la información recopilada en el acuerdo de mejoras de acción: • ¿Qué debemos seguir haciendo? • ¿Qué debemos iniciar hacer? • ¿Qué debemos dejar de hacer? (Cockburn , 2004) **(Anexo D. Plantillas de la Metodología, I. Acuerdo de Mejoras, J. Póster de Reflexión).**

Fuente: (SCRUMstudy, 2017)

5.5 RESULTADOS Y DISCUSIONES

Se desarrolló la Metodología ágil para la Sección de Fábrica de Software (FS) de CIDITIC, basada en la fundamentación del marco teórico, los antecedentes de la FS recopilados y la evaluación de metodologías. La metodología ágil elaborada cuenta con un conjunto de elementos que permitirán al equipo desarrollar proyectos de Software de manera más controlada: principios, valores, roles del equipo, entradas, procesos, actividades y plantillas oficiales.

Por medio de la implementación de la metodología en la FS será posible que la Sección pueda gestionar proyectos de Software: identificando, refinando y controlando los requerimientos (alcance) considerando mayormente al cliente; estimando y controlando el tiempo y costo del proyecto; además, manteniendo un ambiente colaborativo entre todos los actores.

La hipótesis general es aceptada porque la metodología ágil de proyectos de Software propuesta desde un punto de vista conceptual es considerada funcional; debido a que está compuesta por características, técnicas y herramientas efectivas para la gestión adecuada del proyecto. Las metodologías base fueron seleccionadas por medio de una evaluación de metodologías completadas por el equipo de FS y docentes expertos de Ingeniería de Software.

Cabe resaltar que el nivel de funcionalidad de la metodología también dependerá de la experiencia y actitud del equipo en el uso de ésta; igualmente del entorno del proyecto con respecto al Cliente y la aceptación de la metodología por parte de los altos directivos de la institución.

A Continuación, se responden las preguntas de investigación planteadas en el Capítulo I Marco Conceptual en la página 8:

1. ¿Influye el uso de una metodología ágil de proyectos de Software en la Sección de Fábrica de Software en cuanto a la gestión de proyecto en los productos de Software desarrollado?

El marco teórico elaborado (Capítulo II, página 16) conceptualiza un conjunto de términos referentes al desarrollo de Software, enfoques de desarrollo y

metodologías tanto ágiles como tradicionales; y responde que el uso de una metodología ágil de proyectos de Software en la Sección de FS, sí influirá en la gestión de proyectos de los productos de Software desarrollados, porque:

- La metodología funcionará como una guía para conocer los procesos que se deban ejecutar: las entradas, las técnicas, salidas y las plantillas a seguir.
- La metodología cuenta con principios y propiedades a seguir que permiten al equipo de trabajo contar con la misma visión de trabajo.
- Cada integrante del equipo contará con un rol específico y tareas específicas, lo que permitirá que el mismo pueda desarrollarlas al 100%.
- La metodología promueve la colaboración constante en el equipo de trabajo y la comunicación frecuente con el cliente.

Es preciso mencionar que el impacto de una metodología en un proyecto dependerá del nivel de compromiso que tenga equipo de trabajo, el cliente y de la institución con respecto a la misma.

2. ¿Aumentará el nivel de satisfacción del cliente al aplicar una metodología de Proyectos de Software a la Sección de Fábrica de Software?

El nivel de satisfacción del cliente al aplicarse una metodología de proyectos de Software en la Sección de Fábrica de Software aumentará el nivel de satisfacción del cliente debido a que:

- Existe un rol llamado Product Owner dentro del equipo de trabajo que controla el listado de requerimientos y contempla que los requerimientos de los clientes e involucrados sean considerados y que los incrementos del producto cumplan con los criterios de aceptación de la empresa cliente.
- Los clientes e involucrados del proyecto son considerados en 7 significativos procesos de la metodología, donde el mismo puede dar retroalimentación para mejorar el desarrollo del producto de Software:
    1. Creación de la visión del proyecto.
    2. Desarrollo de las historias épicas.
    3. Creación del Product Backlog priorizado.
    4. Reunión de planificación de liberación.

5. Creación de historias de usuarios.
6. Refinamiento del Product Backlog priorizado.
7. Presentación y validación de Sprint (iteración).

Además, se diseñó una propuesta de evaluación de nivel de Satisfacción del Cliente para contar con retroalimentación durante el desarrollo de los proyectos y su seguimiento.

5.6 REFERENCIAS BIBLIOGRÁFICAS

Agile Business Consortium. (2014). agilebusiness.org. Obtenido de https://www.agilebusiness.org/content/principles

Google for Entrepreneurs. (2016). Rapid Prototyping 1 of 3: Sketching & Paper Prototyping. Obtenido de https://www.youtube.com/watch?v=JMjozqJS44M

LimeSurvey. (2018). limesurvey.org. Obtenido de https://www.limesurvey.org/

SCRUMstudy. (2017). Guide to the Scrum Body of Knowledge. Obtenido de https://www.scrumstudy.com/SBOK/SCRUMstudy-SBOK-Guide-3rd-edition.pdf

CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

CONCLUSIONES

Finalizado este trabajo de investigación, se presenta la propuesta de Metodología ágil CIDITIC-Scrum, para la gestión de proyectos de Software en la Sección de Fábrica de Software (FS) del Centro de Investigación CIDITIC. Esta propuesta consiste en una herramienta para la mejora en la elaboración y entrega de productos de Software con respecto a la planificación, seguimiento y control de alcance, tiempo, presupuesto y calidad del producto según el nivel de satisfacción del cliente.

Para apoyar lo anterior, se señalan las conclusiones más relevantes del estudio:

1. Por medio de un diagnóstico a la Sección de FS; se identificó la necesidad de contar con una metodología de proyectos de Software. Este diagnóstico contempló los proyectos desarrollados, los incidentes ocurridos y la clasificación de los incidentes por factores de fracaso como: métodos, tareas, personal y entorno.
2. Para la resolución del problema, se estableció una investigación tipo descriptiva para determinar y documentar el escenario actual en la Sección de FS; y cuasiexperimental pues, aunque no se contó con un grupo control y experimental, si se contó con un planteamiento de problema, variables, una posible solución y una hipótesis propuesta para ser comprobada.
3. Se llevó a cabo una fundamentación teórica que sirvió para establecer las bases de este trabajo de investigación, donde se seleccionaron 10 metodologías ágiles: Adaptive Software Development, Feature Driven Development, Extreme Programming, Scrum, Kanban, Scrumban, Crystal Transparente, Dynamic Systems Development Method, Lean Development, Agile Unified Process; para ser comparadas en nueve criterios: planificación, alcance, costo, tiempo, riesgos, pruebas, comunicación del cliente, comunicación en el equipo trabajo y documentación. Esto se realizó con el objetivo de conocer las características principales de cada una de las metodologías, sus similitudes y atributos principales.
4. Se diseñó un instrumento de medición para evaluar las 10 metodologías en los criterios de gestión de proyectos; los cuales fueron valoradas por el equipo de

FS y docentes expertos de Ingeniería de Software de la universidad; dando como resultado cinco (5) metodologías base: Scrum, DSDM (Dynamic Systems Development Method), Crystal Clear, ASD (Adaptive Software Development) y Kanban.

5. La evaluación tuvo como resultado un porcentaje total de 81%/100%, indicando que las metodologías de mayor ponderación son las más adecuadas en su criterio correspondiente. Esto quiere decir que las metodologías resultantes pueden ser calificadas como válidas y buenas para la propuesta Este porcentaje fue comparado con la escala del Sistema de Calificaciones de la UTP obteniendo un valor categorizado como bueno.
6. Se elaboró la propuesta CIDITIC-Scrum basada en la metodología ágil Scrum, integrando las demás metodologías a la misma, estableciendo: principios, valores (propiedades), roles de trabajo con sus características y funciones, fases del ciclo de vida del proyecto describiendo sus entradas, herramientas, técnicas, salidas e incluyendo las plantillas respectivas para el desarrollo de la metodología.
7. Se diseñaron dos (2) instrumentos de medición con el objetivo de evaluar la funcionalidad de la metodología y el nivel de satisfacción del cliente para comprobar la hipótesis general.
8. Se contempló la ejecución de la metodología CIDITIC-Scrum para comprobar su funcionamiento; pero por motivo de traslado del personal de la Sección de FS a otros departamentos de la universidad, la misma no pudo se probada.
9. Se estableció que la hipótesis general es aceptada, debido a que la metodología ágil propuesta para los proyectos de Software desde un punto conceptual es funcional; con base al Marco Teórico.

RECOMENDACIONES

Presentadas las conclusiones, se procede a realizar un número de recomendaciones con respecto a los resultados obtenidos en este trabajo de investigación:

1. La Universidad Tecnológica de Panamá a través de la Dirección General de Tecnología de la Información y Comunicaciones en conjunto con el Centro de Investigación CIDITIC y la Facultad de Ingeniería de Sistemas Computacionales, deben establecer una estrategia para capacitar e implementar el uso de metodologías ágiles dentro del campus universitario, para así estandarizar los procesos y artefactos generados dentro de los proyectos de Software.
2. Promover proyectos de Software en la Facultad de Ingeniería de Sistemas Computacionales (FISC) con estudiantes de pregrado, con el objetivo de capacitarlos en el uso de metodologías ágiles y motivar la aplicación de la metodología CIDITIC-Scrum.
3. Promocionar el uso de la metodología CIDITIC-Scrum para que la misma pueda ser probada y refinada. Ésta puede ser utilizada por otros equipos de trabajo dentro de la universidad además de la Sección de Fábrica de Software.
4. Para un desarrollo eficiente de proyectos de Software, los equipos de trabajo que apliquen la metodología propuesta deben enfocarse en un proyecto a la vez. Actualmente, la diversidad de proyectos no permite que el equipo se enfoque al 100%, causando retrasos en las entregas.
5. Procurar mantener integrantes del equipo proactivos y con compromiso en el proyecto, donde todos los miembros, sienten una responsabilidad equitativa de completar efectivamente el proyecto. Un miembro del equipo que no tenga la misma visión que los demás puede llegar a desmotivar y afectar negativamente los resultados.
6. El Cliente e involucrados, deben tener un nivel de compromiso con el proyecto: entregar la información requerida, asistir a las reuniones de avances y brindar retroalimentación; para que el proyecto cumpla con los objetivos establecidos y agregue valor a la empresa.

TRABAJOS FUTUROS

Se proponen los siguientes trabajos futuros:

1. Comprobación del funcionamiento y nivel de satisfacción del cliente de la metodología propuesta (CIDITIC-Scrum) con el nuevo equipo de FS u otros equipos de desarrollo. Esto incluiría la publicación de los resultados.
2. Creación de la guía en versión web que contenga la metodología estructurada con sus entradas, herramientas, técnicas, salidas y plantillas, para que sea más sencilla de consultar y permita buscar la información exacta para el equipo de trabajo.
3. Desarrollo e integración del ciclo de vida del Software que se centre en el proceso de diseño, desarrollo, pruebas, integración, implementación y mantenimiento del Software. Esto creará una metodología mucho más completa.

