INFOTEC CENTRO DE INVESTIGACIÓN E INNOVACIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

DIRECCIÓN ADJUNTA DE INNOVACIÓN Y CONOCIMIENTO
GERENCIA DE CAPITAL HUMANO
POSGRADOS

"*FAF*: MARCO DE TRABAJO ÁGIL Y FORMAL, PARA EL DESARROLLO DE *SOFTWARE* EN EL COLEGIO DE MÉXICO"

PROPUESTA DE INTERVENCIÓN
Que para obtener el grado de MAESTRO EN GESTIÓN DE INNOVACIÓN DE LAS TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

Presenta:

Victor Rojas Cortés

Asesor:

Maestro Felipe Alfonso Delgado Castillo

Ciudad de México, agosto de 2020.

Autorización de Impresión

AUTORIZACIÓN DE IMPRESIÓN Y NO ADEUDO EN BIBLIOTECA

MAESTRÍA EN GESTIÓN DE INNOVACIÓN DE LAS TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

Ciudad de México, 27 de noviembre de 2020.
INFOTEC-DAIC-GCH-SE-0628/2020.

La Gerencia de Capital Humano / Gerencia de Investigación hacen constar que el trabajo de titulación intitulado

FAF: MARCO DE TRABAJO ÁGIL Y FORMAL, PARA EL DESARROLLO DE SOFTWARE EN EL COLEGIO DE MÉXICO

Desarrollado por el alumno **Victor Rojas Cortés** y bajo la asesoría del **Mtro. Felipe Alfonso Delgado Castillo**; cumple con el formato de biblioteca. Por lo cual, se expide la presente autorización para impresión del proyecto terminal al que se ha hecho mención.

Asimismo se hace constar que no debe material de la biblioteca de INFOTEC.

Vo. Bo.

Mtra. Julieta Alcibar Hermosillo
Coordinadora de Biblioteca

Anexar a la presente autorización al inicio de la versión impresa del trabajo referido que ampara la misma.

C.p.p Servicios Escolares

Agradecimientos

Hoy me siento satisfecho por un logró más en mi vida profesional, me queda claro que todo es cuestión de fe, voluntad, esfuerzo y disciplina. Quiero agradecer a mi familia, amigos, compañeros de clase y maestros, por el apoyo brindado durante este período.

A mi esposa Claudia, por su apoyo incondicional, motivación y amor. Gracias por el tiempo que me diste y por cuidar a mis hijos, TE AMO.

A mis hijos Karen, Eric y Adrián por ser mi mayor motivación, esperando ser un ejemplo en su trayectoria académica.

A mi padres, Rafaela y Olegario, porque formaron mis cimientos con base en humildad, sencillez y responsabilidad.

A mis herman@s, Rosalba, Yanet, Arnulfo, José Luís y Leticia, por apoyarme siempre.

Al Colegio de México, por brindarme el apoyo para lograr esta meta.

A mis amigos del COLMEX, Gaby, Vic Dueñas, Frank, Jeri y Canek, por la motivación y consejos.

A mis compañeros de clase por compartir sus opiniones, ideas, consejos. Aprendí mucho de ustedes.

A mi asesor de tesis, Maestro Felipe Delgado, por su valiosa aportación en mi trabajo.

A la Dra. Rebeca Barriga, por su apoyo y consejos en la revisión de mi trabajo.

Gracias a dios.

No cabe duda, este trabajo se lo dedico a mi esposa Claudia y a mis hijos Karen, Eric y Adrián, LOS AMO.

Recuerda…
Siempre hay algo nuevo que aprender

Tabla de contenido

Introducción *1*

Capítulo 1. Metodologías de desarrollo de Software *4*

1.1 Conceptos de Metodología, Norma, Modelo, *Framework* y *BOK* **4**

1.2 Metodologías tradicionales **6**

1.2.1 Modelo de Cascada 6

1.2.2 Modelo de Prototipos 7

1.2.3 Modelo Incremental 7

1.2.4 Modelo Espiral 8

1.3 Metodologías ágiles **9**

1.3.1 *Extreme Programming* (*XP*) 10

1.3.2 *Lean Startup* 10

1.3.3 *Kanban* 11

1.3.4 *Scrumban* 12

1.3.5 *SCRUM* 13

1.4 Comparación entre metodologías tradicionales y ágiles **13**

Capítulo 2. SCRUM y PMBOK: desarrollo ágil y gestión de proyectos *16*

2.1 *Framework* de Iacovelli **16**

2.2 Aplicación del *Framework* de Iacovelli **18**

2.3 *SCRUM* **23**

2.4 *PMBOK* **28**

2.5 Comparativa entre *SCRUM* y *PMBOK* **35**

Capítulo 3. Propuesta de marco de trabajo FAF *41*

3.1 Planteamiento del problema **41**

3.2 Objetivo **42**

3.3 Definición del marco de trabajo *FAF* **43**

3.4 Principios **43**

3.5 Roles **44**

3.6 Reuniones **47**

3.7 Entregables **48**

3.8 Diagrama de proceso **48**

3.9 Calidad en *FAF* **57**

3.10 Recomendaciones para implementar *FAF* **58**

3.11 Beneficios esperados **59**

Conclusiones *61*

Bibliografía *64*

Anexos *67*

Anexo I – Formulario de Iacovelli **67**

Anexo II - Project Charter **70**

Anexo III - *Backlog* **72**

Anexo IV – Registro de Impedimentos **72**

Anexo V – Minuta de Reunión **72**

Anexo VI – Acta de cierre de proyecto **73**

Índice de figuras

Figura 1. Modelo de cascada..... 6
Figura 2. Modelo de Prototipos..... 7
Figura 3. Modelo incremental..... 8
Figura 4. Modelo de Espiral..... 9
Figura 5. Cuatro vistas del Framework de Iacovelli 16
Figura 6. Proceso general de SCRUM 25
Figura 7. Grupo de procesos que interactúan en un proyecto 30
Figura 8. Evolución del modelo de cascada hacia SCRUM..... 40
Figura 9. Flujo de comunicación entre todos los roles participantes en FAF. 46
Figura 10. Proceso general de FAF..... 51
Figura 11. Tablero Scrumboard 54

Índice de gráficos

Gráfico 1. Métodos y prácticas ágiles en el reporte del estado ágil no 13..... 23
Gráfico 2. Metodologías de Proyectos a nivel global..... 28

Índice de cuadros

Cuadro 1. Comparación teórica de metodologías tradicionales y ágiles..... 13
Cuadro 2. Resultados de los proyectos de software de acuerdo a CHAOS .. 14
Cuadro 3. Clasificación de procesos ágiles de Iacovelli 19
Cuadro 4. Resultados del COLMEX para la elección de un proceso ágil..... 21
Cuadro 5. Lista de entregables en PMBOK..... 31
Cuadro 6. Relación de procesos y áreas de conocimiento, PMBOK versión 6 33
Cuadro 7. Resultados del cuestionario a especialistas en desarrollo de software..... 36
Cuadro 8. Mapeo de Roles de MSF y SCRUM 37
Cuadro 9. Mapeo de fases y actividades de MSF y SCRUM 37
Cuadro 10. Características deseables para los roles principales de SCRUM .. 45
Cuadro 11. Procesos de FAF 52
Cuadro 12. Herramientas de apoyo para los procesos de FAF 56

Siglas y abreviaturas

COLMEX: El Colegio de México

DEM: Diccionario Español Mexicano

PMI: *Project Manager Institute*

BOK*: Body of Knowledge*

PMBOK: *Project Management Body of Knowledge*

XP: *Extreme programming*

MSF: *Microsoft Solutions Framework*

FAF: Marco de trabajo Ágil y Formal

Introducción

La ausencia de metodologías, definición y gestión de procesos en las organizaciones, aumenta la probabilidad de que surjan problemas, que se traducen en gastos innecesarios y falta de eficiencia. Esto afecta de forma negativa a cualquier organización en el cumplimiento de sus objetivos. Por lo tanto, es fundamental que exista una guía o marco de trabajo que integre y gestione el desarrollo de sistemas, estructura organizacional y personas, lo cual va a facilitar que se trabaje en equipo y organizados bajo un mismo objetivo.

Los resultados efectivos en una organización se deben a la adaptación de los cambios tecnológicos, procesos sistemáticos y a la entrega de valor que se les da a los clientes (Irino, 2017)[1]. En el caso de las instituciones educativas, la incorporación de las tecnologías de la información y la comunicación (TICS) en los desarrollos educativos, implica considerarlas tanto en la definición del currículo, como en el diseño y la implementación de estrategias pedagógicas y recursos didácticos, que apoyen el desarrollo de nuevos aprendizajes, competencias y relaciones con el conocimiento (Santiago Benítez, Caballero Álvarez, Gómez Mayén, & Cuevas, 2013).

En 2017, en El Colegio de México (COLMEX) en el *Tercer Encuentro de Humanidades Digitales* se destacó la importancia, el futuro y la creciente utilidad de los recursos ofrecidos por las TICS para el desarrollo de las ciencias sociales.[2] Además, a fin de facilitar la investigación académica, difusión de publicaciones como artículos, revistas y libros, cursos por medio de plataformas tecnológicas, gestión de aspirantes, control escolar y sistemas administrativos, es fundamental contar con procesos automatizados para garantizar calidad en los resultados.

Dentro de estos procesos está el desarrollo de *software*, según "*The Standish Group*" (2015), el 19% de estos proyectos no se terminan, el 45% no cumple los

---

[1] "La transformación digital avanza a tal velocidad que las empresas deben tener la capacidad de modificar sus modelos de negocio para asegurar una mayor entrega de valor al cliente en menor tiempo". Forbes. Consultado el 25 de julio de 2019 en https://www.forbes.com.mx/empresas-agiles-resultados-rentables/

[2] Informe de actividades de El Colegio de México, Consultado el 2 de diciembre de 2018 en https://www.colmex.mx/assets/pdfs/Informe_Actividades_2018_107_108.pdf?1562886800

requisitos y sólo el 36% tiene éxito, es decir, más del 60% no se concluyen de manera correcta. En este sentido, es fundamental adoptar marcos de trabajo flexibles, que faciliten los entregables de los proyectos en tiempo y forma, e implícitamente garanticen la satisfacción del usuario con entregas de valor.

De acuerdo, con los principios de Lean, si un trabajo de desarrollo de *software* no tiene o no genera valor al cliente, es considerado como no utilizable (Alahyari, Berntsson Svensson, & Gorschek, 2017).

Los modelos de *software* en el desarrollo de un proyecto son de vital importancia para lograr un trabajo de calidad y de forma organizada. En este sentido, el presente estudio tiene el propósito de crear un Marco de Trabajo Ágil y Formal (*FAF*), con el potencial de ser integrado en la Coordinación de Servicios de Cómputo del COLMEX, con el objetivo de favorecer el desarrollo de sistemas basado de buenas prácticas, por un lado, en modelos ágiles como *SCRUM*, el cual proporciona una opción por los cambios inmediatos y, por otro, con la guía de gestión de proyectos, *PMBOK*, que brinda formalidad en el desarrollo de un proyecto.

El COLMEX es una institución pública, autónoma, de carácter universitario, dedicada a la investigación y a la enseñanza superior en ciencias sociales y humanidades[3]. Tiene como misión formar profesionales de excelencia académica, desarrollar investigaciones del más alto nivel y difundir el conocimiento para contribuir a la solución de los problemas nacionales.

El contenido del presente trabajo está estructurado de la siguiente forma. En el Capítulo 1, se explica el marco teórico, la terminología básica sobre metodología, norma, modelo, *Framework* y *BOK,* lo cual permitirá entender y explicar la propuesta de intervención. Se presenta la teoría de las metodologías tradicionales y ágiles, una breve descripción de los modelos más usados, así como, una comparación y análisis de diferentes estudios. En el capítulo 2, se explica cómo se seleccionó *SCRUM* y *PMBOK* para la propuesta de intervención, se expone el marco teórico de cada uno de ellos y se hace un cotejo y análisis de las mejores prácticas y puntos donde convergen. En el capítulo 3, se describen los antecedentes del problema a

[3] El Colegio de México. Consultado el 5 de diciembre de 2018, para obtener más información, visité la página https://www.colmex.mx/

resolver, la situación actual del COLMEX, la definición para el marco de trabajo de desarrollo de *software (FAF)*, basado en las buenas prácticas y estudios de *SCRUM* y *PMBOK*, así como las recomendaciones y beneficios con respecto a la implementación.

Para el desarrollo de la propuesta de intervención se realizaron los siguientes pasos:

1. Revisión de la literatura, respecto a los antecedentes teóricos de las metodologías ágiles y tradicionales, así como una breve descripción de los modelos más convencionales.
2. Exposición del marco teórico del *Framework* ágil de *SCRUM* y la guía de proyectos *PMBOK*, basado en la metodología de administración de proyectos del *PMI* (*Project Manager Institute*). Comparación y análisis de las mejores prácticas y los puntos donde convergen *SCRUM* y *PMBOK*.
3. Aplicación de una encuesta a personal del área de sistemas de el COLMEX con el propósito de definir un modelo ágil para la institución.
4. Propuesta de intervención, la cual será un marco de trabajo de desarrollo de sistemas para la Coordinación de Servicios de Cómputo, específicamente en el área de sistemas del COLMEX.

Capítulo 1

Metodologías de desarrollo de *Software*

Capítulo 1. Metodologías de desarrollo de *Software*

1.1 Conceptos de Metodología, Norma, Modelo, *Framework* y *BOK*

De acuerdo al "Diccionario Español Mexicano (DEM)," (2018), la definición de metodología, es el estudio y exposición de los métodos que se aplican a cierta disciplina, desde el punto de vista de su sistematicidad, sus aplicaciones, entre otros. En el contexto de Ingeniería de *software* se refiere a un *Framework* usado para estructurar, planear y controlar los procesos de desarrollo de sistemas (Pressman & Maxim, 2015). Además, proporciona prácticas y herramientas que facilitan el desarrollo de procesos, dando como resultado un producto de alta calidad, y con las expectativas esperadas del cliente. En términos sencillos, una metodología busca tener un mayor control, con base en etapas secuenciales, es decir, no se puede iniciar una etapa sin haber culminado de forma exitosa la anterior.

La palabra norma, de acuerdo con la Real Academia Española (RAE)[4], significa "regla que se debe seguir o a que se deben ajustar las conductas, tareas, actividades, entre otras", por otro lado, el DEM la define como una idea o juicio que guía la conducta de las personas de acuerdo con ciertos valores y determinados fines. Una norma contiene una serie de requisitos obligatorios y auditables, pero a diferencia de la metodología no son consecutivos ni están establecidos en un orden estricto.

Por otro lado, la RAE define un modelo como un "arquetipo o punto de referencia para imitarlo o reproducirlo". En ámbitos de desarrollo de software, para resolver problemas reales en una organización, es recomendable que los ingenieros de sistemas, adopten una estrategia de desarrollo que esté relacionada con procesos, métodos y herramientas. Esto último se conoce como modelo de proceso o paradigma de ingeniería del software (Pressman, 2002).

A diferencia de los dos anteriores donde las etapas de la metodología y los pasos de la norma deben cumplirse en su totalidad, un modelo no tiene reglas

[4] Real Academia Española. Consultado el 28 de agosto de 2019 en http://dle.rae.es/?id=QcFNGvF

estrictas y permite que dependiendo de la naturaleza del negocio o del proceso que estemos analizando, se aplique un cierto número de elementos, es decir, no es necesario llevar a cabo todos y cada uno de ellos.

Por otro lado, tenemos el concepto de *Framework*, quizá el término más utilizado entre los desarrolladores de *software*. A nivel traducción se refiere a un "marco de trabajo". De acuerdo, con (Bart Verbrugge, 2019), *Framework* es una estructura o sistema para la realización de una meta definida, la cual, está conformada por uno o varios modelos basado de mejores prácticas, y al mismo tiempo, es más flexible respecto al uso total o parcial de las técnicas y pasos que conforman el *Framework*. Es esencial mencionar, que un lenguaje de programación no está determinado por un marco de trabajo.

Por último, el concepto de cuerpo de conocimiento o *body of knowledge* (*BOK*) por sus siglas en Inglés, es una serie de conceptos, términos y actividades que engloban un dominio profesional, el cual establece terminología estandarizada a nivel global e internacional.[5]

Ahora que se tiene información de los conceptos de *Framework* y *BOK*, concluimos que justo es lo que necesitamos para el presente trabajo: mejores prácticas de desarrollo de *software* alineados a términos estandarizados. En este documento, utilizaremos el termino de marco de trabajo el cual incluirá componentes de *SCRUM*, así como las fases de inicio, planeación, ejecución y monitoreo, basado de la guía de *PMBOK*. A partir de esta definición, dependiendo del tamaño y naturaleza del proyecto a ser desarrollado, el líder del proyecto seleccionará las etapas, procesos y pasos que sean pertinentes y aplicables al desarrollo que se esté haciendo en un determinado momento.

En la actualidad, hay diferentes metodologías, las cuales se identifican en dos tipos principales, las metodologías tradicionales y las ágiles. Estas se explican a continuación.