REFERENCIA BIBLIOGRÁFICA

ILX Group. (s.f.). *https://www.prince2.com*. Obtenido de https://www.prince2.com/usa/what-is-prince2#prince2-definition

University of Wisconsin. (15 de 06 de 2006). *Project Management Advisor*. Obtenido de https://pma.doit.wisc.edu/size_what.html

Zavala Ruiz , J. M. (2004). ¿Por Qué Fracasan los Proyectos de Software?; Un Enfoque Organizacional. *Congreso Nacional de Software Libre 2004*.

Agile Alliance. (2001). *www.agilealliance.org*. Obtenido de https://www.agilealliance.org/agile101/the-agile-manifesto/

Agile Alliance. (s.f.). *www.agilealliance.org.* Obtenido de https://www.agilealliance.org/glossary/kanban/#q=~(filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'kanban))~searchTerm~'~sort~false~sortDirection~'asc~page~1)

Agile Business Consortium. (2014). *agilebusiness.org.* Obtenido de https://www.agilebusiness.org/content/principles

Arias, F. G. (2012). Proyecto de Investigación - Introducción a la metodología científica. Caracas, Venezuela: Episteme.

Barzanallana, R. (30 de 12 de 2006). *www.um.es*. (Universidad de Murcia) Obtenido de http://www.um.es/docencia/barzana/IAGP/Iagp2.html

Berkes, O. (2016). *https://www.ca.com/.* Obtenido de https://www.ca.com/content/dam/ca/us/files/ebook/digitally-remastered-building-software-into-your-business-dna.pdf

Berkes, O. (25 de abril de 2017). *http://www.enfasys.net*. Obtenido de http://www.enfasys.net/2017/04/25/la-fabrica-de-software-moderno-es-la-clave-del-exito-empresarial/

Bloch, M., Blumberg, S., & Laartz, J. (2012). Delivering large-scale IT projects on time, on budget, and on value. *Mckinsey&Company*.

BTI360 (Dirección). (2011). *Kanban applied to Scrum* [Película]. Obtenido de https://www.youtube.com/watch?v=0EIMxyFw9T8

Bubernak, C., & Schweikert, M. (23 de marzo de 2012). *cs.colorado.edu.* Obtenido de https://www.cs.colorado.edu/~kena/classes/5828/s12/presentation-materials/schweikertmarcbubernakchris.pdf

Cendejas Valdéz, C. L. (Mayo de 2014). *eumed.net.* Obtenido de http://www.eumed.net/tesis-doctorales/2014/jlcv/

CIDITIC. (2017). *ciditic.utp.ac.pa*. Obtenido de http://www.ciditic.utp.ac.pa/

CIDITIC. (s.f.). *ciditic.utp.ac.pa*. Obtenido de ciditic.utp.ac.pa: http://www.ciditic.utp.ac.pa/

Cockburn , A. (2004). *Crystal Clear - A Human-Powered Methodology For Small Teams, including The Seven Properties of Effective Software Projects.*

Computer Asocciates International Inc. (2017). *https://www.ca.com/.* Obtenido de https://www.ca.com/us/company/newsroom/press-releases/2017/ca-technologies-launches-new-global-marketing-campaign-the-modern-software-factory.html

Court, R. (2006). *http://www.cimaglobal.com*. Obtenido de http://www.cimaglobal.com/Documents/ImportedDocuments/PRINCE2_P5_article.pdf

Craddock, A., Richards, K., Tudor, D., Roberts, B., & Godwin, J. (2012). *The DSDM Agile Project Framework for Scrum.* Obtenido de www.agilebusiness.org: https://www.agilebusiness.org/sites/default/files/the_dsdm_agile_project_framework_v1_11.pdf?token=yqzXtW1a1

Cusumano, M. A. (july de 1988). *dspace.mit.edu.* Obtenido de https://dspace.mit.edu/bitstream/handle/1721.1/2204/SWP-2036-19215282.pdf?sequence=1

Edeki, C. (septiembre de 2013). Agile Unified Process. *International Journal of Computer Science and Mobile Applications*, 13-17 . Obtenido de http://erytheia.nied.unicamp.br:8081/interhad/courses/roberto-pereira/ci163-projeto-de-software-ufpr-1/agenda/auppaper.pdf

Garriga, A. (s.f.). *https://pmi-bcn.org*. Obtenido de https://pmi-bcn.org/index.php/recursos/menu-articulos/733-articulo-cadena-critica-201701-cast

Geambasu, C., Jianu, I., Jianu, I., & Gavrila, A. (2011). Influence Factors for the choice of a Software Development Methodology. *Accounting and Management Information Systems*.

Google for Entrepreneurs. (2016). Rapid Prototyping 1 of 3: Sketching & Paper Prototyping. Obtenido de https://www.youtube.com/watch?v=JMjozqJS44M

Gordillo Polo, E. (21 de octubre de 2014). *inventtatte.com*. Obtenido de inventtatte: https://inventtatte.com/metodologia-tradicional-vs-agil/

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, M. d. (2010). *Metodología de la Investigación.* México D.F: McGRAW-HILL / INTERAMERICANA EDITORES, S.A. DE C.V.

Highsmith, J. A. (2002). Agile Software Development Ecosystems.

IEEE. (28 de September de 1990). *https://www.jyu.fi/en.* Obtenido de http://www.mit.jyu.fi/ope/kurssit/TIES462/Materiaalit/IEEE_SoftwareEngGlossary.pdf

IEEE. (01 de 02 de 2008). *disi.unal.edu.co.* Obtenido de http://disi.unal.edu.co/dacursci/sistemasycomputacion/docs/SystemsEng/ISO_IEC_12270_2008.pdf

ISTQB Certification. (s.f.). *istqbexamcertification*. Obtenido de istqbexamcertification: http://istqbexamcertification.com/istqb-certification-the-definitive-guide/

Joskowicz, J. (10 de febrero de 2008).

Laboratorio Nacional de Calidad de Software de INTECO. (marzo de 2009). *incibe.es.* Obtenido de https://www.incibe.es/file/N85W1ZWFHifRgUc_oY8_Xg

Lars, M. (2012). Survey Shows Why Projects Fail. *Gartner*. Obtenido de http://thisiswhatgoodlookslike.com/2012/06/10/gartner-survey-shows-why-projects-fail/

Lehtinen, T. O., Mäntylä, M. V., Vanhanen, J., Itkonen, J., & Lassenius, C. (2014). Perceived Causes of Software Project Failures - An Analysis of their relationships. *Information and Software Technology*, 3-9.

LimeSurvey. (2018). *limesurvey.org*. Obtenido de https://www.limesurvey.org/

mainss. (7 de febrero de 2017). *mainss.com*. Obtenido de https://mainss.com/las-4-metodologias-de-gestion-de-proyectos-mas-utilizadas/

Mohammed Ali Munassar, N., & Govardhan, A. (2010). A Comparison Between Five Models Of Software Engineering. *IJCSI International Journal of Computer Science Issues*.

Montero Román, J. J. (2013). *usmp.edu.pe*. Obtenido de http://www.usmp.edu.pe/fabrica/

Murillo Torrecilla, F. J. (2006). *uam.es.* Obtenido de https://www.uam.es/personal_pdi/stmaria/jmurillo/Met_Inves_Avan/Materiales/Apuntes%20Instrumentos.pdf

Nomura, L., de Mesquita Spínola, M., Tonini, A. C., & Hikage, O. (2007). A MODEL FOR DEFINING SOFTWARE FACTORY PROCESSES. *19th International Conference on Production Research.*

NORMA ISO 9001. (2008). Obtenido de http://sgc.itchihuahua.edu.mx/Calidad1/ISO9001-2008.PDF

Pahuja, S. (s.f.). *agilealliance.org*. Obtenido de https://www./what-is-scrumban/

Pichler, R. (marzo de 2017). *romanpichler.com*. Obtenido de https://www.romanpichler.com/tools/

PMC, & EXIN. (2015). *https://pmclatam.wordpress.com*. Obtenido de https://pmclatam.wordpress.com/2015/01/09/los7-de-prince2/

Poppendieck, M., & Poppendieck, T. (2003). Obtenido de Lean Software Development: An Agile Toolkit

Prasad Mahapatra, R. (2016). Obtenido de https://books.google.com/books?isbn=9382609687

Pressman, R. S. (2010). *Ingeniería del Software* (Séptima ed.). McGraw-Hill.

Punjab Technical University -PTU. (2006). *Sbsceducation.org.* Obtenido de http://sbsceducation.org/wp-content/uploads/2014/06/BCA-303.pdf

Robert, M., Canuel, V., Benazza, A., & Descossy , C. (29 de mayo de 2012). *https://blog.octo.com/*. Obtenido de https://blog.octo.com/en/toward-a-better-software-factory/

Rodríguez Morillo, R. A. (2013). *pmoinformatica.com*. Obtenido de http://www.pmoinformatica.com/p/plantillas-de-gerencia-de-proyectos.html

Schwaber, K., & Sutherland, J. (2017). *scrumguides.org.* Obtenido de http://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100

Scott W. Ambler + Associates. (2014). *ambysoft.com*. Obtenido de http://www.ambysoft.com/unifiedprocess/agileUP.html

SCRUMstudy. (2017). *Guide to the Scrum Body of Knowledge.* Obtenido de https://www.scrumstudy.com/SBOK/SCRUMstudy-SBOK-Guide-3rd-edition.pdf

SEGUE Technologies. (7 de 08 de 2015). *seguetech.com*. Obtenido de https://www.seguetech.com/benefits-adhering-software-development-methodology-concepts/

Sesento García, L. (Septiembre de 2008). *eumed.net Enciclopedia Virtual.* Obtenido de http://www.eumed.net/tesis-doctorales/2012/lsg/concepto_modelo.html

Sommerville, I. (2005). *Ingeniería del Software* (Séptima ed.). Pearson Addison Wesley.

Standish Group. (4 de octubre de 2015). *www.infoq.com*. Obtenido de https://www.infoq.com/articles/standish-chaos-2015

Stoica, M., Mircea, M., & Ghilic-Micu, B. (2013). Software Development: Agile vs. Traditional. *Informatica Economica*.

Taghavi Dilamani, M. (2014). http://ispe-usa.com. *International Conference on Advanced and Agile Manufacturing*.

Timo O.A. Lehtinen, Mika V. Mäntylä, Jari Vanhanen, Juha Itkonen, & Casper Lassenius. (2014). Perceived Causes of Software Project Failures – An Analysis of their Relationships. *elsevier*, 3-9.

Torres, Z. H., & Torres, H. M. (2014). *Administración de Proyectos.* México: GRUPO EDITORIAL PATRIA.

Trejos, G. (4 de octubre de 2014). Obtenido de https://prezi.com/g7oalhpq1wuo/importancia-del-software-en-las-empresas/

UJA. (30 de octubre de 2015). *ujaen.es*. Obtenido de http://www.ujaen.es/investiga/tics_tfg/estu_cuasi.html

Universidad Tecnológica de Panamá. (2015). *utp.ac.pa*. Obtenido de http://www.utp.ac.pa/documentos/2015/docx/Sistema-de-Calificacion-que-utiliza-la-UTP.docx

Universidad Tecnológica de Panamá. (2017). *http://congreso.utp.ac.pa.* Obtenido de http://congreso.utp.ac.pa/wp-content/uploads/2017/10/cuadernillo-de-congreso_FINAL_last-version.pdf

University of Warwick. (2009). *Feature Driven Development & Empirical Modelling.* Obtenido de https://warwick.ac.uk/fac/sci/dcs/research/em/publications/web-em/04/featurelist.pdf

Verma, I. (2014). *ijcscn.* Obtenido de http://www.ijcscn.com/Documents/Volumes/vol4issue3/ijcscn2014040305.pdf

Voigt, B. J. (2004). Dynamic System.

Wells, D. (8 de october de 2013). *extremeprogramming.org.* Obtenido de http://www.extremeprogramming.org/rules.html

Wrike. (2014). *https://www.wrike.com/*. Obtenido de https://www.wrike.com/project-management-guide/methodologies/

ANEXOS

ANEXO A. EVALUACIÓN DE METODOLOGÍAS POR MEDIO DE CRITERIOS DE GESTIÓN DE PROYECTOS

TABLA 28.
CRITERIOS DE LA EVALUACIÓN

No	CRITERIO	%
	PLANIFICACIÓN	
1	La planificación permite el establecimiento de requerimientos prioritarios enfocados a las necesidades del usuario, permitiendo una gestión efectiva del tiempo, costo, alcance y riesgos en el proyecto.	10%
2	El equipo de trabajo participa activamente de manera continua aportando sus experiencias durante la fase de planificación, fortaleciendo su compromiso en el proyecto.	10%
3	La documentación en la fase de planificación es clara y objetiva, llevando a que todos los involucrados (equipo de trabajo, clientes y usuarios) conozcan plenamente los objetivos y requerimientos a desarrollar del proyecto.	5%
4	Al finalizar cada iteración o proyecto se convocan/realizan reuniones de retroalimentación con el equipo de trabajo para discutir las actividades desarrolladas y las lecciones aprendidas.	5%
	GESTIÓN DE ALCANCE	
5	La priorización oportuna de los requerimientos permite enfocar los esfuerzos en aquellos primordiales para potenciar y maximizar el valor de negocio para el cliente.	9%
6	El refinamiento de los requerimientos (a través de acciones como la priorización, calendarización y diseño de prototipos) permite que estos sean claros, consistentes, realizables y cónsonos con las necesidades del cliente.	7%
7	Los clientes/usuarios participan activamente en las actividades de análisis y diseño proyecto proporcionando conocimiento, necesidades y retroalimentación para el desarrollo del producto.	9%
	GESTIÓN DE TIEMPO	
8	El equipo de trabajo estima el tiempo del proyecto a través de técnicas, experiencias, conocimientos, incrementando las posibilidades de éxito durante la ejecución del mismo.	9%
9	El equipo de trabajo determina en conjunto la cantidad de iteraciones necesarias para completar con éxito el proyecto, aumentando y fortaleciendo el nivel de responsabilidad individual de cada miembro.	7%
10	Los clientes/usuarios participan activamente dentro de las iteraciones desarrolladas, ofreciendo retroalimentación en los avances del proyecto.	9%
	GESTIÓN DE COSTO	
11	El equipo de trabajo estima el costo del proyecto a través de técnicas, conocimientos y experiencias, permitiéndole determinar si es recomendable iniciar o continuar el proyecto; y de llevarse a cabo el proyecto aplica técnicas que reduzcan o mantengan el costo.	10%
	GESTIÓN DE RIESGOS	
12	Para minimizar o evitar riesgos que dificulten la terminación del proyecto, el equipo de trabajo identifica los riesgos y elabora planes de contingencia en caso de que surjan los mismos.	5%
13	La elaboración oportuna de pruebas dentro de las iteraciones permite asegurar la calidad del producto por medio de la detección de fallas, prevención de errores y defectos futuros.	5%
	PREGUNTA GENERAL	
14	¿Cuál metodología o metodologías considera usted que sería recomendable para la Sección de Fábrica de Software?	0%

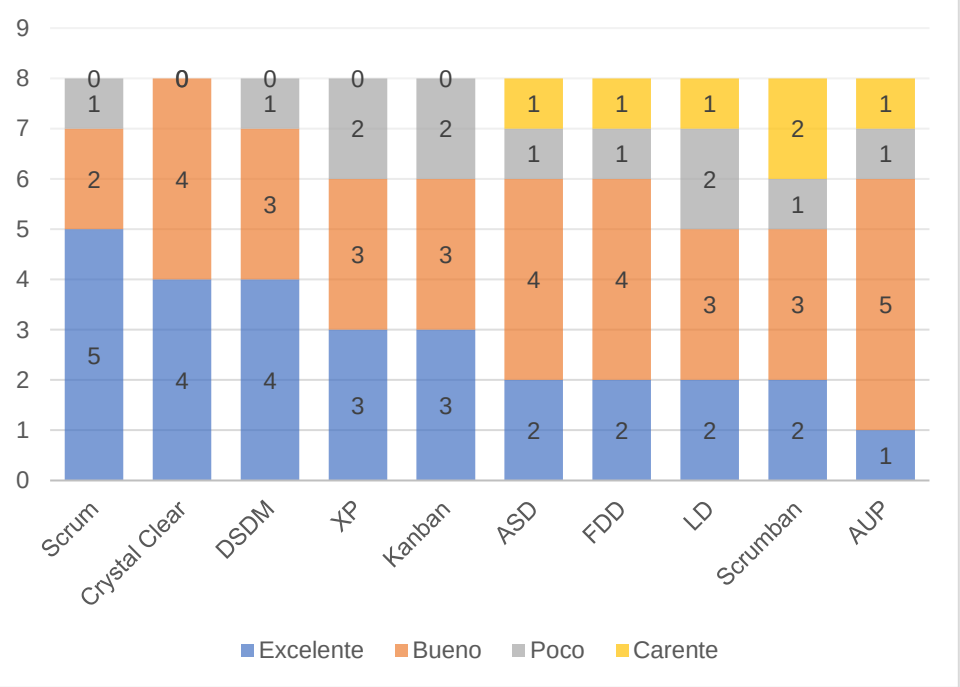
Fuente: elaboración propia

ANEXO B. RESULTADOS DE LAS EVALUACIONES DE METODOLOGÍAS POR PREGUNTA

A. Planificación

Pregunta 1. La planificación permite el establecimiento de requerimientos prioritarios enfocados a las necesidades del usuario, permitiendo una gestión efectiva del tiempo, costo, alcance y riesgos en el proyecto.

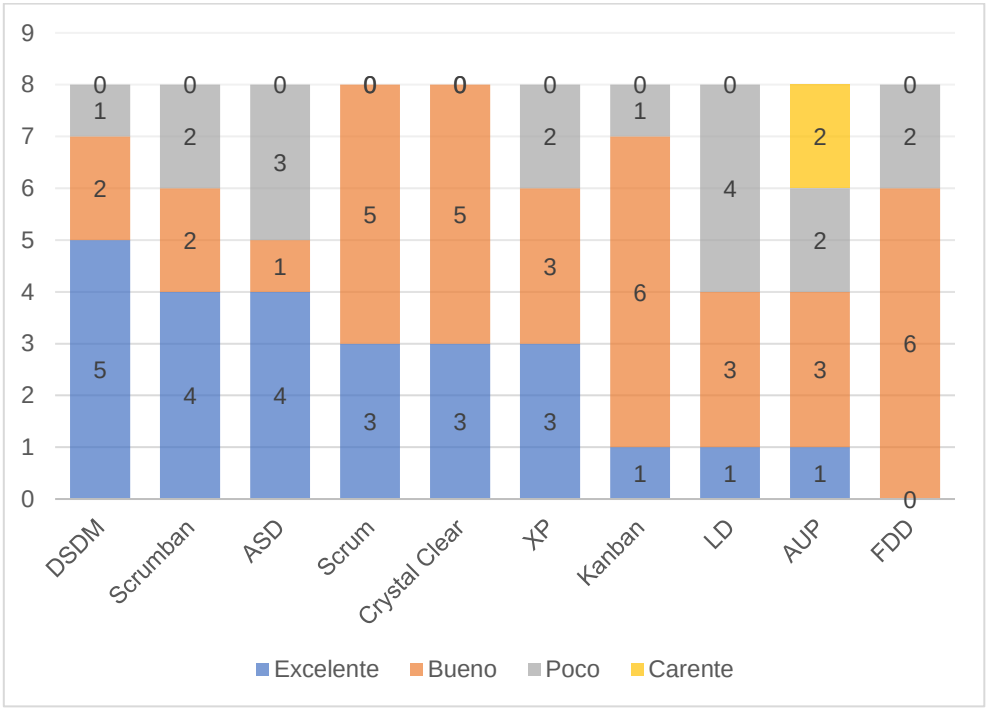
GRÁFICA 6
RESPUESTAS DE LA PREGUNTA 1



No.	Metodología	Excelente	Bueno	Poco	Carente
1	Scrum	5	2	1	0
2	Crystal Clear	4	4	0	0
3	DSDM	4	3	1	0
4	XP	3	3	2	0
5	Kanban	3	3	2	0
6	ASD	2	4	1	1
7	FDD	2	4	1	1
8	LD	2	3	2	1
9	Scrumban	2	3	1	2
10	AUP	1	5	1	1

Pregunta 2. El equipo de trabajo participa activamente de manera continua aportando sus experiencias durante la fase de planificación, fortaleciendo su compromiso en el proyecto.