[5] Van Haren. Consultado el 14 de diciembre de 2019 en https://www.vanharen.net/blog/best-practice-model-framework-method-guidance-standard-towards-consistent-use-terminology/

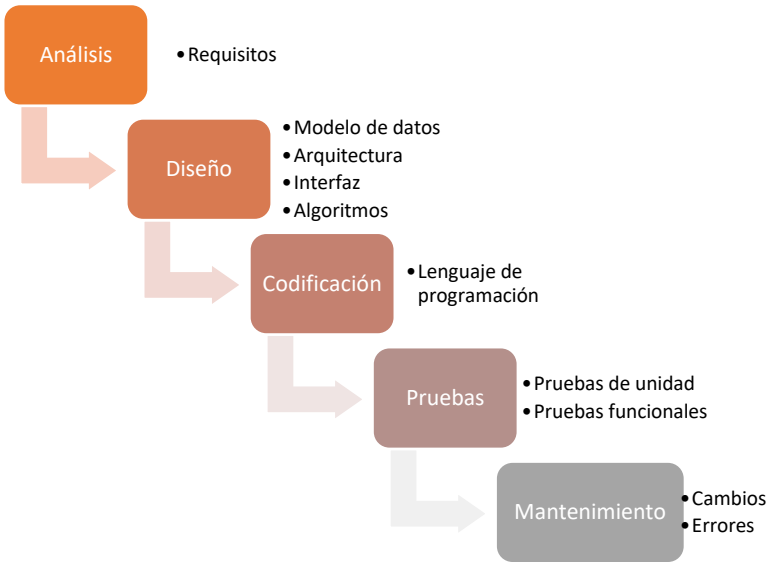
1.2 Metodologías tradicionales

Las metodologías tradicionales de desarrollo de *software* están basadas en una serie de pasos. Inician con una solicitud de requerimientos, diseño, codificación, pruebas y por último la implementación (Awad, 2005). Con esto se trata de asegurar resultados con alta calidad apegados a un calendario. Ejemplos de este tipo de metodologías son la de cascada, prototipos, incremental y espiral (Pressman, 2002), las cuales se resumen a continuación.

1.2.1 Modelo de Cascada

En ingeniería de *software*, también es conocido como modelo lineal o ciclo de vida de un programa, el cual tiene etapas que se deben de realizar de forma secuencial, es decir, el inicio de cada etapa debe esperar hasta que terminé o finalicé la anterior (Pressman, 2002). Es el primer modelo que fue creado y es la base de todos los demás modelos. En la figura 1 se muestran las características que se consideran para este modelo; análisis, diseño, codificación, pruebas y mantenimiento.

Figura 1. Modelo de cascada



Fuente: Pressman, 2002.

1.2.2 Modelo de Prototipos

Este modelo está enfocado a ayudar al usuario a definir y clarificar las actividades del desarrollo de *software*. En la figura 2, se visualiza cómo funciona; inicia con la recolección de requisitos, en conjunto cliente y desarrollador establecen un objetivo global, y en corto tiempo se hace un diseño rápido, es decir, el prototipo. Por último, el usuario final valida de forma visual si se está cumpliendo con la meta. A partir de estos tres pasos se van generando iteraciones, lo cual permite que el programador comprenda mejor lo que se necesita realizar. La clave es establecer las reglas del juego desde el principio, y que ambos comprendan y tengan claro las funcionalidades que se deben de considerar.

Figura 2. Modelo de Prototipos

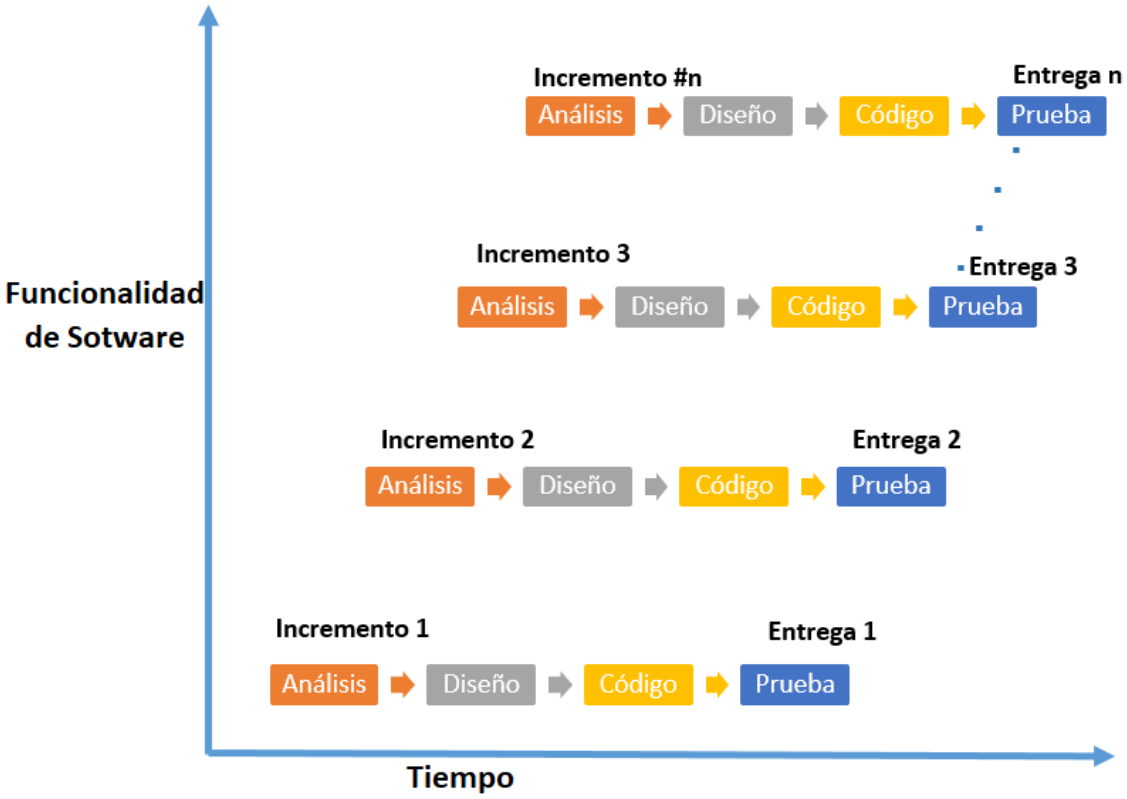
Fuente: Pressman, 2002.

1.2.3 Modelo Incremental

Es una combinación del modelo de cascada con el de prototipos de forma repetida. En la figura 3 se representa como se aplican secuencias lineales de manera escalonada, mientras se avanza en el tiempo. Cada secuencia lineal produce un

incremento del *software* pequeño, pero funcional o utilizable. Este proceso se repite, hasta que se elabore el producto completo.

Figura 3. Modelo incremental



Fuente: Pressman, 2002.

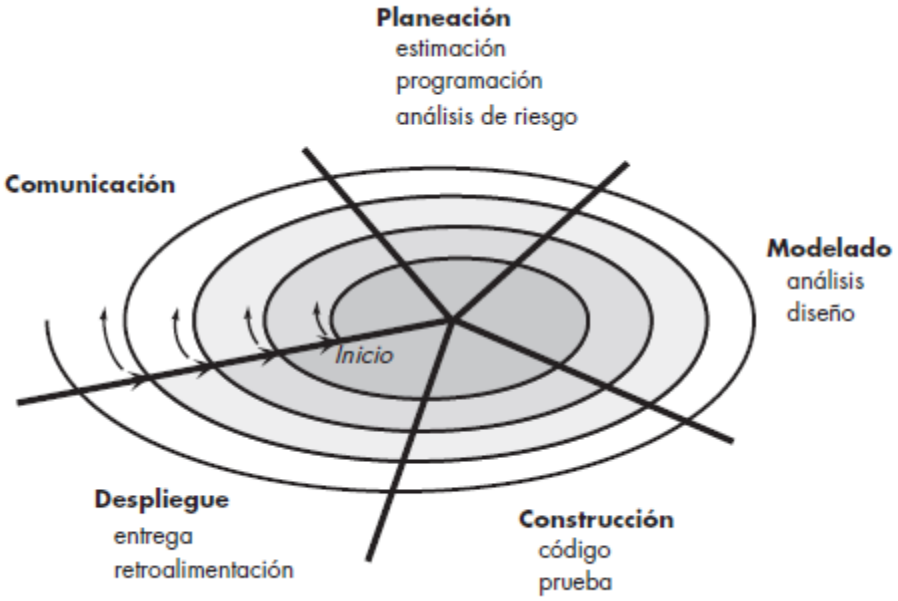
1.2.4 Modelo Espiral

El modelo espiral conjuga la iteración del modelo de prototipos con aspectos controlados del secuencial. El *software* se desarrolla en una serie de versiones incrementales que comprende 5 regiones de tareas, como se muestra en la figura 4.

1. Comunicación: Tareas requeridas para establecer comunicación entre el desarrollador y el cliente.
2. Planificación: Tareas para definir recursos, tiempo, y costos.

3. Modelado: Tareas para analizar riesgos técnicos y de gestión, así como tareas para representar el diseño de la aplicación.
4. Construcción: Tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario.
5. Despliegue: Tareas para obtener la retroalimentación del cliente.

Figura 4. Modelo de Espiral



Fuente: Pressman, 2002.

1.3 Metodologías ágiles

Las metodologías ágiles son flexibles, pueden ser modificadas para que se adapten a la realidad de cada equipo y proyecto. Se genera una lista ordenada de actividades y se hacen en periodos cortos de tiempo, entre dos y seis semanas. La comunicación con el cliente es constante.

La filosofía que siguen estas metodologías es que se centran en el factor humano, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del *software* con iteraciones muy cortas. Este enfoque ágil está mostrando su efectividad en proyectos con requisitos muy cambiantes y

cuando se exige reducir de manera drástica los tiempos de desarrollo, pero manteniendo una alta calidad en el desarrollo de *software* (Letelier & Penadés, 2006).

De acuerdo al reporte del estado ágil de *(CollabNet, 2019)*, las metodologías ágiles usadas a nivel internacional se encuentran la de *extreme programming (XP)*, *lean startup, Kanban, Scrumban y SCRUM*, las cuales se describen de forma breve a continuación.

1.3.1 *Extreme Programming* (*XP*)

Creada por Kent Beck, es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. *XP* se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y actitud para enfrentar los cambios (Letelier & Penadés, 2006). Además, está basada en una serie de valores y de buenas prácticas que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas, así como la satisfacción del cliente.

El segundo objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del *software*.[6]

La programación extrema ayuda a mejorar un proyecto de *software* en cinco formas esenciales; comunicación, simplicidad, retroalimentación, respeto y valor.

1.3.2 *Lean Startup*

Eric Ries diseño un método para crear empresas de éxito utilizando la innovación continua. Los buenos resultados de las *Startups* se pueden diseñar siguiendo un proceso correcto y esto significa que se puede aprender y enseñar.

---

[6] *Extreme programming*. Consultado el 20 de diciembre de 2019 en http://www.extremeprogramming.org/

El concepto original viene de Steve Blank, emprendedor y mentor de Silicon Valley, cuando comenzó a desarrollar una metodología de validación de productos basado en el desarrollo de clientes, que consiste en saber si nuestro producto cubre sus necesidades o deseos (Ries, 2011).

El método *Lean Startup*, está compuesto por cinco principios:

1. Los emprendedores están en todas partes; este método puede funcionar en empresas de cualquier tamaño, incluso en compañías muy grandes, de cualquier sector o actividad.
2. El espíritu emprendedor es *management*; una *Startup* es una institución, donde el crecimiento depende de la innovación.
3. Aprendizaje validado; las *Startups* no solo se crean para producir cosas, ganar dinero o atender a los consumidores, también existen para aprender, generando conocimiento.
4. Crear-Medir-Aprender; la actividad principal es convertir ideas en productos, medir como responden ante los consumidores y aprender cuándo pivotear o perseverar.
5. Contabilidad de la innovación; para mejorar los resultados y medir la innovación, es necesario medir el progreso, establecer hitos y priorizar tareas.

1.3.3 *Kanban*

Formulado por David J. Anderson, es un método productivo que permite mantener el flujo de trabajo, conservando la calidad en la fabricación del producto de forma rápida y garantizada para las organizaciones. *Kanban* aporta control y mejoramiento a los procesos, direcciona la realización de tareas a través del esfuerzo en equipo y ayuda en la administración de los procesos (Fuertes A. & Sepúlveda C., 2016)

Las principales reglas de *Kanban* son las siguientes:

1. Visualizar el trabajo y las fases del ciclo de producción o flujo de trabajo,
2. Determinar el límite de “trabajo en curso” (o *Work In Progress*) y

3. Medir el tiempo en completar una tarea (lo que se conoce como “lead time”).

Si bien *Kanban* no es un método de administración de proyectos, ni una técnica específica de desarrollo de *software*, es aplicado en el desarrollo de sistemas dado su utilidad para el trabajo en equipo. Asimismo, este método se está introduciendo como parte de iniciativas *Lean* para transformar la cultura de las organizaciones y fomentar la mejora continua (Fuertes A. & Sepúlveda C., 2016).

1.3.4 *Scrumban*

Es una metodología ágil combinando el método de *Kanban* y *SCRUM*. Esta metodología híbrida, contempla componentes y conceptos de ambas que se complementan entre sí, para lograr una mejor optimización del proceso de desarrollo.

Scrumban es un modelo de desarrollo adecuado para proyectos de mantenimiento, así como, donde hay historias de usuarios que cambian con frecuencia.

A continuación, se presentan las principales actividades que se hacen en *Scrumban* definidas por Ahmad Khan (2014).

1. Visualizar el flujo de trabajo. Se refiere a visualizar cada historia de usuario en cada una de las etapas del flujo del proceso.
2. Cola de trabajo. Las historias de usuario son priorizadas y seleccionadas para ser realizadas durante un *sprint*, las cuales son alteradas de acuerdo a las necesidades del cliente.
3. Limitar el trabajo en progreso. Solo se asigna trabajo en función de las capacidades del equipo.
4. Reglas explicitas. El objetivo es que todos los miembros del equipo estén facultados para auto-organizarse, con el fin de lograr flujos de trabajo dinámicos y efectivos.

(Ahmad Khan, 2014)

1.3.5 *SCRUM*

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Está indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de *software* se realiza mediante iteraciones, denominadas *sprints*, con una duración máxima de 30 días. La segunda característica fundamental son las reuniones que se llevan a cabo durante el transcurso del desarrollo de *software* (Letelier & Penadés, 2006). Posteriormente, en la sección 2.3 se explicará a detalle.

1.4 Comparación entre metodologías tradicionales y ágiles

Cada proyecto necesita su propia metodología (Cockburn, 2000) de acuerdo con sus características y necesidades. Las metodologías ágiles son adaptables y están enfocadas a las personas, mientras las tradicionales son predictivas y orientadas a procesos (Cadavid, 2013). Ambas tienen sus ventajas, desventajas y son similares en algunos aspectos.

El propósito de este apartado es exponer cuáles son los posibles pros y contras de cada tipo de metodología, sus principales diferencias y hasta qué grado pueden llegar a complementarse. El cuadro 1 muestra una comparación general entre metodologías ágiles y tradicionales.

Cuadro 1. Comparación teórica de metodologías tradicionales y ágiles

Tradicionales	Ágiles
Predictivos	Adaptativos
Orientados a procesos	Orientados a personas
Proceso rígido	Proceso flexible
Se concibe como un proyecto	Un proyecto es subdividido en varios proyectos más pequeños
Poca comunicación con el cliente	Comunicación constante con el cliente
Entrega de *software* al finalizar el desarrollo	Entregas constantes de *software*
Documentación extensa	Poca documentación

Fuente: Khurana & Sohal, 2011.

Como se puede observar, ambas metodologías tienen características que dependiendo la naturaleza del proyecto de desarrollo de *software* pueden resultar útiles o no. En este trabajo no se pretende revisar cuál es mejor o genera más beneficios, sino exponer las particularidades de cada una de ellas para comprender mejor su funcionamiento.

El reporte de "The Standish Group" (2015), consideró una muestra de más de 10,000 proyectos de *software*, en el cual se encontró, que los desarrollos que usan métodos ágiles son cuatro veces más exitosos que aquellos que utilizan métodos de cascada y que, en general, cumplen más con los requisitos originales. (ver cuadro 2).

Cuadro 2. Resultados de los proyectos de software de acuerdo a CHAOS

Tamaño de Proyecto	Método	Exitoso	No cumplió los requisitos	Fallido
Todos	ágil	39%	52%	9%
	cascada	11%	60%	29%
Grande	ágil	18%	59%	23%
	cascada	3%	55%	42%
Mediano	ágil	27%	62%	11%
	cascada	7%	68%	25%
Pequeño	ágil	58%	38%	4%
	cascada	44%	45%	11%

Fuente: The Standish Group, 2015.

Por otro lado, una tendencia relativamente reciente es el manejo de metodologías híbridas, las cuales se crean combinando una serie de elementos de las metodologías tradicionales y las ágiles. Es recomendable, en cualquier caso, tener experiencia y conocimiento de las metodologías para seleccionar aquellos elementos más apropiados con las necesidades en el desarrollo de *software* (Jimenez Hernández & Orantes Jimenez, 2011).

Como hemos visto con anterioridad, las practicas ágiles tienen en común varios aspectos similares, tales como: trabajo en equipo, tableros de actividades, ideas para generar cambios innovadores, equipos pequeños, iteraciones, aprendizaje, respuesta inmediata ante los cambios inesperados, entre otros. En este sentido, la flexibilidad de utilizar y combinar estas prácticas a lo que se conoce como “hibrido” es factible, por lo que no es un requisito seguir con exactitud los lineamientos, si no que se debe de adaptar con el fin de mejorar el marco de trabajo propuesto.