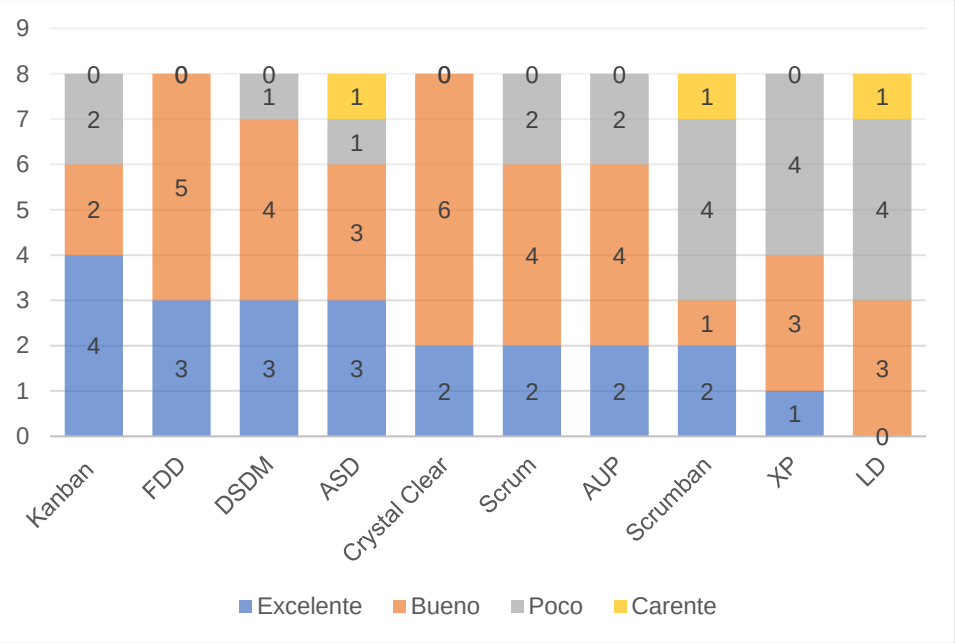
GRÁFICA 7.
RESPUESTAS DE LA PREGUNTA 2



No.	Metodología	Excelente	Bueno	Poco	Carente
1	DSDM	5	2	1	0
2	Scrumban	4	2	2	0
3	ASD	4	1	3	0
4	Scrum	3	5	0	0
5	Crystal Clear	3	5	0	0
6	XP	3	3	2	0
7	Kanban	1	6	1	0
8	LD	1	3	4	0
9	AUP	1	3	2	2
10	FDD	0	6	2	0

Pregunta 3. La documentación en la fase de planificación es clara y objetiva, llevando a que todos los involucrados (equipo de trabajo, clientes y usuarios) conozcan plenamente los objetivos y requerimientos a desarrollar del proyecto.

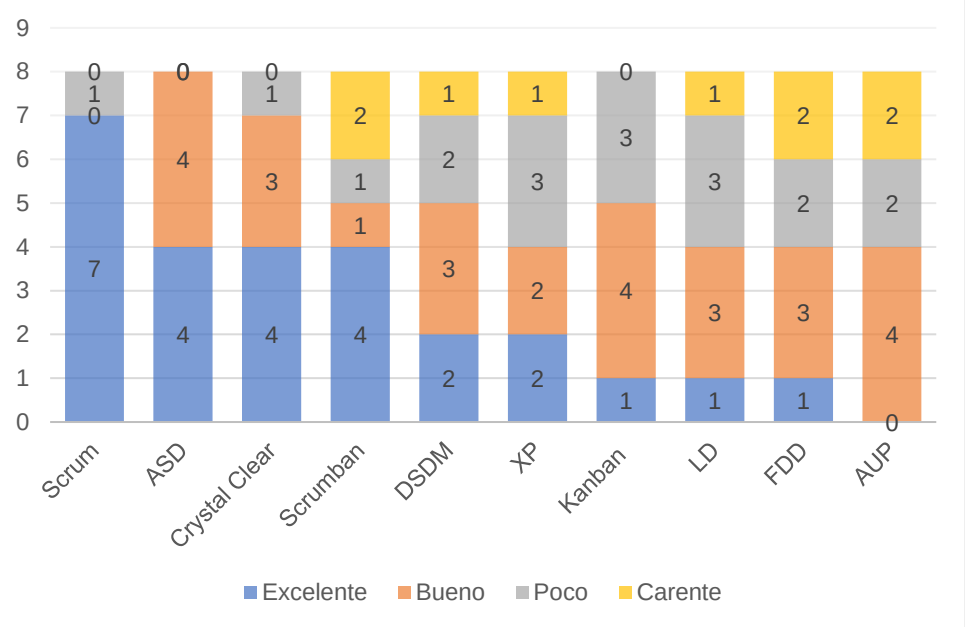
GRÁFICA 8.
RESPUESTAS DE LA PREGUNTA 3



No.	Metodología	Excelente	Bueno	Poco	Carente
1	Kanban	4	2	2	0
2	FDD	3	5	0	0
3	DSDM	3	4	1	0
4	ASD	3	3	1	1
5	Crystal Clear	2	6	0	0
6	Scrum	2	4	2	0
7	AUP	2	4	2	0
8	Scrumban	2	1	4	1
9	XP	1	3	4	0
10	LD	0	3	4	1

Pregunta 4. Al finalizar cada iteración o proyecto se convocan/realizan reuniones de retroalimentación con el equipo de trabajo para discutir las actividades desarrolladas y las lecciones aprendidas.

GRÁFICA 9
RESPUESTAS DE LA PREGUNTA 4

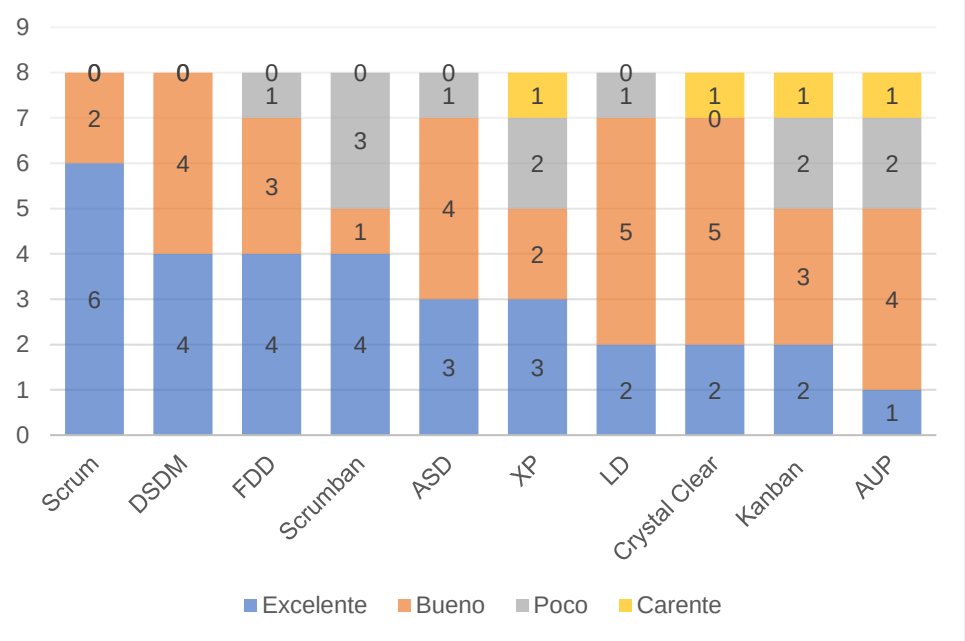


No.	Metodología	Excelente	Bueno	Poco	Carente
1	Scrum	7	0	1	0
2	ASD	4	4	0	0
3	Crystal Clear	4	3	1	0
4	Scrumban	4	1	1	2
5	DSDM	2	3	2	1
6	XP	2	2	3	1
7	Kanban	1	4	3	0
8	LD	1	3	3	1
9	FDD	1	3	2	2
10	AUP	0	4	2	2

B. Gestión de Alcance

Pregunta 5. La priorización oportuna de los requerimientos permite enfocar los esfuerzos en aquellos primordiales para potenciar y maximizar el valor de negocio para el cliente.

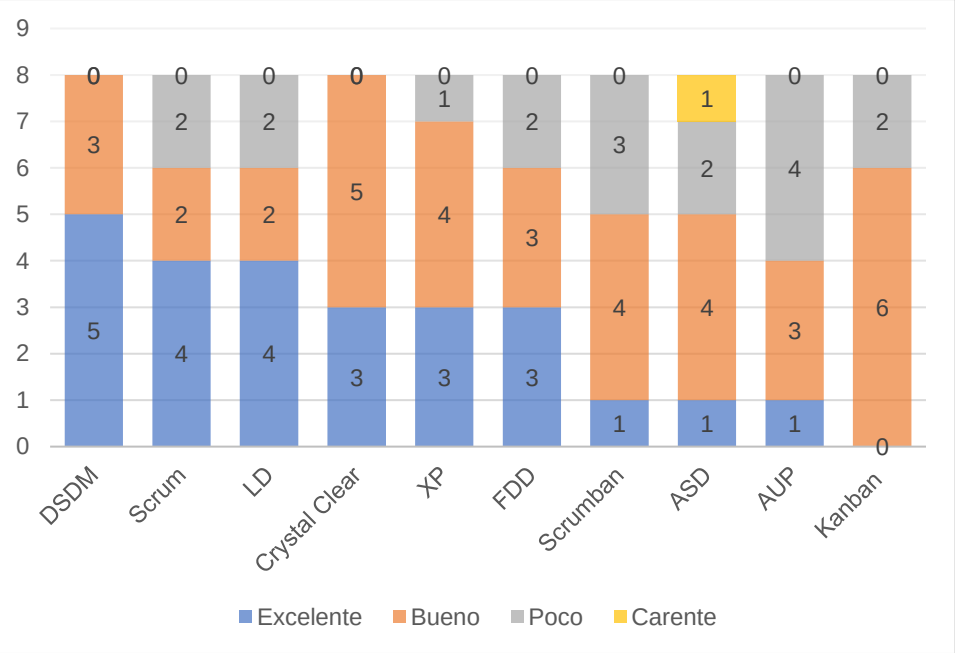
GRÁFICA 10
RESPUESTAS DE LA PREGUNTA 5



No.	Metodología	Excelente	Bueno	Poco	Carente
1	Scrum	6	2	0	0
2	DSDM	4	4	0	0
3	FDD	4	3	1	0
4	Scrumban	4	1	3	0
5	ASD	3	4	1	0
6	XP	3	2	2	1
7	LD	2	5	1	0
8	Crystal Clear	2	5	0	1
9	Kanban	2	3	2	1
10	AUP	1	4	2	1

Pregunta 6. El refinamiento de los requerimientos (a través de acciones como la priorización, calendarización y diseño de prototipos) permite que estos sean claros, consistentes, realizables y cónsonos con las necesidades del cliente.

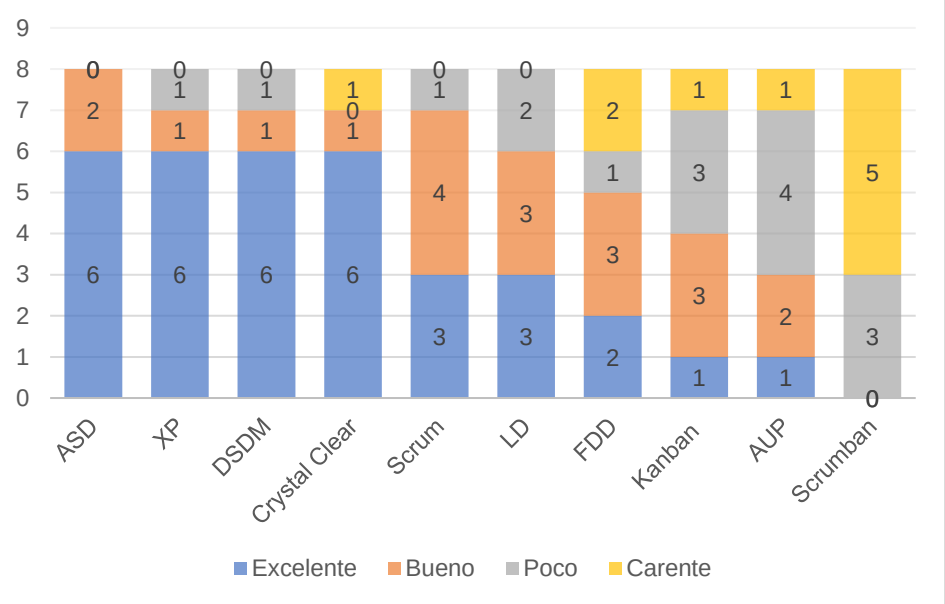
GRÁFICA 11.
RESPUESTAS DE LA PREGUNTA 6



No.	Metodología	Excelente	Bueno	Poco	Carente
1	DSDM	5	3	0	0
2	Scrum	4	2	2	0
3	LD	4	2	2	0
4	Crystal Clear	3	5	0	0
5	XP	3	4	1	0
6	FDD	3	3	2	0
7	Scrumban	1	4	3	0
8	ASD	1	4	2	1
9	AUP	1	3	4	0
10	Kanban	0	6	2	0

Pregunta 7. Los clientes/usuarios participan activamente en las actividades de análisis y diseño proyecto proporcionando conocimiento, necesidades y retroalimentación para el desarrollo del producto.

GRÁFICA 12
RESPUESTAS DE LA PREGUNTA 7

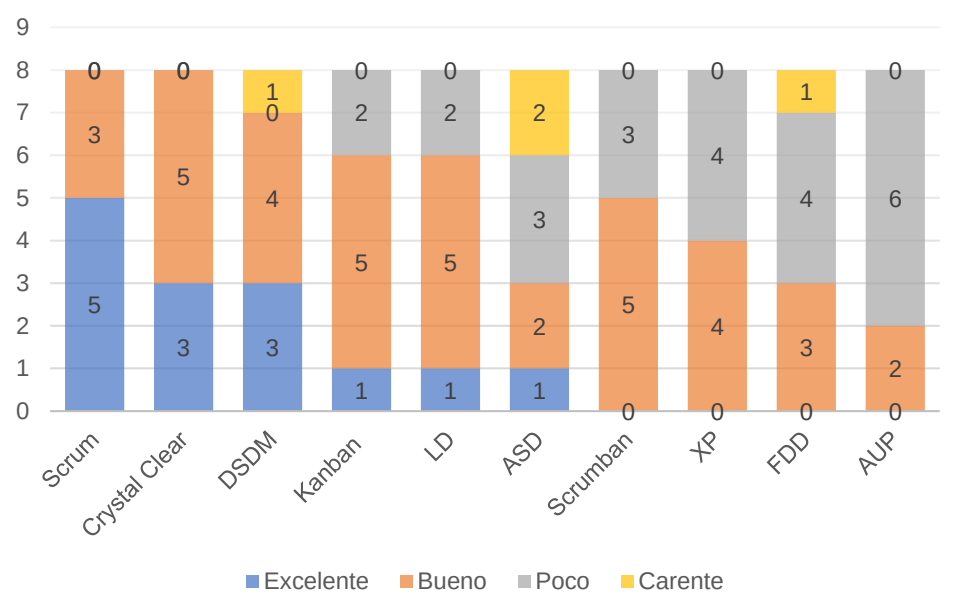


No.	Metodología	Excelente	Bueno	Poco	Carente
1	ASD	6	2	0	0
2	XP	6	1	1	0
3	DSDM	6	1	1	0
4	Crystal Clear	6	1	0	1
5	Scrum	3	4	1	0
6	LD	3	3	2	0
7	FDD	2	3	1	2
8	Kanban	1	3	3	1
9	AUP	1	2	4	1
10	Scrumban	0	0	3	5

C. Gestión de Tiempo

Pregunta 8. El equipo de trabajo estima el tiempo del proyecto a través de técnicas, experiencias, conocimientos, incrementando las posibilidades de éxito durante la ejecución del mismo.

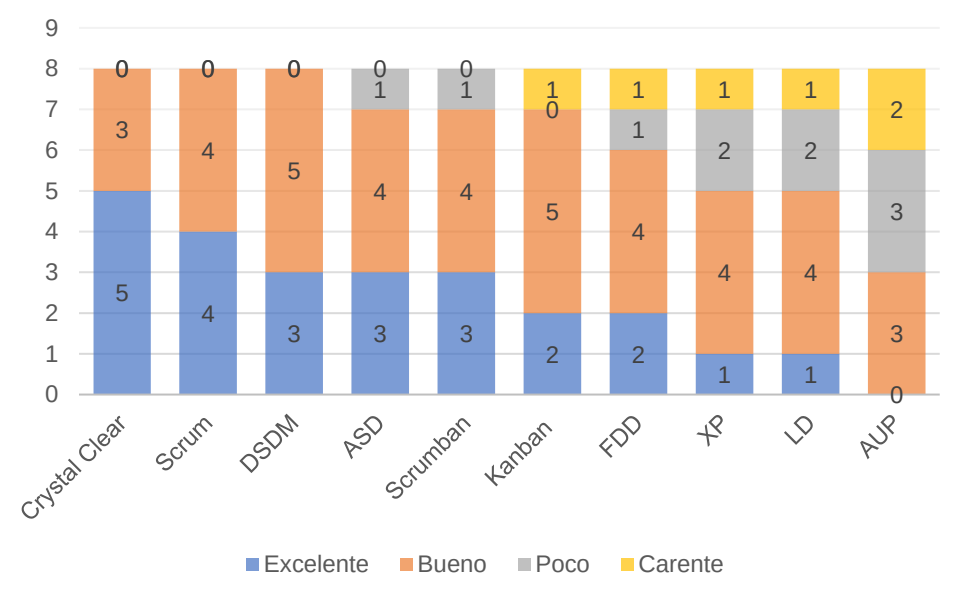
GRÁFICA 13
RESPUESTAS DE LA PREGUNTA 8



No.	Metodología	Excelente	Bueno	Poco	Carente
1	Scrum	5	3	0	0
2	Crystal Clear	3	5	0	0
3	DSDM	3	4	0	1
4	Kanban	1	5	2	0
5	LD	1	5	2	0
6	ASD	1	2	3	2
7	Scrumban	0	5	3	0
8	XP	0	4	4	0
9	FDD	0	3	4	1
10	AUP	0	2	6	0

Pregunta 9. El equipo de trabajo determina en conjunto la cantidad de iteraciones necesarias para completar con éxito el proyecto, aumentando y fortaleciendo el nivel de responsabilidad individual de cada miembro.

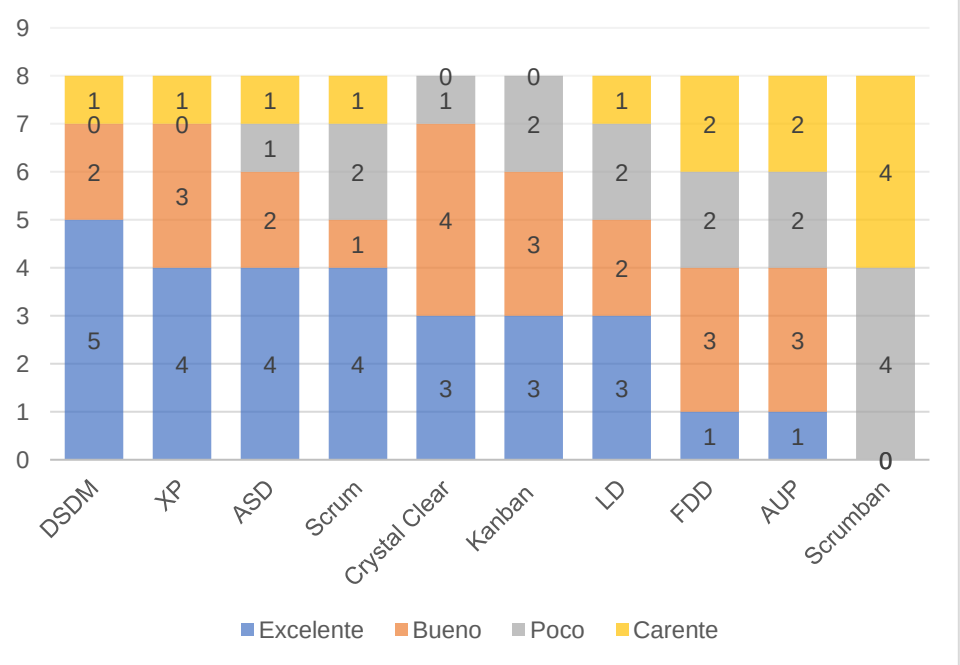
GRÁFICA 14
RESPUESTAS DE LA PREGUNTA 9



No.	Metodología	Excelente	Bueno	Poco	Carente
1	Crystal Clear	5	3	0	0
2	Scrum	4	4	0	0
3	DSDM	3	5	0	0
4	ASD	3	4	1	0
5	Scrumban	3	4	1	0
6	Kanban	2	5	0	1
7	FDD	2	4	1	1
8	XP	1	4	2	1
9	LD	1	4	2	1
10	AUP	0	3	3	2

Pregunta 10. Los clientes/usuarios participan activamente dentro de las iteraciones desarrolladas, ofreciendo retroalimentación en los avances del proyecto.