Capítulo 2

SCRUM y *PMBOK*: desarrollo ágil y gestión de proyectos

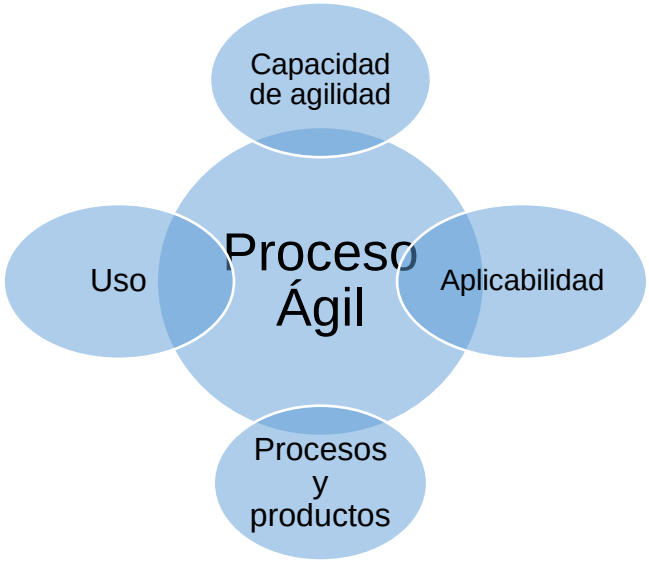
Capítulo 2. *SCRUM* y *PMBOK*: desarrollo ágil y gestión de proyectos

En el presente capítulo se expondrá la justificación de la selección de *SCRUM* como marco de trabajo ágil, tomando como referencia el *Framework* definido por Iacovelli, asimismo, se describirá de forma breve la guía de proyectos *PMBOK*, utilizada de forma constante en la gestión de proyecto tecnológicos.

2.1 *Framework* de Iacovelli

El *Framework* de Iacovelli está enfocado para la clasificación de procesos ágiles (Iacovelli & Souveyet, 2008), considerando sus 4 puntos de vista como se muestra en la figura 5; uso del proceso ágil, capacidad de agilidad, aplicabilidad, procesos y productos. Cada vista con sus respectivos atributos los cuales serán evaluados de acuerdo a cada organización.

Figura 5. Cuatro vistas del Framework de Iacovelli



Fuente: Elaboración propia basado de Iacovelli & Souveyet, 2008.

A continuación, se describe los 4 puntos de vista.

Uso. Se refiere a fundamentar porque es necesario utilizar una metodología ágil, evaluando lo que el equipo de desarrollo y el cliente obtienen al utilizar dicha metodología. Los atributos de este punto de vista son:

- Adaptarse a cualquier tipo de entorno
- Satisfacción del usuario final
- Favorece el *outsourcing*
- Aumento de productividad
- Cumplimiento en calidad
- Cumplimiento de fechas de entrega
- Cumplimiento de requisitos

Capacidad de agilidad. Se refiere a identificar en que parte del proceso se encuentra la agilidad. En este punto de vista se da importancia a la interacción de las personas, las metodologías ágiles tienden a romper las relaciones contractuales entre los clientes y los equipos de desarrollo. Los atributos de este punto de vista son:

- Indicadores de cambio
- Colaboración
- Los requisitos funcionales y no funcionales pueden cambiar
- Los recursos humanos pueden cambiar
- Integración de los cambios
- Intercambio de conocimiento
- Procesos ligeros o sencillos
- Centrado en las personas
- Actitud durante todo el desarrollo del proyecto
- Iteraciones cortas
- Política de pruebas
- El plan de trabajo puede cambiar

Aplicabilidad. Se refiere a cuando el entorno de trabajo es favorable para la aplicación de metodologías ágiles. Los atributos de este punto de vista son:

- Grado de interacción entre los miembros del equipo
- Grado de interacción con el cliente
- Grado de interacción con los usuarios finales
- Grado de integración con la innovación
- Complejidad de los proyectos
- Riesgos de los proyectos
- Tamaño de los proyectos
- Organización del equipo
- Tamaño del equipo

Procesos y productos. Se refiere a como se caracteriza el método usado en el desarrollo de *software*. El proceso se compone de dos dimensiones. La primera dimensión son las actividades de desarrollo y la segunda representa el nivel de abstracción de sus directrices y reglas. Los atributos de este punto de vista son:

- Nivel de abstracción de las normas y directrices
- Actividades cubiertas por el método ágil
- Productos de las actividades del método

En la siguiente sección se explica cómo se aplicó el *Framework* de Iacovelli al personal de sistemas del COLMEX.

2.2 Aplicación del *Framework* de Iacovelli

En esta sección, se muestra como se llevó a cabo la encuesta para elegir un proceso ágil para el desarrollo de *software* en el COLMEX, definido por el *Framework* de Iacovelli.

A partir de los 4 puntos de vista (uso, capacidad de agilidad, aplicabilidad y procesos y productos) explicados en la sección anterior, se definen 4 formularios los cuales se pueden revisar en el Anexo I de este trabajo. Este formulario se aplicó a todo el equipo de desarrollo de sistemas conformado por cinco integrantes (un

coordinador y cuatro ingenieros de sistemas). Enseguida se describen su rol y funciones principales.

a. **Coordinador de sistemas**: Tiene por objetivo planear, supervisar y desarrollar los proyectos asignados de acuerdo con las necesidades de los usuarios, optimizando los recursos y, por ende, contribuir a la mejora de la satisfacción del usuario y al cumplimiento de los objetivos de calidad. Algunas funciones específicas son vigilar el correcto desarrollo, así como, determinar e implementar estrategias de motivación para el personal y de trabajo en equipo.
b. **Ingeniero de sistemas**: Tiene por objetivo desarrollar proyectos de software de acuerdo con las necesidades de los usuarios. Algunas funciones específicas son reunir requerimientos, analizar, diseñar, codificar y dar mantenimiento acorde a las especificaciones solicitadas.

Los resultados de la aplicación de los formularios, se compararon con la versión simplificada de Iacovelli, los cuales se muestran en el cuadro 3.

Cuadro 3. Clasificación de procesos ágiles de Iacovelli

VISTA	Número Pregunta	*Extreme Programming*	*SCRUM*	*KANBAN*	*SCRUMBAN*
USO	A1	FALSO	VERDADERO	FALSO	FALSO
	A2	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	A3	FALSO	FALSO	FALSO	FALSO
	A4	FALSO	VERDADERO	FALSO	FALSO
	A5	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	A6	FALSO	VERDADERO	FALSO	VERDADERO
	A7	VERDADERO	VERDADERO	VERDADERO	VERDADERO
CAPACIDAD DE AGILIDAD	B1	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	B2	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	B3	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	B4	VERDADERO	FALSO	FALSO	FALSO
	B5	VERDADERO	VERDADERO	FALSO	VERDADERO
	B6	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	B7	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	B8	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	B9	FALSO	FALSO	VERDADERO	VERDADERO

	B10	VERDADERO	FALSO	VERDADERO	VERDADERO
	B11	VERDADERO	FALSO	VERDADERO	VERDADERO
	B12	VERDADERO	FALSO	FALSO	FALSO
	B13	ITERACIÓN	ETAPA	ITERACIÓN	ETAPA
	B14	ALTO	BAJO	BAJO	BAJO
APLICABILIDAD	**C1**	PEQUEÑO	GRANDE/PEQUEÑO	PEUQEÑO	GRANDE/PEQUEÑO
	C2	BAJO	ALTO	BAJO	ALTO
	C3	BAJO	ALTO	BAJO	ALTO
	C4	PEQUEÑO	PEQUEÑO	PEQUEÑO	PEQUEÑO
	C5	ALTO	ALTO	BAJO	BAJO
	C6	BAJO	ALTO	BAJO	BAJO
	C7	ALTO	ALTO	BAJO	ALTO
	C8	ALTO	ALTO	BAJO	ALTO
	C9	AUTO-ORGANIZACIÓN	AUTO-ORGANIZACIÓN	AUTO-ORGANIZACIÓN	AUTO-ORGANIZACIÓN
PROCESOS Y PRODUCTOS	**D1**	FALSO	VERDADERO	VERDADERO	VERDADERO
	D2	VERDADERO	FALSO	FALSO	VERDADERO
	D3	VERDADERO	FALSO	FALSO	FALSO
	D4	FALSO	FALSO	FALSO	FALSO
	D5	VERDADERO	VERDADERO	FALSO	VERDADERO
	D6	VERDADERO	VERDADERO	FALSO	FALSO
	D7	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	D8	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	D9	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	D10	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	D11	FALSO	FALSO	FALSO	FALSO
	D12	FALSO	FALSO	FALSO	FALSO
	D13	FALSO	FALSO	FALSO	FALSO
	D14	FALSO	VERDADERO	FALSO	VERDADERO
	D15	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	D16	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	D17	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	D18	VERDADERO	VERDADERO	VERDADERO	VERDADERO
	D19	VERDADERO	FALSO	VERDADERO	VERDADERO
	D20	FALSO	FALSO	FALSO	FALSO
	D21	FALSO	FALSO	FALSO	FALSO
	D22	FALSO	FALSO	FALSO	FALSO

Fuente: Elaboración propia basado de Iacovelli & Souveyet, 2008.

Los resultados obtenidos por parte del equipo del COLMEX se mapearon con el cuadro 3 de clasificación de Iacovelli, y de esta forma se identificó el proceso ágil que mejor se adapta a la forma de trabajo de la institución. En este sentido, el proceso ágil adecuado es el que mayor número de coincidencias tiene, si la respuesta coincide con el valor asociado al cuadro 3 se suma 1 en caso contrario tiene el valor de 0, por lo tanto, se selecciona la que tenga mayor puntuación. Los resultados se resumen en el cuadro 4.

Cuadro 4. Resultados del COLMEX para la elección de un proceso ágil

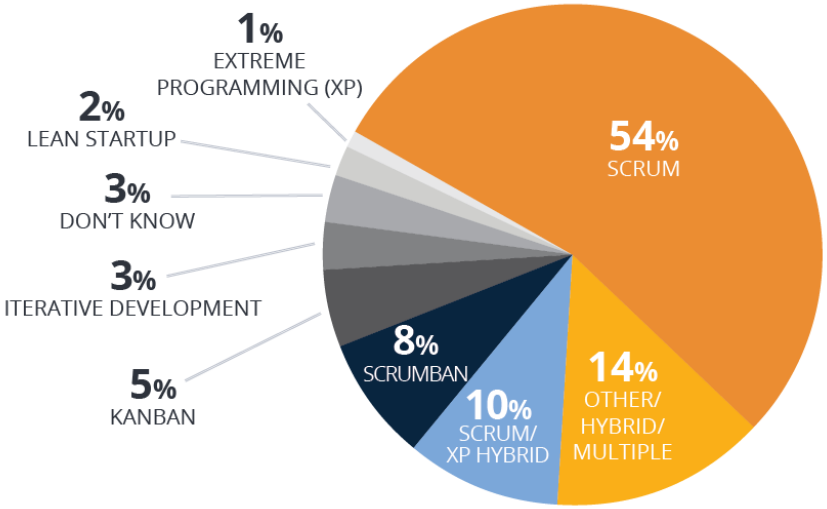
VISTA	Número Pregunta	*Extreme Programming*	*SCRUM*	*KANBAN*	*SCRUMBAN*
USO	A1	1	4	1	1
	A2	5	5	5	5
	A3	0	0	0	0
	A4	0	5	0	0
	A5	4	4	4	4
	A6	0	5	0	5
	A7	5	5	5	5
CAPACIDAD DE AGILIDAD	B1	4	4	4	4
	B2	5	5	5	5
	B3	4	4	4	4
	B4	3	2	2	2
	B5	4	4	1	4
	B6	5	5	5	5
	B7	3	3	3	3
	B8	5	5	5	5
	B9	2	2	3	3
	B10	5	0	5	5
	B11	3	2	3	3
	B12	3	2	2	2
	B13	1	4	1	4
	B14	5	0	0	0
APLICABILIDAD	C1	2	5	2	5
	C2	1	4	1	4
	C3	3	2	3	2
	C4	5	5	5	5
	C5	4	4	1	4

	C6	2	3	2	2
	C7	5	5	0	5
	C8	3	3	2	3
	C9	2	2	2	2
PROCESOS Y PRODUCTOS	**D1**	3	2	2	2
	D2	4	1	1	4
	D3	3	2	2	2
	D4	2	2	2	2
	D5	1	1	4	1
	D6	4	4	1	1
	D7	5	5	5	5
	D8	3	3	3	3
	D9	4	4	4	4
	D10	5	5	5	5
	D11	1	1	1	1
	D12	4	4	4	4
	D13	0	0	0	0
	D14	2	3	2	3
	D15	3	3	3	3
	D16	1	1	1	1
	D17	4	4	4	4
	D18	4	4	4	4
	D19	2	3	2	2
	D20	2	2	2	2
	D21	5	5	5	5
	D22	2	2	2	2
	TOTALES	**158**	**164**	**135**	**161**

Fuente: Elaboración propia

De acuerdo con los resultados del *Framework* de Iacovelli, el proceso ágil que mejor puede adaptarse al COLMEX, según las personas encuestadas, es *SCRUM*. Este resultado está en consonancia con algunos estudios, puesto que este marco de trabajo es el más utilizado, acorde a una encuesta realizada por *VersionOne* (CollabNet, 2019), le sigue un hibrido múltiple con el 14% y en tercer lugar esta una parte de *SCRUM* con *XP* con el 10%, como se puede apreciar en el gráfico 1.

Gráfico 1. Métodos y prácticas ágiles en el reporte del estado ágil no 13



Fuente: CollabNet, 2019.

SCRUM es uno de los marcos de trabajo ágiles más populares. Es un *Framework* adaptable, iterativo, rápido, flexible y eficaz, diseñado para ofrecer un valor considerable en forma rápida a lo largo del proyecto (Satpathy, 2017).

A partir de los resultados del cuestionario de Iacovelli aplicado al personal del COLMEX, y tomando como referencia los datos de la encuesta de *VersionOne*, se determinó utilizar *SCRUM* para la presente propuesta de intervención, metodología que se detalla en la siguiente sección.

2.3 *SCRUM*

La palabra *SCRUM* alude a un concepto del *rugby*, relacionado con la formación requerida para la recuperación rápida del juego ante una infracción menor. En 1986, Takeuchi y Nonaka, definen un nuevo enfoque en el desarrollo de productos, dirigido a incrementar su flexibilidad y rapidez, a partir de la integración de un equipo interdisciplinario (Cadavid, 2013). *SCRUM* se reduce a un marco de trabajo ágil para el desarrollo de *software* el cual busca otorgar un alto valor del negocio en un corto tiempo.

Otras características son que el negocio fija las prioridades, hay equipos auto organizados, los cuales determinan que tareas quieren realizar, el *software* funcional se muestra cada 2 semanas, en el cual se decide si se libera o se mejora, es decir, se fundamenta con el trabajo visible.

Este modelo parte de supuestos que hay en el desarrollo de sistemas, tales como:

- El cliente sabe lo que quiere y realiza definiciones correctas.
- El desarrollo de *software* es predecible y se puede determinar un tiempo específico para el plan del proyecto.
- Los desarrolladores trabajan con procesos definidos y estos no cambian.
- El equipo y el cliente se comunican con documentos.
- Los retrasos se pueden estimar.
- Los desarrolladores disfrutan leer documentación técnica.

Para contrarrestar los supuestos mencionadas, el modelo está basado en un manifiesto ágil, compuesto por los siguientes valores:

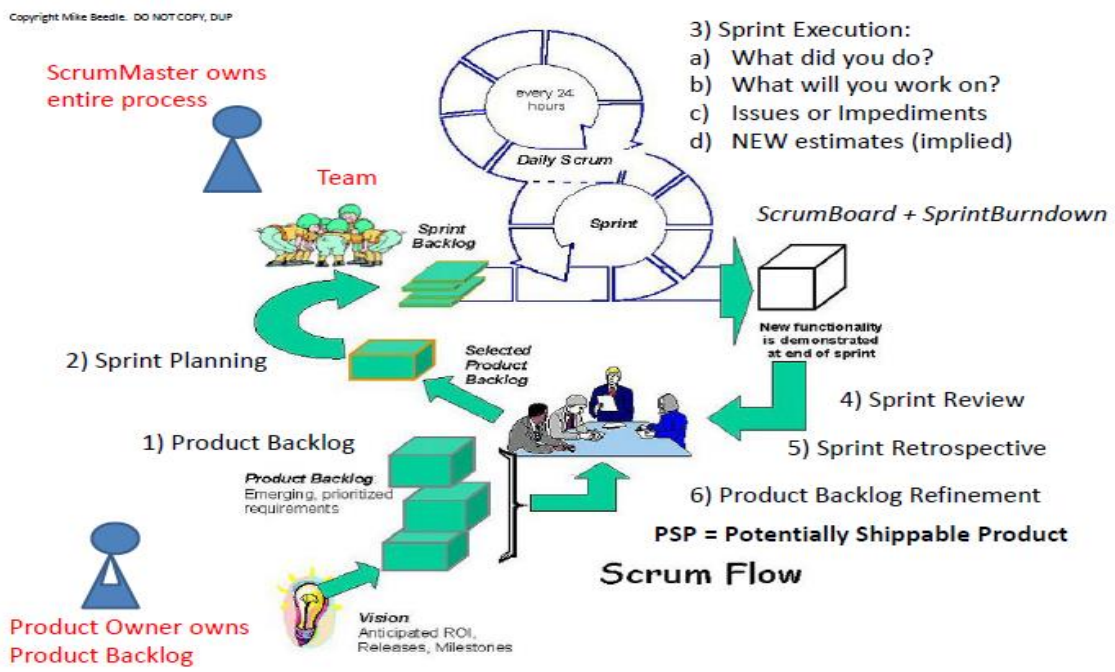
- Individuos e interacciones sobre procesos y herramientas. La gente es el principal factor de éxito de un proyecto *software*. Si se sigue un buen proceso de desarrollo, pero el equipo falla, los resultados positivos no están asegurados; sin embargo, si el equipo funciona, es más fácil conseguir el objetivo final, aunque no se tenga un proceso bien definido. No se necesitan desarrolladores brillantes, sino personas que se adapten bien al trabajo en equipo (Letelier & Penadés, 2006).
- Desarrollar *software* que funcione más que conseguir una documentación extensiva. Aunque se parte de la base de que el *software* sin documentación es un desastre, la regla a seguir es “no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante”. Estos documentos deben ser cortos y centrarse en lo fundamental (Letelier & Penadés, 2006).
- La colaboración con el cliente debe ser más importante que hacer un contrato. La característica particular del desarrollo de *software* hace que muchos proyectos hayan fracasado por intentar cumplir unos plazos y

unos presupuestos preestablecidos al inicio del mismo, según los requisitos que el cliente manifestaba en ese momento (Letelier & Penadés, 2006). Para evitar esto, el enfoque es centrase en lo que el cliente vaya requiriendo durante el desarrollo del *software*.

- Respuesta ante un cambio más que seguir un plan específico. La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, entre otros.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta puesto que hay muchas variables en juego, debe ser flexible para poder adaptarse a los cambios que puedan surgir (Letelier & Penadés, 2006).

A partir de los valores mencionados, se derivan 12 principios del manifiesto ágil, que se mencionan en la sección 3.4 de la presente propuesta. El proceso general de *SCRUM* se resume en la figura 6.

Figura 6. Proceso general de SCRUM



Fuente: Beedle, Mike. Taller *SCRUM Master*, 2013.

A continuación, se describe el marco de trabajo de acuerdo a documentación del taller *SCRUM Master*, impartido por Mike Beedle en 2013.

Se inicia con el *Product Backlog*, es decir recopilar todo el conjunto de requerimientos necesarios para la elaboración del proyecto. Después, se realiza una reunión (*Meeting Backlog*), para establecer la prioridad con la que se llevaran a cabo las solicitudes. Posteriormente, acorde a los requerimientos priorizados, en una reunión (*Release Planning*) se analizan y se determina el alcance del plan de entrega.

Con los requerimientos detallados se hace otra reunión (*Sprint Planning*) en la cual se crean las tareas necesarias para cada requerimiento y de igual forma se establece una prioridad con la estimación de tiempo y el recurso que hará la actividad.

Posterior al inicio de la ejecución de las tareas, diario hay una reunión (*Daily SCRUM*) de máximo 15 minutos, con el objetivo de que los miembros del equipo comenten que hicieron un día anterior, que van hacer el día actual y que problemas o impedimentos tiene para llevar a cabo sus tareas.

Los avances de las tareas se van actualizando en un tablero de control (*Burndown*) donde se visualiza cuantas unidades faltan para terminar el proyecto.

Al término de cada *sprint*, se hace una reunión (*Sprint Review*), para revisar trabajo visible y determinar si hay que hacer adecuaciones. Cuando, se da por terminado el proyecto, todos los participantes se reúnen (*Sprint Retrospective*) con el propósito de mejorar el proceso de desarrollo de *SCRUM*, dar su opinión sobre lo que funciono y podría mejorarse.

Los puntos que hay que considerar son roles, reuniones y documentos.

Los roles se conforman de la siguiente manera:

- *Product Owner*: Es la persona que puede representar a *stakeholders*, clientes o usuarios. Prioriza los requerimientos, define las funcionalidades y revisa los entregables.

- *SCRUM Master*: Es la persona responsable de promover los valores y prácticas de *SCRUM*. Es un *coach*, remueve impedimentos, elimina obstáculos y resuelve problemas. Fomenta la creatividad, la colaboración e intercambio de conocimientos.
- *Team*: Es un conjunto de personas que se caracterizan por ser auto organizados, cuentan con habilidades técnicas, son personas multifuncionales, es decir, analizan, diseñan, programan, validan y también pueden ser líderes de proyecto. Son voluntarias de auto asignarse el trabajo.

Las reuniones la podemos resumir como:

- *Meeting Backlog*: Reunir requerimientos y priorizarlos.
- *Release Planning*: Determinar alcance de entrega y detallar requerimientos.
- *Sprint Planning*: Generar tareas, estimar tiempos y asignar recursos.
- *Daily SCRUM*: Responde a tres preguntas; que hiciste ayer, que vas hacer hoy y si tienes problemas para realizar tus actividades.
- *Sprint Review*: Determinar si se requieren hacer cambios en las tareas correspondientes al *sprint* en curso.
- *Sprint Retrospective*. Revisar que se hizo bien o no, para mejorar el proceso.
- *PB Refinement*. Identificar si se cumplieron los objetivos del proyecto y validar si hacen falta más recursos.

Los documentos utilizados son:

- *Product Backlog*: Listado de requerimientos.
- *Sprint Backlog*: Listado de tareas a ejecutar en cada *Sprint*.
- *Sprint Burndown*: Gráfico para visualizar el trabajo pendiente.
- *Release Burndown*: Gráfico para visualizar los *Sprints* pendientes.
- *Scrumboard*: Tablero que muestra las tareas por hacer, en progreso y las que ya están terminadas.

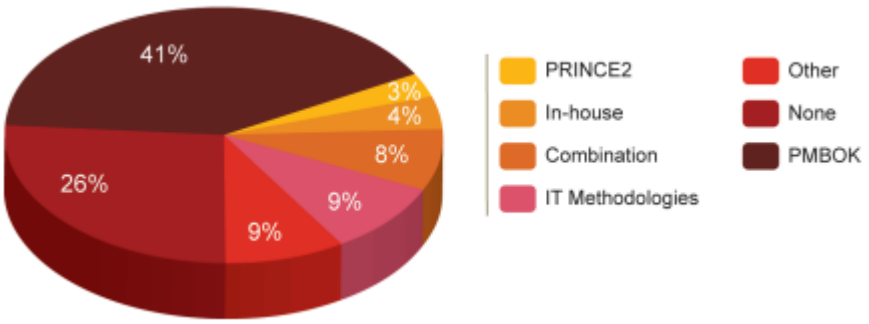
Como podemos apreciar, este marco de trabajo está enfocado en hacer los requerimientos y tareas más rápido con entregables inmediatos, sin embargo, es conveniente tener un control en la administración de los proyectos de *software*, por lo que es necesario incluir en la propuesta una guía de gestión de proyectos.

2.4 *PMBOK*

Las metodologías más usadas, entre las compañías y organizaciones, para la administración de proyectos de desarrollo de *software* son *PMBOK* y *PRINCE2* (Matos & Lopes, 2013). De acuerdo a una encuesta realizada por PWC global (2012), *PMBOK* es la más utilizada.

El gráfico 2 muestra diferentes metodologías y la que predomina es la que este cargo por el *Project Managment Institute* (PMI), basada en la guía del *PMBOK* con el 41%.

Gráfico 2. Metodologías de Proyectos a nivel global



Fuente: Pwc, 2012.

En la presente propuesta se utilizará *PMBOK*, por la razón que es una guía que está fundamentada en el “conocimiento” generado por las mejores prácticas, técnicas, herramientas y se enfoca a los requerimientos del cliente, a diferencia de *PRINCE2* que está basado en la definición de su propia metodología y orientado en los casos de negocio (Jaziri, El-Mahjoub, & Boussaffa, 2018).

El *PMBOK*[7] (*Project Management Body of Knowledge*) es una guía de gestión de proyectos que recoge las mejores prácticas del sector, gestionado y actualizado de forma periódica por el PMI (*Project Management Institute*). Esta disciplina formal de proyectos se encuentra en su sexta edición publicada en 2017.

Un error común es que se considera a *PMBOK* como una metodología de gestión de proyectos, pero en realidad es una guía que recopila las mejores prácticas del sector y que por tanto es tarea del *Project Manager* decidir que procesos aplica y con qué nivel de detalle.

La definición de un proyecto está dada por un esfuerzo temporal para crear un producto, servicio o resultado único con un inicio y fin de término (PMI, 2017).

La guía de *PMBOK* contempla un marco amplio de aspectos industriales y de soluciones; desde proyectos de construcción, de gobierno y hasta esfuerzos de desarrollo de *software* dinámicos y rápidos. Además, proporciona buenas prácticas validada por años de investigación de organizaciones profesionales y de alta disciplina (Briscoe & Staikoff, 2006).

La administración de un proyecto corresponde a la aplicación de conocimiento, habilidades, herramientas y técnicas para cumplir con los requerimientos iniciales, de acuerdo al *PMBOK* 6ta edición, está conformada por 10 áreas de conocimiento y por 49 procesos agrupados en 5 fases:

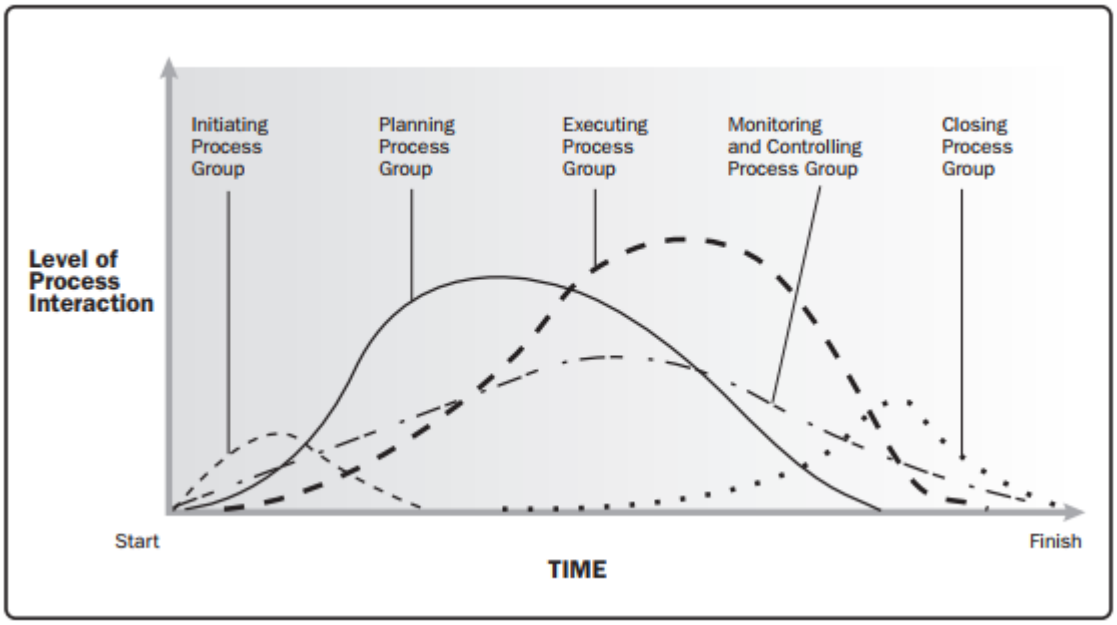
1. Iniciación,
2. Planificación,
3. Ejecución,
4. Monitoreo y Control
5. Cierre de proyecto.

Lo anterior ayudara a los profesionales del proyecto a planificar, organizar y entregar proyectos a tiempo, con el costo presupuestado y de calidad (Gonçalves, Drumond, Méxas, & Méxas, 2017).

[7] Project Management Institute. Consultado el 25 de enero de 2019, para obtener más información visité el sitio https://www.pmi.org

La figura 7 muestra como los proyectos se dividen en fases.

Figura 7. Grupo de procesos que interactúan en un proyecto



Fuente: *PMBOK* 2017.

Podemos visualizar que los procesos no son en su totalidad secuenciales entre sí: la fase de inicio toma relevancia al principio del proyecto, que es cuando se determina que es lo que se va hacer. La fase de planeación y ejecución son las que tiene mayor interacción con los procesos, se deben coordinar personas y otros recursos para realizar el plan. La fase de monitoreo está presente en todos los procesos y en esta se asegura que los objetivos del proyecto se están cumpliendo, en caso contrario tomar acciones correctivas de ser necesario. Al final, en la fase de cierre, siempre debemos concluir los proyectos formalizando la aceptación del proyecto.

Cada fase de un proyecto ésta marcada por la finalización de uno o varios entregables. Un entregable es un trabajo tangible y verificable, un diseño detallado o un prototipo, los cuales son revisados con la finalidad de determinar si el proyecto debe continuar hacia la siguiente fase y/o detectar errores para corregirlos.

Los entregables en cada una de las fases están resumidos en el cuadro 5 de acuerdo a un curso de capacitación en 2012 que se impartió por en la empresa Alpha Consultoría[8]

Cuadro 5. Lista de entregables en PMBOK

Fase	Documentos
Inicio	• Minuta de acuerdos de reunión de entendimiento • Acta de Proyecto • Declaración de alcance
Planeación	• Plan de gestión de proyecto • Descripción de puestos • Matriz de habilidades requeridas • Matriz de responsabilidades • Plan de calidad • Plan de riesgos • *Work Breakdown Structure* (WBS) • Cronograma • Estructura de directorios
Ejecución	• Solicitud de cambio de alcance • Minuta de acuerdos • Reporte de actividades • Reporte de avance • Actividades en adelanto
Seguimiento y Control.	• Seguimiento al Cronograma • Plan de gestión de Riesgos
Cierre	• *Checklist* de cierre • Acta de cierre de proyecto • Minuta de acuerdos de Cierre.

Fuente: Elaboración propia.

El *PMBOK* define 10 áreas de conocimiento en función de sus prácticas y sus procesos, las cuales se mencionan a continuación.

[8] AlphaConsultoria. Consultado el 5 de febrero de 2019, para obtener más información visité el sitio https://www.alpha-consultoria.com/

1. **Gestión de integración**: Describe los procesos requeridos para garantizar que todos los elementos del proyecto están coordinados e integrados.
2. **Gestión de alcance**: Describe los procesos requeridos para asegurar que el proyecto incluye todo el trabajo requerido para completarlo con éxito.
3. **Gestión de tiempo**: Describe los procesos correspondientes para cumplir con las fechas de entrega y finalizar el proyecto a tiempo.
4. **Gestión de costos**: Describe los procesos requeridos para asegurar que el proyecto se cumpla dentro del presupuesto aprobado.
5. **Gestión de calidad**: Describe los procesos para incorporar la política de calidad de la organización, para asegurar que el proyecto cubra las necesidades y satisfacer las expectativas de los interesados.
6. **Gestión de recursos humanos**: Describe los procesos para identificar, adquirir y gestionar los recursos necesarios para un término exitoso del proyecto.
7. **Gestión de comunicación**: Procesos para asegurar la generación, recopilación, difusión, almacenamiento y disposición de la información del proyecto de una manera puntual y adecuada.
8. **Gestión de riesgos**: Proceso para identificar, analizar y dar respuesta a posibles riesgos durante el desarrollo del proyecto.
9. **Gestión de adquisiciones**: Describe los procesos requeridos para adquirir bienes y servicios de proveedores externos.
10. **Gestión de los interesados**: Procesos para identificar las personas, grupos u organizaciones que puedan afectar o ser afectados por el proyecto.

(PMI, 2017)

En el cuadro 6 se resume como es la relación de los procesos con las áreas de conocimiento.

Cuadro 6. Relación de procesos y áreas de conocimiento, PMBOK versión 6

<table>
<tr><th>Inicio</th><th>Planeación</th><th>Ejecución</th><th>Monitoreo y control</th><th>Cierre</th></tr>
<tr><td colspan="4">Gestión de los interesados</td><td></td></tr>
<tr><td></td><td colspan="3">Gestión de las adquisiciones</td><td></td></tr>
<tr><td></td><td colspan="3">Gestión de los riesgos</td><td></td></tr>
<tr><td></td><td colspan="3">Gestión de las comunicaciones</td><td></td></tr>
<tr><td></td><td colspan="3">Gestión de los recursos</td><td></td></tr>
<tr><td></td><td colspan="3">Gestión de la calidad</td><td></td></tr>
<tr><td></td><td>Gestión de los costos</td><td></td><td>Gestión de los costos</td><td></td></tr>
<tr><td></td><td>Gestión del tiempo</td><td></td><td>Gestión del tiempo</td><td></td></tr>
<tr><td></td><td>Gestión del alcance</td><td></td><td>Gestión del alcance</td><td></td></tr>
<tr><td colspan="5">Gestión de la integración</td></tr>
</table>

Fuente: Elaboración propia basado de Todopmp[9]

A continuación, se mencionan de forma agrupada a la fase correspondiente, los 49 procesos de acuerdo a la versión 6 del *PMBOK*:

Inicio

1. Desarrollar el acta de constitución del proyecto.
2. Identificar a los interesados.

Planeación

3. Desarrollar el plan para la dirección del proyecto.
4. Planificar el involucramiento de los interesados.
5. Planificar la gestión del alcance.
6. Recopilar los requisitos.
7. Definir el alcance.
8. Crear la EDT/WBS.
9. Planificar la gestión del cronograma.
10. Definir las actividades.
11. Secuenciar las actividades.
12. Planificar la gestión de los riesgos.