GRÁFICA 15
RESPUESTAS DE LA PREGUNTA 10

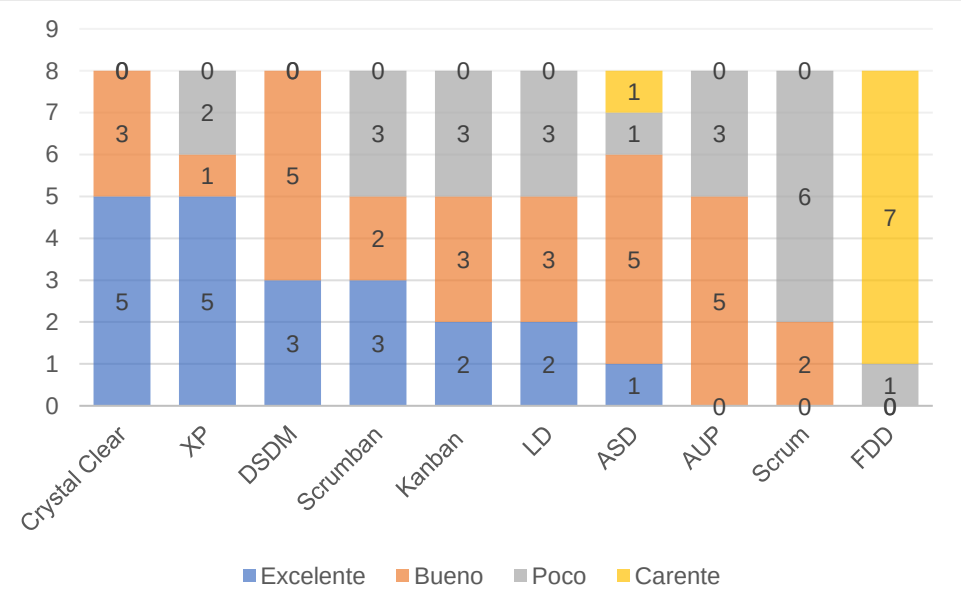


No.	Metodología	Excelente	Bueno	Poco	Carente
1	DSDM	5	2	0	1
2	XP	4	3	0	1
3	ASD	4	2	1	1
4	Scrum	4	1	2	1
5	Crystal Clear	3	4	1	0
6	Kanban	3	3	2	0
7	LD	3	2	2	1
8	FDD	1	3	2	2
9	AUP	1	3	2	2
10	Scrumban	0	0	4	4

D. Gestión de Costo

Pregunta 11. El equipo de trabajo estima el costo del proyecto a través de técnicas, conocimientos y experiencias, permitiéndole determinar si es recomendable iniciar o continuar el proyecto; y de llevarse a cabo el proyecto aplica técnicas que reduzcan o mantengan el costo.

GRÁFICA 16
RESPUESTAS DE LA PREGUNTA 11

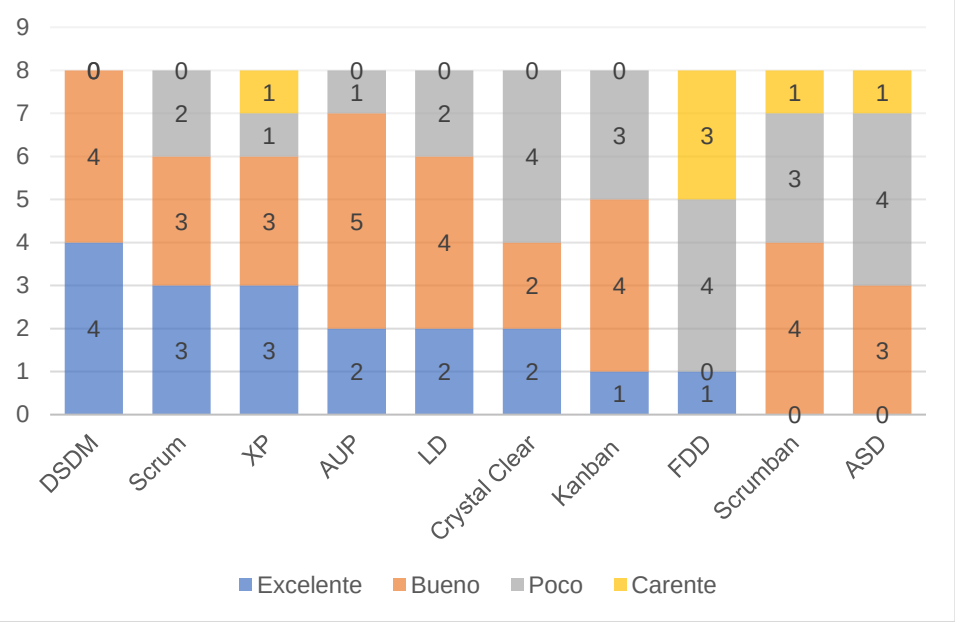


No.	Metodología	Excelente	Bueno	Poco	Carente
1	Crystal Clear	5	3	0	0
2	XP	5	1	2	0
3	DSDM	3	5	0	0
4	Scrumban	3	2	3	0
5	Kanban	2	3	3	0
6	LD	2	3	3	0
7	ASD	1	5	1	1
8	AUP	0	5	3	0
9	Scrum	0	2	6	0
10	FDD	0	0	1	7

E. Gestión de Riesgos

Pregunta 12. Para minimizar o evitar riesgos que dificulten la terminación del proyecto, el equipo de trabajo identifica los riesgos y elabora planes de contingencia en caso de que surjan los mismos.

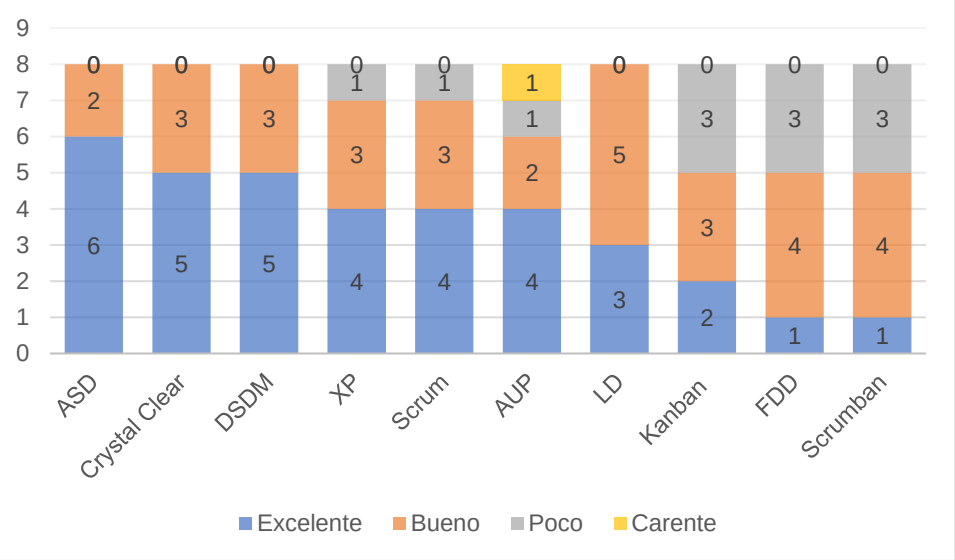
GRÁFICA 17
RESPUESTAS DE LA PREGUNTA 12



No.	Metodología	Excelente	Bueno	Poco	Carente
1	DSDM	4	4	0	0
2	Scrum	3	3	2	0
3	XP	3	3	1	1
4	AUP	2	5	1	0
5	LD	2	4	2	0
6	Crystal Clear	2	2	4	0
7	Kanban	1	4	3	0
8	FDD	1	0	4	3
9	Scrumban	0	4	3	1
10	ASD	0	3	4	1

Pregunta 13. La elaboración oportuna de pruebas dentro de las iteraciones permite asegurar la calidad del producto por medio de la detección de fallas, prevención de errores y defectos futuros.

GRÁFICA 18
RESPUESTAS DE LA PREGUNTA 13



No.	Metodología	Excelente	Bueno	Poco	Carente
1	ASD	6	2	0	0
2	Crystal Clear	5	3	0	0
3	DSDM	5	3	0	0
4	XP	4	3	1	0
5	Scrum	4	3	1	0
6	AUP	4	2	1	1
7	LD	3	5	0	0
8	Kanban	2	3	3	0
9	FDD	1	4	3	0
10	Scrumban	1	4	3	0

F. Pregunta General

Pregunta 14. ¿Cuál metodología o metodologías considera usted que sería recomendable para la Sección de Fábrica de Software?

RESPUESTAS
Consideraría optar no solo por una única metodología esto es debido a que la Sección de Fábrica de Software cuenta con algunas características que he podido divisar en las diferentes metodologías presente en la encuesta: - De momento no estamos involucrados en el desarrollo de estudios de factibilidad - En la mayoría de los proyectos excluimos actividades mantenimiento final - La interacción con el equipo al inicio y en cada incremento se lleva a cabo la retroalimentación con los usuarios - Se considera el Prototipado, Modelo de negocios General, modelo de datos, Lista de requerimientos, detalles del proyecto (Antecedentes, objetivos, reuniones, etc.) - Pruebas con el usuario a nivel modular - Se intenta involucrar a todo el equipo en la toma de decisiones. - Asignación de actividades según experiencia de los desarrolladores Las metodologías con características más parecidas a Fábrica de Software son: ASD, FDD y DSDM.
Scrum
En mi opinión la metodología que recomendaría para Fabrica de Software seria la Lean Development debido a que muchas veces una persona tiene más de 2 proyectos que son prioridad y luego al momento de hacer una distribución de tiempo pueden llegar a suceder situaciones que un proyecto que creías importante de repente hay un cambio de emergencia y entonces debes sacrificar tiempo de un proyecto para dedicárselo a otro y me gusta la idea de esta metodología que un desarrollador tenga enfocado de 1 a 2 proyectos máximos y

también que el cliente está involucrado junto al equipo de trabajo en cada momento.
Considero recomendable usar Scrum debido a la gran versatilidad en materia de procesos que se tendría en la sección de fábrica de software.
Es relativo a los recursos con que se cuentan, el tipo de proyecto, las herramientas a utilizar y el tiempo con que se cuenta.
Considero que un factor importante es la planificación del proyecto de software y específicamente la factibilidad de hacer el proyecto. Además, la definición de roles del equipo de trabajo y algo muy importante la gestión de proyectos. Veo la metodología de Dynamic Systems Development (DSDM) como la gran administradora de proyecto apoyada de Extreme Programming (XP) y Scrum para entregables específicos.
Para un mejor rendimiento dentro de la fábrica de software, las metodologías que yo recomendaría utilizar son: -Feature Driven Development -Crystal -Kanban -DSDM
Recomendaría el uso del Dynamic Systems Development Method (DSDM)

Fuente: elaboración propia

ANEXO C. RESULTADOS CON BASE A LA ESCALA PROPUESTA

Criterio	Metodología	Excelente	Bueno	Poco	Carente	Total
1	Scrum	10	2	0.5	0	12.5
1	Crystal Clear	8	4	0	0	12
1	DSDM	8	3	0.5	0	11.5
1	XP	6	3	1	0	10
1	Kanban	6	3	1	0	10
1	ASD	4	4	0.5	0	8.5
1	FDD	4	4	0.5	0	8.5
1	LD	4	3	1	0	8
1	Scrumban	4	3	0.5	0	7.5
1	AUP	2	5	0.5	0	7.5
2	DSDM	10	2	0.5	0	12.5
2	Scrumban	8	2	1	0	11
2	ASD	8	1	1.5	0	10.5
2	Scrum	6	5	0	0	11
2	Crystal Clear	6	5	0	0	11
2	XP	6	3	1	0	10
2	Kanban	2	6	0.5	0	8.5
2	LD	2	3	2	0	7
2	AUP	2	3	1	0	6
2	FDD	0	6	1	0	7
3	Kanban	8	2	1	0	11
3	FDD	6	5	0	0	11
3	DSDM	6	4	0.5	0	10.5
3	ASD	6	3	0.5	0	9.5
3	Crystal Clear	4	6	0	0	10
3	Scrum	4	4	1	0	9
3	AUP	4	4	1	0	9
3	Scrumban	4	1	2	0	7
3	XP	2	3	2	0	7
3	LD	0	3	2	0	5
4	Scrum	14	0	0.5	0	14.5
4	ASD	8	4	0	0	12
4	Crystal Clear	8	3	0.5	0	11.5
4	Scrumban	8	1	0.5	0	9.5
4	DSDM	4	3	1	0	8
4	XP	4	2	1.5	0	7.5
4	Kanban	2	4	1.5	0	7.5
4	LD	2	3	1.5	0	6.5