[9] TodoPMP. Consultado el 25 de enero de 2020 en https://todopmp.com/

13. Identificar los riesgos.
14. Realizar el análisis cualitativo de riesgos.
15. Realizar el análisis cuantitativo de riesgos.
16. Planificar la respuesta a los riesgos.
17. Planificar la gestión de recursos.
18. Planificar la gestión de los costos.
19. Estimar los costos.
20. Estimar los recursos de las actividades.
21. Estimar la duración de las actividades.
22. Desarrollar el cronograma.
23. Determinar el presupuesto.
24. Planificar la gestión de la calidad.
25. Planificar la gestión de las comunicaciones.
26. Planificar la gestión de las adquisiciones.
27. Dirigir y gestionar el trabajo del proyecto.

Ejecución

28. Gestionar el conocimiento del proyecto
29. Gestionar la participación de los Interesados.
30. Adquirir recursos.
31. Desarrollar el equipo.
32. Dirigir al equipo.
33. Gestionar las comunicaciones.
34. Efectuar las adquisiciones.
35. Gestionar la calidad.
36. Implementar la respuesta a los riesgos.

Monitoreo y control

37. Monitorear y controlar el trabajo del proyecto.
38. Realizar el control integrado de cambios.
39. Monitorear el involucramiento de los interesados.
40. Controlar el cronograma.

41. Controlar los costos.
42. Monitorear las comunicaciones.
43. Monitorear los riesgos.
44. Controlar la calidad.
45. Controlar los recursos.
46. Validar el alcance.
47. Controlar el alcance.
48. Controlar las adquisiciones.

Cierre

49. Cerrar el proyecto o fase.

Para lograr reunir todas las actividades en el marco de trabajo que se va a proponer, se revisarán algunos estudios comparativos entre *SCRUM* y *PMBOK*, a fin de obtener las buenas prácticas y determinar los puntos donde convergen.

2.5 Comparativa entre *SCRUM* y *PMBOK*

En esta sección se presentan estudios y análisis que se han hecho en relación a la evaluación de las mejores prácticas de *SCRUM*, *PMBOK* y otras, con el objetivo de revisar posibles puntos de convergencia.

Goncalvez (Gonçalves et al., 2017), hizo un estudio donde se aplicó un cuestionario a 90 especialistas relacionados al desarrollo de *software* en compañías localizadas en Rio de Janeiro, Brasil. Los resultados se muestran en el cuadro 7 considerando una escala del 1 al 5.

1. En desacuerdo
2. Parcialmente en desacuerdo
3. Ni en desacuerdo y ni en acuerdo (neutral)
4. Parcialmente de acuerdo
5. De acuerdo.

Cuadro 7. Resultados del cuestionario a especialistas en desarrollo de software

	Puntos calificados	**Escala *Likert* [%]**				
		5	**4**	**3**	**2**	**1**
1	Detalle de planificación y control de acuerdo a *PMBOK*	26	54	7	12	1
2	Mejor comunicación entre los miembros del equipo con *SCRUM* y *PMBOK*	52	36	8	4	0
3	Aprendizaje a corto plazo mediante iteraciones retrospectivas con *SCRUM*	52	37	8	3	0
4	Facilidad de uso de *SCRUM*	40	43	12	5	0
5	Documentación adicional con *PMBOK*	36	30	16	16	2
6	Producto funcionando en períodos cortos en *SCRUM*	46	36	13	5	0
7	Aumento de motivación del equipo debido al uso de las prácticas de *SCRUM*	38	36	21	5	0
8	Aumento de la productividad del equipo, como resultado de las prácticas de *SCRUM*	40	44	12	4	0
9	Mayor control de las actividades del proyecto, como resultado de las reuniones diarias de acuerdo a las prácticas de *SCRUM*	49	32	8	10	1
10	Mejor colaboración entre *stakeholders* a partir del uso de las prácticas de *SCRUM*	44	39	14	3	0
11	Visibilidad de las actividades por parte de todos los miembros del equipo como resultado de las prácticas de *SCRUM*	60	27	11	2	0
12	Mayor agilidad para justar requisitos durante el diseño en *SCRUM*	34	41	14	8	3
13	Las prácticas de *SCRUM* pueden contribuir al cierre del proyecto más rápido	34	42	17	4	3
14	Las prácticas de *SCRUM* pueden contribuir a reducir los costos del proyecto	23	41	24	6	6

Fuente: Gonçalves et al., 2017.

Los resultados muestran que tanto *PMBOK* y *SCRUM* son de utilidad. *PMBOK* en la parte de planeación y control y *SCRUM* para comunicación y entregas en periodos cortos de tiempo. La apreciación es que *SCRUM* es más fácil de aplicar, mientras que en *PMBOK* valoran la importancia de la documentación.

Por otro lado, (Alavandhar & Ṇikiforova, 2017) plantean la pregunta acerca de que si las metodologías ágiles son mejor que las tradicionales, en el cual hace

una comparación considerando a *SCRUM* y *Microsoft Solutions Framework (MSF)*. De este planteamiento solo se considera un mapeo de roles y de fases.

En el cuadro 8 se muestra el mapeo de Roles, en general en *SCRUM* el *Product Owner* es el representante del producto y es responsable de la calidad del proyecto. El equipo de desarrollo en *SCRUM* es multidisciplinario y multifuncional, lo que significa que pueden analizar, diseñar la interfaz gráfica, definir la arquitectura, codificar, validar e implementar el sistema solicitado de tal forma que brinde una mejor experiencia para el usuario.

Cuadro 8. Mapeo de Roles de MSF y SCRUM

Microsoft Solutions Framework	***SCRUM***
Product Management	*Stakeholders*
Program Management	*Product Owner*
Testing Release / Operations	
Architecture	*Development Team*
User Experience	
Development	*SCRUM Master*

Fuente: Alavandhar & Ṇikiforova, 2017.

En el cuadro 9 se muestra el mapeo de Fases, se inicia con la apertura del proyecto, generar un plan, estimación de tiempo y trabajar en el desarrollo de tareas para implementar el correcto funcionamiento. Por último, se hace una retroalimentación del trabajo desarrollado por todos los miembros del equipo, y así, dar por terminado con el proyecto.

Cuadro 9. Mapeo de fases y actividades de MSF y SCRUM

MSF	***SCRUM***
Visualizar	Inicio
Planear	Planeación y estimación
Desarrollar	Implementar

Estabilizar	Implementar (Parte de la implementación del proceso)
(No hay fase equivalente)	Revisión y retrospectiva
Liberación del producto	Lanzamiento del producto

Fuente: Alavandhar & Nikiforova, 2017.

En *SCRUM* no hay una disciplina específica para definir los riesgos, pero como es un proceso continuo, estos se identifican en las reuniones diarias de 15 minutos. En este sentido, podemos omitir el documento de plan de riesgos, puesto que, se van a identificar y documentar en la reunión diaria.

En este estudio, se concluye que el marco de trabajo *SCRUM* no tiene relevancia sobre *MSF* en términos de desarrollo de soluciones de *software*, aunque se menciona que *SCRUM* muestra desorganización, contradicciones e inconsistencias. Se recomienda aplicar la solución que mejor se adapte a las especificaciones del negocio y utilizar el equipo de desarrollo adecuado con modelos híbridos, es decir usar las mejores prácticas tradicionales y ágiles.

De acuerdo a (Briscoe & Staikoff, 2006) los puntos de convergencia se da en las siguientes actividades:

- Actividades de planificación del producto. *PMBOK* identifica la necesidad de una estructura de desglose de producto (*WBS*), como una buena forma de organizar el producto deseado de un proyecto. *SCRUM* identifica un listado de requerimientos conocido como *Product Backlog*.
- Actividades de planificación y gestión de tareas. *PMBOK* identifica la necesidad de un *WBS*, como una buena forma de unir los pasos necesarios para obtener el producto deseado. *SCRUM* identifica tareas realizadas durante un *Sprint*.
- Actividades de informe de progreso. *PMBOK* identifica la necesidad de un informe periódico de estado o progreso como una buena forma de comunicar el avance y mantener la confianza del cliente a medida que se desarrolla el proyecto. *SCRUM* identifica un *Sprint Backlog* para este propósito.

Los puntos donde no convergen:

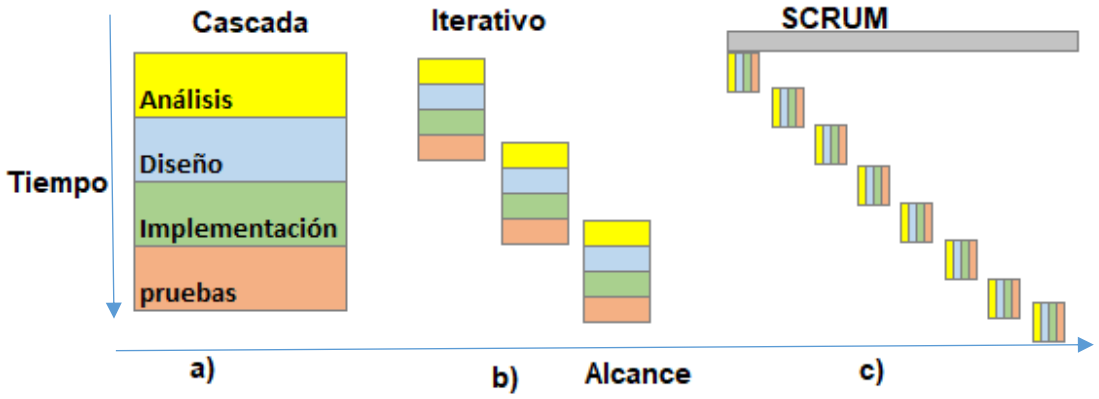
- En *SCRUM* hay facilitadores que ayudan al equipo a obtener los recursos necesarios para que lleven a cabo sus tareas, se denominan *SCRUM Masters*, mientras en *PMBOK* hay administradores de proyecto que están más orientados a la asignación, dirección y medición del desempeño del proyecto.
- En *PMBOK* se establecen entregables y reuniones en tiempos dispersos y diversos, mientras en *SCRUM* el trabajo visible se muestra con periodicidad de 30 días máximo.
- En *PMBOK* identifica buenas prácticas para la gestión de proveedores, contratos y riesgos, por el contrario, *SCRUM* excluye la gestión de proveedores y contratos. Respecto a la mitigación de riegos lo maneja de forma implícita en su proceso.

Después de revisar las comparaciones estudiadas por los autores mencionados, identificaremos cuales son las mejores prácticas, convenientes, para definir un marco de trabajo que ayude al COLMEX a facilitar sus actividades y hacer entregas de calidad.

Se considera entonces, que las técnicas y recomendaciones en *PMBOK* y *SCRUM*, pueden integrarse de manera efectiva a fin de lograr una entrega de *software* más rápida, flexible y, al mismo tiempo, proporcionar una planificación detallada, instrumentos para monitorear, controlar y comunicar durante el desarrollo de un proyecto de *software*.

Con base en la Figura 8, en el cual se muestra la evolución del proceso del modelo de cascada para *SCRUM*, se muestra la adaptabilidad de usar metodologías ágiles dentro de modelos tradicionales. El inciso a, muestra el modelo en cascada en el cual se aprecia que el ciclo tiene una duración extensa, por otro lado, en el inciso b, los ciclos son más cortos. La evolución del desarrollo iterativo a modelos ágiles está inmersa en *SCRUM* (c), donde el análisis, diseño, codificación y pruebas son combinadas en periodos cortos durante todo el proyecto.

Figura 8. Evolución del modelo de cascada hacia SCRUM



Fuente: Elaboración Propia basada de Gonçalves et al., 2017.

Considerando la evolución anterior, podemos decir que *SCRUM* está relacionado de manera tácita con la metodología tradicional, por lo que queda claro que no son opciones separadas, sino, todo lo contrario, complementarias.

Se considera, entonces que es posible gestionar un proyecto teniendo en cuenta las mejores prácticas y el marco conceptual definido en la guía de proyectos *PMBOK*, aplicando prácticas y metodologías ágiles para su desempeño. Por lo tanto, no es mejor una que la otra, dado que no son ni comparables, ni excluyentes (Gonçalves et al., 2017).

La efectividad de prácticas ágiles como *SCRUM*, han demostrado sus bondades en el desarrollo de *software*, en comparación con las metodologías tradicionales. De acuerdo con Pete Deemer, el agilísimo ha ganado popularidad y ofrece datos relevantes y positivos en productividad, capacidad, adaptabilidad, responsabilidad, colaboración y cooperación (Fuertes A. & Sepúlveda C., 2016).

Además, *SCRUM* ayuda a generar entregas rápidas y con valor a los clientes finales y, por su parte, *PMBOK* es útil por el seguimiento de un proyecto apoyado de formatos para su documentación.

A partir de estos resultados y con el conocimiento general de las diferentes propuestas de marcos de trabajo del capítulo 1, en el siguiente capítulo se expone la propuesta de intervención para el desarrollo de *software* en el COLMEX.

Capítulo 3

Propuesta de marco de trabajo *FAF*

Capítulo 3. Propuesta de marco de trabajo *FAF*

A partir de la teoría revisada y estudios comparativos entre las metodologías ágiles, y tradicionales, así como, el marco de trabajo de *SCRUM* y la guía de *PMBOK*, enseguida se presenta el planteamiento del problema y la propuesta de marco de trabajo para el desarrollo de *software,* para el uso de la Coordinación de Servicios de Cómputo en el área de sistemas para el COLMEX.

3.1 Planteamiento del problema

Hoy en día, se considera que la falta de adopción o implementación de modelos, metodologías, gestión de procesos y la no adaptación a los cambios tecnológicos en las organizaciones, puede derivar en ineficiencias y costos.

En la actualidad en el COLMEX, existen procesos que no están sistematizados y no tienen documentación, por lo que es conveniente iniciar con la implementación de un *Framework* para el desarrollo de *software*, a fin de hacer más eficientes los procesos de admisión, intercambio académico, gestión escolar, proyectos académicos y administrativos, entre otros.

El papel de los *Frameworks,* es fundamental en el desarrollo de *software* para potenciar, por ejemplo, la reutilización de los diferentes elementos del *software* y facilitar la labor de todos los roles que participan en el proceso. Para maximizar las ganancias en una organización, se deben considerar tres puntos básicos, trabajar más rápido, trabajar más ágil y trabajar menos (Quintero & Anaya, 2007). En el ámbito de desarrollo de *software*, el último punto se refiere, a cambiar la forma de trabajar, es decir, no invertir tiempo y trabajo en diseño, codificación y pruebas, si no programar en el nivel de ingeniería de modelos y requisitos. En este sentido, las guías y/o marcos de trabajo para el desarrollo de *software* son de vital importancia, porque permiten lograr un trabajo de manera organizada y de calidad, lo cual potencialmente ayudará a incrementar su efectividad y mejorar la satisfacción de los usuarios.

La Coordinación de Servicios de Cómputo del COLMEX tiene una demanda de desarrollo de *software*, que estimula la necesidad de implementar un marco de trabajo, ya que en la actualidad no se aplica una metodología, modelo o guía que ayude a entregar *software* en tiempo y forma. Asimismo, los programadores tienen tendencia en enfocarse en cuestiones técnicas, hablar de lenguajes de programación, bases de datos, entornos de desarrollo, pero con frecuencia restando importancia a temas de ingeniería de *software*, es decir, la manera en la que se debe de hacer el desarrollo de sistemas.

Cada día requerimos crear desarrollos en menor tiempo, en este sentido, las prácticas ágiles tienen aceptación en el proceso de desarrollo de productos de *software*, lo mismo que en la generación de conocimiento (Fuertes A. & Sepúlveda C., 2016).

En este estudio, se plantea una propuesta de intervención para establecer los lineamientos, integrar las mejores prácticas de forma estratégica con un *Framework* de desarrollo ágil, como *SCRUM*, y una guía de administración de proyectos, tal como *PMBOK*, para proponer un marco de trabajo ágil y formal de desarrollo de *software*. Se espera que este marco de trabajo permita que el equipo de desarrollo del COLMEX trabaje de manera más unificada bajo un mismo enfoque, con el objetivo de brindar entregables de valor en menor tiempo, mostrando avances visibles de las funcionalidades del sistema, documentación mínima necesaria con manuales de usuario y manteniendo calidad en el desarrollo de *software*.

3.2 Objetivo

El objetivo de la presente propuesta de intervención es proponer un marco de trabajo de desarrollo de *software* para la Coordinación de Servicios de Cómputo en el área de sistemas de El Colegio de México, basado en el *Framework* de *SCRUM* y la guía de proyectos *PMBOK*, con el fin de facilitar el desarrollo de productos y servicios de *software* con entregas de valor y de calidad, para los distintos usuarios (estudiantes, académicos, administrativos y proveedores externos del Colegio).

3.3 Definición del marco de trabajo *FAF*

FAF, se define en función de la guía de proyectos de *PMBOK*, con base al PMI, con el objetivo de brindar formalidad al proyecto con la realización mínima necesaria de documentación, para cumplir con los requerimientos y hacer entregas de calidad. Por otro lado, se hará uso del marco de trabajo de *SCRUM* considerando la terminología estandarizada por el organismo SCRUMstudy[10] publicada en la tercera edición de *SBOK* (Satpathy, 2017).

*FAF***,** es un marco de trabajo ágil y formal para el desarrollo de proyectos de *software*, enfocado para implementarse en El Colegio de México. Acopla buenas prácticas de *SCRUM* y *PMBOK*, haciendo un hibrido entre lo tradicional, debido a que considera las etapas de requerimientos, diseño, desarrollo y pruebas, y, por otro lado, lo ágil, es decir, entregas al usuario en periodos cortos de tiempo.