Criterio	Metodología	Excelente	Bueno	Poco	Carente	Total
4	FDD	2	3	1	0	6
4	AUP	0	4	1	0	5
5	Scrum	12	2	0	0	14
5	DSDM	8	4	0	0	12
5	FDD	8	3	0.5	0	11.5
5	Scrumban	8	1	1.5	0	10.5
5	ASD	6	4	0.5	0	10.5
5	XP	6	2	1	0	9
5	LD	4	5	0.5	0	9.5
5	Crystal Clear	4	5	0	0	9
5	Kanban	4	3	1	0	8
5	AUP	2	4	1	0	7
6	DSDM	10	3	0	0	13
6	Scrum	8	2	1	0	11
6	LD	8	2	1	0	11
6	Crystal Clear	6	5	0	0	11
6	XP	6	4	0.5	0	10.5
6	FDD	6	3	1	0	10
6	Scrumban	2	4	1.5	0	7.5
6	ASD	2	4	1	0	7
6	AUP	2	3	2	0	7
6	Kanban	0	6	1	0	7
7	ASD	12	2	0	0	14
7	XP	12	1	0.5	0	13.5
7	DSDM	12	1	0.5	0	13.5
7	Crystal Clear	12	1	0	0	13
7	Scrum	6	4	0.5	0	10.5
7	LD	6	3	1	0	10
7	FDD	4	3	0.5	0	7.5
7	Kanban	2	3	1.5	0	6.5
7	AUP	2	2	2	0	6
7	Scrumban	0	0	1.5	0	1.5
8	Scrum	10	3	0	0	13
8	Crystal Clear	6	5	0	0	11
8	DSDM	6	4	0	0	10
8	Kanban	2	5	1	0	8
8	LD	2	5	1	0	8
8	ASD	2	2	1.5	0	5.5
8	Scrumban	0	5	1.5	0	6.5
8	XP	0	4	2	0	6

Criterio	Metodología	Excelente	Bueno	Poco	Carente	Total
8	FDD	0	3	2	0	5
8	AUP	0	2	3	0	5
9	Crystal Clear	10	3	0	0	13
9	Scrum	8	4	0	0	12
9	DSDM	6	5	0	0	11
9	ASD	6	4	0.5	0	10.5
9	Scrumban	6	4	0.5	0	10.5
9	Kanban	4	5	0	0	9
9	FDD	4	4	0.5	0	8.5
9	XP	2	4	1	0	7
9	LD	2	4	1	0	7
9	AUP	0	3	1.5	0	4.5
10	DSDM	10	2	0	0	12
10	XP	8	3	0	0	11
10	ASD	8	2	0.5	0	10.5
10	Scrum	8	1	1	0	10
10	Crystal Clear	6	4	0.5	0	10.5
10	Kanban	6	3	1	0	10
10	LD	6	2	1	0	9
10	FDD	2	3	1	0	6
10	AUP	2	3	1	0	6
10	Scrumban	0	0	2	0	2
11	Crystal Clear	10	3	0	0	13
11	XP	10	1	1	0	12
11	DSDM	6	5	0	0	11
11	Scrumban	6	2	1.5	0	9.5
11	Kanban	4	3	1.5	0	8.5
11	LD	4	3	1.5	0	8.5
11	ASD	2	5	0.5	0	7.5
11	AUP	0	5	1.5	0	6.5
11	Scrum	0	2	3	0	5
11	FDD	0	0	0.5	0	0.5
12	DSDM	8	4	0	0	12
12	Scrum	6	3	1	0	10
12	XP	6	3	0.5	0	9.5
12	AUP	4	5	0.5	0	9.5
12	LD	4	4	1	0	9
12	Crystal Clear	4	2	2	0	8
12	Kanban	2	4	1.5	0	7.5
12	FDD	2	0	2	0	4

Criterio	Metodología	Excelente	Bueno	Poco	Carente	Total
12	Scrumban	0	4	1.5	0	5.5
12	ASD	0	3	2	0	5
13	ASD	12	2	0	0	14
13	Crystal Clear	10	3	0	0	13
13	DSDM	10	3	0	0	13
13	XP	8	3	0.5	0	11.5
13	Scrum	8	3	0.5	0	11.5
13	AUP	8	2	0.5	0	10.5
13	LD	6	5	0	0	11
13	Kanban	4	3	1.5	0	8.5
13	FDD	2	4	1.5	0	7.5
13	Scrumban	2	4	1.5	0	7.5

ANEXO D. PLANTILLAS DE LA METODOLOGÍA

A. Declaración de Visión de Proyecto

DECLARACIÓN DE VISIÓN DEL PROYECTO

Información General del Proyecto	
Nombre del Proyecto	*<Nombre del Proyecto>*
Nombre del Cliente	*<Nombre de empresa o persona solicitante>*
Nombre del Patrocinador	*<Nombre de organización o persona patrocinadora del proyecto>*
Interesados del Proyecto	*<Nombres de interesados o involucrados del proyecto>*
Fecha de inicio	*<DD/MM/YYYY>*
Fecha de cierre	*<DD/MM/YYYY>*

Visión del Producto	
Para	*<Cliente o usuarios meta>*
Quién	*<Necesidades y problemas>*
El	*<nombre del producto>*
Es un	*<Categoría del producto>*
Que	*<Beneficios del producto>*
No como	*<Competencia>*
Nuestro Producto	*<Propuesta de valor y alcance>*

Fuente: Elaboración propia basada de (Pichler, 2017)

B. Lista de Involucrados

LISTA GENERAL DE INVOLUCRADOS DEL PROYECTO

No	Título	Nombre del Involucrado	Cargo	Rol en el Proyecto
1	*<Lic., Ing. Tec….>*	*<nombre y apellido>*	*<cargo o puesto en la organización*	*<cliente, patrocinador, usuario>*
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				

Fuente: Elaboración propia basada del (SCRUMstudy, 2017)

C. Personas

PERSONAS

No	Foto	Nombre	Características	Meta
1	*<Foto del personaje ficticio>*	*<nombre del personaje ficticio>*	*<características del personaje> Actividades, comportamiento, actitud*	*<Necesidad o problema que será abordado, o el beneficio que se proveerá>*
2				
3				
4				
5				
6				
7				
8				
9				
10				

Fuente: Elaboración propia basado en (Pichler, 2017) y (Rodríguez Morillo, 2013)

D. Historias épicas

HISTORIAS ÉPICAS

No	Tema	Historias épicas	Prioridad	Riesgo	Dependencia
1	*<nombre de módulo, actividad o categoría del producto> Ejemplo: Lista de Carrito, Venta de artículos*	*<descripción de la historia épica>* *Yo como <rol/persona> quisiera poder <requerimientos> para así <beneficio>* *Ejemplo: Como <cliente> quiero tener <una lista de deseos> para comprar luego*	*<alto/medio/bajo>*	*<alto/medio/bajo>*	*<la historia épica depende de otra historia, colocar la historia a la que depende>*
2					
3					
4					
5					
6					
7					
8					
9					
10					

Fuente: Elaboración propia basada del (SCRUMstudy, 2017)

E. Product Backlog priorizado

PRODUCT BACKLOG PRIORIZADO

No	Tema	Historia épica	Historia de Usuario			Criterio de Aceptación	Prioridad	Estimación de esfuerzo	Miembro responsable
			Yo como	quisiera poder	para así				
1	*<nombre de módulo, actividad o categoría del producto> Ejemplo: Lista de Carrito, Venta de artículos*	*<descripción de historia épica>*	*<rol/persona>*	*<requerimientos>*	*<beneficio>*	*<criterios que debe cumplir para considerar que está terminado>*	*<alta/media/baja>*	*<estimación de la historia de usuario>*	*<miembro del equipo responsable de la Historia de Usuario>*
2									
3									
4									
5									
6									
7									
8									
9									
10									

Fuente: Elaboración propia basada del (SCRUMstudy, 2017)

F. Lista de Tareas

LISTA DE TAREAS

No	Historia de Usuario	Tarea	Story Points	Total de horas estimadas	Miembro responsable
1	*<Número o descripción de Historia de Usuario>*	*<tarea por realizar de la Historia de Usuario*	*<puntos de historia por esfuerzo y complejidad>*	*<número de horas requeridas al proyecto>*	*<Nombre del miembro de equipo de desarrollo responsable de la tarea>*
2					
3					
4					
5					
6					
7					
8					
9					
10					

Fuente: Elaboración propia basada del (SCRUMstudy, 2017)

G. Bitácora de impedimentos

BITÁCORA DE IMPEDIMENTOS

No	Fecha	Impedimento	Reportado por	Resuelto
1	*<dd/mm/yyyy>*	*<descripción del impedimento>*	*<nombre del miembro que reporta el impedimento*	*<si/no>*
2				
3				
4				
5				
6				
7				
8				
9				
10				

Fuente: Elaboración propia basada del (SCRUMstudy, 2017)

H. Registro de Entregables aceptados

LISTA DE ENTREGABLES

No	Fecha	Entregable	Aceptado	Comentarios
1	*<dd/mm/yyyy>*	*<nombre de entregable>*	*<si/no>*	*<comentarios de entregables*
2				
3				
4				
5				
6				
7				
8				
9				
10				

Fuente: elaboración propia basada en (Rodríguez Morillo, 2013)

I. Acuerdo de mejoras

ACUERDO DE MEJORAS DE ACCIÓN

No	Fecha	¿Qué debemos seguir haciendo?	¿Qué debemos iniciar hacer?	¿Qué debemos dejar de hacer?
1	*<dd/mm/yyyy>*	*<buenas prácticas>*	*<procesos de mejora>*	*<problemas de procesos y cuellos de botella*
2				
3				
4				
5				
6				
7				
8				
9				
10				

Fuente: Elaboración propia basada del (SCRUMstudy, 2017)

J. Póster de Reflexión

¿Qué debemos seguir haciendo? *<conserva esto>*	**¿Qué debemos iniciar hacer?** *<prueba esto>*
¿Qué debemos dejar de hacer? *<problemas>*	

(Cockburn , 2004)

ANEXO E. EVALUACIÓN DE PROPUESTA DE METODOLOGÍA

INSTRUMENTO DE EVALUACIÓN:
METODOLOGÍA ÁGIL PARA LA GESTIÓN DE PROYECTOS DE SOFTWARE

Objetivo: Registrar el seguimiento de los avances alcanzados y experiencias del equipo de trabajo utilizando la metodología propuesta durante el desarrollo de un proyecto de Software para medir el nivel de funcionalidad de la metodología.

A quién va dirigido: Instrumento dirigido al Scrum Master del proyecto o miembro encargado de gestionar o dar seguimiento a las fases, procesos y actividades a desarrollarse en el proyecto por desarrollarse o en desarrollo.

Materiales y recursos requeridos:

1-Caso de estudio.
2-Propuesta de Metodología del Trabajo de Investigación.
3-Lápiz o pluma de escribir.
4-Equipo computacional.
5-Equipo de Trabajo de Software (Mínimo 3).

Instrucciones: Desarrollar el caso de estudio siguiendo los procesos definidos en la metodología propuesta del trabajo de investigación. Durante la ejecución del proyecto, por iteración debe llenar la siguiente evaluación y marcar con una X la respuesta más cercana a su respuesta.

Escala de Evaluación	**4**	**3**	**2**	**1**
	Muy de acuerdo	De acuerdo	En desacuerdo	Muy desacuerdo

Fase en la que se encuentra			
Proceso en el que se encuentra			
# de iteración *(si aplica)*			
# de requerimientos hasta la fecha			
Costo presupuestado		**Costo actual**	

#	Rúbrica	Escala				Comentarios u observaciones
		1	2	3	4	
1	La descripción de los procesos en las fases (definición del proceso, entradas, técnicas y herramientas y salidas) son lo suficientemente claras y comprensibles para la ejecución de los procesos por los miembros del equipo.					
2	Las plantillas de las salidas de los procesos son comprensibles y fáciles de utilizar para los miembros del equipo.					

3	Los procesos de la metodología cumplen con las actividades suficientes para seguir avanzando en el proyecto.					
4	Por medio de la Metodología todos los miembros del equipo conocen sus funciones y responsabilidades a cumplir durante todo el proyecto.					
5	Las salidas de los procesos ofrecen la suficiente información para tomar decisiones en el proyecto por el equipo de trabajo y el cliente.					

Fuente: elaboración propia

ANEXO F. EVALUACIÓN DE NIVEL DE SATISFACCIÓN DEL CLIENTE

INSTRUMENTO DE EVALUACIÓN:
NIVEL DE SATISFACCIÓN DEL CLIENTE

Objetivo: Medir el nivel de satisfacción del cliente a final de cada iteración durante la ejecución de un proyecto de Software.

A quién va dirigido: Instrumento dirigido a Involucrados relevantes del proyecto que participan activamente brindando información al equipo de trabajo que desarrolla el caso de estudio o proyecto.

Materiales:
1-Lápiz o pluma de escribir.

Instrucciones: Según la experiencia semanal recibida durante la ejecución del proyecto, conteste marcando con una X la respuesta más acertada los siguientes criterios utilizando la escala sugerida a continuación.

La escala por utilizar es:

Escala de Evaluación	**4**	**3**	**2**	**1**
	Excelente	Bueno	Poco	Carente

Cantidad de reuniones hasta el momento	
Su cargo	

No.	**Rúbrica**	**Escala**			
		4	**3**	**2**	**1**
1	Fue considerado para la identificación, refinamiento de los problemas, necesidades y requerimientos identificados.				
2	Se comunicaron con usted presencialmente o por otros medios de comunicación para informarle los avances obtenidos hasta la fecha.				
3	Es visible que las retroalimentaciones que brindaron usted y los interesados en las sesiones de avances son consideradas posteriormente en el proyecto o si no, recibe la explicación u otra solución al respecto.				
4	Se siente satisfecho con la forma en que se está ejecutando el proyecto con respecto a los procesos, técnicas y herramientas.				
5	Usted se encuentra conforme con los resultados de avance del proyecto (salidas como: requerimientos, cronogramas, informes, incrementos del producto, etc.)				

Comentarios *u observaciones*	

Fuente: elaboración propia

ANEXO G. PUBLICACIÓN DE PÓSTER EN CONGRESO ESTEC-UTP 2017

Metodología de gestión para proyectos de Software en el Centro de Investigación, Desarrollo e Innovación en TICs de la Universidad Tecnológica de Panamá

Nichol Sánchez, Ramfis Miguelena, Kexy Rodríguez
Universidad Tecnológica de Panamá, Panamá, Panamá

INTRODUCCIÓN

La Universidad Tecnológica de Panamá – **UTP** posee hoy en día una diversa cantidad de Sistemas Informáticos que facilitan las actividades administrativas, financieras, operacionales y de investigación de la institución. El Centro de Investigación, Desarrollo e Innovación en Tecnologías de la Información y las Comunicaciones – **CIDITIC** cuenta con una Sección llamada Fábrica de Software – FS, encargada de desarrollar proyectos de Software pequeños y medianos para el área investigación y que en los últimos años ha sido utilizada también para desarrollar aplicaciones institucionales.
Debido a la gran demanda de sistemas informáticos existente en la UTP, se requiere un desarrollo eficiente de Software, donde sea posible asegurar una entrega en el tiempo determinado, el cumplimiento de los requisitos solicitados por el cliente, la disminución de sobrecostos y una mejor calidad.
Por esta razón nació la iniciativa de elaborar una metodología de gestión para proyectos de Software por medio de una evaluación y comparación de metodologías existentes tradicionales y ágiles; seleccionando aquellas metodologías que se adaptan a las necesidades actuales que posee la Sección de Fábrica de Software de CIDITIC de la UTP que permitan un análisis, diseño, desarrollo, implementación, control y seguimiento efectivo de los proyectos.

METODOLOGÍA DEL PROYECTO

Para la elaboración de este proyecto se establecieron cuatro importantes fases:

Figura No. 1: Las cuatro fases que conforman el proyecto

FASE 1. Situación actual de la Sección de Fábrica de Software. Se evalúa la situación actual de la Sección de Fábrica de Software; se identifican sus necesidades y oportunidades de mejora.

FASE 2. Desarrollo del Marco Teórico. Se realiza una revisión bibliográfica de modelos de procesos de software, metodologías, frameworks (marcos de trabajo) tradicionales y ágiles existentes de Software.

FASE 3. Diseño de la metodología de gestión de proyectos de Software. Se propone la metodología según la revisión bibliográfica realizada y los requerimientos de la Sección de Fábrica de Software de CIDITIC.

FASE 4. Aplicación de la metodología propuesta a un caso de estudio real. Se establece un caso de estudio real; donde el mismo será aplicado a 2 equipos de trabajo: uno capacitado para utilizar la metodología y el otro utilizando la metodología actual de la Fábrica de Software para desarrollar el caso. Finalizado el caso de estudio, se hará una comparación de los resultados y se medirá la efectividad de la metodología propuesta.

Literatura citada

Timo O.A, Lehtinen, Mika V. Mäntylä, Jari Vanhanen, Juha Itkonen, & Casper Lassenius. (2014). Perceived Causes of Software Project Failures – An Analysis of their Relationships. elsevier, 3-9.

Pressman, R. S. (2010). Ingeniería del Software (Séptima ed.). McGraw-Hill.

CIDITIC. 2017. cidític.utp.ac.pa. Obtenido de http://www.cidític.utp.ac.pa/

AVANCES DEL PROYECTO

I. Solicitudes en la Sección de Fábrica de Software

La Sección de Fábrica de Software actualmente trabaja en 4 tipos de solicitudes: Páginas web informativas (Congresos, foros, jornadas de investigación, páginas informativas en general), aplicaciones web (Gestión de inventario, incidentes, automatización de flujos de trabajo, sistemas de consulta, entre otros); implementación de plataformas de recepción de artículos y formularios web de inscripciones para eventos variados. La figura No.2 muestra el porcentaje de proyectos realizados por solicitudes 2012-2017, donde se puede resaltar que los porcentajes más altos son Páginas y Aplicaciones Web.

Figura No. 2: Porcentaje de proyectos por solicitudes en la Sección de Fábrica de Software
2012 - 2017

II. Incidencias ocurridas en la Sección de Fábrica de Software

Figura No. 3: Porcentaje de fallo en proyectos en la Sección de Fábrica de Software
2012 - 2017

Para identificar cuál es la causa principal por la que los proyectos no fueron finalizados según lo planeado, se seleccionaron las 11 aplicaciones web realizadas en la Sección de Fábrica de Software y se categorizaron según las incidencias ocurridas en cada desarrollo, utilizando las causas de fallos determinados por Timo Lehtinen de la Universidad Aalto de Finlandia. Las categorías que el autor indica son: el entorno, personal, métodos y tareas. Por medio de estas categorías se clasificaron las aplicaciones web según las incidencias ocurridas en cada una de ellas. Como resultado se obtuvo un porcentaje mayor en la categoría de Métodos y Tareas; estas dos categorías se centran en el uso de las actividades requeridas en las fases y en la ejecución correcta de las actividades.

III. Metodología para la Gestión del Proyecto

De acuerdo a las necesidades de la Sección de Fábrica de Software y el porcentaje resultante de fallo en la categoría de métodos y tareas, se tomó la decisión de desarrollar una metodología que se adapte a los requerimientos de la Fábrica de Software que permita gestionar los proyectos de Software de manera efectiva. Para seleccionar cuál es la metodología más conveniente para la sección, se realizó una comparación entre el enfoque tradicional y ágil; para así direccionar la investigación a un enfoque en particular. En base a la evaluación se realizará una comparación entre las metodologías del enfoque seleccionado.

CONCLUSIONES

Implementar una metodología para proyectos para la Sección de Fábrica de Software de CIDITIC, permitirá elaborar Software de mayor calidad, en un tiempo y presupuesto más razonable. Además, facilitará el seguimiento de las actividades e informes de avances para la dirección, cliente y usuarios.

Iimplementar una metodología en la Sección ayudará a mejorar la imagen del Centro de Investigación CIDITIC ya que será posible desarrollar Software aplicando buenas prácticas a los procesos y así, poder resolver una mayor cantidad de necesidades en el área de investigación, brindando apoyo mediante el desarrollo de aplicaciones web, páginas web y aplicaciones móviles.

Adicionalmente, contar con una metodología con su respectiva documentación permitirá que a futuro la misma pueda ser actualizada y mejorada ajustándose mayormente a las necesidades que se presenten a la Sección de Fábrica de Software más adelante.

www.cidític.utp.ac.pa
info.cidític@utp.ac.pa

Octubre 2017

Fuente: elaboración propia, Póster publicado en memoria del congreso (Universidad Tecnológica de Panamá, 2017).

Este documento ha sido analizado con la herramienta de análisis de plagio:
https://www.plagscan.com/

Resultando un Grado de Similitud de un 15%