3.4 Principios

El marco de trabajo se basará en los 12 principios comúnmente adoptados del manifiesto ágil (Beedle & Bennekum, 2001). Los cuales se enuncia a continuación:

1. La prioridad es el cliente.
2. Bienvenidos los cambios en cualquier momento del desarrollo de *software*.
3. Entregas continuas de trabajo visible, por lo menos cada 2 semanas.
4. Todas las personas involucradas deben trabajar juntos durante todo el proyecto.
5. Mantener equipos motivados.
6. Comunicación presencial, cara a cara.
7. El trabajo visible es la medida de progreso.

[10] SCRUMstudy. Consultado el 14 de enero de 2020, para obtener más información, visite la página https://www.scrumstudy.com/

8. Trabajo sostenible. Mantener el mismo ritmo de trabajo durante todo el proyecto.
9. Mejora continua.
10. Esencial, hacer notar el trabajo no realizado.
11. Las mejores arquitecturas, requisitos y diseños surgen de equipos auto organizados.
12. Retroalimentación del equipo en intervalos regulares para ser más efectivos.

3.5 Roles

Se compone de cinco roles, los cuales, uno (cliente/usuario final) corresponde a la figura, que hace la solicitud del proyecto de desarrollo de *software* y puede ser cualquier persona u organización sin considerar un perfil específico. Los cuatro roles restantes serán los involucrados y responsables de hacer cumplir con los objetivos del desarrollo del proyecto. A continuación, se describen los cinco roles:

1. Cliente/Usuario Final (**C**): Es la persona u organización que especifica los requerimientos o establece la solicitud del desarrollo de *software*, además paga por el proyecto, si es que aplica. Se convierten en usuarios finales una vez que el *software* es liberado a producción. Dentro de este rol, también pueden estar los s*takeholders,* los cuales intervienen e influyen en el desarrollo del proyecto al proveer recursos y apoyo para que el proyecto pueda llevarse a cabo sin ningún problema. El cliente puede pertenecer a la organización o también puede ser externo.
2. *Product Owner (***PO***)*: Es la persona responsable de maximizar el valor del negocio para el proyecto (Satpathy, 2017). Prioriza los requerimientos, define las funcionalidades y revisa los entregables acordes a los criterios de aceptación. Se le conoce como la voz del cliente.
3. *SCRUM Master (***SM***)*: Es la persona quien modera y facilita las interacciones del equipo como coach técnico y motivador del mismo (Satpathy, 2017), además, facilita el desarrollo de las tareas y coordina al equipo de desarrollo. Este rol es responsable de que el equipo tenga un

ambiente de trabajo productivo, eliminando obstáculos y haciendo que se cumplan los principios, características y procesos de *SCRUM*. También hará las funciones de administrador del proyecto, es decir, coordinará las reuniones con el cliente y con el *Product Owner*. Adicional, supervisa las actividades de documentación y será el encargado de llevar una buena comunicación con el cliente.

4. *Team* (**T**): Consiste en un grupo de personas que trabajan en las historias de usuario en el *Sprint Backlog* para crear los entregables del proyecto (Satpathy, 2017). Las historias de usuario se desglosan en tareas de análisis, diseño, desarrollo, pruebas e implementación del sistema. Debe estar formado entre 2 y 10 personas, con el objetivo de tener una comunicación simple y fluida.
5. Documentador (**D**): Es la persona que apoyará en generar documentos que ayuden a darle formalidad al proyecto y al mismo tiempo, generar documentos técnicos que faciliten el entendimiento para el equipo de desarrollo.

En el cuadro 10 se definen características deseables de los cuatro roles que se requieren, para llevar a cabo el desarrollo del proyecto de *software*. Todos deben tener conocimientos del marco de trabajo de *SCRUM*.

Cuadro 10. Características deseables para los roles principales de SCRUM

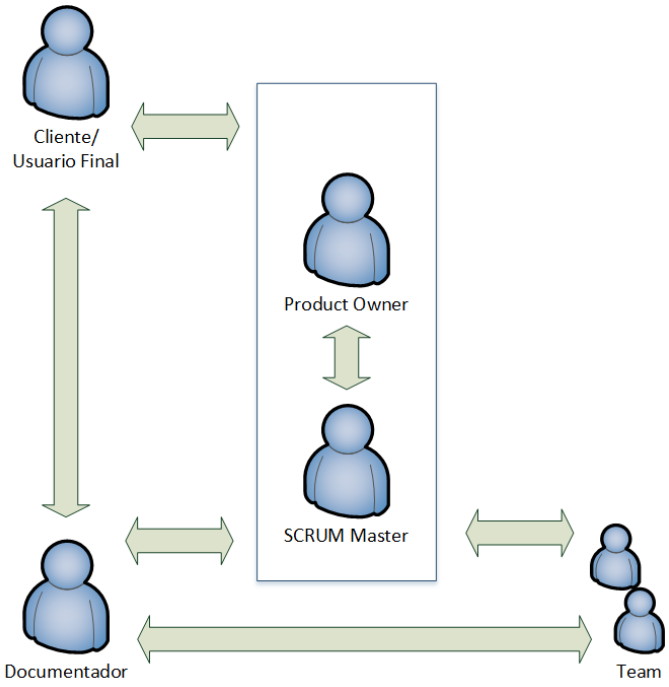
Product Owner	*SCRUM Master*	*Team*	Documentador
• Conocimiento de scrum • Conocimiento del negocio • Habilidad de comunicación • Habilidades de negociación • Accesible • Proactivo • Práctico	• Certificado en scrum *Master* • Lider servicial • Moderador • Habilidad para solucionar problemas • Accesible • Motivador • Perceptivo • Consejero • Habilidad de coordinación • Introspectivo	• Conocimiento de scrum • Colaborativos • Auto-organizados • Motivados • Proactivos • Expertos técnicos • Trabajo en equipo • Independientes • Intuitivos • Participativos • Introspectivos	• Conocimiento de scrum • Colaborativo • Habilidad de comunicación • Accesible • Proactivo • Perceptivo • Análitico • Organizado • Habilidad para hacer diagramas de flujo y de procesos

Fuente: Elaboración propia basado de Satpathy, 2017.

El flujo de comunicación es el siguiente; el cliente y/o usuario final revisará con el *Product Owner* y el *SCRUM Master* la solicitud de requerimientos, así como la validación de los impactos en el negocio y sobre todo los beneficios que generará el desarrollo del proyecto. El *Product Owner* estará en constante comunicación con el *SCRUM Master* para priorizar las tareas, así como definir el detalle de las actividades. El *SCRUM Master* coordinará las reuniones con todos los roles, revisará la documentación y las tareas del *Team*. El documentador en conjunto con el *Team* harán la documentación técnica, así como los documentos formales.

El flujo de comunicación entre todos los roles se muestra en la Figura 9.

Figura 9. Flujo de comunicación entre todos los roles participantes en FAF



Fuente: Elaboración propia

3.6 Reuniones

Se compone de nueve reuniones, en las cuales, se debe de respetar la duración (*time-boxing*[11]) de la reunión, estas se describen a continuación:

1. Reunión inicial con el cliente: El objetivo es obtener la solicitud de requerimientos, en el cual participan en conjunto el *Product Owner*, *SCRUM Master* y documentador. Esta reunión tiene un time-box de 2 a 4 horas.
2. *Meeting Backlog*: Reunión con el cliente, en conjunto con el *Product Owner*, *SCRUM Master* y documentador, con el objetivo de detallar los requerimientos y definir el *Product Backlog* priorizado. Esta reunión tiene un time-box de 4 horas.
3. *Release Planning*: Reunión con el equipo de trabajo, *SCRUM Master*, documentador y *Product Owner* para crear las historias de usuario y los criterios de aceptación. Esta reunión tiene un time-box de 2 horas.
4. *Sprint Planning*: Reunión con el equipo de trabajo, *SCRUM Master* y documentador para definir tareas específicas, estimación de tiempo de cada tarea definida y auto-asignación de recursos para crear el *Sprint* Backlog.

   Con los tiempos estimados se hará el cronograma de actividades. Esta reunión tiene un time-box de 8 horas considerando un *Sprint* de un mes.
5. *Daily Standup*: Reunión diaria con el *SCRUM Master* y *Team* con un time-box de 15 minutos para mencionar que se realizó, cuales son las actividades a realizar y cuáles son las problemáticas o impedimentos para llevar a cabo sus tareas. Esta reunión se debe hacer por las mañanas y de pie, por lo que el *Team* debe ser breve y conciso.
6. *Sprint Review*: Reunión con el cliente para revisar trabajo visible, es decir, demostrar y validar que lo que se pidió es lo que se está mostrando. Esta reunión tiene un time-box de 4 horas considerando un *Sprint* de un mes.

[11] *Scrum* introduce un concepto de *time-boxing* (o asignación de un bloque de tiempo), que propone la fijación de una cierta cantidad de tiempo para cada proceso y actividad en un proyecto *Scrum*. Esto garantiza que los miembros del equipo no ocupen demasiado o muy poco tiempo para un trabajo determinado, y que no desperdicien su tiempo y energía en un trabajo para el cual tiene poca claridad (Satpathy, 2017).

7. *Sprint Retrospective*: Reunión con el *Team* para retroalimentar lo bien y lo mal que se ha hecho en cada *Sprint*. Esta reunión tiene un time-box de 4 horas considerando un *Sprint* de un mes.
8. Reunión de cierre con el cliente: Tiene por objetivo presentar el proyecto de desarrollo finalizado, y al mismo tiempo generar el acta de cierre con lo que se da por finalizado el proyecto. Esta reunión tiene un time-box de 2 a 4 horas.
9. Retroalimentación: Reunión con todos los roles para revisar si se cumplió con los objetivos del proyecto y que se puede mejorar. El objetivo es generar un documento con las lecciones aprendidas con todos los roles participantes. Esta reunión tiene un time-box de 2 a 4 horas.

3.7 Entregables

Se deben de generar 11 documentos o entregables en el desarrollo de un proyecto de *software*.

1. Minuta de acuerdo inicial
2. *Project Charter*
3. *Product Backlog*
4. *Sprint Backlog*
5. Plan del proyecto
6. Registro de impedimentos
7. Control de Cambios
8. Minuta de acuerdos
9. Acta de Cierre
10. Minuta de acuerdo de Cierre
11. Lecciones aprendidas

3.8 Diagrama de proceso

Con base en la guía de *PMBOK* se seguirán las fases de Inicio, Planeación, Ejecución, Control y Monitoreo y Cierre. Dentro de cada una de ellas

estableceremos las técnicas empleadas por *SCRUM* y los documentos necesarios para el buen seguimiento del proyecto, la cual está conformada por 13 procesos.

Este modelo tiene cinco fases en su proceso las cuales se describen a continuación.

1. **Inicio:** Esta fase consiste en establecer los requerimientos que el *software* debe tener, por medio de una reunión con el cliente. Los involucrados en la reunión serán el cliente, stakeholders, *Product Owner*, *SCRUM Master* y documentador. Dicha reunión deberá ser guiada por el *SCRUM Master* en comunicación directa con el cliente, los demás solo podrán intervenir para dudas y consideraciones importantes.

   Durante la reunión, el documentador deberá elaborar la minuta de acuerdos, anotando a los participantes, puntos tratados y acuerdos establecidos. Adicional, el documentador en colaboración con el *SCRUM Master* deben elaborar el *Project Charter* del proyecto, el cual se considerará como parte inicial del contrato o compromiso a desarrollar. Es recomendable que se firme el *Project Charter* por parte del usuario, *Product Owner* y *SCRUM Maste*r. Por último, se lleva a cabo una reunión (*Meeting Backlog*) para definir los detalles de los requerimientos solicitados y las prioridades por parte del *Product Owner* para definir el *Product Backlog* priorizado.
2. **Planeación:** En esta fase, se lleva a cabo una reunión (*Release Planning*), con el *Product Owner*, documentador y moderada por el *SCRUM Master.* Todos los integrantes participan por igual, generando lluvia de ideas, con el objetivo de crear historias de usuario y definir los criterios de aceptación, es decir, el listado de requerimientos necesarios para la elaboración del *software*.

   Después, se lleva a cabo una reunión (*Sprint Planning*) donde participan *SCRUM Master* y el equipo de desarrollo. De acuerdo al *Product Backlog* priorizado, se detallan las tareas necesarias para la elaboración de los requerimientos, asignando recursos y tiempos. El documentador redacta el *Sprint Backlog* con el objetivo de establecer fechas de seguimiento para los *Sprints* siguientes. Para finalizar, con el detalle de las tareas se establece el

plan del proyecto, donde todos los roles están enterados y comprometidos con sus actividades.

3. **Ejecución**: En esta fase, se llevan a cabo todas las tareas establecidas en el *Sprint Backlog* y en el plan del proyecto.

   Todos los días debe haber una reunión (*Daily Standup*) de 15 minutos con el equipo de desarrollo, para responder a 3 preguntas, ¿Qué hiciste ayer?, ¿Qué vas hacer hoy?, ¿Qué problemas tienes? Adicional, el *SCRUM Master* debe ir actualizando o ajustando el cronograma dentro del plan del proyecto, así como, actualizar el tablero (*Scrumboard*) en un lugar visible para el conocimiento del equipo de desarrollo y facilitar la comunicación para todos los miembros del equipo. El *SCRUM Master* debe llevar el seguimiento con un grafica (*Sprint Burdown*) para registrar las horas pendientes de trabajo.

4. **Seguimiento y Control:** En esta fase, se lleva a cabo la revisión de los avances del desarrollo de software, así como también, determinar si deben hacer cambios en las prioridades y funcionalidades del sistema.

   Cada que se concluye un *Sprint*, es necesario hacer una reunión (*Sprint Review*) guiada por el *SCRUM Master,* en conjunto con el cliente y *Team,* para revisar y validar las funcionalidades del sistema.

   Si en algún momento del desarrollo, hay cambios o adiciones por parte del cliente, estos se deben de registrar en un formato de control de cambios. Asimismo, si es conveniente se pueden hacer reuniones con el cliente de no más de 30 minutos, con su correspondiente minuta de acuerdos.

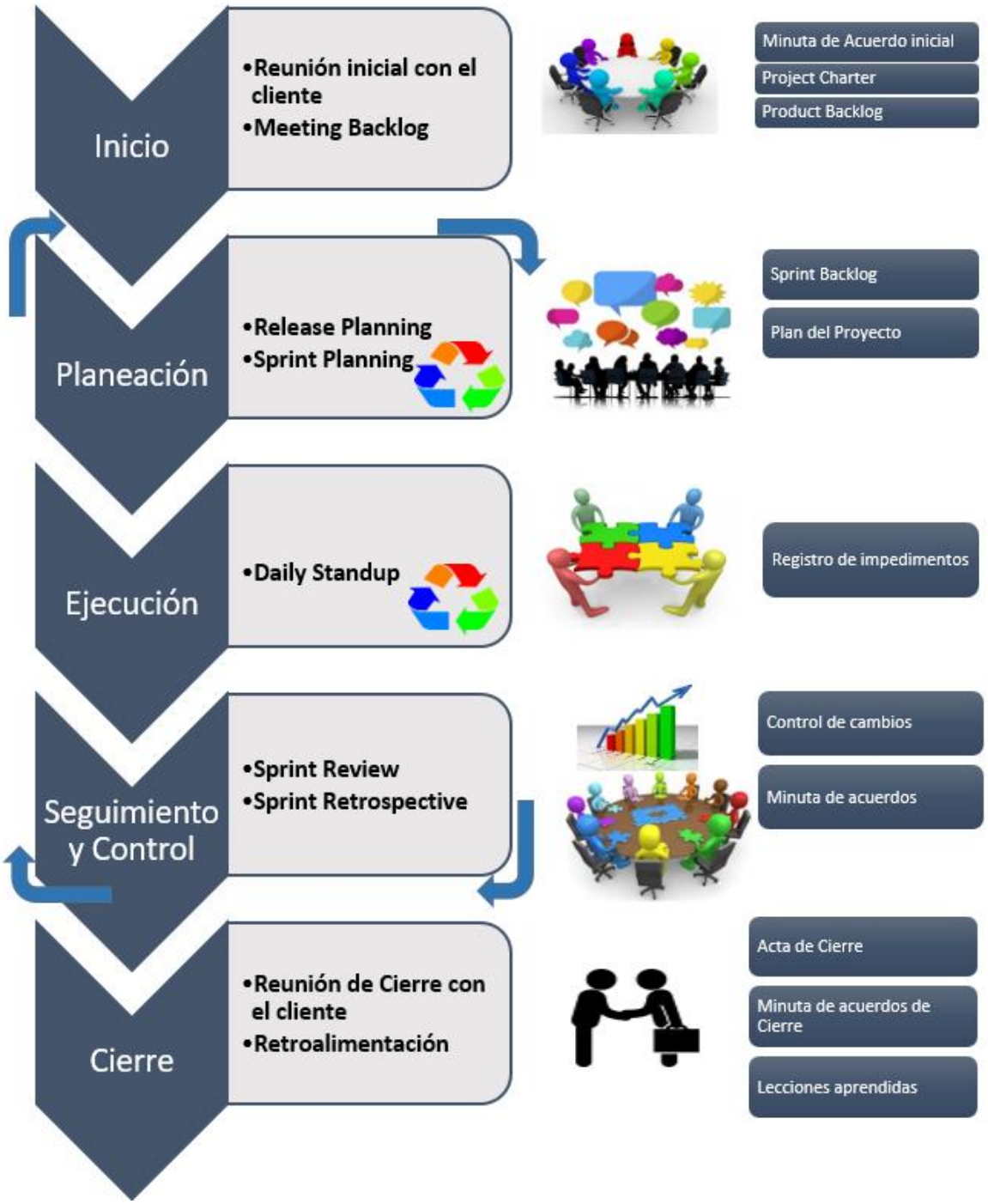
   Para concluir esta fase, se realiza una reunión (*Sprint Retrospective*), con todos los miembros del equipo, para revisar que se puede mejorar (aspectos buenos y malos) del proceso de desarrollo.

5. **Cierre**: En esta fase, se lleva a cabo una reunión con el cliente para dar por terminado el desarrollo del sistema solicitado, y firmar un acta de cierre del proyecto.

   De forma adicional, debe de haber una reunión de retroalimentación con todos los miembros del equipo para revisar lo sucedido durante todo el desarrollo del proyecto y registrar las lecciones aprendidas.

En la Figura 10, podemos apreciar el proceso general de *FAF*.

Figura 10. Proceso general de FAF



Fuente: Elaboración Propia.

Los 13 procesos que se tienen que aplicar a cualquier proyecto de desarrollo de *software* se muestran en el cuadro 11, así como las reuniones, entregables y responsables de cada proceso.

Cuadro 11. Procesos de FAF

Fase	Proceso	Reunión	Entregable/Responsable
Inicio		1. Inicial con cliente (**C, PO, SM, D**) TB: **2-4 horas**	1. Minuta de acuerdo inicial / **D**
	1. Desarrollar project Charter		2. Project Charter / **D, SM, PO**
	2. Crear *Product Backlog*	2. Meeting *Backlog* (**C, PO, SM, D**) TB: **4 horas**	3. *Backlog* priorizado / **D, SM, PO**
Planeación	3. Crear historias de usuario	3. Release Planning (**SM, D ,T, PO**) TB: **2 horas**	3. *Backlog* H.U. / **D, SM, PO**
	4. Especificar tareas	4. *Sprint* Planning (**SM, D, T**) TB: **2 horas**	4. *Sprint Backlog* / **D, SM, T**
	5. Estimar y asignar tareas		
	6. Crear *Sprint Backlog*		
	7. Crear el plan del proyecto		5. Plan del proyecto / **D, SM**
Ejecución	8. Desarrollar tareas		/ **T**
	9. Realizar Daily Standup	5. Daily Standup (**SM, T**) TB: **15 min.**	6. Registro de impedimentos / **SM**
Seguimiento y Control	10. Demostrar y validar *Sprint*	6. *Sprint* Review (**PO, SM, D, T**) TB: **2 horas**	7. Control de Cambios / **D**
			8. Minuta de acuerdos / **D**
	11. Retrospectiva del *Sprint*	7. *Sprint* Retrospective (**SM, D, T**) TB: **2 horas**	/ **D, SM, T**
Cierre	12. Presentación y entrega	8. Cierre con cliente (**C, PO, SM, D**) TB: **2-4 horas**	9. Minuta de acuerdo de cierre / **D**
			10. Acta de cierre / **D**
	13. Retroalimentación	9. Retroalimentación (**C, PO, SM, D, T**) TB: **2 -4 horas**	11. Lecciones aprendidas / **D**

Fuente: Elaboración propia

Enseguida se detallan los **13 procesos de *FAF***, mencionando técnicas y herramientas para llevar a cabo las actividades correspondientes.

1. Desarrollar *Project Charter*. En este proceso se revisa el caso de negocio del proyecto para crear los objetivos, alcances y riesgos del proyecto, que servirá como base y, además, proporcionará un enfoque general para todo el proyecto. Así mismo, se debe identificar a los *stakeholders*, el *Product Owner*, *SCRUM Master* y el *Team* que va a participar en el desarrollo del proyecto. El *Project Charter* debe estar definido por el *Product Owner* en conjunto con el *SCRUM Master* y escrito por el documentador. En el Anexo II del presente trabajo, se muestra la plantilla propuesta para este proceso.
2. Crear *Product Backlog*. En este proceso se genera un listado de los requerimientos y/o funcionalidades generales, los cuales se priorizan para crear el *Product Backlog* priorizado. Es importante considerar que el *Product Backlog* nunca va estar completo, este cambiará a medida que el desarrollo vaya avanzando (Schwaber & Sutherland, 2013). En el Anexo III del presente trabajo, se muestra la plantilla propuesta para este proceso.
3. Crear historias de usuario. En este proceso se crean las historias de usuario y los criterios de aceptación por parte del *Product Owner*. Una historia de usuario incluye tres elementos sobre el requerimiento: ¿Quién? ¿Qué? y ¿Por qué? Los requerimientos expresados en las historias de usuario son oraciones breves, sencillas y fáciles de entender (Satpathy, 2017). El formato es el siguiente: Como <rol> yo debería <requerimiento> a fin de <beneficio>. Los criterios de aceptación deben estar alineados de forma explícita a las historias de usuario y deben ser entendibles, ya que son de suma importancia para la entrega eficaz y oportuna de la funcionalidades definidas, lo cual, determinan el éxito del proyecto (Satpathy, 2017).
4. Especificar tareas. En este proceso, las historias de usuario se detallan por parte del *Team* en tareas específicas, es decir, a un entendimiento

técnico. Se debe considerar que cada tarea no debe llevar más de 4 horas en realizarse, en caso de que la tarea se estime en más de 4 horas, se sugiere dividir la tarea.

5. Estimar y asignar tareas. En este proceso, el *Team* estima el esfuerzo o tiempo requerido para completar una tarea. Luego, se auto-asignan las tareas y se comprometen a entregarlas en el *Sprint* correspondiente. Se recomienda utilizar horas como unidad de tiempo.
6. Crear *Sprint Backlog.* En este proceso, el *Team* determina la cantidad de tareas que se llevaran a cabo en los diferentes *Sprints* para ser completadas.
7. Crear el plan del proyecto. En este proceso se realiza el cronograma de actividades, en función de las historias de usuario priorizadas en el *Product Backlog*, así como las tareas correspondientes.
8. Desarrollar tareas. En este proceso, el *Team* realiza las tareas del *Sprint* correspondiente. Para dar seguimiento a las tareas, se utiliza un *Scrumboard*, el cuál puede estar definido por 3 columnas; pendiente, en progreso, y terminado como se muestra en la figura 11.

Figura 11. Tablero Scrumboard

Pendiente	En progreso	Terminado
Tarea 1		
	Tarea 2	
Tarea 4		Tarea 3
.		
.		
.		
Tarea n		

Fuente: Elaboración propia

9. Realizar *Daily Standup*. En este proceso, el *SCRUM Master* lleva a cabo una reunión diaria, de preferencia de pie, para que los integrantes del equipo se actualicen de lo que están haciendo los demás ya sea en su

avance o en los impedimentos o problemas que han tenido. Se deben de contestar 3 preguntas:

- ¿Qué hiciste ayer?
- ¿Qué vas hacer hoy?
- ¿Qué problemas o impedimentos tienes para llevar a cabo tus actividades?

En caso de que haya problemas, el *SCRUM Master* debe de anotarlo en el documento de registro de impedimentos. En el Anexo IV del presente trabajo, se muestra la plantilla propuesta.

10. Demostrar y validar *Sprint*. En este proceso, el *Team* muestra los entregables del *Sprint* al *Product Owner* y al usuario, supervisados por el *SCRUM Master*. El objetivo es presentar trabajo visible con el propósito de obtener la aprobación y aceptación del *Product Owner*. En esta reunión, también se puede revisar la prioridad de las actividades, así como cualquier cambio o actualización al *Product Backlog*. El documentador debe redactar una minuta con los acuerdos generados.

    En el Anexo V del presente trabajo, se muestra la plantilla propuesta para la redacción de minutas.

11. Retrospectiva del *Sprint*. En este proceso, el *SCRUM Master* y el *Team* analizan las lecciones aprendidas durante el *Sprint*, una vez que este ha terminado. El objetivo es revisar que se está haciendo bien, que se puede mejorar respecto a las personas, procesos y herramientas.

12. Presentación y entrega. En este proceso, el desarrollo del sistema es presentado al usuario y al *Product Owner*, con la finalidad de dar por terminado el proyecto, y al mismo tiempo generar el acta de cierre del proyecto.

    En el Anexo VI del presente trabajo, se muestra la plantilla propuesta para este proceso.

13. Retroalimentación. En este proceso, se reúnen todos los que participaron en el proyecto, con el objetivo de registrar las lecciones aprendidas durante el desarrollo del proyecto.

Con el objetivo de facilitar la realización de los procesos, en el cuadro 12 se muestran algunas herramientas que pueden ser de utilidad en la creación de entregables o desarrollo de tareas.

Cuadro 12. Herramientas de apoyo para los procesos de FAF

Núm.	Proceso	Herramientas
1	Desarrollar *project Charter*	*FODA*, *Join Application Design* (*JAD*), *SMART*, Cuestionario de estilos de aprendizaje (CEA), entrevistas al usuario, Matriz de poder
2	Crear *Product Backlog*	*Brainstorming*, 6 sombreros, priorización Moscow, comparación por pares, método de los 100 puntos
3	Crear historias de usuario	Cuestionarios, *Mockups*, talleres de historias de usuario, *INVEST*
4	Especificar tareas	Técnicas de descomposión, Tipos de dependencias
5	Estimar y asignar tareas	*Wideband Delphi*, *Planning Poker*, puño de cinco, estimación por afinidad, *PERT* de 3 valores
6	Crear *Sprint Backlog*	*Jira*, *Vivify*, *Asana*, *Trello*
7	Crear el plan del proyecto	*Microsoft Project*, *Open Project*
8	Desarrollar tareas	*Jira*, *Vivify*, *Asana*, *Trello*, *Git*, *ButBucket*, *Team Foundation*
9	Realizar *Daily Standup*	Reunión presencial, videoconferencias, *Skype*, *Microsoft Teams, BlueJeans, Gotomeeting, Zoom*
10	Demostrar y validar *Sprint*	Reunión mostrando el *software* funcionando.
11	Retrospectiva del *Sprint*	Análisis del valor ganado, Retroalimentación de 360 grados.
12	Presentación y entrega	Reunión presencial, videoconferencias, *Skype, Microsoft Teams, BlueJeans, Gotomeeting, Zoom*
13	Retroalimentación	ECVP (Explorador, Comprador, Vacacionista, Prisionero)

Fuente: Elaboración propia.

La propuesta de intervención planteada es un marco de trabajo, que integra elementos tanto de *SCRUM* como de *PMBOK*, y no una metodología o modelo, es decir, se puede usar lo que considere pertinente para el desarrollo del *software* e inclusive plantear adiciones. Se debe recordar que el objetivo es facilitar el trabajo y lograr la satisfacción del cliente y/o usuario final.

Asimismo, se hace énfasis en integrar dentro del equipo de trabajo a los documentadores, ya que servirán de apoyo para reducir tiempos. Con la documentación se podrá atender mantenimientos, posteriores al lanzamiento a producción, además, dichos documentos servirán como transferencia del

conocimiento para todo el equipo de desarrollo. Una documentación incompleta genera una rápida degradación y envejecimiento del *software* (Cadavid, 2013).

3.9 Calidad en *FAF*

Generalmente se considera que un desarrollo es de calidad cuando el usuario está satisfecho con el resultado final, en términos de valor aportado. Ahora bien, ¿cómo lograr un desarrollo de software de calidad?

De acuerdo con la guía *SBOK*, la calidad es la capacidad con la que cuenta un producto terminado, o los entregables, para cumplir con los criterios de aceptación y lograr el valor del negocio que espera el cliente (Satpathy, 2017).

FAF, de forma implícita adopta la mejora continua, es decir, cada que finaliza un *Sprint*, el *Team* revisa, analiza y continúa haciendo las actividades que influyeron en los buenos resultados. Por lo general, esto se realiza en ciclos de 2 semanas, lo cual ayuda a identificar riesgos, oportunidades, cambiar prioridades y, por consiguiente, evitar que se afecte la calidad del proyecto.

Por otra parte, durante el desarrollo de tareas de codificación, se hacen pruebas, con el objetivo de detectar errores y/o defectos en la funcionalidad. Esto fomenta el aprendizaje con base a la práctica y al mismo tiempo, se adquiere conocimiento para mejorar. También el papel del documentador es fundamental, ya que debe mantener actualizados los diagramas de procesos, el manual de usuario y técnico, considerando todos los cambios requeridos en cada *Sprint*. Esto asegura que la calidad sea inherente en los requerimientos solicitados por el usuario.

En cualquier desarrollo de software, siempre se debe pensar en mitigar la brecha entre lo que espera el usuario y lo que estamos entregando. Es primordial informar, cuando no es posible lograr el alcance solicitado; por ejemplo, sí no se tiene la capacidad de recursos necesarios o cuando hay dependencia con proveedores que influyan de forma directa en los tiempos de entrega.

De igual forma, las historias de usuario y los criterios de aceptación tienen que ser claros, es decir, se recomienda comprender específicamente qué es lo que

espera el usuario, por lo tanto, la definición de un requerimiento “terminado” debe contemplar lo siguiente:

- Completar las pruebas de funcionalidad acorde a lo solicitado.
- Conclusión de la documentación, relacionada con la historia de usuario entregada.
- Corrección de todos los errores que hayan surgido.
- Validación y aprobación del requerimiento por parte del usuario.
- Satisfacción del usuario en la entrega de los requerimientos.

El usuario o cliente siempre va a ser la persona primordial en cualquier proyecto de desarrollo de software, por lo que es esencial entender sus necesidades y requerimientos (Satpathy, 2017).

3.10 Recomendaciones para implementar *FAF*

Para que el marco de trabajo propuesto funcione de manera efectiva se recomienda desarrollar las siguientes actividades:

- Capacitar a todo el equipo de trabajo en el marco de trabajo *SCRUM*[12] y en los elementos básicos de *PMBOK*, exponiendo con claridad los beneficios y ventajas que presentan en el desarrollo de *software*.
- Dar a conocer la nueva forma de trabajo a los usuarios que realizan solicitudes de desarrollo de *software*, explicando en qué consisten los cambios en los procesos y las mejoras que se buscan obtener.
- Implementar *FAF* de forma gradual en los proyectos de desarrollo de *software*, con el propósito de obtener retroalimentación por parte de los usuarios, y así concentrar opiniones o ideas para mejorar los desarrollos o el propio marco de trabajo.

[12] *SCRUMstudy*. El curso de los fundamentos *SCRUM* puede hacerlo en *SCRUMstudy* de forma gratuita.

- Ubicar al equipo de sistemas de el COLMEX en una misma zona física, lo cual puede favorecer una mejor comunicación entre los integrantes del equipo. Se busca, con ello, generar una mayor confianza entre sus miembros.
- Considerar de alta importancia los requerimientos priorizados en primer orden, ya que estos son los de mayor valor para el usuario. No olvidar que la calidad y el valor están estrechamente vinculados.
- Utilizar herramientas colaborativas que den visibilidad del trabajo, por ejemplo, *Jira, Vivify, Asana,* o *Trello,* así como formatos similares a los mencionados en los Anexos II, III, IV, V, y VI del presente trabajo.
- Establecer una estrategia de mantenimiento y revisión continúa del marco de trabajo *FAF*, con el fin de que este se ajuste a las nuevas necesidades del área u organización.

3.11 Beneficios esperados

Al implementar la propuesta de intervención en el COLMEX los beneficios esperados son los siguientes:

- Mayor flexibilidad a cambios y adaptación a las necesidades de los usuarios finales. Posibilidad de disminuir los tiempos de desarrollo.
- Mitigación de riesgos e identificación de cambios anticipados antes de la implementación en producción.
- Identificación de mejoras para mantener sistemas más efectivos y amigables para el usuario.
- Fortalecimiento de la comunicación y relaciones interpersonales del equipo de trabajo.
- Posibilidad de mayor motivación del equipo de trabajo. Adoptar un nuevo esquema de trabajo, puede ayudar a concretar nuevos retos para el personal del área, favoreciendo su crecimiento profesional y al mismo tiempo, ofreciendo productos y servicios de calidad.

Hoy en día, la metodología que se aplica en el COLMEX es secuencial, y con frecuencia no se sigue conforme a lo definido en procesos de calidad para el desarrollo de *software*, por lo que suele convertirse en trabajo adicional que aporta bajo valor. En este sentido, el presente marco de trabajo busca ayudar a mantener una comunicación fluida entre los integrantes del equipo, detectar los riesgos de forma oportuna, y lo más importante, adoptar este marco de trabajo como parte de las actividades diarias en el desarrollo de *software*.

La presente propuesta de intervención busca que su adopción se pueda realizar con facilidad y flexibilidad en el COLMEX, a fin de que sea el medio de trabajo del día a día para el personal de desarrollo de la Coordinación de Servicios de Cómputo.

Finalmente, se considera que esta propuesta es susceptible de adaptarse en organizaciones académicas similares al COLMEX (Centros de investigación, universidades, institutos de educación superior, entre otros) que desarrollen proyectos de TI internos, en especial de desarrollo de *software*, y que busquen formalizar y mejorar dicha actividad.

Conclusiones

Conclusiones

Las metodologías ágiles para el desarrollo de *software* son herramientas que pueden ayudar a organizar, administrar y controlar con mayor efectividad las actividades, a fin de implementar un sistema de forma correcta.

Con la generalización de las TICS en las actividades organizacionales, es primordial que una institución educativa y de investigación, impulse su uso efectivo y estratégico, a fin de que estas puedan ser un factor que otorgue valor a la organización. En este caso, las áreas de tecnologías de información, a partir del esfuerzo generado, pueden tomar también un papel más protagónico dentro de la operación institucional, elevando el número de los proyectos y servicios realizados en el tiempo y costo planeados en un inicio, cumpliendo con las necesidades y expectativas de los usuarios y entregando productos de calidad.

Se considera, entonces, que el uso de modelos de trabajo, ya sea metodologías, normas, estándares, *Frameworks* o guías, en el desarrollo de *software*, son indispensables en cualquier tipo de organización, para ayudar a lograr la satisfacción del cliente o usuario final.

En el presente trabajo, después de revisar la teoría, estudios de comparación, y haber realizado una propuesta de intervención, se considera que la guía de proyectos *PMBOK* y el *Framework* de *SCRUM,* pueden ser complementarios para facilitar el desarrollo de *software*, con el objetivo de que se entreguen en menor tiempo y costo, con mayor satisfacción de los usuarios.

Se puede establecer que no existe una metodología global o universal para implementarse en cualquier proyecto de desarrollo de *software*, debido a que cada una debe de ser aplicada de acuerdo al contexto de la organización, considerando factores como recursos humanos, tiempo, presupuesto, condiciones laborales, ambiente de trabajo e infraestructura. Todas las metodologías tienen ventajas que se pueden aprovechar al máximo, pero también desventajas, lo conveniente es utilizar lo que más se ajuste a esta organización, aunque, en principio, debe otorgarse valor al usuario final, pues es quien se beneficiará de este marco de trabajo.

En gran medida, el éxito de un proyecto se debe a una buena planeación; lo valioso es tomar las mejores decisiones, sin olvidar el triángulo de la calidad, es decir, considerar alcance, tiempo y costo, siempre enfocándonos a no afectar los requerimientos del usuario.

FAF, la propuesta de marco de trabajo expuesta en este estudio, cumple con una combinación de elementos de la guía de proyectos de *PMBOK* y el *Framework* de *SCRUM*, por lo tanto, es un marco híbrido, que ofrece una descripción breve y concisa acerca de la manera en que se debe de hacer en el COLMEX, ajustándose a la estrategia de la Coordinación de Servicios de Cómputo para realizar desarrollos dentro del triángulo de la calidad, y sobretodo, buscando mayor valor y satisfacción en los usuarios finales.

Es necesario mencionar que el personal que participe en la integración de *FAF* deben ser capacitados en *SCRUM* y en *PMBOK*, por lo que será indispensable un plan de capacitación. Esto podrá traducirse en una mejor comunicación, confianza y cohesión con todos los miembros del equipo.

Los cambios implican esfuerzo, de igual modo, potencial resistencia y, por consiguiente, mayor tiempo en las actividades, por lo que, al principio de la integración de un marco de trabajo, como el presentado, es posible que se aumenten las tareas, sin embargo, conforme se vaya adaptando la nueva propuesta, los resultados se podrán plasmar en mejores entregas de valor.

Esta propuesta de intervención tiene como característica que se ajusta a las necesidades del COLMEX, pues, aunque la institución no tiene por objetivo producir desarrollo de *software*, al implementar un marco de trabajo ágil, bajo un esquema flexible y dinámico, puede facilitar y mejorar actividades institucionales, administrativas y académicas. En este sentido, la presente propuesta, está dirigida a apoyar la gestión de proyectos de desarrollo de la Coordinación de Servicios de Cómputo.

Por último, la bibliografía especializada muestra que es factible combinar varias metodologías de desarrollo de *software* de forma exitosa. Otro *Framework* ágil que podría complementar esta propuesta de intervención es la programación extrema (*XP*), por lo que en futuros trabajos se puede ampliar dicha investigación a

fin de considerar a *FAF* como el marco de trabajo para la gestión de proyectos de *software* en grupos pequeños, y *XP*, como la definición de prácticas técnicas más específicas dirigidas al equipo de programadores.

En suma, se considera que este marco de trabajo es una contribución a la consolidación de áreas y equipos de desarrollo de software en instituciones, como el COLMEX, con el propósito final de apoyar las actividades organizacionales y administrativas de manera más efectiva y, por lo tanto, posibilitando un impacto positivo en las actividades sustantivas, docencia e investigación.

Bibliografía

Ahmad Khan, Z. (2014). *Scrumban-adaptive agile development process: using scrumban to improve software development process. Master's Tesis*. Finland. Tomado de https://www.theseus.fi/bitstream/handle/10024/77014/Khan_Zahoor.pdf?sequence=1

Alahyari, H., Berntsson Svensson, R., & Gorschek, T. (2017). A study of value in agile software development organizations. *Journal of Systems and Software*, *125*, 271–288. https://doi.org/10.1016/J.JSS.2016.12.007

Alavandhar, J. V., & Ṇikiforova, O. (2017). Several Ideas on Integration of SCRUM Practices within Microsoft Solutions Framework. *Applied Computer Systems*, *21*(1). https://doi.org/10.1515/acss-2017-0010

Awad, M. A. (2005). *A Comparison between Agile and Traditional Software Development Methodologies*. Australia. Tomado de http://pds10.egloos.com/pds/200808/13/85/A_comparision_between_Agile_and_Traditional_SW_development_methodologies.pdf

Beedle, M., & Bennekum, A. van. (2001). Manifesto for Agile Software Development. Tomado de http://agilemanifesto.org/

Briscoe, T. J., & Staikoff, R. L. (2006). *PMBOK® guide versus SCRUM mastery: points of convergence and divergence*. Paper presented at PMI® Global Congress 2006—North America, Seattle, WA. Newtown Square. Tomado de https://www.pmi.org/learning/library/pmbok-guide-scrum-convergence-divergence-8089

Cadavid, A. N. (2013). Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*, *11*(2), 30. https://doi.org/10.15665/rp.v11i2.36

Cockburn, A. (2000). Selecting a project's methodology. *IEEE Software*, *17*(4), 64–71. https://doi.org/10.1109/52.854070

CollabNet, V. (2019). 13th Annual State Of Agile Report. Tomado de https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508

Diccionario Español Mexicano (DEM). (2018). Tomado de http://dem.colmex.mx/

Fuertes A., Y., & Sepúlveda C., J. (2016). Scrum, Kanban and Canvas in the commercial, industrial and educational sector - A literature review. *Revista Antioqueña de Las Ciencias Computacionales*, *6*(1), 46–50. Tomado de http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=117348344&lang=es&site=ehost-live&scope=site

Gonçalves, E. F., Drumond, G. M., Méxas, M. P., & Méxas, M. P. (2017). Evaluation of PMBOK and scrum practices for software development in the vision of specialists. *Independent Journal of Management & Production*, *8*(5), 569–582. https://doi.org/10.14807/ijmp.v8i5.598

Iacovelli, A., & Souveyet, C. (2008). Framework for Agile Methods Classification. In *CEUR Workshop Proceedings* (Vol. 341, pp. 91–102).

Jaziri, R., El-Mahjoub, O., & Boussaffa, A. (2018). Proposition of A Hybrid Methodology of Project Management. *American Journal of Engineering Research*, *7*, 113–127. Tomado de https://www.researchgate.net/publication/324517405_Proposition_of_A_Hybrid_Methodology_of_Project_Management

Jimenez Hernández, M., & Orantes Jimenez, S. (2011). META: a new hybrid methodology to software development created to suit the current needs in Mexico for ICTA 2011. *ICTA*. Tomado de http://www.iiis.org/cds2011/cd2011idi/icta_2011/paperspdf/ct002wn.pdf

Khurana, H., & Sohal, D. J. S. (2011). Agile: The Necessitate Of Contemporary Software Developers. *International Journal of Engineering Science and Technology*, *3*(2), 1031–1039. Tomado de http://www.ijest.info/docs/IJEST11-03-02-181.pdf

Letelier, P., & Penadés, M. C. (2006). Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Ciencia y Técnica Administrativa*, *5*(26). Tomado de http://www.cyta.com.ar/ta0502/v5n2a1.htm

Matos, S., & Lopes, E. (2013). Prince2 or PMBOK – A Question of Choice. *Procedia Technology*, *9*, 787–794. https://doi.org/10.1016/j.protcy.2013.12.087

PMI. (2017). *A guide to the Project Management of Body of Knowledge (PMBOK ®*

Guide) (Sixth Ed.). Project Management Institute, Inc.

Pressman, R. S. (2002). *Ingeniería del software Un enfoque práctico*. (McGRAW-HILL, Ed.) (Quinta Edición). Tomado de www.FreeLibros.org

Pressman, R. S., & Maxim, B. R. (2015). *SOFTWARE ENGINEERING: A PRACTITIONER'S APPROACH.* (EIGHTH EDITION). New York, NY: McGraw-Hill. Tomado de https://downloadnema.com/wp-content/uploads/2017/02/Software Engineering A Practitioner's Approach eighth edition-(www.downloadnema.com).pdf

Pwc. (2012). Insights and Trends: Current Portfolio, Programme, and Project Management Practices. Tomado de https://mosaicprojects.com.au/PDF/pwc-global-project-management-report-2012.pdf

Quintero, J. B., & Anaya, R. (2007). MDA Y EL PAPEL DE LOS MODELOS EN EL PROCESO DE DESARROLLO DE SOFTWARE. *Revista EIA*, (8), 131–146. Tomado de http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1794-12372007000200011

Ries, E. (2011). *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. New York: Crown Business.

Santiago Benítez, G., Caballero Álvarez, R., Gómez Mayén, D., & Cuevas, A. D. (2013). El uso didáctico de las TIC en escuelas de educación básica en México. *Revista Latinoamericana de Estudios Educativos*, *XLIII*(3), 99–131.

Satpathy, T. (2017). *Una guía para el cuerpo de conocimiento de SCRUM (SBOK)*. (SCRUMstudy, Ed.) (3ra ed.). Arizona, USA: SCRUMstudyTM, una marca de VMEdu, Inc.

Schwaber, K., & Sutherland, J. (2013). La Guía de Scrum La Guía Definitiva de Scrum: Las Reglas del Juego. Tomado de http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf

The Standish Group. (2015). *The Standish International: CHAOS Manifesto 2015*. Tomado de https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

Anexos

Anexos

Anexo I – Formulario de Iacovelli

Formulario 1: Uso

Marque con una “X” Verdadero (V) o Falso (F) de cada uno de los siguientes puntos, considerando cuando participa en el desarrollo de un proyecto de *software* en el COLMEX.

- A1. Se cumplen las fechas de entrega (V)(F)
- A2. Se cumple con los requisitos solicitados por el cliente (V)(F)
- A3. El *software* es de calidad (V)(F)
- A4. Satisfacción del usuario final (V)(F)
- A5. Adaptación a cualquier tipo de entorno (V)(F)
- A6. Flexibilidad para adaptarse a otros entornos de trabajo (*outsourcing*) (V)(F)
- A7. Aumento en la productividad (V)(F)

Formulario 2: Capacidad de agilidad

Marque con una “X” Verdadero (V) o Falso (F) de cada uno de los siguientes puntos, considerando como hace el trabajo durante el desarrollo de un proyecto de *software*.

- B1. Iteraciones cortas (entregas en menos de un mes) (V)(F)
- B2. Colaboración con su equipo de trabajo (V)(F)
- B3. Centrado en las personas, más que en los procesos (V)(F)
- B4. Política de refactorización (simplificar y mejorar el código) (V)(F)
- B5. Política de pruebas (Probar el sistema durante todo el desarrollo) (V)(F)
- B6. Facilidad para integrar cambios durante el desarrollo del proyecto (V)(F)
- B7. Los procesos son simples y sencillos (V)(F)
- B8. Los requisitos funcionales pueden cambiar (V)(F)

- B9. Los requisitos no funcionales pueden cambiar (V)(F)
- B10. El plan de trabajo puede cambiar (V)(F)
- B11. Los recursos humanos pueden cambiar (V)(F)
- B12. Los indicadores pueden cambiar (métricas de objetivos) (V)(F)
- B13. En qué fase del desarrollo tiene mayor capacidad de reacción ante cualquier cambio
    a) Inicio del proyecto
    b) Cada que se concluye una etapa (análisis, diseño, pruebas, etc.)
    c) Cada iteración, es decir al concluir una serie de tareas en un tiempo máximo de 2 semanas
- B14. El intercambio de conocimiento con el equipo de trabajo es:
    a) Alto
    b) Bajo

Formulario 3: Aplicabilidad

De forma general, indique con una “X” como es el entorno y los proyectos de desarrollo de *software* en el COLMEX.

- C1. Tamaño de los proyectos (Pequeño) (Grande)
- C2. Complejidad de los proyectos (Baja) (Alta)
- C3. Riesgos del proyecto (Baja, Alta)
- C4. Tamaño del equipo de desarrollo (Pequeño) (Grande)
- C5. Grado de interacción con el cliente (Baja) (Alta)
- C6. Grado de interacción con los usuarios finales (Baja) (Alta)
- C7. Grado de interacción entre los miembros del equipo (Baja) (Alta)
- C8. Grado de integración con tecnologías nuevas (Baja) (Alta)
- C9. Como es la organización del equipo (Auto-organizados) (Jerárquico)

Formulario 4: Procesos y productos

Marque con una “X” Verdadero (V) o Falso (F) de cada uno de los siguientes puntos, considerando el proceso que se utiliza en el desarrollo de un proyecto de *software* en el COLMEX.

Nivel de abstracción de las normas y directrices.

- D1. Gestión de proyectos (V)(F)
- D2. Descripción de procesos (V)(F)
- D3. Se aplican reglas y lineamientos al realizar actividades (V)(F)

Actividades cubiertas por el método ágil.

- D4. Project Charter (V)(F)
- D5. Definición de requerimientos (V)(F)
- D6. Modelado de base de datos (V)(F)
- D7. Código (V)(F)
- D8. Pruebas unitarias (V)(F)
- D9. Pruebas de integración (V)(F)
- D10. Pruebas del sistema (V)(F)
- D11. Pruebas de aceptación (V)(F)
- D12. Control de calidad (V)(F)
- D13. Los sistemas son liberados a producción (V)(F)

Productos de las actividades del método.

- D14. Se hacen modelos de diseño (V)(F)
- D15. Comentarios al código fuente (V)(F)
- D16. Se generar paquetes .EXE (V)(F)
- D17. Entregables de pruebas unitarias (V)(F)
- D18. Entregables de pruebas de integración (V)(F)
- D19. Entregables de pruebas del sistema (V)(F)
- D20. Entregables de pruebas de aceptación (V)(F)

- D21. Informes de calidad (V)(F)

D22. Documentación para el usuario final (V)(F)

Anexo II - Project Charter

Cliente:	
Nombre del proyecto:	
Tipo de proyecto:	
SteakHolders:	
Product Owner:	
SCRUM Master:	

Propósito del documento
Este documento define la descripción general, los objetivos y los participantes del proyecto. Se relaciona con la autorización del inicio del proyecto. Asimismo este documento brinda una descripción de la situación actual, los requisitos de alto nivel, criterios de éxito, riegos y oportunidades.

Propósito / Justificación:
Breve descripción del proyecto:
Alcance preliminar del proyecto:

Resultados esperados del proyecto / Beneficios:
Riesgos
Costo Preliminar Estimado / Presupuesto

Interesados en el proyecto				
Interesado	***Cargo***	***Rol***	***Teléfono***	***Email***

Supuestos
Restricciones

Autorización de proyecto			
Nombre	**Cargo**	**Firma**	**Fecha**
		_____	
		_____	

Anexo III - *Backlog*

Requerimiento	Historia de usuario				Detalles					
Descripción	**ID**	**Como (Rol)**	**Deseo…**	**Para…**	**Criterios de aceptación**	**Prioridad**	**Estimación [hrs.]**	**Dependencias**	**Núm. *Sprint***	**Estatus**
[Descripción del requerimiento]	H1	*[puesto]*	*[Detalle]*	*[Objetivo del detalle]*	*[Reglas de negocio]*	1	4	*[Dependencia de una historia de usuario]*	1	Por hacer
	H2					2	6		1	En progreso
	H3					3	5		1	Terminado
	H1					1	2		2	Eliminado

Anexo IV – Registro de Impedimentos

REGISTRO DE IMPEDIMENTOS						
PROYECTO:						
***SCRUM MASTER*:**						
Núm. Impedimento	**Fecha de Registro**	**Descripción**	**Reportado por**	**Estatus**	**Fecha de respuesta**	**Solución / Comentarios**
1				Pendiente		
2				En proceso		
3				Terminado		

Anexo V – Minuta de Reunión

Nombre del Proyecto:	
Objetivo de la Reunión:	
Participantes:	
Fecha y Hora:	
Lugar:	

Puntos Tratados

Acuerdos		
Responsable	**Actividad**	**Fecha compromiso**

Notas

Fecha de seguimiento: _____

Anexo VI – Acta de cierre de proyecto

Nombre del Proyecto:

Fecha:

Cierre por parte del *Product Owner*		
Pregunta	**Sí/No**	**Justificación**
¿Se han cumplido los objetivos del proyecto?		
¿Se han culminado todos los entregables del proyecto?		

¿Está el *Product Owner* satisfecho del resultado global del proyecto?		
Comentarios		

Firmas correspondientes: