ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

VICERRECTORADO DE INVESTIGACIÓN

INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DEL SOFTWARE

IV PROMOCIÓN

TESIS PREVIO A LA OBTENCIÓN DEL TÍTULO DE

MAGISTER EN INGENIERÍA DE SOFTWARE

TEMA: "IMPLEMENTACIÓN DE UN MARCO DE TRABAJO BASADA EN LA NORMA ISO/IEC 12207 Y LA METODOLOGÍA ÁGIL SCRUM PARA EL MEJORAMIENTO DE LA CALIDAD DE LOS PRODUCTOS DE SOFTWARE EN LA COOPERATIVA DE AHORRO Y CRÉDITO EL SAGRARIO LTDA."

AUTOR: ING. JAIRO ANDRÉS BEJARANO MONTESDEOCA

DIRECTOR: ING. LUCAS GARCÉS GUAYTA MSc.

LATACUNGA

2015

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

MAESTRÍA EN INGENIERÍA DE SOFTWARE

CERTIFICACIÓN

Certifico que el trabajo de titulación, "IMPLEMENTACÍON DE UN MARCO DE TRABAJO BASADA EN LA NORMA ISO/IEC 12207 Y LA METODOLOGÍA ÁGIL SCRUM PARA EL MEJORAMIENTO DE LA CALIDAD DE LOS PRODUCTOS DE SOFTWARE EN LA COOPERATIVA DE AHORRO Y CRÉDITO EL SAGRARIO LTDA.", realizado por el Ing. JAIRO ANDRÉS BEJARANO MONTESDEOCA, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo que cumple con los requisitos teóricos científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor JAIRO ANDRÉS BEJARANO MONTESDEOCA para que lo sustente públicamente.

Latacunga, 25 de noviembre de 2015

Ing. Lucas Garcés Guayta MSc.
Director

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

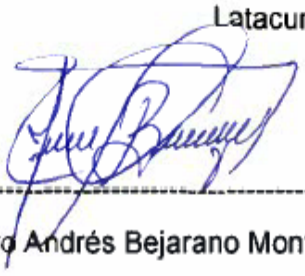
MAESTRÍA EN INGENIERÍA DE SOFTWARE

AUTORÍA DE RESPONSABILIDAD

Yo, JAIRO ANDRÉS BEJARANO MONTESDEOCA, con cédula de identidad N°. 1803115870, declaro que este trabajo de titulación "IMPLEMENTACIÓN DE UN MARCO DE TRABAJO BASADA EN LA NORMA ISO/IEC 12207 Y LA METODOLOGÍA ÁGIL SCRUM PARA EL MEJORAMIENTO DE LA CALIDAD DE LOS PRODUCTOS DE SOFTWARE EN LA COOPERATIVA DE AHORRO Y CRÉDITO EL SAGRARIO LTDA." ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 25 de noviembre de 2015



Ing. Jairo Andrés Bejarano Montesdeoca

C.C.: 1803115870

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

MAESTRÍA EN INGENIERÍA DE SOFTWARE

AUTORIZACIÓN

Yo, JAIRO ANDRÉS BEJARANO MONTESDEOCA, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación "IMPLEMENTACIÓN DE UN MARCO DE TRABAJO BASADA EN LA NORMA ISO/IEC 12207 Y LA METODOLOGÍA ÁGIL SCRUM PARA EL MEJORAMIENTO DE LA CALIDAD DE LOS PRODUCTOS DE SOFTWARE EN LA COOPERATIVA DE AHORRO Y CRÉDITO EL SAGRARIO LTDA.", cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Latacunga, 25 de noviembre de 2015

Ing. Jairo Andrés Bejarano Montesdeoca
C.C 1803115870

DEDICATORIA

Dedico este trabajo de investigación a mi amada esposa Silvia por su paciencia, comprensión y amor.

A mis adorados hijos Andrés y Sebastián que son el motor que me impulsa a superarme cada día.

A mis padres Luisa y Humberto por el ejemplo de perseverancia y amor a la vida.

A mí querida hermana Eva, por su apoyo incondicional.

Jairo.

AGRADECIMIENTO

Agradezco a Dios, por darme su bendición cada día y brindarme la posibilidad de superarme.

Un agradecimiento especial al director de mi trabajo de investigación y coordinador de la maestría de Ingeniería de Software Ing. Lucas Garcés, que gracias a su guía y conocimiento me condujo al feliz término de esta investigación.

A mis compañeros y amigos del departamento de tecnología de la Cooperativa de Ahorro y Crédito El Sagrario, que con su trabajo profesional me ayudaron a implantar el trabajo de investigación realizado.

A mis familiares y amigos por su apoyo incondicional

Jairo.

ÍNDICE DE CONTENIDOS

CAPÍTULO IV

CAPÍTULO V

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En la actualidad puede resultar difícil imaginarse la ejecución de un proceso sin la ayuda de un software que permita realizar un trabajo de forma más ágil y eficiente. Es por este motivo, la lucha por encontrar mecanismos que permitan desarrollar software de calidad se hace cada vez más necesario. Muchas son las metodologías que se han desarrollado a lo largo de estas décadas en pos de obtener productos de software confiables, cumpliendo estándares de calidad y con los tiempos de entrega acordados, sin embargo los diferentes imprevistos que se presentan al desarrollar software, no ha permitido que dichas metodologías sean 100% confiables para el desarrollo de cualquier proyecto. El presente trabajo de investigación tuvo como propósito elaborar un marco de trabajo basado en una de las metodologías ágiles más utilizadas a nivel mundial como Scrum; complementada con la norma ISO/IEC 12207, la cual permitió establecer lineamientos sobre los cuales se puedo desarrollar software de calidad en la Cooperativa de Ahorro y Crédito El Sagrario Ltda. Los procesos que se establecieron en este proyecto de investigación se adaptaron a las necesidades de la institución, obteniendo el mayor provecho de los procedimientos y personal con los que la institución cuenta. Con la fusión entre Scrum y la norma ISO / IEC 12207 se obtuvo una metodología robusta sobre la cual la institución se apalanca para poder desarrollar software de calidad mejorando sus procedimientos y haciéndolos más eficientes.

PALABRAS CLAVES:

- **METODOLOGÍA ÁGIL SCRUM**
- **SOFTWARE - ESTÁNDARES DE CALIDAD**
- **SOFTWARE - NORMA ISO/IEC 12207**
- **INGENIERÍA DE SOFTWARE**

ABSTRACT

Today it may be difficult to imagine the execution of a process without the help of a software that allows to work faster and more efficiently. This is the reason why finding mechanisms to develop quality software is becoming increasingly necessary. There are many methodologies that have been developed over the last decades in pursuit of obtaining reliable software products. These products have achieved quality standards within the agreed delivery times; however, the different incidental situations that appear during the development of the software have not allowed that the used methodologies are 100% reliable for the development of any project. Agile methodologies show up as an alternative in the software development process. They give a different approach to the traditional development that has been applied over several decades. The present research is focused on developing a framework based on one of the most used agile methodologies worldwide which is Scrum. It is complemented with the ISO / IEC 12207 standard. This methodology will allow to establish guidelines to develop a quality software at “El Sagrario Ltda.” credit union. The processes that will be established in this research paper will be adapted to the needs of the institution, getting the best benefits from the procedures and staff of the institution.

KEY WORDS:

- **AGILE SCRUM METHODOLOGIES**
- **SOFTWARE - QUALITY STANDARS**
- **SOFTWARE – ISO / IEC 12207**
- **SOFTWARE ENGINEERING**

CAPÍTULO I

DEFINICIÓN DEL PROBLEMA

1. Introducción

Este capítulo se enfoca en analizar los problemas que causa la falta de procedimientos, normas y estándares, dentro del proceso de desarrollo de Software, y como estos afectan el buen desenvolvimiento de la empresa en estudio.

Además se presenta la causa por la cual se debe investigar este evento, con el fin de dar una solución que permita optimizar el proceso de desarrollo de Software en la Cooperativa de Ahorro y Crédito El Sagrario Ltda. Obteniendo productos de Software confiables y con el menor número de defectos posibles.

1.1 Problemas de investigación abordada

En la actualidad muchas de las empresas optan por tener un departamento propio de desarrollo de software que les permita tener un mejor tiempo de respuesta al cambio, esta decisión sin lugar a duda conlleva algunas ventajas entre las cuales podemos citar las siguientes:

- El costo por hora del personal es más económico.
- El tiempo de respuesta para realizar cambios en los aplicativos es bajo.
- El control y mantenimientos de los aplicativos informáticos son responsabilidad del Departamento de TIC de las empresas.

Sin embargo esta forma de administrar los aplicativos informáticos de una empresa, también supone desventajas entre las cuales citamos las siguientes:

- El costo por errores ocasionados por los aplicativos informáticos los asume la empresa.
- La empresa se responsabiliza por la capacitación del personal.

- La rotación del personal puede ocasionar graves inconvenientes en la culminación de los proyectos en ejecución.

Al momento de realizar un análisis entre las ventajas y desventajas que tiene este modelo de administración de desarrollo de software, se puede resaltar el costo que las empresas deben asumir por los defectos que puedan existir en un producto de software que es liberado en producción.

Watts Humphrey denominado por muchos el padre de la calidad, en uno de sus principios menciona que *"La calidad de un producto la determina el proceso usado para desarrollarlo".* Bajo este principio las empresas deben enfocarse en tener procesos que les permitan obtener software de calidad con el menor número de defectos posibles. Lastimosamente por la poca experiencia que tienen las empresas de nuestro país esto no ha sido posible.

Hasta hace unos años, el proceso de desarrollo de software se enfocaba en la definición exhaustiva de actividades, roles y artefactos que basaban el éxito de un proyecto en documentación bien detallada. Esta forma de desarrollar software ha resultado ser necesario y eficiente en proyectos grandes, en los cuales se tiene la colaboración de bastante personal. Lastimosamente para empresas pequeñas que no cuentan con un gran número de personas, no resulta eficiente la aplicación de estas metodologías tradicionales.

1.2 Descripción del problema

El Departamento de Tecnología de la Cooperativa de Ahorro y Crédito El Sagrario Ltda. posee un área de desarrollo de software, el cual se encarga del desarrollo y mantenimiento del aplicativo informático con el que cuenta la institución.

El Departamento de Tecnología, no cuenta con una metodología de desarrollo de software que le permita ser más productivo en los procesos de desarrollo y mantenimiento de software. Este problema ocasiona que no

exista una sinergia entre el dueño del proceso y el equipo de desarrollo, dando como resultado la inconformidad en los productos de software por parte de los usuarios y retraso en el tiempo de entrega de dichos productos de software.

Al momento de publicar una versión en producción se producen múltiples errores, los mismos que causan insatisfacción en los usuarios y han ocasionado pérdidas económicas para la institución. Esto se produce por no tener un adecuado proceso de desarrollo de software que garantice la calidad de los productos de software elaborados.

El no tener un marco de desarrollo de software en áreas tan críticas como las del sector financiero puede ocasionar muchos inconvenientes, como por ejemplo fraudes informáticos, los mismos que en el Ecuador son los delitos con más incidencia según datos de Kaspersky Lab compañía especializada en seguridad informática, estos fraudes se lo pueden lograr manipulando la información a través de la modificación de los programas informáticos, estos delitos ocurren usualmente por no segmentar adecuadamente los ambientes de desarrollo de software sean estos: ambiente de desarrollo de software, control de calidad y producción, o por no tener políticas y procedimientos que ayuden a mejorar la organización de los departamentos de desarrollo de software.

La calidad del producto de software es otro factor a tomar en cuenta, ya que puede ocasionar pérdidas económicas tanto a la institución como a los socios o clientes que son parte de las mismas, debido a cálculos mal realizados por los aplicativos informáticos.

Los productos de software usualmente no son lo que el dueño del producto necesitaba, lo que provoca re procesos en la elaboración del software y posteriores mejoras.

Entre las causas que originan los problemas mencionados podemos citar las siguientes:

- Falta de normas, estándares, políticas y procedimientos en el proceso de desarrollo de software.
- No existe un marco de trabajo para la gestión, planificación y seguimiento de los productos de software.
- No existe un área de aseguramiento de la calidad que verifique la calidad de los procesos y de los productos.
- No existe indicadores que permita medir la calidad del software para posteriormente mejorar.

1.3 Descripción de la solución

Para obtener software de calidad, se debe recurrir a la utilización de normas, estándares, metodologías y procedimientos que ayuden a regular y estandarizar el proceso de desarrollo de software, con la finalidad de obtener productos de software confiables, que permita a las empresas ser cada vez más competitivas.

Las metodologías ágiles, nacen como una posible respuesta para las empresas que cuentan con un número reducido de personal y tienen la necesidad de adoptar procesos de desarrollo de software que le ayude a ser más eficiente.

El informe CHAOS 2013 elaborado por Standish Group, menciona que los proyectos pequeños tienen un menor índice de fracaso y se adapta de mejor manera a los cambios utilizando metodologías ágiles y realiza la siguiente comparación con las metodologías tradicionales.

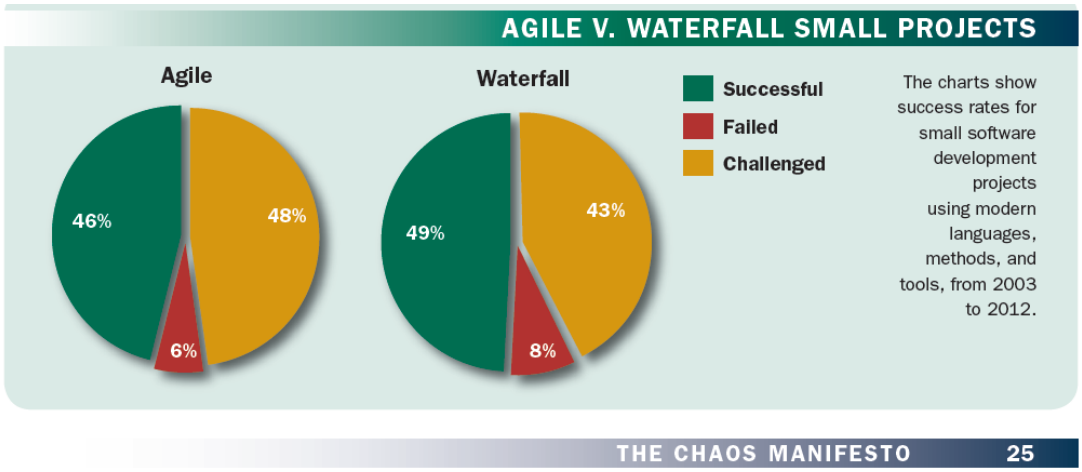


Figura 1 Comparación entre metodologías ágiles y tradicionales

Fuente: (e Standish Group International, 2013)

Scrum es una buena alternativa para empresas que tienen un número pequeño de colaboradores, siendo estos entre 4 y 7 personas ya que esta metodología no posee muchos roles y permite realizar seguimientos diarios. Scrum nos ayuda a verificar de manera oportuna como se está construyendo el software (de dos semanas a un mes) permitiendo a las empresas establecer prioridades.

1.4 Descripción resumida del proyecto

La propuesta del presente tema de tesis consiste en elaborar un marco de trabajo que permita establecer procedimientos, normas y estándares que se deben seguir en las fases del desarrollo de software, desde la concepción misma de un problema a resolver, hasta la liberación en producción y posteriores mantenimientos, para lo cual se elaborarán las siguientes acciones:

Elaborar un diagnóstico de la institución para conocer el proceso actual de desarrollo, organigrama estructural, número de personas que intervienen en el proceso de desarrollo de software y roles que cumple cada persona.

Construir el marco teórico sobre el cual se sustentará el trabajo investigativo de esta tesis.

Desarrollar el marco de trabajo para la Cooperativa de Ahorro y Crédito ‘El Sagrario’ Ltda. para lo cual se va a elaborar un organigrama estructural con las personas que van a intervenir en el proceso de desarrollo de software, luego se establecerá los procedimientos para cada etapa del desarrollo de software; los mismos que comprenderán las etapas de requerimientos, análisis, diseño, pruebas, liberación en producción, errores y mantenimiento de software. Posteriormente se establecerá la configuración de software estableciendo los diferentes ambientes que deben existir en la institución y describiendo las actividades que va a desarrollar cada uno de ellos. Además se elaboraran formatos que evidenciarán la aplicación de esta metodología, se establecerá indicadores que permitan evaluar la calidad del desarrollo de software y por último se validará el marco de trabajo con la norma ISO/IEC 12207.

Una vez implementado el marco de trabajo en el proceso de desarrollo de software de la Cooperativa de Ahorro y Crédito El Sagrario Ltda. se establecerá un análisis de seis meses, con la finalidad de evaluar el impacto que produjo el marco de trabajo en la institución.

Se validará los resultados obtenidos con la implementación del marco de trabajo en los proyectos que se estén realizando dentro de la Cooperativa de Ahorro y Crédito El Sagrario Ltda.

1.5 Justificación e importancia

Las Instituciones financieras están consideradas dentro de las denominadas “infraestructuras críticas del país” y las Cooperativas de Ahorro y Crédito son una de las partes más vulnerables de este sector, ya que el mismo se encuentra dentro de una economía globalizada, dinámica y competitiva, el cual obliga a implementar nuevas estrategias basadas en productos innovadores que permita diferenciarse de la competencia, manteniendo un incremento sostenido del negocio a través de la fidelización de los socios y clientes.

Para enfrentar estos desafíos la Tecnología de Información se convierten en un área estratégica para conseguir dichos objetivos, esta es la razón fundamental para que esta área cuente con una sólida gestión en sus procesos, con un marco de trabajo capaz de adaptarse fácilmente a los cambios del entorno, priorizando la satisfacción de los usuarios a través de revisiones y entregas continuas, entendiendo que el éxito de un proyecto depende de tres factores: éxito técnico, éxito personal y éxito organizacional.

El trabajar con metodologías ágiles nos permite tener una mejor comunicación con la persona responsable del proceso y con las personas que utilizaran el sistema informático, obteniendo una retroalimentación oportuna, derivando en productos de software de calidad y satisfaciendo las expectativas de los usuarios finales, este proceso se logra estableciendo estándares, políticas y procedimientos claros para todas las fases que intervienen en el proceso de desarrollo de software, de igual manera se debe tomar en cuenta los roles que van a tener cada uno de los actores dentro de este proceso.

Este proyecto tiene una gran importancia técnica ya que permitirá organizar adecuadamente los procesos del área de tecnología de la Cooperativa.

Además tendrá una repercusión económica positiva en la institución, ya que al tener un área de tecnología más productiva, se contribuye al desarrollo de nuevos productos que permita ser más competitivo en el mercado, mitigando el riesgo que esto puede ocasionar.

Socialmente este proyecto tiene una gran importancia, ya que con los resultados obtenidos se puede extrapolar hacia otras Cooperativas y posteriormente para empresas que tengan similares características.

1.6 Objetivos

1.6.1 Objetivo general

Implementar un marco de trabajo basada en la norma ISO/IEC 12207 y la metodología ágil Scrum para el mejoramiento de la calidad de los productos de software en la Cooperativa de Ahorro y Crédito 'El Sagrario' Ltda.

1.6.2 Objetivos específicos

- Construir el marco teórico que analice los procesos, normas y estándares de calidad, que existen para desarrollar software.
- Desarrollar la propuesta del marco de trabajo tomando en cuenta todas las fases del desarrollo de software.
- Implementar la propuesta del marco de trabajo en la institución.
- Validar el marco de trabajo en los procesos y productos de software de la Cooperativa de Ahorro y Crédito El Sagrario Ltda.

1.7 Metas

La meta a corto plazo de este proyecto es proporcionar a la Cooperativa de Ahorro y Crédito El Sagrario Ltda., un marco de trabajo ágil que permita desarrollar productos de software de una manera eficiente, implementando estándares de calidad, segmentando las funciones de las personas involucradas en el proceso y creando una configuración de software adecuada para la organización.

La meta a mediano plazo es tomar este marco de trabajo como base para implementar en las Cooperativas de Ahorro y Crédito de la ciudad de Ambato y difundir las ventajas encontradas.

La meta a largo plazo es implementar este marco de trabajo en instituciones que tengan desarrollo in house con un número pequeño de colaboradores en el área de tecnología de información.

1.8 Hipótesis

Si se implementa un marco de trabajo basada en la norma ISO/IEC 12207 y la metodología ágil Scrum, entonces se mejorará la calidad del producto de software en la Cooperativa de Ahorro y Crédito ‘El Sagrario’ Ltda.

1.9 Variables de la investigación

1.9.1 Variable dependiente

Se mejora la calidad de los productos de software en la Cooperativa de Ahorro y Crédito El Sagrario Ltda.

1.9.2 Variable independiente

Se implementa un marco de trabajo basado en la norma ISO/IEC 12207 y Scrum en la Cooperativa de Ahorro y Crédito El Sagrario Ltda.

1.10 Conclusión del capítulo

Cooperativa de Ahorro y Crédito El Sagrario Ltda. tiene problemas en el desarrollo de software. Los mismos que por el ámbito de su negocio pueden llegar a causar importantes pérdidas económicas. Sin embargo el compromiso de la Administración en mejorar estos aspectos, conjuntamente con la elaboración de un marco de trabajo, van a permitir establecer procedimientos, normas y estándares que le ayuden al departamento de desarrollo de software a elaborar productos de software de calidad.

CAPÍTULO II

MARCO TEÓRICO

2. Introducción

La Ingeniería de Software es una disciplina relativamente joven, sin embargo ha tenido una gran evolución en este corto tiempo. En este capítulo señalaremos los aspectos más relevantes que han ocurrido en este proceso y la contribución que ha dejado en el proceso de desarrollo de software en cada una de estas etapas.

2.1 Antecedentes Históricos

Evolución de Normas y estándares de calidad en el proceso de desarrollo de software

Software según la IEEE, "*es la suma total de los programas de ordenador, procedimientos, reglas, la documentación asociada y los datos que pertenecen a un sistema de cómputo*" y "*un producto de software es un producto diseñado para un usuario*". En este contexto, la Ingeniería de Software (SE del inglés "Software Engineering") es un enfoque sistemático del desarrollo, operación, mantenimiento y retiro del software.

En 1950 la ingeniería del software inicia su ámbito de aplicación, una de sus principales funciones “estaba basada en el desarrollo de software para la ingeniería aérea y espacial” (Villa, 2013). El término ingeniería del software fue “acuñado en 1968, en la primera conferencia organizada por la OTAN sobre desarrollo de software” (Wikipedia, Crisis del software, 2014), como consecuencia de la denominada crisis del software. “La crisis se fundamentó en el tiempo de creación de software, ya que en la creación del mismo no se obtenían los resultados deseados” (Wikipedia, Crisis del software, 2014), además “*expresaba las dificultades del desarrollo frente al rápido crecimiento de la demanda, y la complejidad de los problemas a ser resueltos y de la inexistencia de técnicas establecidas para el desarrollo de sistemas que*

funcionaran adecuadamente o pudieran ser validados" (Universidad Politécnica de Valencia, 2011).

A finales de los años 50 y principios de los 60, el procesamiento de información de los equipos de cómputo era limitado, estos equipos ejecutaban "simples" programas construidos (Desarrollados con lenguajes de programación de bajo nivel) sin metodología alguna. A finales de los 60, el procesamiento de información de los equipos de cómputo aumentó considerablemente, apareciendo los lenguajes de programación de alto nivel. Como consecuencia de un mayor procesamiento de información, fue necesario construir programas complejos.

En esta época, se empezó a concebir el Software como producto, y se empezaron a desarrollar algunos proyectos para que funcionaran en las máquinas de la época. Pero aparecieron importantes problemas: los productos excedían la estimación de costes, había retrasos en las entregas, las prestaciones no eran las solicitadas, el mantenimiento se hacía extremadamente complicado y a veces imposible, las modificaciones tenían un coste prohibitivo, en resumen se desarrollaba software de mala calidad, ya que la técnica utilizada para desarrollar pequeños programas para maquinas con mucho menos potencial se quedaba desfasada, y muchas veces este software acababa en el olvido" (Universidad Politécnica de Valencia, 2011).

"*Una de las principales causas de todo esto, si no la principal, era el enfoque dado al proceso de desarrollo de software, el cual era malo e incluso a veces era inexistente. En este proceso, solo ¼ del tiempo de desarrollo se dedicaba a las fases de análisis, diseño, codificación y pruebas, y más de ¾ del tiempo se dedicaba a correcciones y mantenimiento. Es evidente que dedicándole solo ¼ del tiempo a las primeras fases, se arrastran errores graves, sobre todo procedentes de las fases de análisis y diseño, lo que dificultaba muchísimo la implementación, produciendo constantes paradas y retrocesos para revisar este análisis/diseño*" (Universidad Politécnica de Valencia, 2011).

Los graves problemas que acarreó la crisis del software, exigió contar con metodologías que permitan establecer técnicas aplicadas al desarrollo con la finalidad de proveer productos que satisfagan las necesidades de los interesados y caracterizados por un alto grado de confiabilidad. Así, surgió la ingeniería del software.

La ingeniería del software es considerada como una división de la ingeniería “que crea y mantiene las aplicaciones de software usando tecnologías y prácticas de las ciencias de la computación, manejo de proyectos, ingeniería, el ámbito de la aplicación, y otros campos” (Universidad de Murcia, 2014). Tiene como objetivo “apoyar el desarrollo de software profesional, en lugar de la programación individual. Incluye técnicas que apoyan la especificación, el diseño y la evolución del programa, ninguno de los cuales son normalmente relevantes para el desarrollo de software personal” (Sommerville, Ingeniería del software, 2011)

Actualmente, la ingeniería del software se enfrenta a una constante y creciente demanda, limitada por los cortos tiempos de entrega y con la exigencia de proporcionar un producto con un alto grado de confiabilidad. “Aproximadamente el 60 % de los costos del software son de desarrollo, y un 40 % de prueba. Para el software elaborado específicamente, los costos de evolución superan con frecuencia los costos de desarrollo” (Sommerville, Ingeniería del software, 2011).

A pesar de la existencia de técnicas para la gestión y el desarrollo de proyectos de software, estas técnicas no son empleadas correctamente. Así lo reflejó un estudio de 280 000 proyectos de software (Schach, 2006), realizado por Jonhson, Boucher, Connors y Robison, resumido en la figura 2:

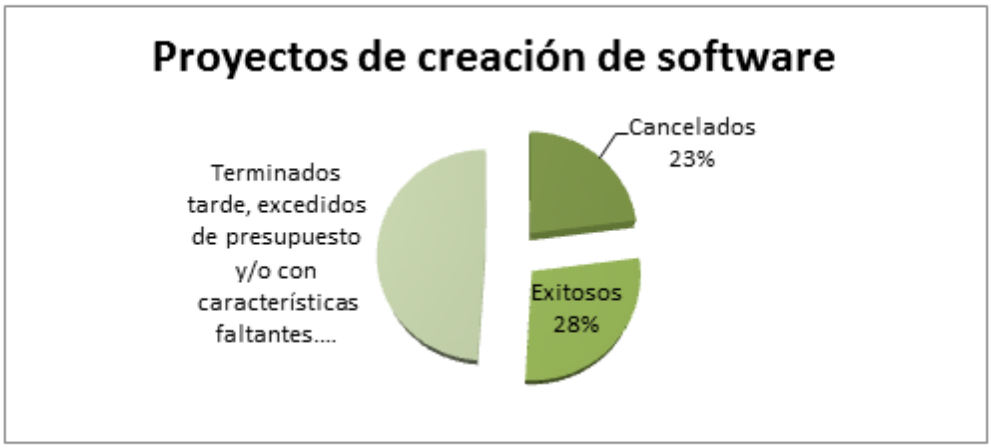


Figura 2 Proyectos de software

Fuente: (Schach, 2006)

Schach considera que la crisis del software aún perdura, presentados dos parámetros: “el proceso de producción de software es semejante a la ingeniería tradicional, tiene sus propiedades y problemas únicos” (Schach, 2006) y “la crisis del software tal vez se deba renombrar como la depresión del software, en vista de su larga duración y pronóstico” (Schach, 2006).

La crisis del software y las consecuencias financieras de la crisis del software son enormes. En una encuesta realizada por el Cutter Consortium [2002], se informó lo siguiente:

- “Un 78 % de las organizaciones de tecnologías de la información se han involucrado en disputas que terminaron en litigio” (Schach, 2006).
- “En 67 % de estos casos, la funcionalidad o el desempeño de los productos software entregados no cumplió con los ofrecimientos” (Schach, 2006).
- “En 56 % de estos casos, la fecha de entrega prometida se postergó varias veces” (Schach, 2006).
- “En 45 % de estos casos, las fallas eran tan graves que el producto software no podía utilizarse”. (Schach, 2006)

A continuación citaremos las diversas etapas que ha tenido que pasar el desarrollo de software para mejorar la calidad del mismo.

Primera etapa (1950 - 1970)

El año de 1950 fue trascendental para la era informática, ya que las computadoras estaban disponibles en universidades y centros de investigación, siendo utilizados principalmente en el área de ingeniería y física. Para poder utilizar el potencial de las computadoras nacieron los llamados programadores. Los programadores escribían código y trasladaban sus programas a un mostrador en donde un despachador recogía los programas y los colocaba en una cola para posteriormente ser procesados, los resultados de los programas se daban a conocer horas o días después; no existía ningún tipo de interactividad entre el hombre y el ordenador, la programación y la informática eran tareas separadas. Para facilitar este proceso fueron creados los que hoy llamamos lenguajes de programación, el primer lenguaje fue Fortran creado por IBM en 1957, seguido por Algol en 1958 y su sucesor oficial en 1960. En aquellos años las computadoras eran utilizadas para cálculos y no para el procesamiento ni almacenamiento de información. En 1962 nació Cobol creado por el Departamento de Defensa de Estados Unidos, específicamente para aplicaciones de negocios. Con la aparición de este lenguaje de programación, las empresas comenzaron a contratar programadores para que automaticen sus procesos, se comenzó a desarrollar proyectos sin ningún tipo de proceso, al cual se le conoce como "*code and fix*" codifica y prueba. Este método de construir software tuvo muchos problemas ya que la calidad de software era muy mala.

En la conferencia realizada por la OTAN en 1968, se discutió todos los problemas que estaban generando la falta de procesos al momento de desarrollar software, en esa reunión surgieron algunas alternativas para mejorar la llamada "*crisis del software*". (Wirth, 2008)

Segunda etapa (1970 - 1991)

La imperiosa lucha por salir de la crisis del software, dio como resultado la creación de modelos y metodologías para el desarrollo de software, el primero fue el modelo en cascada presentado en 1970 por Royce, este modelo presentaba una serie de pasos para lograr el desarrollo de software. Tenía una estructura lineal es decir para comenzar una fase se debía terminar la anterior, lo que ocasionó muchas críticas a este modelo, posteriormente surgió el modelo incremental creado por Lehman en 1984, este modelo corregía la linealidad del modelo en cascada, este modelo resultaba útil cuando los requerimientos eran poco precisos, pero realizar modificaciones resultaba igual de costoso que el modelo en cascada. Para suplir las deficiencias del modelo en cascada y el incremental, Boehm crea el modelo en espiral en 1988 cuya principal característica era su adaptabilidad a los continuos cambios que esta disciplina conlleva, una de las mayores desventajas es que su mayor utilidad es en proyectos grandes. La mayor desventaja que tienen los modelos de desarrollo de software es la falta de procedimientos para el desarrollo de software.

La creación de metodologías fue otra alternativa para mejorar la calidad del software, así nacen las primeras metodologías de software orientados a procesos, que fueron creadas por DeMarco en 1979, Gane y Searsons en 1979 y yourdon en 1989. Estas metodologías se basaban en la descomposición funcional del problema, en el año de 1991 Booch revoluciona el desarrollo de software creando la metodología orientada a objetos. Estas metodologías ayudaron a mejorar en algo el desarrollo de software, pero no cumplieron totalmente con su cometido. (Cataldi, 2000)

Tercera etapa (1991 - 1999)

Las empresas y el gobierno al no encontrar los resultados esperados con la utilización de modelos y metodologías de desarrollo de software, y una encuesta realizada sobre los proyectos de software cuyo resultado fue que el

30% de los proyectos se cancelaban; el 54% de los proyectos exceden ampliamente costos y tiempo, y solo el 16% de los proyectos finalizan exitosamente, llevan al ejército de EEUU a tomar una decisión y a través de Software Engineering Institute (SEI) en colaboración de Mitre Corporation empezaron a desarrollar un framework para medir la madurez de proceso en una organización en el año de 1986 y en agosto de 1991 publica su primera versión de SW-CMM(Capabitity Maturity Model for Software), este modelo proporcionó inicialmente una guía para el desarrollo e implementación de software, su creación tuvo el objetivo de proporcionar a las empresas estrategias para mejorar sus procesos enfocándose en un conjunto de actividades que permitan logros continuos y capacidad perdurable (Albariz, 2006) . A esta etapa también se le atribuye la creación del estándar de calidad de producto de software ISO/IEC 9126 a finales de 1991, esta norma permitía la evaluación del software, este estándar señala que la calidad debe estar enfocada a cumplir las necesidades de los usuarios, por lo que los usuarios finales son los encargados de percibir la calidad del software. En 1995 se crea una norma para estandarizar el ciclo de vida de software esta fue la ISO/IEC 12207, esta norma se preocupaba desde la definición de requerimientos hasta la liberación en producción, tomando en cuenta la provisión y adquisición de software. En 1998 se publicó oficialmente la primera familia de la norma ISO/IEC 15504 conocida también como SPICE cuyas siglas se traducen como Software Process Improvement and Capability Determination en español Determinación de la capacidad y mejora de los procesos de software. Esta norma proviene de la norma ISO/IEC 12207 y los modelos de madurez Bootstrap, Trillium y CMM, esta norma proporciona lineamientos para realizar la evaluación de calidad. (Palacio, 2006)

Cuarta etapa (1999 - actualidad)

Esta etapa nace con la publicación oficial de la metodología RUP creada por Grady Booch, Ivar Jacobson y James Jacobson a inicios de 1999, se centra en la interacción de personas, procesos y herramientas. Esta

metodología se basa en una amplia documentación con la cual se pretende tener claro el desarrollo que se va a realizar (Rational Unified Process, 2012). En el mismo año Kent Beck desarrolla una metodología la cual se conoce como Programación Extrema, esta metodología contiene buenas prácticas para la planificación, gestión, diseño y codificación basándose en historias de usuario y planificación de iteraciones. (PMOinformatica, 2013).

En el año 2000 se realiza la integración de SW-CMM que toma el nombre de CMMI, fue creado para simplificar, facilitar y mejorar la usabilidad de varios modelos de madurez de forma simultánea en un solo framework, en agosto del 2006 se publica el modelo CMMI-DEV el cual propone una solución integral para el desarrollo y mantenimiento de software enfocados a productos y servicios.

En el año 2001 se lleva a cabo una reunión con diecisiete expertos en modelos de desarrollo de software, en esta reunión se acuño el término “Métodos ágiles” y realizaron un resumen al cual denominaron manifiesto ágil y surge como una alternativa a las metodologías denominadas formales como CMMI o RUP.

En el año 2002 la norma ISO/IEC 12207 sufre su primera enmienda y el en año 2004 la segunda, en el cual se estableció un modelo de referencia conforme los requisitos de la norma ISO/IEC 15504-2, en el año 2006 el comité técnico de normalización de ingeniería de software y sistemas de información elabora otra modificación a esta norma. La última modificación realizada en esta norma es en el año 2008 cuyo objetivo principal es establecer un marco de referencia común para los proveedores, compradores, desarrolladores, gestores y personal involucrado en el desarrollo de software. La norma 15504 se termina de publicar en el año 2006 y su objetivo principal es evaluar los procesos del ciclo de vida del software definidos en la norma ISO/IEC 12207. En el año 2002 surge una norma mexicana focalizada a empresas que poseen áreas de desarrollo de software esta norma fue llamada MoProSoft y en el año 2003 se crea EvalProSoft que es un modelo

de avaluación de MoProSoft. Esta norma fue reconocida como norma internacional en el año 2006. En el año 2011 tomando como base MoProSoft, ISO/IEC12207, ISO/IEC15504 se crea la norma ISO/IEC 29110 la cual contiene guías y estándares para el ciclo de desarrollo de software de pequeñas organizaciones que realicen desarrollo y mantenimiento de aplicativos que no sean riesgosos.

2.2 Antecedentes conceptuales y referenciales

2.2.1 Caracterización tecnológica del proceso de ingeniería del software

A. Conceptos

A.1 Ingeniería del software

"*La ingeniería del software es una disciplina de ingeniería que se interesa por todos los aspectos de la producción del software, desde las primeras etapas de la especificación del sistema hasta el mantenimiento del sistema después de que se pone en operación*" (Sommerville, Ingeniería del software, 2002).

Según el autor de la definición anterior, es posible considerar un "buen software" si el producto está caracterizado por los siguientes atributos esenciales: mantenimiento, confiabilidad y seguridad, eficiencia y aceptabilidad.

El mantenimiento es la capacidad que dispone el software para adaptarse a los nuevos requerimientos y necesidades de los clientes, el mantenimiento es esencial en un producto de software. La confiabilidad es un atributo que garantiza la protección y brinda la seguridad de la información que es almacenada o gestionada; el software confiable evita en lo posible, accesos no autorizados y malintencionados a más de evitar daños económicos o físicos.

La eficiencia es un atributo que exige a un producto de software aprovechar correctamente los recursos del sistema (Memoria, almacenamiento, entre otros); así, como los tiempos de respuesta respecto a las diversas peticiones

que efectuarán los usuarios finales del aplicativo informático. Finalmente, la aceptabilidad refiere a un atributo que exige que el sistema debe ser aceptable para quienes lo utilizan, es decir, un producto de software de fácil comprensión, sencillo de utilizar y que pueda intercambiar información con otros sistemas.

A.2 Proceso de desarrollo de software

El proceso de desarrollo de software es “un conjunto de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto del trabajo” (Pressman, Ingeniería del software, un enfoque práctico, 2010). Este proceso tiene como propósito la generación eficaz de un producto de software, que cumplirá con los requisitos establecidos por el cliente.

“*En el contexto de la ingeniería de software, un proceso no es una prescripción rígida de cómo elaborar software de cómputo. Por el contrario, es un enfoque adaptable que permite que las personas que hacen, busquen y elijan el conjunto apropiado de acciones y tareas para el trabajo*” (Pressman, Ingeniería del software, un enfoque práctico, 2010).

El proceso de software es “intensamente intelectual, afectado por la creatividad y juicio de las personas involucradas” (Sommerville, Ingeniería del software, 2002). La administración de un proyecto de software, posee características similares a la administración de proyectos de otras ramas de la ingeniería, la diferencia está en la complejidad del producto a ser entregado, que por su intangibilidad presenta desafíos adicionales como: alcanzar un 100 % de confiabilidad, ambiguas definiciones por el nivel de abstracción del software (La carente experiencia o el desconocimiento de otros productos de software de similares características impiden una vasta, correcta y concreta definición de requerimientos); y, durante el desarrollo de un producto de software es posible que se presenten requerimientos o especificaciones no contempladas en la definición inicial.

Sommerville, establece que el proceso de desarrollo de software dispone de cuatro actividades esenciales: especificación, desarrollo, validación y evolución del software. A diferencia de Pressman quien afirma que el proceso de la ingeniería del software consta de las siguientes actividades: comunicación, planeación, modelado, construcción y despliegue.

Ambos autores coinciden en que el proceso de software es un conjunto de actividades, que permitirá en un plazo determinado entregar un producto de software. La experiencia de ambos autores permite definir los elementos o actividades consideradas necesarias para ejecutar un proceso, que deberá conllevar en lo posible la entrega de un producto que cumplirá con las especificaciones establecidas.

Pressman describe en cinco las actividades que se deberán cumplir para llevar a cabo un proceso, en una compresión más amplia de estas actividades y de las actividades propuestas por Sommerville, la experiencia nos conduce a definir un elemento muy esencial y de gran relevancia como es la comunicación. En sí, la comunicación es la actividad que se ejecutará previo al inicio de cualquier trabajo técnico asociado al proyecto; a más de definir los "objetivos de los participantes respecto del proyecto, y reunir los requerimientos que ayuden a definir las características y funciones del software" (Pressman, Ingeniería del software, un enfoque práctico, 2010)

Para la presente investigación, se considerará la definición de actividades o elementos estructurales del proceso de software descrita por Pressman.

B. Componentes o elementos del proceso de desarrollo de software

B.1 Comunicación

La comunicación se constituye como parte inicial de un proceso de software. La comunicación, tiene como objetivo primordial el intercambio de información entre los participantes de un proyecto de desarrollo de software,

con la finalidad de conglomerar los requerimientos o especificaciones que deberá cumplir el producto a ser entregado en el plazo establecido.

B.2 Planeación

La planeación permite la definición de las funcionalidades y restricciones (controles operacionales) del producto de software; a más de, describir las tareas a ejecutarse, la identificación de los riesgos inherentes del desarrollo, la asignación de los recursos, los entregables y el conjunto de actividades (Ciclo) que se ejecutarán (Utilización de metodologías de desarrollo de software).

B.3 Modelado

En el modelado se diseñan las interfaces de usuario, el modelado de base de datos; y, la selección de la arquitectura de la aplicación. El modelado definirá los requerimientos del programa y el diseño que los integrará.

B.4 Construcción

La construcción representará las definiciones establecidas en el modelado, en el producto de software a ser entregado. A través de ciclos de interacción (Empleando metodologías de desarrollo de software) los responsables de la programación del software generarán productos entregables que deberán ser supervisados y verificados para asegurar la calidad y que cumpla con los requerimientos y expectativas del cliente.

B.5 Despliegue

Es la entrega parcial o completa del producto de software al cliente. El producto entregado lo emplearán los usuarios finales, con el propósito de evaluar el sistema y generar la retroalimentación correspondiente. La retroalimentación se realizará en un corto plazo.

C. Métodos

C.1 Métodos basados en planes

Los métodos basados en planes, están caracterizados por seguir una estricta planificación del proyecto. El método divide el proceso en fases, que

son ejecutadas en actividades y tareas. El análisis, diseño y desarrollo son fases destacas del método; a más de la generación de la documentación en cada una de las fases.

Entre los métodos basados en planes se destacan: CDM, RUP y UNC-method.

CDM.- Es un método propietario de Oracle, que cubre las todas las fases del ciclo de vida. Cada fase está constituida por un conjunto de actividades agrupadas por procesos. Cada proceso, genera un documento denominado “entregable”. Los “entregables” generados poseen estándares y deben ser tramitados por los responsables antes de iniciar la fase siguiente.

RUP (Proceso Unificado Racional, Rational Unified Process).- es un proceso orientado al producto, estratégicamente documentada y definida para el proceso de desarrollo. RUP abarcar todas las fases del ciclo de vida, cuyo objetivo prioritario es construir productos de software de alta calidad, es ajustable a las necesidades de los interesados, flexible en tiempos y con presupuestos predecibles/manejables.

UNC-method.- tiene como “objetivo conservar la trazabilidad entre los requisitos y la solución informática”. El método está constituido por cuatro fases: contexto del software (Identificación de las características de la organización, responsabilidades y roles), análisis del problema (Representación de los procesos de la organización y los mayores problemas y causas), propuesta de solución (Definición de un repositorio de soluciones) y esquema conceptual (Perfeccionamiento de la solución seleccionada).

C.2 Métodos Ágiles

Los métodos ágiles constituyen un enfoque en el desarrollo de aplicaciones que se orientan a proyectos en contextos cambiantes y adaptativos y que requieren la entrega de productos en tiempos reducidos. Así, se busca lograr

la calidad, procurando la simplicidad y la creación de aplicaciones de software flexibles (Orjuela & Rojas, 2008).

Los métodos ágiles dan importancia al individuo y al equipo de desarrollo, sobre los procesos y las herramientas tecnológicas, considerando que es vital proporcionar software funcional con buena documentación; en el marco de una colaboración con el cliente sobre la negociación de un contrato; y, la respuesta inmediata a los cambios.

Entre los métodos ágiles, se destacan XP y SCRUM.

XP (Programación Extrema, Extreme Programming).- Es una metodología ágil basada en cuatro valores: comunicación, simplicidad, retroalimentación y coraje. El método busca la generación rápida de software basada en la colaboración de todos los integrantes del equipo de desarrollo. XP busca satisfacer las necesidades de los clientes, brindar un continuo aprendizaje de los desarrolladores y un buen clima laboral.

SCRUM.- Es un método basado en iteraciones para equipos de desarrollo en un entorno, posiblemente cambiante. SCRUM está caracterizado por realizar reuniones diarias del equipo de desarrollo con la finalidad de solventar inquietudes y conocer el porcentaje de cumplimiento de las actividades; desarrolla actividades mediantes iteraciones conocidas como "sprint". Al finalizar un "sprint" se dispone de un "entregable".

D. Modelos del proceso de desarrollo de software

Un modelo de proceso de software es "*una representación simplificada de un proceso de software, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de procesos del software es una abstracción de un proceso real.*" (Sommerville, Ingeniería del software, 2002).

Los modelos de desarrollo de software se clasifican en modelos tradicionales y orientados a objetos. Algunos de los modelos tradicionales son:

- Codificar y corregir.
- Modelo en cascada.
- Desarrollo evolutivo.
- Desarrollo formal de sistemas.
- Desarrollo basado en reutilización.
- Desarrollo incremental.
- Desarrollo en espiral.

Algunos de los modelos orientados a objetos son:

- Modelo de agrupamiento
- Modelo fuente
- Modelo remolino
- Modelo Pinball

D.1 Modelo en cascada

"Es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior" (Sommerville, 2005).

Las actividades fundamentales de este modelo de proceso de software son: análisis y definición de requerimientos, diseño del sistema y del software, implementación y prueba de unidades, integración y prueba del sistema; y, funcionamiento y mantenimiento.

- Análisis y definición de requerimientos: se constituyen los objetivos del proyecto y se definen detalladamente todos los requerimientos del cliente.

- Diseño del sistema y del software: definición de la arquitectura de la aplicación y de las relaciones de los componentes del sistema.
- Implementación y prueba de unidades: construcción de módulos del software.
- Integración y prueba del sistema: integración de todos los módulos desarrollados, ejecución de las pruebas de aseguramiento de la calidad.
- Funcionamiento y mantenimiento: instalación del sistema, detección y corrección de errores (Retroalimentación). Es posible, la definición de nuevos requerimiento por parte del cliente.

D.2 Desarrollo evolutivo

"El desarrollo evolutivo se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta la que se desarrolla un sistema adecuado" (Sommerville, 2011).

Existen dos tipos de desarrollo evolutivo: desarrollo exploratorio y prototipos desechables. El desarrollo exploratorio trabaja con el cliente para conocer sus requerimientos. El desarrollo de prototipos desechables, comprende los requerimientos del cliente que permiten una definición mejorada, representado a través de definiciones concretas y compresibles.

D.3 Desarrollo en espiral

"Las actividades de este modelo se conforman en una espiral, en la que cada bucle o iteración representa un conjunto de actividades. Las actividades no están fijadas a ninguna prioridad, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior" ("Desarrollo en espiral - Wikipedia, la enciclopedia libre," 2015)

El modelo está constituido por las siguientes actividades: comunicación con el cliente, la planificación, el análisis de riesgos, la ingeniería, la evaluación del cliente y la construcción/entrega.

Comunicación con el cliente.- permite la definición de las tareas requeridas, la actividad es desarrollada entre el cliente y el responsable/líder del proyecto desarrollo de software.

Planificación.- establece el tiempo de las tareas definidas en el ítem anterior. A más de la asignación de los recursos técnicos tecnológicos y del personal responsable de la ejecución de cada una de las tareas.

Análisis de riesgos.- evaluación de los riesgos técnicos y de gestión del proyecto de desarrollo de software.

Ingeniería.- definición de las tareas que permitirán la construcción de uno o varios módulos del software.

Construcción y acción.- ejecución de las tareas ya definidas en el proceso de ingeniería. La actividad incluye pruebas de certificación y la generación de documentación para los usuarios finales.

Evaluación del cliente.- conjunto de tareas que permitirá generar la retroalimentación por parte del cliente y que serán empleadas posteriormente.

D.4 Modelo de agrupamiento

El modelo de agrupamiento está caracterizado por ser un modelo constituido por un componente secuencial y un componente concurrente. El modelo permite el trabajo en forma continua desde el inicio del proceso (análisis) hasta su finalización (generación), lo que permite el trabajo en distintas partes del producto final a ser entregado. Una de las ventajas del modelo es que permite la modularidad y la reutilización.

D.5 Modelo remolino

El modelo remolino, es un modelo orientado a objetos multiciclo en forma de remolino en lugar de una cascada. El modelo está constituido por cinco dimensiones: amplitud (Tamaño del desarrollo), profundidad (Nivel de abstracción o detalle), madurez (Asociado a la usabilidad, diseño, interfaces amigables, etc.), alternativas (Diferentes soluciones a un problema), alcance

(Objetivos del sistema, los requisitos cambian en función del avance del proyecto).

D.6 Modelo pinball

El modelo fuente es una representación gráfica del alto grado de iteración y solapamiento. Cada iteración puede estar en una fase diferente del ciclo de vida durante el desarrollo del sistema. El ciclo inicia en el conjunto de requisitos desde el cual se avanza, volviendo abajo solo al realizar el mantenimiento.

E. Organismos que regulan el proceso de desarrollo de software

E.1 International Standarization Organization, ISO.

La “International Standarization Organization” (ISO) es la institución rectora a nivel internacional encargada de definir y actualizar las normas de comercio, comunicación y fabricación para empresas a nivel mundial. La aplicación de las normas ISO son voluntarias, considerando que el instituto es una organización no gubernamental, por tanto sus decisiones no están influenciadas por ningún gobierno.

La Organización Internacional de Normalización, nació tras la Segunda Guerra Mundial, está constituida por los institutos de normalización de 163 países. La Secretaría General radica en Suiza.

Para el desarrollo de software, ISO define las siguientes normas: ISO/IEC 9000-3:2004, ISO/IEC 12207:1995, ISO/IEC 12207:2008, ISO/IEC 9126:2001, ISO/IEC 15939:2007, ISO/IEC 15504:2004, ISO/IEC 14598:1999, ISO/IEC 25000:2005.

E.1.1 IEEE

El Instituto de Ingeniería Eléctrica y Electrónica (IEEE, por sus siglas en inglés) es una organización destinada a la estandarización y el desarrollo en áreas técnicas. (*IEEE 2009 Annual Report*, 2009)

Para el desarrollo de software, el IEEE define las siguientes normas: norma 1633 (Práctica recomendada en la fiabilidad del software), norma 14471 (Directrices para la adopción de herramientas CASE).

El Instituto de Ingeniería Eléctrica y Electrónica fue constituido en el año de 1884, entre algunos de sus miembros fundadores destacan Thomas Edison, Alexander Graham y Franklin Leonard Pope. En el año de 1963 se adopta el nombre actual, al fusionar el instituto con el American Institute of Electrical Engineers (AIEE, Instituto Americano de Ingenieros Eléctricos) y el Institute of Radio Engineers (IRE).

E.1.2 SEI

El Instituto de Ingeniería de Software (SEI, por sus siglas en inglés), es una organización de los Estados Unidos de América, que tiene como objetivo el desarrollo de modelos de evaluación y mejora en el desarrollo de productos de software. Actualmente está financiado por el Departamento de Defensa de los EEUU y administrado por la Universidad Carnegie Mellon. El Software Engineering Institute (SEI) fue fundado por Congreso de los Estados Unidos en el año de 1984. (“Software Engineering Institute - Wikipedia, la enciclopedia libre,” 2015)

La SEI, ha diseñado el modelo CMM (Capability Maturity Model) que permite le evaluación de los procesos de una organización.

F. Herramientas para el proceso de desarrollo de software

Las herramientas tecnológicas del proceso de ingeniería del software, nacieron con la finalidad de brindar una solución a los problemas que generaba el desarrollo de productos de software. El desarrollo rápido de aplicaciones (RAD por sus siglas en inglés, Rapid Application Development) fue diseñado originalmente por James Martin, un reconocido consultor de tecnologías de la información, en la década de los años 80. (Martin, 1990)

Las herramientas CASE (Por sus siglas en inglés: Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora), son herramientas de carácter tecnológico que contribuyen a la automatización del proceso de ciclo de vida de una o varias fases del desarrollo de productos de software. Las herramientas CASE permiten analizar datos, crear interfaces gráficas y generar código fuente. La generación de código fuente es considerada como una de las mayores ventajas de emplear este tipo de herramientas. De acuerdo a las funciones que se cumplen durante el ciclo de vida del proyecto de software, es posible clasificar las herramientas CASE en: I-CASE (Integrated CASE, CASE integrado) empleadas en todas las fases del ciclo de vida del proceso de software; y, U-CASE (Upper CASE - CASE de alto nivel o superior) automatización de las primeras fases del desarrollo del producto que software.

Tabla 1

Clasificación de herramientas CASE

Tipo	Ventaja	Desventaja
I-CASE	- Integra el ciclo de vida. - Mejoras de productividad a mediana plazo. - Eficiente mantenimiento de sistemas.	- Son eficientes solo para soluciones complejas. - Costoso.
U-CASE	- Mejoras de productividad a corto plazo. - Eficiente mantenimiento de sistemas.	- No garantiza la consistencia de los resultados a nivel corporativo. - No garantiza la eficiencia del análisis y diseño.

Fuente: (“HERRAMIENTAS CASE: 3. CLASIFICACION,” 2008)

A nivel de la experiencia del cliente, la utilización de las herramientas CASE, supone los siguientes beneficios: facilidad para realizar la revisión por parte de los usuarios de las aplicaciones a través del empleo de prototipos, lo

que permite la modificación de los prototipos sin alterar el código fuente generado; y, aumento de la satisfacción de cumplimiento de los requerimientos; por disponer de prototipos que brindan una concepción amplia del producto final a ser entregado.

El mercado actual disponible de una alta gama de herramientas CASE, destacamos las siguientes: Microsoft Project, Rational Rose, MagicDraw, JDeveloper, entre otras.

Microsoft Project.- Permite la administración de proyectos, considerándose la mejor herramienta para la administración de recursos, de tareas, presupuesto y más. Esta herramienta CASE es propiedad de Microsoft.

Rational Rose.- "IBM Rational Rose Enterprise proporciona un conjunto de prestaciones controladas por modelo para desarrollar muchas aplicaciones de software, incluidas aplicaciones Ada, ANSI C++, C++, CORBA, Java, Java EE, Visual C++ y Visual Basic. El software permite acelerar el desarrollo de estas aplicaciones con código generado a partir de modelos visuales mediante el lenguaje UML (Unified Modeling Language)". ("IBM - Rational Rose Enterprise," 2015)

MagicDraw.- Es una herramienta con atributos que permiten el modelado basado en UML. MagicDraw, permite el análisis y diseño de los productos de software y del modelado de base de datos.

JDeveloper: Es un entorno de desarrollo integrado desarrollado por Oracle Corporation para los lenguajes Java, HTML, XML, SQL, PL/SQL, Javascript, PHP, Oracle ADF, UML y otros". ("JDeveloper - Wikipedia, la enciclopedia libre," 2015). Esta herramienta CASE es propiedad de Oracle Corporation.

2.2.2 Caracterización gnoseologica de las normas y estándares de calidad en el proceso de desarrollo de software

A. Conceptos

A.1 Norma y estándar

La Real Academia Española, define la palabra norma como “Regla que se debe seguir o a que se deben ajustar las conductas, tareas, actividades, etc.”. (“Real Academia Española. Diccionario Usual.,” n.d.-a) Un estándar, según la RAE, “sirve como tipo, modelo, norma, patrón o referencia” (“Real Academia Española. Diccionario Usual.,” n.d.-b)

La ingeniería del software es una rama de la ingeniería cuyo fin es la elaboración de software que cumpla y satisfaga los requerimientos de los interesados o clientes. En la medida de satisfacer el cumplimiento de los requerimientos, un estándar en la ingeniería del software se constituye como un conjunto de criterios de desarrollo que permiten la entrega, en lo posible, de un producto de software que garantice la calidad del mismo.

B. Principales estándares

“International Standarization Organization” define para la evaluación de los productos y mejoras de proceso las siguientes normas:

ISO/IEC 9000-3:2004.- Guía para el desarrollo, la aplicación y mantenimiento de productos de software.

ISO/IEC 12207:1995.- Específica los procesos que constituyen el ciclo de vida del software.

ISO/IEC 12207:2008.- Define las actividades y tareas que se aplicarán durante la adquisición de un producto de software o similares.

ISO/IEC 9126:2001.- Evalúa la calidad del software y permita establecer las características de la calidad.

ISO/IEC 15504:2004.- Define un conjunto de actividades para la evaluación y mejorar de los procesos. Esta norma se aplica en conjunto con ISO/IEC 12207, que permite la evaluación y la mejora en la calidad del proceso de desarrollo y mantenimiento de software.

ISO/IEC 14598:1999.- Define un conjunto de actividades que permiten la evaluación de un producto de software.

B.1 Estándar ISO/IEC 9126

El estándar ISO9126 (2001), es una guía que define a un conjunto de características que deben ser cumplidas para que un producto de software sea considerado como de calidad. El estándar ISO/IEC 9126 (2001), está dividido en:

ISO/IEC 9126-1 (2001): un modelo de calidad del software, estructurado en características y sub características.
ISO/IEC TR 9126-2 (2003): métricas externas que miden los atributos de las características de calidad externa (seis características) definidas en la norma ISO/IEC 9126-1 (2001), a más de representar una guía para su correcta aplicación.
ISO/IEC TR 9126-3 (2003): métricas internas para medir atributos de seis características de calidad interna definidas en la ISO/IEC 9126-1 (2001).
ISO/IEC TR 9126-4: métricas de calidad en uso para medir los atributos definidos en la ISO/IEC 9126-1 (2001).

Las características descriptas en la ISO/IEC 9126 (2001) son: usabilidad (Fácil de usar), funcionalidad (Cumple con los requerimientos de los usuarios), confiabilidad (Disponibilidad), eficiencia (Relación entre el rendimiento y la cantidad de recursos asignados), mantenimiento (Cambios en el productos) y portabilidad (Cambios a otros ambientes).

B.2 ISO / IEC 12207: 2008

La norma ISO / IEC 12207:2008 es un estándar que define los lineamientos generales para el ciclo de vida de los procesos de desarrollo de productos de software. La norma define los procesos, las actividades y tareas que se ejecutarán durante el desarrollo e implementación de un software. (“ISO/IEC 12207:2008 - Systems and software engineering -- Software life cycle processes,” 2008)

Se considera como proceso, a un conjunto de actividades ejecutadas secuencialmente para cumplir un propósito específico. Las actividades son un conjunto de tareas, y una "tarea es una acción que transforma entradas en salidas".

“*Los procesos se clasifican en tres tipos: Procesos principales, procesos de soporte y procesos de la organización. Los procesos de soporte y de organización deben existir independientemente de la organización y del proyecto ejecutado. Los procesos principales se instancian de acuerdo con la situación particular*”. (“ISO/IEC 12207 - Wikipedia, la enciclopedia libre,” 2015)

Procesos principales, constituido por: adquisición, suministro, desarrollo, operación y mantenimiento.

Procesos de soporte, constituido por: documentación, gestión de la configuración, aseguramiento de calidad, verificación, validación, revisión conjunta, auditoría y resolución de problemas.

Procesos de la organización, constituido por: gestión, infraestructura, mejora y recursos humanos.

El estándar está constituido de la siguiente manera:

Figura 3 Procesos de la norma ISO/IEC 12207

Fuente: (Orna, 2013)

B.3 CMMi (Capability Maturity Model Integration o Integración de modelos de madurez de capacidades)

"Es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software". ("Capability Maturity Model Integration - Wikipedia, la enciclopedia libre," 2015)

El modelo agrupa las áreas del proceso en cinco niveles de madurez: inicial, repetible, definido, gestionado y optimizado:

Inicial: la organización no dispone de ambientes para el desarrollo y mantenimiento de aplicaciones de software.

Repetible: la organización dispone de un repositorio de prácticas para la gestión de proyectos.

Definido: a más de buenas prácticas, la organización dispone de procedimientos para la coordinación entre grupos, capacitación, técnicas de ingeniería y más.

Gestionado: la organización disponen de un repositorio de métricas significativas de calidad y productividad.

Optimizado: la organización está dedicada a la mejora continua de los procesos organizacionales.

El modelo de madurez proporción a las organizaciones, generadoras de software, un conjunto de guías que permitirán controlar los procesos para el desarrollo y mantenimiento de software, con la finalidad de establecer una cultura basada en la ingeniería de software y una correcta administración de procesos.

2.3 Antecedentes contextuales

La Cooperativa de Ahorro y Crédito 'El Sagrario' Ltda., es una institución financiera sólida, que tiene 51 años sirviendo a la colectividad del centro del país. Fue creada el 10 de Junio de 1964 y mantiene su oficina matriz en la ciudad de Ambato. Posee 10 agencias a nivel nacional ubicadas en las ciudades de: Ambato, Riobamba, Latacunga, Guaranda, Quito, Babahoyo y Milagro, posee alrededor de 95.000 socios, con un total de activos de casi 121 millones de dólares americanos ("boletines_mensuales - SEPS," 2015), se encuentra catalogada por la SEPS (Superintendencia de economía popular y solidaria) en el segmento 1, donde se encuentran las Cooperativas más grandes del Ecuador, segmentadas por el monto de activos que posee las instituciones financieras del país. Esta institución se caracteriza por brindar productos y servicios innovadores a sus socios y clientes, basadas en una estructura financiera sólida, con personal capacitado y tecnología confiable manteniendo siempre los principios cooperativos.

El siguiente gráfico muestra la distribución de socios en las ciudades del Ecuador donde se encuentran ubicadas las oficinas de la Cooperativa de Ahorro y Crédito El Sagrario Ltda.

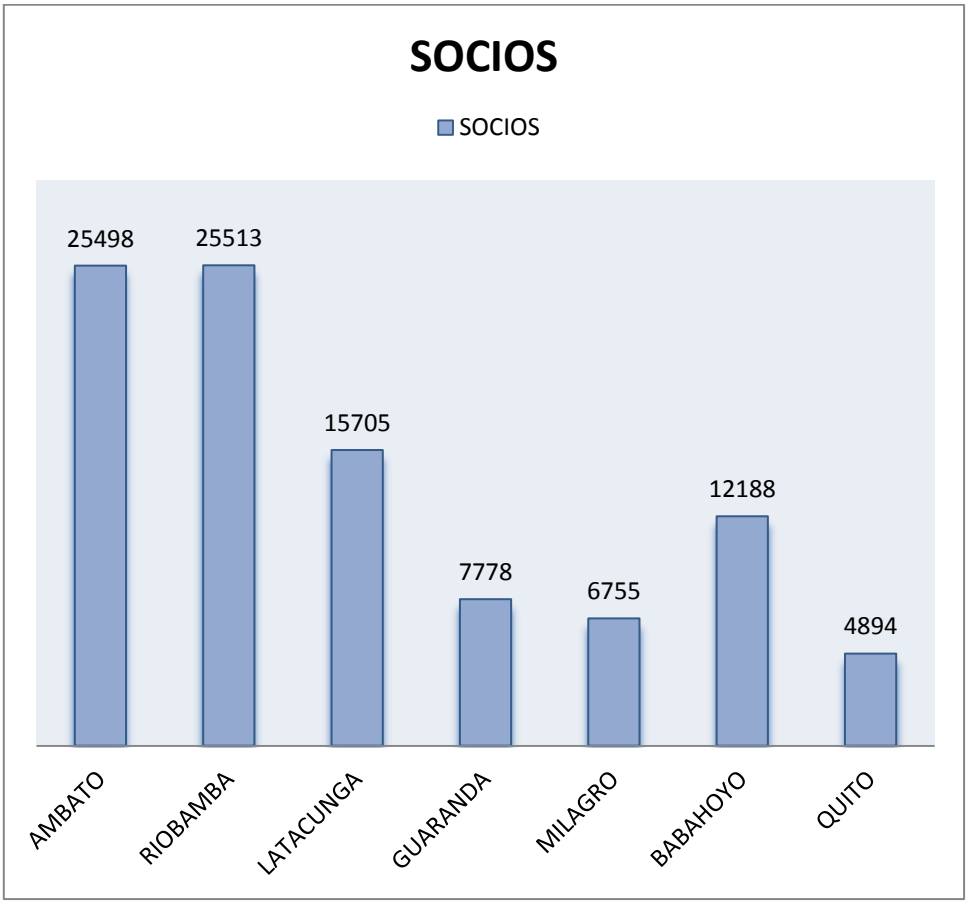


Figura 4 Distribución de socios en el Ecuador

Para la justificación científica del problema del problema se ha escogido a la encuesta como instrumento de investigación. Se elaboró dos tipos de encuestas, la primera enfocada a los miembros que conforman el departamento de tecnología de la Cooperativa de Ahorro y Crédito El Sagrario Ltda, para lo cual se encuestó a 6 miembros del área y la otra encuesta dirigida a 10 usuarios que utilizan el software elaborado por el departamento de tecnología de la institución. Los resultados obtenidos se muestran a continuación.

Encuesta a usuarios

1. **¿Al momento de colocar una actualización del Sistema informático cumple con los requerimientos que solicitó?**

Nunca	1
Pocas veces	6
Frecuentemente	2
Siempre	1

Resultados: Se obtuvo como resultado, que un 10% de respuestas manifiestan que no cumplen con los requerimientos solicitados, el 60% pocas veces, el 20% que es frecuente y un 10 % manifiesta que siempre cumple con los requisitos, lo cual confirma una deficiencia en el proceso de desarrollo de software.

2. **¿El tiempo ofrecido por el Departamento de tecnología de la Cooperativa, para el desarrollo de un requerimiento se cumple?**

Nunca	7
Pocas veces	2
Frecuentemente	1
Siempre	0

Resultados: Se obtuvo como resultado, que un 70% de respuestas manifiestan que no se cumple con el tiempo estipulado, el 20% pocas veces y un 10% frecuentemente, lo cual confirma una deficiencia en la estimación de tiempos para el desarrollo de software.

3. **¿Al momento de colocar una actualización del Sistema informático tiene fallos?**

Nunca	0
Pocas veces	0

Frecuentemente	5
Siempre	5

Resultados: Se obtuvo como resultado, que un 50% de respuestas manifiestan que los aplicativos informáticos siempre tienen fallos mientras que el otro 50% es frecuente, lo cual confirma una deficiencia en el control de calidad de los productos de software.

4. ¿Participa activamente en el proceso de desarrollo de software?

Nunca	1
Pocas veces	5
Frecuentemente	3
Siempre	1

Resultados: Se obtuvo como resultado, que un 10% de respuestas manifiestan que nunca han participado del proceso, el 50% pocas veces, el 30% frecuentemente y solo un 10 % siempre participan, lo cual confirma una deficiencia en la recopilación de requerimiento.

5. ¿Al momento de colocar una actualización del Sistema informático recibe capacitación y manuales de usuario para utilizar el Sistema?

Nunca	9
Pocas veces	1
Frecuentemente	0
Siempre	0

Resultados: Se obtuvo como resultado, que un 90% de respuestas manifiestan que nunca han recibido capacitación ni manuales y el 10% pocas veces, lo cual confirma un problema al momento de utilizar los productos de software.

6. ¿Cuándo el Sistema informático tiene un fallo recibe ayuda de forma oportuna?

Nunca	4
Pocas veces	4
Frecuentemente	1
Siempre	1

Resultados: Se obtuvo como resultado, que un 40% de respuestas manifiestan que nunca reciben ayuda oportuna y el 40% pocas veces, el 10% frecuentemente y solo un 10% siempre, lo cual confirma un problema al momento de brindar soporte en el aplicativo informático.

Encuesta para los miembros del departamento de tecnología de la institución

1. ¿En el proceso de desarrollo de software que modelo aplican habitualmente?

Modelo en V	6
R.U.P	0
Metodologías Ágiles	0
Ninguno	0

Resultados: Se obtuvo como resultado, que un 100% de respuestas manifiestan que utilizan el modelo en V como modelo de desarrollo de software, lo cual confirma que no se tiene un proceso específico para desarrollar de software.

2. ¿Utilizan normas o estándares para el desarrollo de software?

Si	0
No	6

Resultados: Se obtuvo como resultado, que un 100% de respuestas manifiestan que no utilizan normas ni estándares, lo cual evidencia uno de los motivos por los cuales el proceso de desarrollo de software es deficiente.

3. ¿La entrega de un proyecto se lo realiza en los tiempos establecidos?

Nunca	6
Pocas veces	0
Frecuentemente	0
Siempre	0

Resultados: Se obtuvo como resultado, que un 100% de respuestas manifiestan que nunca se entregan los proyectos a tiempo, lo cual confirma un problema en la administración de recursos y tiempos en el proceso de desarrollo de software.

4. ¿Una vez liberado un producto de software, el mismo cumple con las expectativas del usuario?

Nunca	0
Pocas veces	4
Frecuentemente	2
Siempre	0

Resultados: Se obtuvo como resultado, que un 66% de respuestas manifiestan que pocas veces cumplen las expectativas de los usuarios y un 44% que es frecuente, lo cual confirma que no existe sinergia entre el dueño del proceso y el área de desarrollo de Software.

5. ¿Quién realiza las pruebas de software desarrollado?

El área de calidad de software	0
Cada desarrollador	6

El usuario final	0
Nadie	0

Resultados: Se obtuvo como resultado, que un 100% de respuestas manifiestan que solo el desarrollador realiza control de calidad, lo cual confirma la razón de los fallos que tiene el sistema informático al momento de ser liberado en producción.

6. ¿Existen normas o estándares para la verificación de cada etapa de desarrollo de software?

Si	0
No	6

Resultados: Se obtuvo como resultado, que un 100% de respuestas manifiestan que no existen normas ni estándares para verificar cada etapa, lo cual confirma que no existen lineamientos para verificar cada etapa del desarrollo de software.

7. ¿Existen normas o estándares para la validación de cada etapa de desarrollo de software?

Si	0
No	6

Resultados: Se obtuvo como resultado, que un 100% de respuestas manifiestan que no existen normas ni estándares para validar cada etapa, lo cual confirma que no existen lineamientos para validar cada etapa del desarrollo de software.

En esta sección de preguntas se puede evidenciar los problemas que tiene el área de desarrollo de software al momento de elaborar productos de

software y cuál es la percepción de los usuarios que utilizan el Sistema informático de la institución.

La siguiente sección de preguntas nos ayudará a discernir las soluciones a estos inconvenientes.

Encuesta a usuarios

7. ¿Considera usted que si tuviera reuniones permanentes con el departamento de desarrollo de software el producto de software cumpliría con sus necesidades?

Si	10
No	0

Resultados: Se obtuvo como resultado, que un 100% de respuestas manifiestan que manteniendo reuniones permanentes con los usuarios se puede mejorar la satisfacción de los productos de software elaborados por el departamento de TIC.

8. ¿Considera usted que si existiera una mejor planificación de actividades por parte del área de desarrollo de software, el tiempo de entrega de los productos de software sería mejor?

Si	10
No	0

Resultados: Se obtuvo como resultado, que un 100% de respuestas manifiestan que con una mejor planificación se puede entregar a tiempo los productos de software elaborados por el departamento de TIC.

9. ¿Considera usted que si existiera un departamento de control de calidad el número de fallos al momento de realizar una actualización en el Sistema reduciría?

Si	10
No	0

Resultados: Se obtuvo como resultado, que un 100% de respuestas manifiestan que con la existencia de un departamento de control de calidad el número de fallos al momento de la liberación en producción va a disminuir.

Encuesta para los miembros del departamento de tecnología de la institución.

8. ¿Si se aplica un modelo o marco de trabajo que estructure todos los procesos de desarrollo de software, serían más eficientes las entregas de los productos de software?

Si	6
No	0

Resultados: Se obtuvo como resultado, que un 100% de respuestas manifiestan que con la elaboración de un marco de trabajo se va a mejorar la calidad de los productos de software.

9. ¿Si se definen las normas, estándares y modelos a ser aplicados en todo el proceso de desarrollo de software, los resultados obtenidos serían más óptimos?

Si	6
No	0

Resultados: Se obtuvo como resultado, que un 100% de respuestas manifiestan que con la implementación de normas y estándares el proceso de desarrollo de software va a ser más óptimo.

10. ¿Considera aplicable en nuestros productos de software, un marco de trabajo que integre todo el proceso de desarrollo de software?

Si	6
No	0

Resultados: Se obtuvo como resultado, que un 100% de respuestas manifiestan que es viable la elaboración de un marco de trabajo que nos ayude a organizar, planificar y desarrollar productos de software de buena calidad.

El instrumento (encuesta) aplicado tanto al personal del área de tecnología y a los usuarios que utilizan el Sistema informático de la Cooperativa de Ahorro y Crédito El Sagrario Ltda. se puede apreciar en los anexos A y B.

Como se puede observar en las encuestas aplicadas tanto a los usuarios que utilizan el software elaborado en la institución, como a los integrantes del departamento de tecnología, un 90% manifiesta que no se cumple con los requerimientos solicitados por el usuario, 100% de los usuarios afirma que no se cumplen con los tiempos establecidos, los aplicativos tienen fallos al momento de ser liberados en producción y no se realiza la capacitación debida para utilizar los aplicativos informáticos desarrollados; un 90% de los usuarios manifiestan que no reciben ayuda oportuna cuando el Sistema falla, un 100% de los integrantes del departamento de tecnología, informan que no se utilizan normas y estándares para el desarrollo de software, no poseen un área de control de calidad, no se verifica ni se valida cada etapa del desarrollo de software. Demostrando de esta manera que el departamento de tecnología de la Cooperativa de Ahorro y Crédito El Sagrario Ltda. tiene un grave problema al momento de desarrollar software.

Al consultar sobre la necesidad de crear un marco de trabajo en el proceso de desarrollo de software que permita mejorar la calidad de los productos de software elaborados. El 100% de los encuestados estuvieron de acuerdo, ya que consideran que es factible aplicarlo en la institución y va a permitir entregar productos de software en el tiempo acordado, con los requerimientos solicitados y con la cantidad mínima de errores. Esto se puede lograr validando y verificando cada etapa del desarrollo de software, estableciendo normas y estándares que ayuden a mejorar el proceso.

2.4 Conclusiones del capítulo

Los aplicativos informáticos se han convertido en una herramienta fundamental para la evolución de las empresas a nivel mundial, por este motivo se ha destinado mucho esfuerzo para crear mecanismos que permitan mejorar la calidad del desarrollo de software a través de métodos, metodologías, normas y estándares que se puedan aplicar dependiendo de las necesidades y circunstancias que cada organización tiene.

CAPÍTULO III

DESARROLLO DEL MARCO DE TRABAJO

3. Introducción

El establecer un marco de trabajo para el desarrollo de software en una empresa es fundamental, puesto que nos permite establecer los lineamientos sobre los cuales se van a elaborar los productos de software. El propósito de crear un marco de trabajo, es mejorar la calidad del software estableciendo estándares, políticas y buenas prácticas que permitan mejorar la productividad del equipo que interviene en el proceso de desarrollo de software.

La calidad de software se basa en la interacción de tres pilares como son productos, procesos y personas tal como lo muestra el siguiente gráfico.

Figura 5 Calidad del Software
Fuente: (Roger Pressman, 2005)

En la actualidad los principales esfuerzos para mejorar la calidad de software se centran en los procesos. Los modelos de mejora para la calidad de procesos se dividen en modelos de procesos y modelos de evaluación de procesos.

Los modelos de procesos describen la utilización de buenas prácticas, la definición de procesos y las relaciones que deben tener entre estos.

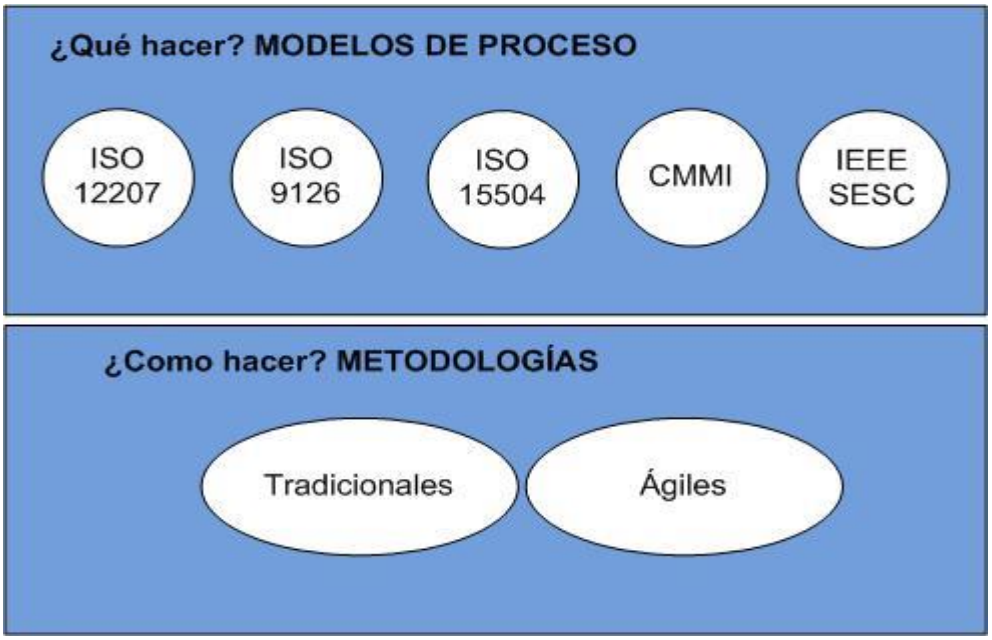


Figura 6 Calidad en el proceso de software

En este capítulo se crea el marco de trabajo para La Cooperativa de Ahorro y Crédito El Sagrario Ltda. para lo cual empezaremos seleccionando los estándares del proceso de desarrollo de software, posteriormente la metodología de desarrollo, para finalmente construir el marco de trabajo que integre el estándar y la metodología a las condiciones específicas del contexto estudiado.

3.1. Selección del estándar para el proceso de desarrollo de software

La implementación de normas y estándares dentro de una organización juega un papel importante dentro del proceso de desarrollo de software, ya que permite organizar de mejor forma los recursos con los que cuenta la empresa, con la finalidad de optimizar los procesos y desarrollar productos de software de calidad, estableciendo sinergia entre todos los miembros que intervienen en el proceso.

Para elegir el modelo que mejor se adapte a las necesidades de la Cooperativa de Ahorro y Crédito El Sagrario Ltda. nos vamos ayudar del estudio realizado por (Vega Zepeda, Gasca Hurtado, & Echeverry Arias, 2012).

Para lo cual nos vamos a centrar en 3 estándares de calidad: ISO 9126, ISO 15504, IEEE/IEC 12207.

Este modelo propone varios criterios a ser analizados por cada uno de los estándares que se van analizar y se calificar de acuerdo a lo siguiente:

- “H” (3) representa un alto grado de cumplimiento del criterio.
- “M” (2) representa que el criterio se cumple parcialmente.
- “L” (1) representa que el criterio no satisface lo suficiente.

Aplicando estos criterios a nuestro contexto, se obtienen los siguientes resultados:

Tabla 2

Comparación entre normas de desarrollo de software

Criterios	ISO 9126	ISO 15504	IEEE/IEC 12207
Es ampliamente conocido y utilizado.	M	M	M
Ha sido reconocido por un organismo de reconocido prestigio internacional.	H	H	H
La información y estructura propia es pública y se encuentra disponible.	H	L	H
Incluye recomendaciones para la gestión de calidad y el desarrollo de software.	M	H	M
Proporciona información actualizada.	M	L	M
Relevancia de la institución proponente.	H	H	H

CONTINÚA

Criterios	ISO 9126	ISO 15504	IEEE/IEC 12207
Enfoque de la propuesta en términos de la calidad.	L	M	H
Enfoque de la propuesta en términos de la calidad del proceso.	L	L	H
Mayor permeabilidad de la propuesta en términos de otros procesos o buenas prácticas	L	L	M
Hace referencia a los procesos para planificación	M	L	H
Estimación de recursos	L	L	M
Puntos clave para estimar	M	L	H
Realizar un adecuado y sencillo análisis de riesgos	L	L	M
Establece un ámbito de proyecto	M	M	H
Sumatoria	**18**	**17**	**19**

Los resultados obtenidos de la aplicación del modelo que se ajustan a nuestro contexto, nos arroja que la norma ISO/IEC 12207:2008 es la que mejor se ajusta, ya que esta norma es una herramienta de gestión para todos los procesos vinculados con el desarrollo de software, que establece un marco de referencia común para ser utilizado dentro del proceso de software con la finalidad de hablar el mismo lenguaje entre todos los entes que intervienen en el proceso de desarrollo de software, esta norma abarca desde la conceptualización de ideas hasta la liberación del producto en producción y el mantenimiento del mismo.

Organizar los procesos de desarrollo de software con la norma ISO/IEC 12207:2008 en una institución es importante ya que nos permite dar el primer paso hacia una adecuada estandarización de procesos para posteriormente comenzar a evaluar dicho proceso a través de otra norma como la ISO 15504 o pensar a futuro en implementar niveles de madurez a través de CMMI.

3.2. Selección de la metodología para el proceso de desarrollo de software

En la actualidad las metodologías ágiles han sido bien acogidas por pequeñas y medianas empresas, ya que permiten administrar equipos de desarrollo de software pequeños sin la necesidad de invertir un gran presupuesto.

Las metodologías ágiles se caracterizan por el desarrollo iterativo e incremental; la simplicidad de la implementación; las entregas frecuentes; la priorización de los requerimientos o características a desarrollar a cargo del cliente; y la cooperación entre desarrolladores y clientes. Las metodologías ágiles dan como un hecho que los requerimientos van a cambiar durante el proceso de desarrollo (Cadavid, Daniel, Martínez, & Vélez, 2013)

A continuación se muestra un cuadro comparativo de las metodologías ágiles más utilizadas a nivel mundial.

Figura 7 Metodologías ágiles más utilizadas

Fuente: (VersionOne, 2013) p.4

Los resultados obtenidos en este gráfico fueron el resultado de una encuesta realizada entre agosto y octubre del 2013 a 3501 personas vinculadas con el desarrollo ágil de software, el cual muestra que Scrum es la metodología más utilizada con un 55% y aplicando otras variantes ocupa el 73% dentro de las metodologías ágiles, esto se debe a su gran flexibilidad.

Para la elegir cuál de estas metodologías ágiles es la más adecuada para implementarla en la Cooperativa de Ahorro y Crédito El Sagrario Ltda. se va a utilizar el método propuesto en el trabajo de investigación de (Sáez Martínez, 2013).

Este estudio propone cuatro secciones: Aplicabilidad, nivel de abstracción de normas y directrices, actividades cubiertas por la metodología y productos de las actividades de la metodología, cada una de estas secciones tiene preguntas relacionadas a cada ítem; las mismas que deben ser contestadas de acuerdo a las opciones que cada pregunta propone. Este estudio contiene una plantilla base con las características de cada metodología que se ponderan con uno y cero de acuerdo a la necesidad de cada organización, en esta matriz se analiza las siguientes metodologías ágiles: ASD, AUP, Crystal, DSDM, XP, FDD, LSD, Kanban, Scrum, Scrumban.

Para aplicar este estudio primero se contesta el cuestionario, posteriormente se elabora una matriz con las metodologías ágiles que se van a evaluar, cada respuesta tiene un valor de uno o cero de acuerdo a la respuesta obtenida. A continuación se visualiza la matriz sobre la cual se pondera los datos.

			ASD	AUP	Crystal	DSDM	XP	FDD	LSD	Kanban	Scrum	Scrumban
PROCESOS Y PRODUCTOS	¿Cómo están caracterizados los procesos de la metodología y cuáles son los productos de sus actividades?	Nivel de abstracción de las normas y directrices										
		Gestión de proyectos	V	V	V	V	F	V	F	F	V	F
		Descripción de procesos	V	V	V	V	V	V	F	F	V	V
		Normas y orientaciones concretas sobre las actividades y productos	F	V	F	F	V	V	F	F	V	V
		Actividades cubiertas por la metodología										
		Puesta en marcha del proyecto	V	V	F	V	V	V	F	F	V	F
		Definición de requisitos	V	V	F	V	V	V	V	F	V	F
		Modelado	V	V	V	V	V	V	F	F	V	F
		Código	V	V	V	V	V	V	V	V	V	V
		Pruebas unitarias	V	V	V	V	V	V	V	V	V	V
		Pruebas de integración	V	V	V	V	V	V	V	V	V	V
		Prueba del sistema	V	V	V	V	V	V	V	V	V	V
		Prueba de aceptación	V	V	V	V	V	V	V	V	V	V
		Control de calidad	V	V	F	F	F	V	V	F	F	F
		Uso del sistema	F	V	F	V	F	F	F	F	F	F
		Productos de las actividades de la metodología										
		Diseño del modelo	F	V	F	V	F	V	F	F	V	F
		Código fuente comentado	V	V	V	V	V	V	V	F	V	V
		Ejecutable	V	V	V	V	V	V	V	V	V	V
		Pruebas unitarias	V	V	V	V	V	V	V	V	V	V
		Pruebas de integración	V	V	V	V	V	V	V	V	V	V
		Pruebas de sistema	V	V	V	V	V	V	V	V	V	V
		Pruebas de aceptación	V	V	V	V	V	V	V	V	V	V
		Informes de calidad	V	V	F	F	F	V	V	F	F	F
		Documentación de usuario	F	V	F	V	F	V	V	F	V	F

Figura 8 Valoración de los procesos y productos de las metodologías

Fuente: (Sáez Martínez, 2013), p.98

Tomando en consideración las metodologías más utilizadas de la Figura3.3, las cuales van a ser: Scrum, Scrumban, Kanban y XP. El primer paso que propone dicha investigación es llenar el siguiente cuestionario.

Tabla 3

Análisis de metodologías para la organización en estudio

Aplicabilidad		Respuesta
Tamaño del proyecto(Pequeño, Grande)		P
Complejidad del proyecto(Bajo, Alto)		A
Riesgo del proyecto(Bajo, Alto)		A
Tamaño del equipo(Pequeño, grande)		P
Grado de interacción con el cliente(Bajo, Alto)		A
Grado de interacción con los clientes finales(Bajo, Alto)		A
Grado de interacción entre los miembros del equipo(Bajo, Alto)		A
Grado de Innovación(Bajo, Alto)		B
Organización del equipo (auto-organización, jerárquica)		AO
Nivel de abstracción de las normas y directrices		**Respuesta**
Gestión de proyectos(Verdadero, Falso)		V
Descripción de procesos(Verdadero, Falso)		V
Normas y orientaciones concretas sobre las actividades y productos(Verdadero, Falso)		V
Actividades cubiertas por la metodología		**Respuesta**
1	Puesta en marcha del proyecto(Verdadero, Falso)	V
2	Definición de requisitos(Verdadero, Falso)	V
3	Modelado(Verdadero, Falso)	V
4	Código(Verdadero, Falso)	V
5	Pruebas unitarias(Verdadero, Falso)	V
6	Pruebas de integración(Verdadero, Falso)	V
7	Prueba del Sistema(Verdadero, Falso)	V
8	Prueba de aceptación(Verdadero, Falso)	V

CONTINÚA

	Actividades cubiertas por la metodología	Respuesta
9	Prueba de aceptación(Verdadero, Falso)	V
10	Control de calidad(Verdadero, Falso)	V
	Productos de las actividades de la metodología	**Respuesta**
	Diseño del modelo(Verdadero, Falso)	V
	Código fuente comentado(Verdadero, Falso)	V
	Ejecutable(Verdadero, Falso)	V
	Pruebas unitarias(Verdadero, Falso)	V
	Prueba del Sistema(Verdadero, Falso)	V
	Prueba de aceptación(Verdadero, Falso)	V
	Informes de calidad(Verdadero, Falso)	V
	Documentación del usuario(Verdadero, Falso)	V

El segundo paso es elaborar la matriz comparativa, tomando en consideración las metodologías analizar y la matriz base de la Figura 8.

Tabla 4

Matriz comparativa de metodologías.

		XP	Kanban	Scrumban	Scrum
Aplicabilidad	Tamaño del proyecto	1	1	1	1
	Complejidad del proyecto	0	0	1	1
	Riesgo del proyecto	0	0	1	1
	Tamaño del equipo	1	1	1	1
	Grado de interacción con el cliente	1	1	1	1
	Grado de interacción con los clientes finales	0	0	0	1

CONTINÚA

	XP	Kanban	Scrumban	Scrum
Grado de interacción entre los miembros del equipo	1	0	1	1
Grado de Innovación	1	0	1	1
Organización del equipo	1	1	1	1
Nivel de abstracción de las normas y directrices				
Gestión de proyectos	0	0	0	1
Descripción de procesos	1	0	1	1
Normas y orientaciones concretas sobre las actividades y productos	1	0	1	1
Actividades cubiertas por la metodología				
Puesta en marcha del proyecto	1	0	0	1
Definición de requisitos	1	0	0	1
Modelado	1	0	0	1
Código	1	1	1	1
Pruebas unitarias	1	1	1	1
Pruebas de integración	1	1	1	1
Prueba del Sistema	1	1	1	1
Prueba de aceptación	1	1	1	1
Control de calidad	1	0	0	0
Uso del Sistema	0	0	0	0
Productos de las actividades de la metodología				
Diseño del modelo	0	0	0	1
Código fuente comentado	1	0	1	1
Ejecutable	1	1	1	1
Pruebas unitarias	1	1	1	1
Prueba del Sistema	1	1	1	1
Prueba de aceptación	1	1	1	1
Informes de calidad	0	0	0	0
Documentación del usuario	0	0	0	1
Sumatoria :	**22**	**13**	**20**	**27**

De acuerdo a los datos obtenido, aplicando el método propuesto por (Sáez Martínez, 2013), la metodología que se debe aplicar en la institución es Scrum, esto se debe por que la Cooperativa de Ahorro y Crédito El Sagrario

Ltda. es una institución financiera que por la criticidad de su actividad económica requiere una metodología ágil que permita combinar una adecuada gestión de proyectos con documentación que nos permita la trazabilidad de los cambios realizados; pero sin que llegue a ser tan exhaustivo. Esta metodología nos debe permitir interactuar tanto las fases de desarrollo de software como los ambientes de desarrollo que son pruebas, control de calidad y producción.

3.3. Elaboración del marco de trabajo en la Cooperativa de Ahorro y Crédito El Sagrario Ltda.

Aplicando el modelo propuesto por (Vega Zepeda et al., 2012). para establecer el estándar de calidad y el modelo propuesto por (Sáez Martínez, 2013), para elegir la metodología de software, se obtuvo que el marco de trabajo a ser desarrollado debe estar compuesto por la metodología Scrum basado en el estándar de calidad ISO/IEC 12207:2008. El estándar de calidad ISO/IEC 12207:2008, contiene la siguiente estructura:

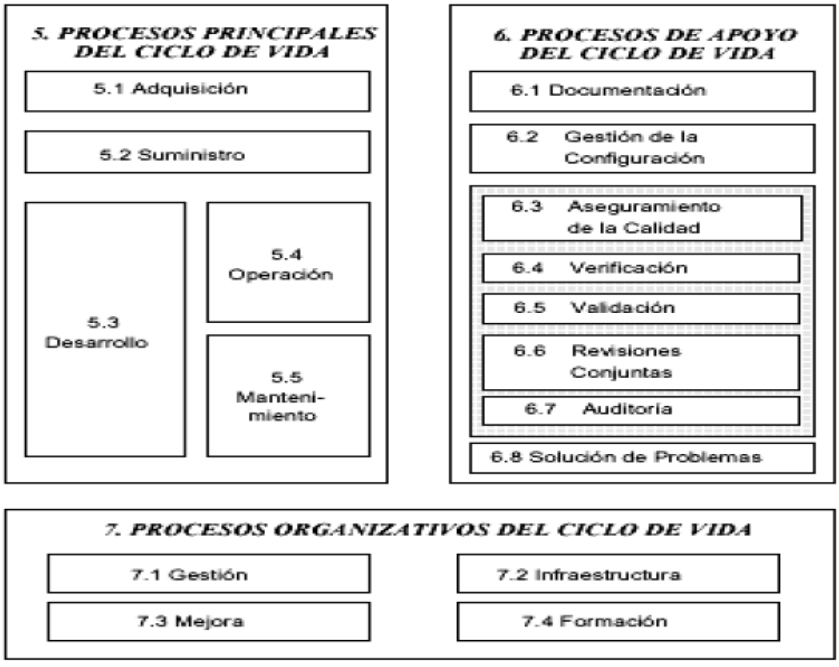


Figura 9 Esquema estructural del IEEE 12207

Fuente: (Orna, 2013)

Tomando en consideración los procesos que establece la norma y aplicándolos al contexto de la organización en estudio podemos eliminar en los procesos principales las actividades de adquisición y suministro, dado que la institución no compra software a proveedores externos ni suministra software a otros entes, adicionalmente se va a fusionar los procesos de desarrollo y mantenimiento ya que dentro de la institución el manejo es el mismo.

Dentro de los procesos de apoyo vamos a utilizar todas las actividades, ya que cada actividad aporta de forma valiosa al proceso de desarrollo de software y contribuye a la calidad del mismo.

Al analizar los procesos organizativos, se encuentra que las actividades de gestión y mejora se adaptan a las necesidades de la organización. Por lo tanto el modelo resultante de la Cooperativa de Ahorro y Crédito se muestra en el siguiente gráfico.

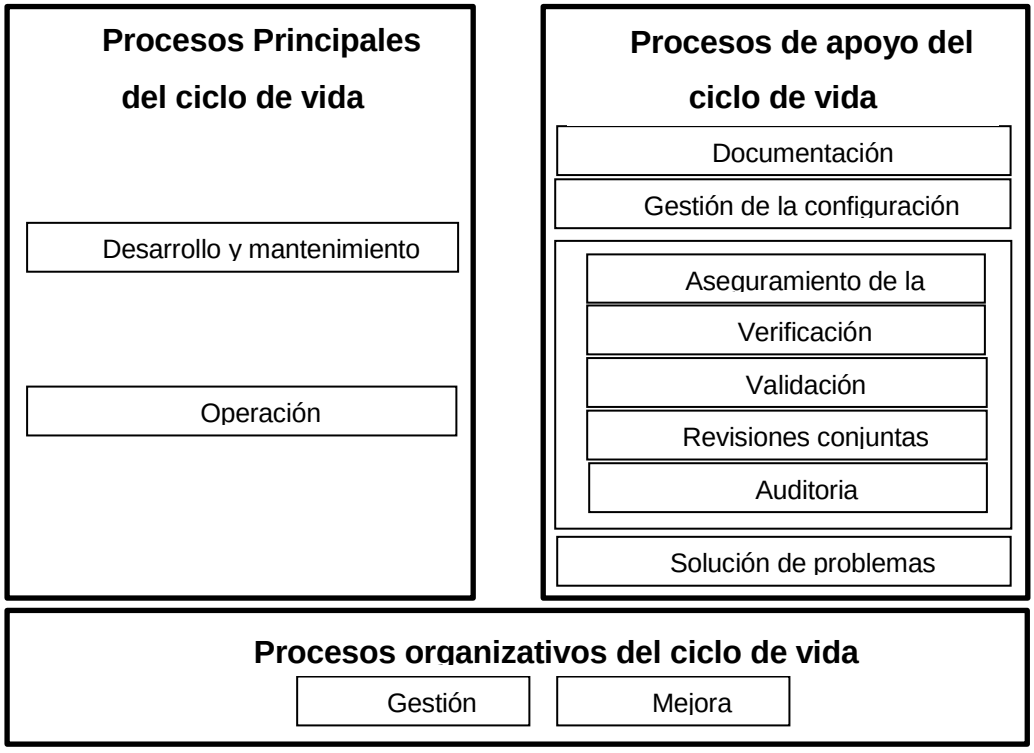


Figura 10 Esquema estructural del IEEE 12207 propuesto

Para evaluar la metodologia Scrum se propone los siguientes roles:

Product Owner.- Es el encargado de gestionar el producto de software, transmite la visión y metas del sprint, ayuda a priorizar los proyectos de acuerdo a las necesidades del negocio. Es la persona que representa al cliente, por lo que es fundamental la interacción continua con el mismo para garantizar que el equipo esté construyendo el producto que necesita.

Scrum Master.- Es el encargado de gestionar el proceso de software, tiene la responsabilidad de resolver cualquier impedimento que ponga en riesgo el avance del proyecto. Es la persona que ayuda al equipo a llegar a un consenso de lo que se puede realizar en un sprint.

Equipo de trabajo.- Se encarga de construir el producto que va a ser utilizado por el cliente, el equipo de trabajo debe descomponer el product backlog en un serie de actividades que deben realizarse en el sprint.

Scrum propone receptar todos los requerimientos que se deben desarrollar para luego priorizar de acuerdo a las necesidades del negocio. Una vez que se tiene claro los requerimientos que se van a desarrollar estos se descomponen en pequeñas funcionalidades que al igual que los requerimientos, se priorizan de acuerdo a las necesidades del dueño del producto para luego poder formar el Sprint con las actividades que se van a desarrollar.

Esta metodología se basa en la interacción continua de todos los miembros que intervienen en el proceso de desarrollo de software, para lo cual se efectúa una serie de reuniones conocidas también como rituales y son las siguientes:

Reunión de planificación del sprint.- En esta reunión intervienen el dueño del proceso, el product owner, el Scrum master y el equipo. El dueño del producto debe explicar las funcionalidades que necesita que estén listas en el próximo incremento.

Revisión del sprint.- En esta reunión se informa las actividades que componen el sprint y en qué orden deben ser desarrolladas.

Scrum diario.- Es una reunión corta de máximo 15 minutos en la cual se exponen los avances e impedimentos para efectuar el trabajo, en esta reunión se realizan compromisos entre los miembros del equipo para efectuar su trabajo.

Retrospectiva del sprint.- Es una reunión que se realiza al finalizar el Sprint y tiene como objetivo principal analizar el cumplimiento del sprint que culminó y los aspectos que se deben mejorar para aplicarlos en el próximo sprint.

El modelo que Scrum propone el siguiente modelo.

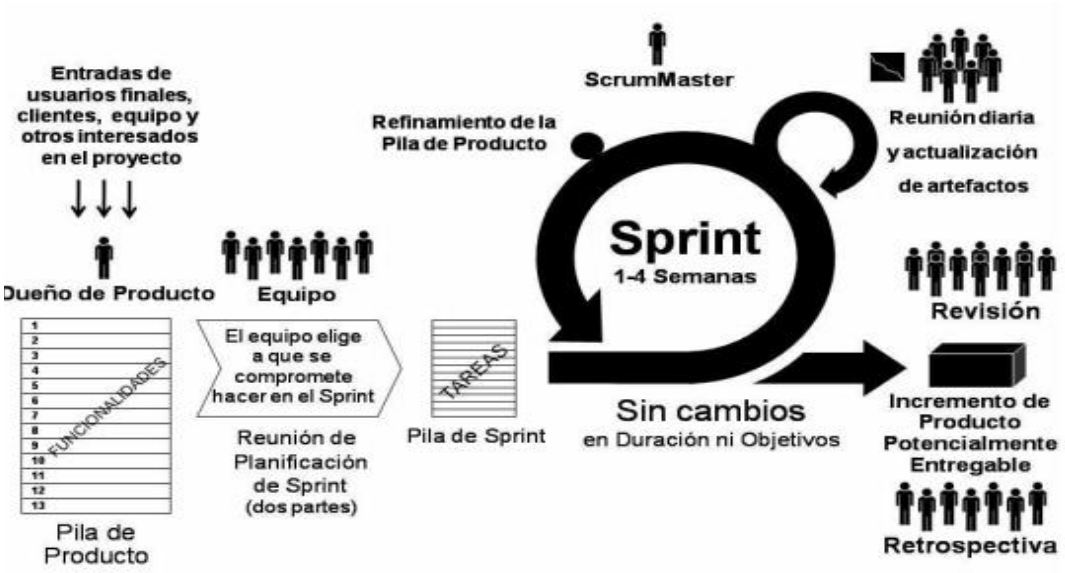


Figura 11 Modelo de la metodología Scrum

Fuente: (Deemer, Por Pete, Gabrielle Benefield, Craig Larman, 2009) p.5

Para adaptar esta metodología a nuestro contexto primero definimos los roles de acuerdo al organigrama estructural de la organización el cual es el siguiente:

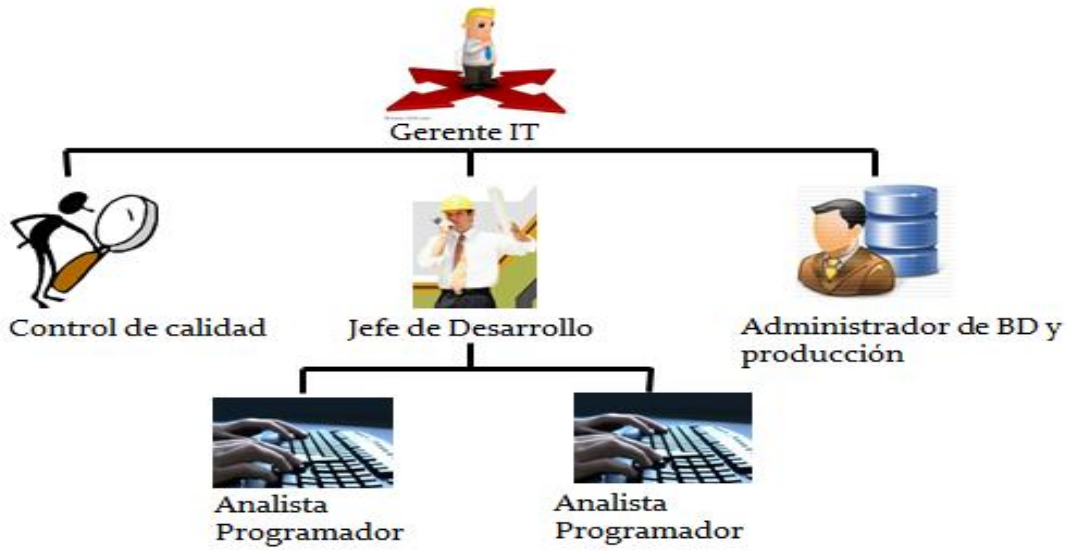


Figura 12 Organigrama del Departamento de tecnología

A continuación se describe las funciones que cumple cada cargo dentro de la institución.

Gerente IT.- Tiene la misión de asegurar que la Institución cuente con tecnología de información y comunicación adecuada para el desarrollo de sus operaciones, contribuyendo a los objetivos institucionales.

Jefe de desarrollo.- Tiene como misión dirigir y supervisar el análisis, diseño e implementación de los sistemas informáticos para la Institución.

Control de calidad.- Es el encargado de asegurar y controlar la calidad de los aplicativos informáticos desarrollados para la Cooperativa, previo a su liberación en ambientes de producción.

Administrador de BD y producción.- Es el encargado de asegurar el correcto funcionamiento de los aplicativos informáticos en producción en cumplimiento a las políticas y procedimientos vigentes en la institución.

Analista programador.- Es el encargado de construir sistemas informáticos que contribuyan a la automatización de los procesos internos en base a requerimientos establecidos.

Tomando en consideración los roles que propone Scrum y los miembros del departamento de tecnología con los que cuenta la institución se establece la siguiente categorización.

Tabla 5

Ajuste de cargos existentes en la institución con roles de Scrum

Cargo	Product Owner	Scrum master	Equipo de trabajo
Gerente de tecnología	X		
Jefe de desarrollo		X	
Ingeniero de control de calidad			X
Administrador de BD y producción			X
Analista programador			X

Con la finalidad de poder filtrar y priorizar los proyectos de acuerdo a las necesidades de la organización se propone crear un comité tecnológico el mismo que estará conformado por el gerente general, gerente de tecnología, control de calidad, auditor interno y jefe de procesos. Este comité se va a encargar de aprobar, negar y establecer las prioridades de los proyectos propuestos.

El modelo metodológico que se propone para la Cooperativa es el siguiente:

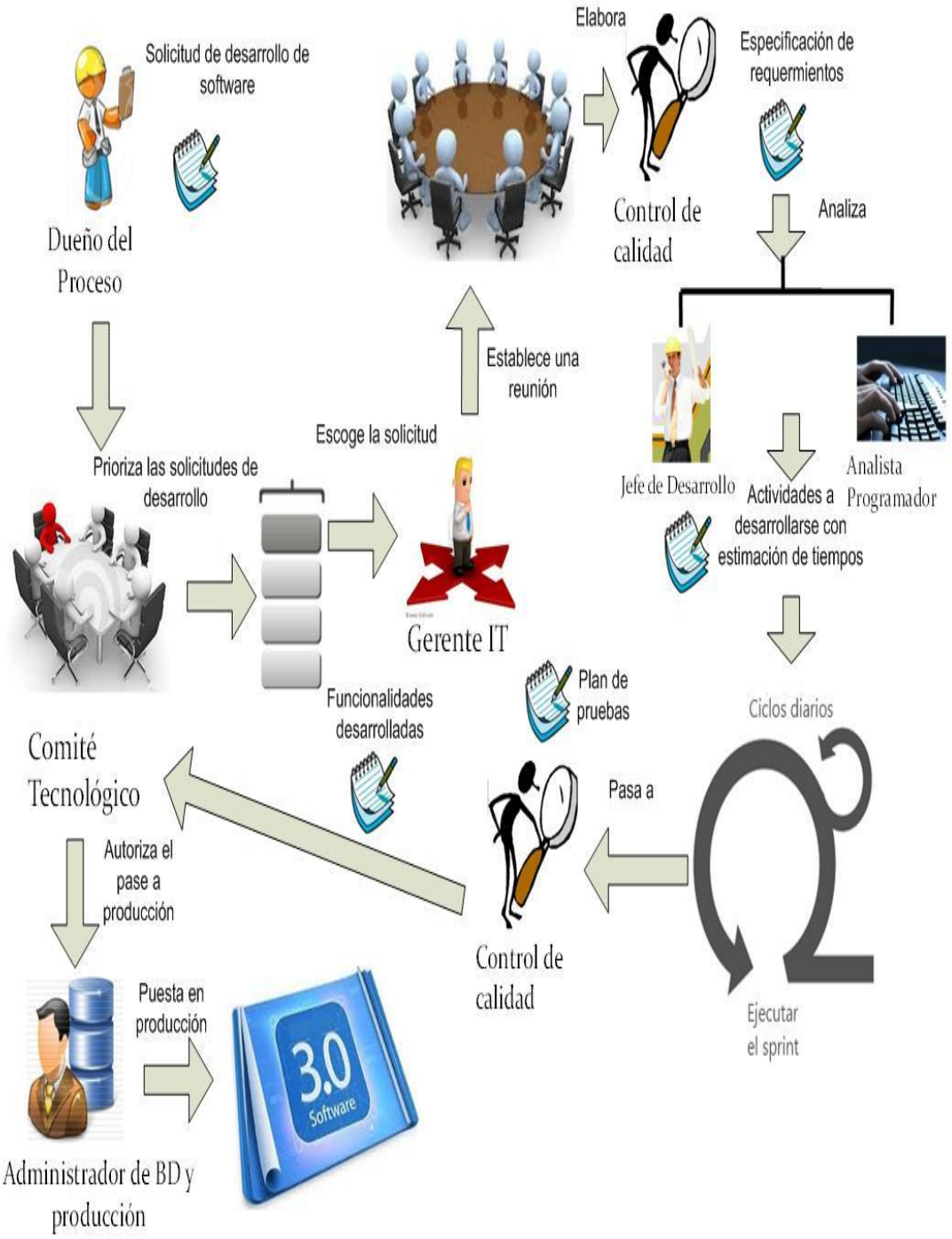


Figura 13 Metodología Scrum propuesto

3.4. Proceso de desarrollo de software propuesto

Procesos principales del ciclo de vida

Desarrollo y mantenimiento

- Elaboración de la solicitud de software.
- Aprobación o negación de la solicitud.
- Reunión de planificación del sprint.
- Análisis y estimación de tiempos de ERS
- Revisión del sprint
- Desarrollo del sprint
- Reunión de scrum diario
- Reunión para validar con el dueño del proceso.
- Finalización del sprint y reunión de retrospectiva.
- Verificación en el área de control de calidad.
- Liberación del producto de software en producción.

Operación

- Reporte de incidentes.
- Atención de incidentes.
- Desarrollo de incidentes.
- Calificación del servicio.

Procesos de apoyo del ciclo de vida

Documentación

- Solicitud de desarrollo de Software.
- Acta de comité tecnológico.
- Especificación de requerimientos.
- Estimación de tiempos.
- Planificación diaria.
- Plan de pruebas.
- Acta de cierre de sprint
- Detalle de funcionalidades para producción.
- Registro de fallas en producción.

Gestión de la configuración

- Versionamiento de especificación de requerimientos.
- Versionamiento del aplicativo informático cada vez que sale a producción.
- Acta de comité tecnológico.

Solución de problemas

- Se resuelven con las reuniones conjuntas.

Aseguramiento de la calidad

- El área de QA es el encargado de verificar el producto de software.

Verificación

- Tanto el jefe de desarrollo y el área de QA verifican el producto de software.
- Indicadores de calidad.

Validación

- El dueño del proceso valida el producto de software

Revisiones conjuntas

- Revisión de ERS con el dueño del proceso y miembros del equipo.
- Revisión del producto de software con dueño del proceso.

Auditoria

- Se realizan auditorías internas y externas

CONTINÚA

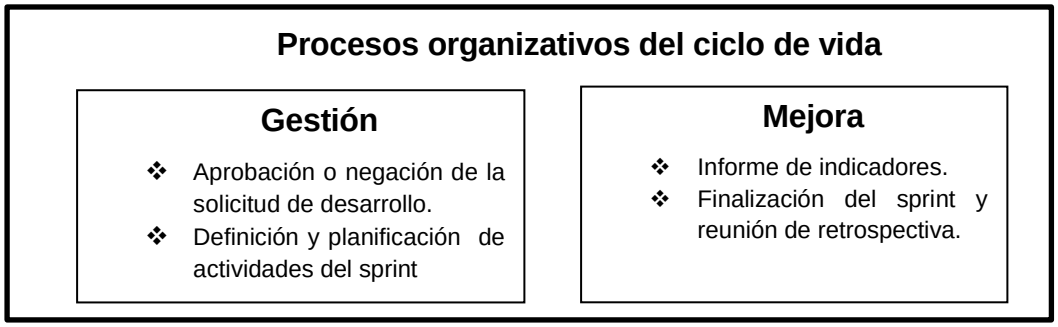


Figura 14 Propuesta de desarrollo y mantenimiento

3.4.1. Procesos principales del ciclo de vida del software

A. Desarrollo y mantenimiento

En la Cooperativa de Ahorro y Crédito El Sagrario Ltda. tanto el desarrollo como el mantenimiento de software se lo trata de igual manera, ya que al tener una plataforma informática ya establecida, el departamento de tecnología continuamente implementa nuevos módulos; lo cual a la larga resulta una modificación al software informático.

A.1 Elaboración de la solicitud de desarrollo de software

La etapa de desarrollo de software de la Cooperativa de Ahorro y Crédito El Sagrario Ltda. comienza con la elaboración de una solicitud de desarrollo de software, la cual es realizada por el dueño del proceso. La plantilla de la solicitud de desarrollo de software se muestra a continuación:

COOPERATIVA DE AHORRO Y CREDITO EL SAGRARIO

SOLICITUD DE REQUERIMIENTOS DE DESARROLLO DE SOFTWARE

INFORMACIÓN DE USUARIO SOLICITANTE

NOMBRES _____
CARGO : _____
FECHA: _____

INFORMACIÓN DE LA SOLICITUD

ACCION A SOLICITAR Nuevo desarrollo ☐ Actualización : ☐

APLICATIVO _____ MODULO _____

DETALLE DE LA SOLICITUD

Solicitado por: Aprobado por:

_____ Acta N° _____
USUARIO SOLICITANTE COMITÉ TECNOLOGICO

DETALLE DE DESARROLLO Y PUESTA EN PRODUCCIÓN

PROGRAMADOR ASIGNADO _____
FECHA DE INICIO _____ FECHA ESTIMADA DE IMPLANTACIÓN _____

REVISIONES REALIZADAS

Fecha	Revisado Por	Observaciones

FECHA DE PUESTA EN PRODUCCION _____ VERSION DEL APLICATIVO ____

Aprobador por:

_____ Acta N°: _____
USUARIO SOLICITANTE COMITE TECNOLOGICO

Figura 15 Solicitud de desarrollo de software

Esta solicitud consta de tres secciones: información de usuario solicitante, información de la solicitud, detalle de desarrollo y puesta en producción

Información de usuario solicitante.- En esta sección se coloca los datos del dueño de proceso que está solicitando un requerimiento. Los datos a llenar son: nombres del solicitante, cargo que ocupa en la organización y la fecha en la cual fue llenada dicha solicitud.

Información de la solicitud.- En esta sección se coloca información del requerimiento que se está solicitando, los datos a ser llenados son los siguientes:

Acción a solicitar.- Se especifica marcando con una “X” si el requerimiento solicitado es un nuevo desarrollo de software, o una actualización de un módulo específico.

Aplicativo.- Se especifica a que aplicativo afecta dicha solicitud, esta puede ser: aplicativo financiero, aplicativo web o aplicativo móvil.

Módulo.- Se especifica a que módulo afecta la solicitud, esta puede ser: Clientes, captaciones vista, captaciones plazo, contabilidad, talento humano, etc.

Detalle de la solicitud.- En esta sección se detalla brevemente el requerimiento que se desea implementar.

En esta sección debe constar la firma de la persona que está realizando la solicitud de desarrollo y el número de acta de comité tecnológico que fue aprobada o negada la solicitud.

Detalle de desarrollo y puesta en producción.- En esta sección se coloca información del desarrollo del requerimiento, las revisiones realizadas y la fecha de puesta en producción, los datos a ser llenados son los siguientes:

Programador asignado.- Se especifica el analista programador responsable del desarrollo del requerimiento.

Fecha de inicio.- Se especifica la fecha en la cual se comenzó el desarrollo de la solicitud.

Fecha estimada de implantación.- Se especifica una fecha tentativa en la cual se estima terminar el desarrollo.

Revisiones realizadas.- Se especifica todas las revisiones que se realizan para validar que el producto de software cumpla con los requerimientos y necesidades del dueño del producto, en esta sección se anota la fecha de

revisión, la persona que realizó la revisión y las observaciones que se generaron de dicha revisión.

Fecha de puesta en producción.- Se especifica la fecha en la cual dicho requerimiento fue liberado en producción

Versión del aplicativo.- Se especifica la versión con la cual se liberó a producción dicha solicitud.

Al final de esta sección debe constar la firma de aceptación de la persona que realizó la solicitud y el número de acta del comité tecnológico que aprobó la liberación en producción.

A.2 Aprobación o negación de la solicitud de desarrollo de software.

Una vez que el dueño del proceso elabora la solicitud de desarrollo de software, esta es presentada al comité tecnológico para que este organismo analice la viabilidad de la solicitud y apruebe o niegue la misma. En caso que la solicitud sea aprobada el comité tecnológico asignará la prioridad de dicha solicitud con relación a las solicitudes que se encuentren pendientes de desarrollo. Esta priorización debe ser gestionada por el gerente de tecnología para que las solicitudes de desarrollo se vayan desarrollando de acuerdo a las prioridades establecidas por el comité tecnológico. La plantilla de la reunión de comité tecnológico se muestra a continuación:

COOPERATIVA DE AHORRO Y CREDITO
EL SAGRARIO

ACTA DE COMITÉ TECNOLÓGICO
SAG-CT-20XX-XXX

INFORMACIÓN DE LA REUNIÓN

FECHA: _____ HORA DE INICIO : _____ HORA DE FINALIZACIÓN : _____
CONVOCADO POR : _____
LUGAR DE LA REUNIÓN : _____

MIEMBROS CONVOCADOS

Nombres	¿Asiste?	Cargo

SOLICITUDES DE DESARROLLO

Detalle de la solicitud	Dueño del proceso	Resolución del comité

PRIORIZACIÓN DE SOLICITUDES

Prioridad	Descripción	Dueño del proceso

APROBACIÓN DEL ACTA

Nombre	Firma

Figura 16 Acta de comité tecnológico

Este registro nos permite constatar las solicitudes de desarrollo presentadas, cuales fueron aprobadas o negadas y la prioridad para desarrollar dichas solicitudes está conformada de un código de acta, el cual se forma por las siglas SAG-CT-año de la reunión-número de acta por ejemplo: SAG-CT-2015-001. Adicionalmente consta de cinco secciones, las mismas que se describen a continuación:

Información de la reunión.- En esta sección se coloca la fecha en la cual fue efectuada la reunión, la hora que inició, la hora de finalización, la persona que convocó dicha reunión y el lugar en donde tuvo efecto la misma.

Miembros convocados.- En esta sección se coloca los miembros que fueron citados para la reunión, el detalle de esta sección se describe a continuación:

Nombres.- Se especifica los nombres y apellidos del funcionario convocado.

Asiste.- Se especifica si dicho funcionario asistió a la reunión los valores son SI o NO.

Cargo.- Se especifica el cargo que desempeña en la institución.

Solicitudes de desarrollo.- En esta sección se coloca la descripción de la solicitud que se está presentando al comité tecnológico, el dueño del proceso que presenta y la resolución de comité tecnológico sobre la solicitud, esta puede ser: “APROBADA” o “NEGADA”.

Priorización de solicitudes.- Esta sección contiene las solicitudes pendientes de desarrollo con la prioridad a ser desarrollada, esta sección se debe actualizar cada vez que se libera en producción una solicitud o cuando se apruebe por comité tecnológico una solicitud, colocando la prioridad que comité tecnológico le dé a las mismas.

Aprobación del acta.- Esta sección contiene las firmas de todos los miembros que asistieron a la reunión.

A.3 Reunión de planificación del sprint

Una vez que se tiene priorizado las solicitudes de desarrollo de software, el gerente de tecnología convoca a una reunión a todos los involucrados en la solicitud de desarrollo que se va a implementar, el dueño del proceso explica al personal de desarrollo de software todas las funcionalidades que se requiere desarrollar, el resultado de esta reunión es un documento de especificación de requisitos, cuyo formato se especifica a continuación.

Figura 17 Carátula de la especificación de requerimientos

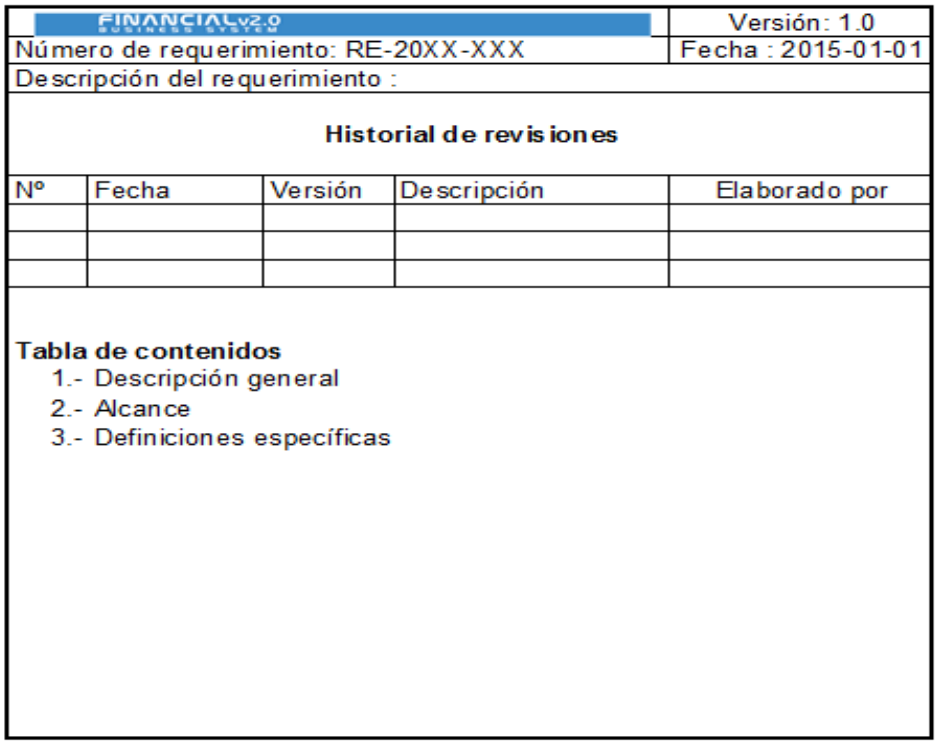


Figura 18 Contenido de la especificación de requerimientos

La especificación de requerimientos debe ser elaborada por la persona encargada de control de calidad y el dueño del proceso. Consta de una carátula la misma que se muestra en la figura 17 y está compuesta por el logo de la institución, un código único que identifica el requerimiento. Está conformado por ER - año de elaboración - secuencial, ejemplo: RE-2015-001. En la segunda sección detalla el historial de revisiones que se ha mantenido a lo largo de la especificación, la tercera sección detalla las especificaciones que debe tener el producto de software consta de los siguientes ítems:

Descripción general.- Se describe de forma global lo que el software debe hacer.

Alcance.- Describe los módulos que van a ser afectados al desarrollar este requerimiento.

Definiciones específicas.- Describe cada funcionalidad, reporte que debe tener el producto de software que se va a elaborar, el lenguaje de estas especificaciones debe ser el lenguaje cotidiano, sin embargo se debe especificar claramente las entradas, salidas y el proceso que debe efectuar cada funcionalidad, de tal manera que tanto las personas involucradas en el área operativa como el personal técnico puedan entender lo que se requiere desarrollar.

Al final del documento deben existir las firmas de las personas que levantaron la especificación de requerimientos.

A.4 Análisis y estimación de tiempos de la especificación de requerimientos.

Una vez que se ha mantenido la reunión con los involucrados y con el documento de especificación de requerimientos, se asigna a un analista programador como el encargado de desarrollar ese requerimiento. El analista programador debe dividir los requerimientos solicitados en actividades no

mayores a 8 horas, con la finalidad de que el seguimiento de cada actividad sea más eficiente, para lo cual se debe llenar el siguiente formato:

Nº Especificación de requerimienos	Proceso	Actividad	Responsable	Horas del desarrollador	Horas del jefe de desarrollo	Promedio

Figura 19 Estimación de tiempos

A continuación se detalla cada campo de la estimación de tiempos:

Nº especificación de requerimientos.- Se coloca el identificador del documento de especificación de requerimiento junto a la descripción, por ejemplo: RE-2015-001 Recaudación de IESS en ventanillas.

Proceso.- describe una categorización de la actividad a ser desarrollada por ejemplo diseño, reportes, parametrización, etc. Nos ayuda a agrupar una serie de actividades para que sea más comprensible lo que se desea desarrollar.

Actividad.- Se describe el proceso que se va a desarrollar.

Responsable.- Se coloca la persona responsable de efectuar la actividad descrita.

Una vez que se ha llenado esta primera sección, el analista programador envía este documento al Jefe de desarrollo para su revisión. Tanto el analista programador como el jefe de desarrollo deben llenar las horas que creen se

van a demorar en realizar cada actividad de acuerdo a la experiencia que tienen en desarrollar funcionalidades similares. Para lo cual se llena la siguiente sección:

Horas del desarrollador.- Cada responsable de la actividad debe colocar el tiempo en horas que cree se puede demorar, este tiempo no debe ser mayor a 8 horas.

Horas del Jefe de desarrollo.- El jefe de desarrollo debe colocar el tiempo en horas que cree se puede demorar el analista programador para terminar la actividad propuesta.

Promedio.- Es el promedio entre las horas del desarrollar y las horas estimadas por el jefe de desarrollo.

En esta etapa el jefe de desarrollo, debe analizar tanto los tiempos estimados por el analista programador y sus tiempos impuestos, con la finalidad de revisar que no existan mucha dispersión en la estimación, el tiempo máximo permitido será de 2 horas, en caso que la diferencia de horas sea mayor se establecerá un dialogo entre las dos partes para poder analizar el motivo de cada estimación, puede ser que alguna de las partes desconozca algún aspecto de dicha actividad.

Una vez que se tiene el promedio de la estimación de tiempo de cada actividad, se procede a establecer el Sprint de cada miembro del equipo, tomando en consideración que el tiempo total del Sprint. En el caso de la Cooperativa de Ahorro y Crédito El Sagrario Ltda. va a ser de 120 horas que corresponden a 15 días laborables.

Esta actividad la realizan todos los miembros del departamento de tecnología, el encargado de control de calidad debe planificar sus actividades tomando en cuenta las solicitudes de desarrollo pendiente que se deben elaborar la especificación de requerimientos y las funcionalidades que se deben probar del sprint anterior y que deben ser liberadas en producción. De

igual manera el ingeniero de producción y base de datos debe planificar y estimar sus actividades de acuerdo a las funcionalidades que van a ser liberadas en producción y otras actividades que el gerente de tecnología le haya indicado en la reunión de planificación.

A.5 Definición y planificación de actividades del sprint

Con el levantamiento de las actividades y tiempos que se van a demorar, el gerente de tecnología se reúne con un representante de cada área del departamento de tecnología que son: jefe de desarrollo, control de calidad e ingeniero de producción y base de datos, con la finalidad de analizar las actividades que se pueden cumplir en el sprint. El resultado de esta revisión son todas las actividades que cada miembro del departamento de tecnología va a realizar en el sprint.

A.6 Desarrollo del sprint

Definidas las actividades que cada miembro del departamento va a realizar en el sprint, cada persona desarrolla las actividades que tiene asignado. La primera tarea que deben desarrollar los analistas programadores es el diseño de interfaces y base de datos de las funcionalidades que van a programar, estos diseños son aprobados por el jefe de desarrollo y posteriormente aprobados por el dueño del proceso. Esta actividad asegura que el producto terminado cumpla con las expectativas del cliente o se pueda modificar el diseño en etapas tempranas y no genere re procesos ni pérdida de tiempo.

Mientras tanto la primera actividad que el ingeniero de control de calidad debe realizar, es la elaboración de la versión alpha de las funcionalidades programadas en el sprint anterior por parte del equipo de desarrollo de software, esta versión va a permitir realizar la verificación del producto de software que va a ser liberado en producción.

A.7 Scrum diario

Todos los días el jefe de desarrollo mantiene una reunión con todos los miembros del equipo con una duración máxima de 15 minutos, en esta reunión se aclaran ciertas dudas que se tengan y analizan los impedimentos que puedan tener los miembros del equipo para realizar eficientemente su trabajo. Cada miembro del equipo debe ingresar la planificación diaria en el siguiente formato.

Fecha	Actividad	Horas	Responsable

Figura 20 Planificación diaria

Este formato permite conocer las actividades que cada miembro planifica realizar en el día, este formato debe ser llenado luego de la reunión de sprint diario. Este formato consta de la fecha en la que se va a realizar la actividad, la actividad del sprint que se planifica realizar, las horas reales que se tardaron en desarrollar la actividad y el responsable de la actividad.

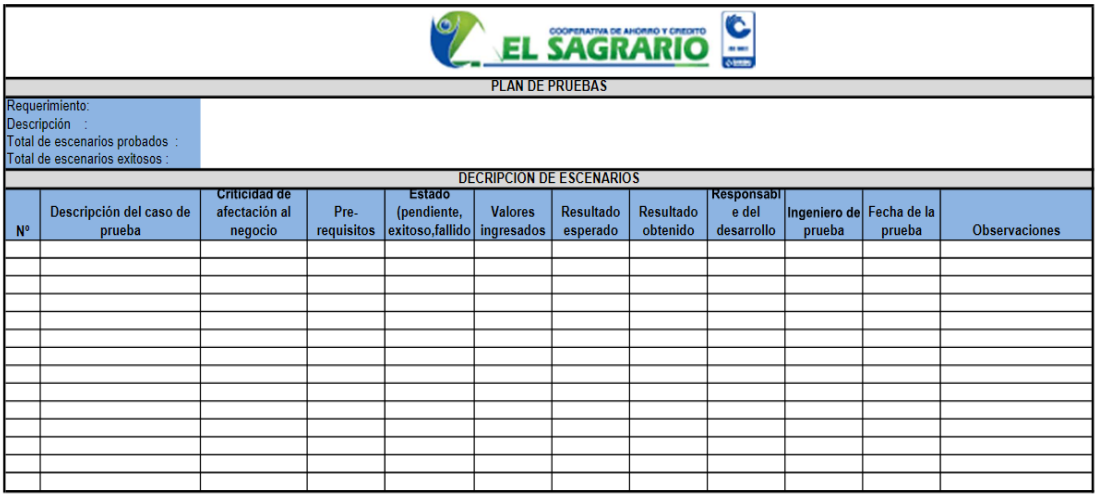
Esta información es importante para analizar la diferencia de tiempo, entre el estimado y el real. De esta manera se puede ajustar los tiempos en próximos desarrollos que contengan actividades similares.

A.8 Revisión de actividades

Cuando los analistas programadores terminen una actividad del sprint, le comunicarán al jefe desarrollo para que certifique que dicha actividad cumpla con los requerimientos solicitados en la especificación de requerimientos y de esta manera se dará como cumplida la actividad.

A.9 Certificación técnica de las funcionalidades desarrolladas

Todos los analistas programadores deben certificar las funcionalidades que desarrollen, el resultado de esta certificación será el plan de pruebas lleno, el mismo que será entregado al encargado de control de calidad para que pueda revisar los escenarios probados e incluir otros que no se han analizado. El formato del plan de pruebas se muestra a continuación:



EL SAGRARIO
COOPERATIVA DE AHORRO Y CREDITO

PLAN DE PRUEBAS

Requerimiento:
Descripción :
Total de escenarios probados :
Total de escenarios exitosos :

DECRIPCIÓN DE ESCENARIOS

Nº	Descripción del caso de prueba	Criticidad de afectación al negocio	Pre-requisitos	Estado (pendiente, exitoso,fallido	Valores ingresados	Resultado esperado	Resultado obtenido	Responsable del desarrollo	Ingeniero de prueba	Fecha de la prueba	Observaciones

Figura 21 Plan de pruebas

El plan de pruebas consta de dos secciones, la primera referente al resumen del plan de pruebas que se está realizando, es esta sección constan los siguientes campos:

Requerimiento.- Se coloca el código del requerimiento que se está realizando la certificación, por ejemplo: RE-2015-001.

Descripción.- Se coloca la descripción del requerimiento que se está realizando la certificación, por ejemplo: recaudación de IESS a través de ventanilla.

Total de escenarios probados.- Se coloca número total de escenarios que se probaron del requerimiento que se está certificando.

Total de escenarios exitosos.- Se coloca número total de escenarios que resultaron exitosos en las pruebas.

La segunda sección del plan de pruebas se refiere al detalle de cada caso de prueba realizado, consta de los siguientes campos:

Nº.- Es un número secuencial que identifica cada escenario

Descripción del caso de prueba.- Describe el escenario que se va a probar.

Criticidad de afectación al negocio.- Describe el afectación que tiene la funcionalidad que se está evaluando al negocio, esta puede ser: Alta, media, baja.

Pre-requisitos.- Describe los requisitos previos que debe tener el escenario que se va a testear.

Estado.- Describe el estado en que se encuentra el escenario, este puede ser: pendiente, exitoso o fallido.

Valores ingresados.- Detalla los valores de entrada con los cuales se realizaron las pruebas.

Resultado esperado.- Describe el resultado que se espera obtener al evaluar el escenario.

Resultado obtenido.- Describe el resultado que se obtuvo al evaluar el escenario, si el resultado obtenido es igual al resultado esperado se concluye que el estado de la prueba es exitoso, caso contrario será fallido.

Responsable del desarrollo.- Se coloca la persona encargada de desarrollar el requerimiento que se está evaluando.

Ingeniero de prueba.- Se coloca la persona que está realizando la evaluación del escenario.

Fecha de prueba.- Se coloca la fecha en la cual se prueba el escenario.

Observaciones.- Se detalla algún tipo de observación sobre el escenario evaluado.

Cada analista programador debe realizar el manual de usuario de las funcionalidades desarrolladas, y debe ser entregada al jefe de desarrollo para que este consolide todos los manuales de usuario y posteriormente sea entregado al ingeniero de producción y base de datos para modificar el manual de usuario general de aplicativo con estas nuevas funcionalidades.

A.10 Reunión de validación con el dueño del proceso

El analista programador debe realizar al menos una reunión con el dueño del proceso, con la finalidad de validar que el producto de software cumpla con las expectativas del cliente, esta reunión debe ser registrada en la solicitud de desarrollo de software en el apartado de revisiones realizadas vea ítem 3.5.1.1. De esta manera se asegura que el software que se está construyendo cumpla con las necesidades del dueño del proceso.

A.11 Finalización del sprint y reunión de retrospectiva

Al siguiente día de culminado el sprint, se debe realizar una reunión de finalización y retrospectiva del sprint. En esta reunión se da a conocer el cumplimiento de cada miembro del departamento y se analiza los planes de mejora para que la metodología se siga afianzando.

Es importante para optimizar el tiempo, que esta reunión también sirva como reunión de planificación ver ítem 3.5.1.3. El resultado de esta reunión es el acta de cierre de sprint, el mismo que se describe a continuación.

COOPERATIVA DE AHORRO Y CREDITO EL SAGRARIO

CIERRE DE SPRINT
SPRINT 20XX-XXX

DATOS DE LA CONVOCATORIA

Asunto :
Fecha de la reunión: Hora :
Lugar :
Convocado por :

ASISTENTES

Nombre	Cargo

AGENDA DE LA REUNIÓN

Punto	Descripción
1	Conocimiento y resolución del cumplimiento del sprint 20XX-XX
2	Planificación de actividades del sprint 20XX-XX

DESARROLLO DE LA AGENDA

1.- Conocimiento y resolución del cumplimiento del sprint 20XX-XX

2.- Planificación de actividades del sprint 20XX-XX

Figura 22 Acta de cierre de sprint

En el acta de cierre de sprint se registra las novedades del sprint que acaba de finalizar en la cabecera se escribe el sprint que se está analizando por ejemplo sprint 2015-001, consta de cuatro secciones que se detallan a continuación:

La primera sección describe la información de la convocatoria y está compuesta por los siguientes campos:

Asunto.- Se detalla la razón por la cual fue convocada la reunión.

Fecha de la reunión.- Se detalla la fecha en que tuvo efecto la reunión.

Lugar.- Se detalla el sitio en la cual se efectúa la reunión.

Convocado por.- Se coloca la persona que efectúa la convocatoria.

La segunda sección corresponde a las personas que asistieron a la reunión y el cargo que mantiene en la institución.

En la tercera sección se detalla la agenda que tendrá la reunión. En esta agenda al menos deben constar dos puntos, el primero referente al conocimiento y resolución del cumplimiento del sprint y el segundo punto referente a las actividades que se van a realizar en el siguiente sprint. En esta sección se puede aumentar más puntos a tratarse en la reunión, como aspectos de revisión del área, revisión de indicadores, etc.

En la cuarta sección se detalla la agenda, en el primer punto se describe cual fue el porcentaje de cumplimiento de cada miembro y en caso de no haber cumplido, el motivo que ocasionó dicho incumplimiento. El segundo punto describe el inicio, finalización, actividades a desarrollarse en el próximo sprint y adicionalmente las horas laborables con el que cuenta cada miembro del equipo, ya que es probable que algún miembro del equipo tenga planificado vacaciones o algún tipo de permiso por el cual tenga que ausentarse.

A.12 Paso de funcionalidades del área de desarrollo al área de control de calidad.

Una vez que se ha terminado el sprint, el área de desarrollo debe pasar las funcionalidades elaboradas en el sprint anterior al área de control de calidad con la finalidad de probar cada funcionalidad desarrollada y eliminar la mayor cantidad de defectos producidos en el área de desarrollo. El responsable de las pruebas debe elaborar un plan de pruebas igual al elaborado por los analistas programadores incluyendo todos los escenarios que crea conveniente, es importante que también se tome en cuenta los escenarios propuestos por el área de desarrollo, ya que puede darse el caso que el responsable de control de calidad omita ciertos escenarios, el formato del plan

de pruebas se puede ver en el ítem 3.4.1.1.6.3 figura 21 Formato del plan de pruebas.

En caso que se detecte fallos en las funcionalidades que se están evaluando, el responsable de control de calidad debe comunicar al responsable de la funcionalidad que tiene fallos, para que este sea corregido inmediatamente, es importante asignar un cierto tiempo en el próximo sprint del área de desarrollo para corregir este tipo fallos que por lo general siempre se dan en cualquier proceso de desarrollo de software. Cuando el fallo detectado sea corregido el responsable de control de calidad debe volver a probar ese escenario y dependiendo de la criticidad del escenario fallido, se deberá probar los escenarios dependientes.

Al finalizar las pruebas de control de calidad se tiene como resultado el plan de pruebas lleno tanto con el número de escenarios probados como con el número de escenarios exitosos.

A.13 Autorización para liberación a producción.

Una vez que el responsable de control de calidad ha certificado que todas las funcionalidades desarrolladas cumplan con los requerimientos solicitados y no tengan fallos, debe presentar una lista de funcionalidades aptas para ser liberadas en producción, el formato que debe llenar el responsable de control de calidad es el siguiente.

COOPERATIVA DE AHORRO Y CREDITO EL SAGRARIO

DETALLE DE FUNCIONALIDADES PARA CAMBIO DE VERSIÓN

Fecha de elaboración :

Aplicativo	Versión actual		Nueva versión
Financial	2.0.0.X		2.0.0.X
Descripción del requerimiento	Dueño del proceso	N° de requerimiento	Responsable del Desarrollo

Atentamente,

Personal involucrado en los desarrollos

CERTIFICACIÓN DE PASO A PRODUCCIÓN

Fecha de puesta en producción :..... Versión: .....

Fecha de aprobación del comité tecnológico : .....

Observaciones :.....

.....

Figura 23 Detalle de funcionalidades para cambio de versión

Este registro debe entregar el responsable de control de calidad al gerente de tecnología, para que este convoque a una reunión al comité tecnológico y se apruebe o niegue la liberación en producción de las funcionalidades descritas.

Cuando el comité tecnológico apruebe la liberación en producción, el registro de funcionalidades para cambio de versión debe ser entregado al ingeniero de producción para que proceda a liberar el producto de software en producción.

A.14 Capacitación a los usuarios

En los casos que amerite capacitación el personal que va a utilizar las nuevas funcionalidades desarrolladas, el área de talento humano en conjunto con el dueño del proceso deberá planificar la capacitación para las personas que se considere deben recibirla.

A.15 Liberación en producción

Una vez que el comité tecnológico ha aprobado la liberación de las funcionalidades desarrolladas en producción, el ingeniero de producción y base de datos debe realizar la publicación del aplicativo y actualizar la base de datos con los scripts que son enviados desde control de calidad.

A.16 Difusión a los usuarios

Una vez realizada la liberación en producción, se debe enviar un mail a todo el personal de la institución informándoles de los cambios realizados en el aplicativo informático.

A.17 Seguimiento de la versión en producción

Una vez realizada la liberación en producción, cada dueño del proceso debe revisar que las funcionalidades liberadas en producción, no tengan fallos y funcionen correctamente, en caso que exista algún fallo en las funcionalidades, se debe reportar al ingeniero de producción y base de datos para que en conjunto con el encargado de control de calidad verifiquen este inconveniente. En caso que exista un fallo en la programación se debe llenar el siguiente formulario.

Figura 24 Registro de fallas en producción

Este formulario permite registrar todas las fallas que el sistema ha producido luego de que se libere en producción nuevas funcionalidades, consta de los siguientes campos:

Descripción.- Se detalla el fallo que tuvo el sistema informático.

Origen.- Se coloca el código del requerimiento al que se asocia el fallo, por ejemplo RE-2015-001.

Fecha.- Se coloca la fecha en la cual fue reportado el fallo.

Criticidad.- Se coloca la criticidad del fallo, puede ser baja, media o alta.

¿Requiere una nueva publicación?.- Se utiliza para especificar si para resolver el fallo encontrado se tuvo que realizar una nueva publicación en producción.

Persona que reportó el incidente.- Se utiliza para especificar el nombre de la persona que reportó en fallo encontrado.

Todo el proceso descrito para el desarrollo de software y la interacción entre la norma ISO/IEC 12207:2008 con la metodología Scrum se resume en la siguiente tabla.

Tabla 6

Resumen de integración entre ISO/IEC 12207:2008 y Scrum

INTEGRACIÓN DE ISO/IEC 12207:2008 CON SCRUM DESARROLLO Y MANTENIMIENTO		
ISO/IEC 12207:2008	**SCRUM**	**ACTIVIDADES**
Gestión	Fase de planificación	Elaboración de la solicitud de desarrollo de software
		Aprobación o negación de la solicitud de desarrollo
		Reunión de planificación del sprint
Análisis		Análisis y estimación de tempos de la especificación de requerimientos
		Definición y planificación de las actividades del sprint
Diseño	Fase de desarrollo	Desarrollo del sprint
Codificación		
Integración		Scrum diario
		Reunión de validación con el dueño del proceso
		Finalización del Sprint y reunión de retrospectiva
Pruebas	Fase de finalización	Paso de funcionalidades del área de desarrollo al área de control de calidad
Implantación		Autorización para liberación a producción
		Capacitación a los usuarios
		Liberación en producción
		Difusión a los usuarios
		Seguimiento de la versión

La siguiente figura muestra gráficamente el proceso de desarrollo fusionando la ISO/IEC 12207:2008 y Scrum.

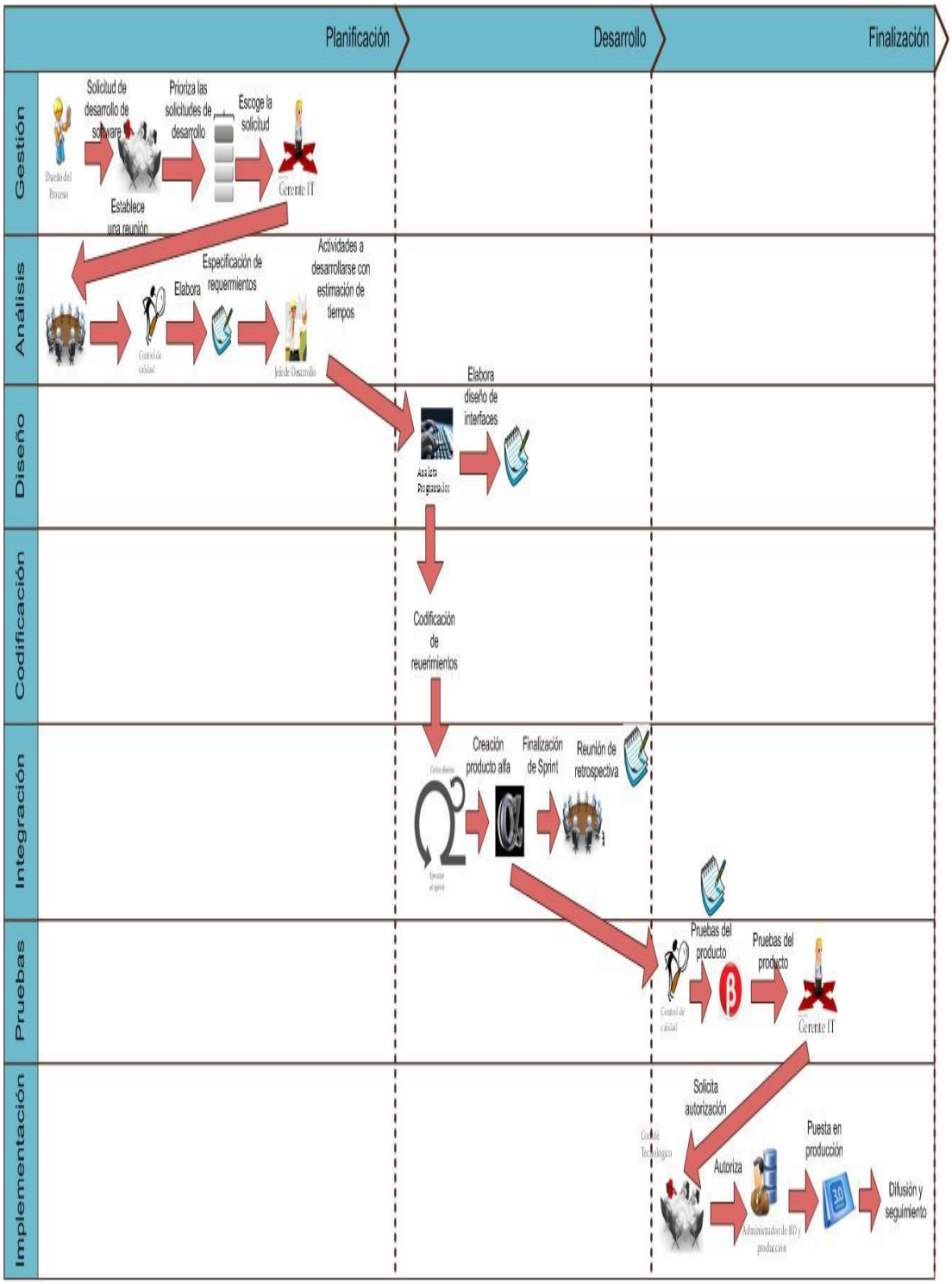


Figura 25 Proceso de desarrollo de software fusionando las fases de ISO/IEC 12207 y Scrum

B. Operación

B.1 Reporte de incidentes

Para atender de mejor manera los incidentes o necesidades ocasionadas por la utilización de los productos de software en la institución, se instaló un software gratuito llamado help desk en la intranet institucional, este software permite reportar incidentes, requerimientos rápidos o solicitar soporte de algún tema en específico.

El primer paso para gestionar correctamente las necesidades de los usuarios, es establecer los acuerdos de niveles de servicios conocidos también como SLA. El SLA es un compromiso entre el negocio y el departamento de tecnología, tiene la finalidad de comprometer al departamento de tecnología a brindar un servicio eficiente a sus usuarios dentro de los plazos establecidos. Para brindar un mejor soporte a los usuarios de la institución se establecen las siguientes prioridades:

Urgente.- Son los servicios tecnológicos cuya interrupción afecta gravemente al negocio, no se puede brindar atención en una o varias oficinas.

Alto.- Son los servicios tecnológicos cuya interrupción afecta al negocio, no se puede brindar atención en un proceso o puesto de trabajo.

Medio.- Son los servicios tecnológicos cuya interrupción afecta parcialmente al desempeño normal de las funciones de los puestos de trabajo.

Normal.- Esta prioridad se utiliza para atender servicios de tecnología relacionados con requerimientos específicos para apoyar o mejorar ciertas actividades o tareas.

De acuerdo a las prioridades establecidas, se asigna el tiempo máximo para que se pueda atender una petición.

Tabla 7

Tiempos de prioridades de SLA

Prioridad	Tiempo máximo acordado (Horas)
Urgente	**4**
Alto	**8**
Medio	**40**
Normal	**240**

Tomando en consideración los requerimientos más frecuentes en la institución, se establece las prioridades definidas para cada caso.

Tabla 8

Categorización de incidentes más comunes

Servicio	Descripción	Prioridad
Errores presentados en los aplicativos	Cuando existen errores de programación que no permite realizar una transacción en el área de ventanillas, afectando el servicio al socio o cliente.	Urgente
	Cuando existen errores de programación que no permite realizar una transacción en los puestos de atención al cliente.	Alto
	Cuando la información arrojada en un reporte es inconsistente.	Normal

CONTINÚA

Servicio	Descripción	Prioridad
	Cuando existen errores de programación en las áreas de apoyo, pero no afecta al desempeño de las funciones cotidianas.	Normal
Requerimientos en los aplicativos	Cuando se requiera modificaciones pequeñas en un proceso.	Normal
	Cuando se requiera modificaciones en los reportes.	Normal
Soporte a usuarios	Solicitud de información no automatizada en el aplicativo para entidades de control.	Urgente
	Solicitud de información no automatizada en el aplicativo para usuarios internos.	Alto
	Ayuda en la utilización de los aplicativos informáticos	Alto

Tomando en cuenta los acuerdos de niveles de servicio creados en la institución, cuando los usuarios deseen solicitar un servicio al área de desarrollo de software, lo primero que deben realizar es crear un ticket en el aplicativo de help desk, a continuación se muestra la manera de crear un ticket.

Figura 26 Creación de tickets

En el gráfico anterior se muestra la pantalla que permite a los usuarios internos de la institución crear tickets para que el área de desarrollo de software pueda brindar el soporte requerido, los campos que se llenan en el formulario son los siguientes:

Grupo de soporte.- Este campo se utiliza para informar qué tipo de ayuda necesita el usuario, puede ser soporte en caso que requiera realizar algún tipo de consulta acerca de la utilización de una funcionalidad del aplicativo informático o aplicativos en caso que requiera informar algún incidente o mejora rápida en el aplicativo.

Agente.- Se utiliza para especificar el nombre de la persona que se desea que le brinde el servicio.

Ticket prioridad.- Se utiliza para especificar la prioridad del ticket de acuerdo a los acuerdos de niveles de servicio establecidos.

Ticket estado.- Se utiliza para especificar el estado actual en el que se encuentra el ticket, los estados posibles son: pendiente, abierto, en proceso, esperando respuesta, cerrado

Tipo de problema.- Este campo se utiliza para informar que tipo de problema que tiene el usuario, este puede ser requerimiento cuando requiera realizar algún tipo de consulta acerca de la utilización de una funcionalidad del aplicativo informático o requiera informar alguna mejora rápida en el aplicativo. También puede ser falla tecnológica, cuando se requiera informar de un error en el aplicativo.

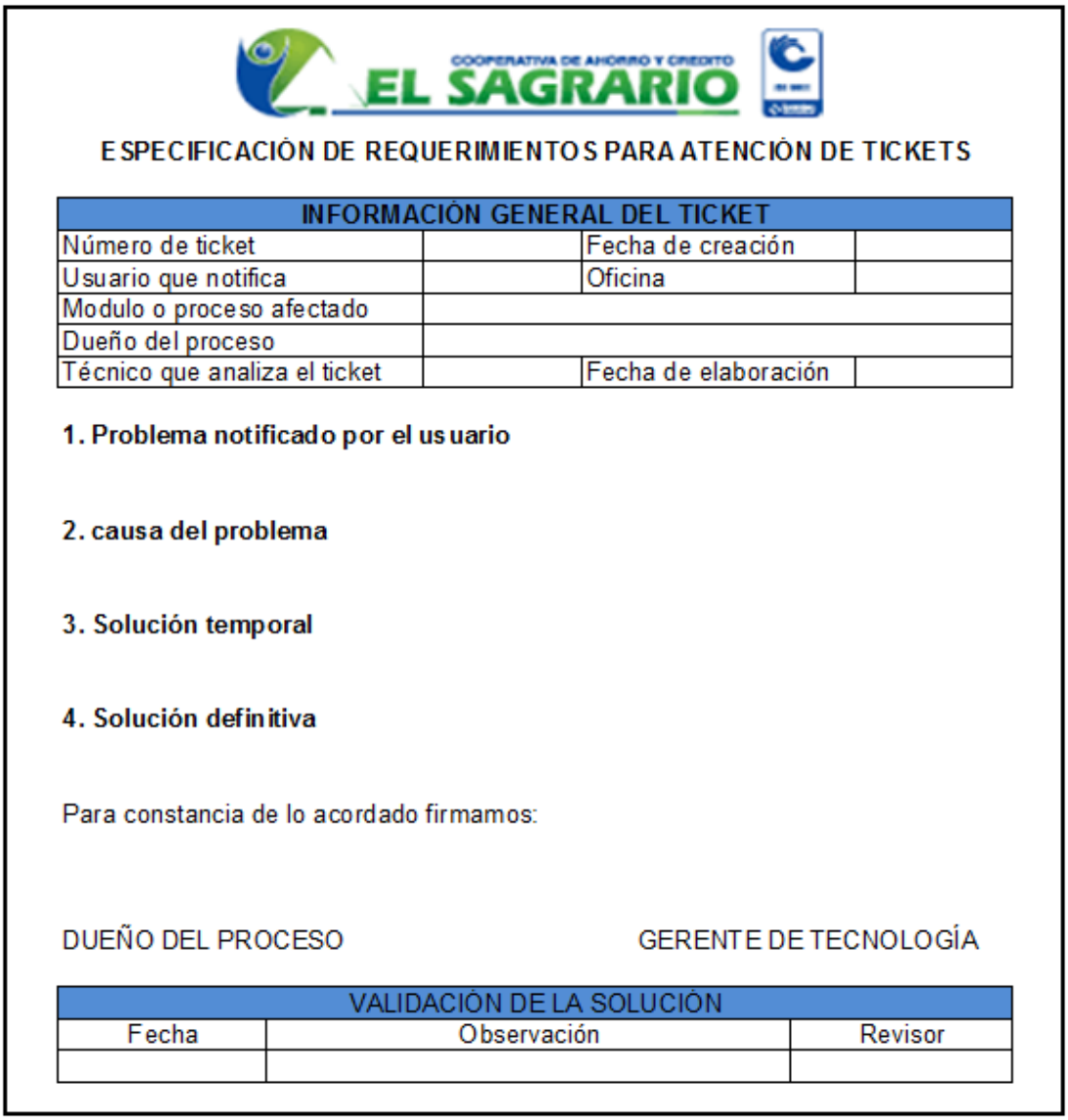
Título.- Se especifica una descripción corta del incidente.

Descripción del problema.- Se especifica una descripción detallada del incidente o ayuda que el usuario requiere que se brinde de parte del departamento de tecnología.

B.2 Atención de incidentes

Una vez que el usuario ha reportado el incidente a través de la creación de tickets, este llega al agente que se especificó en el ticket mediante un mail, el agente o miembro del departamento de tecnología analiza y se contacta con el usuario para informarle el tiempo en el cual va a ser resuelto el requerimiento, tomando en consideración los niveles de acuerdo de servicio.

En caso que el incidente reportado sea una falla en el aplicativo informático o un requerimiento de mejora, el analista programador debe elaborar un documento de solución de incidentes, cuyo formato se muestra a continuación.



COOPERATIVA DE AHORRO Y CRÉDITO EL SAGRARIO

ESPECIFICACIÓN DE REQUERIMIENTOS PARA ATENCIÓN DE TICKETS

INFORMACIÓN GENERAL DEL TICKET			
Número de ticket		Fecha de creación	
Usuario que notifica		Oficina	
Modulo o proceso afectado			
Dueño del proceso			
Técnico que analiza el ticket		Fecha de elaboración	

1. Problema notificado por el usuario

2. causa del problema

3. Solución temporal

4. Solución definitiva

Para constancia de lo acordado firmamos:

DUEÑO DEL PROCESO GERENTE DE TECNOLOGÍA

VALIDACIÓN DE LA SOLUCIÓN		
Fecha	Observación	Revisor

Figura 27 Especificación de requerimientos para atención de tickets

Este registro nos permite levantar la especificación de requerimientos de una manera ágil, los campos a llenar son los siguientes:

Número de ticket.- Es el número que el aplicativo de help desk genera para cada incidente.

Usuario que notifica.- En este campo se especifica el usuario que notifica el incidente.

Fecha de creación.- Se especifica la fecha en la cual fue creado el ticket.

Oficina.- En este campo se especifica la oficina a la cual pertenece el usuario que notificó el incidente

Módulo o proceso afectado.- Se especifica el módulo que va a sufrir modificaciones.

Dueño del proceso.- Se coloca el nombre del dueño del proceso.

Técnico que analiza el ticket.- Se coloca el nombre de la persona que realiza la atención del requerimiento.

Fecha de elaboración.- Se coloca la fecha en la cual fue elaborado el documento.

Problema notificado por el usuario.- Se coloca la descripción detallada del ticket que fue llenada por el usuario que requiere soporte.

Causa del problema.- la persona que está atendiendo el ticket debe especificar la causa que origina el problema reportado.

*Solución temporal.-*En caso de existir alguna solución temporal debe ser especificada, de lo contrario se colocará la palabra ninguna.

*Solución definitiva.-*Se describe de forma detallada la solución que se va adoptar para resolver el incidente reportado.

Una vez llenado e impreso este registro se debe revisar en conjunto con el dueño del proceso con la finalidad de verificar que la solución propuesta sea la más conveniente. Este documento debe ser firmado tanto por el dueño del proceso como por el gerente de tecnología.

La siguiente sección del registro, se refiere a la revisión realizada por el dueño del proceso, con el objetivo de validar la corrección efectuada, los campos que se deben llenar son los siguientes:

*Fecha.-*Se coloca la fecha en la cual tuvo efecto la reunión.

*Observación.-*Se coloca algún tipo de observación que realice el revisor.

Revisor.- Se coloca el nombre y una sumilla de la persona que realizó la validación de la solución.

B.3 Desarrollo de incidentes que requieren programación

Una vez que se ha elaborado el documento de especificación de requerimientos para atención de tickets y el dueño del proceso está de acuerdo con la solución planteada, el analista programador debe programar la solución planteada, al terminar el desarrollo, se debe establecer una reunión con el dueño del proceso para validar que la modificación realizada cumpla con las necesidades del dueño del producto. Realizada esta revisión se debe completar la información en el documento de especificación de requerimientos para atención de tickets en la sección de validación de la solución.

Al finalizar el sprint todas las modificaciones realizadas por el reporte de incidentes, deben ser pasadas al área de control de calidad conjuntamente con las funcionalidades desarrolladas en el sprint para que sean certificadas y posteriormente liberadas en producción.

B.4 Calificación del servicio

Una vez que el usuario ha verificado que la solución al incidente reportado se ha solucionado, se procede a calificar el ticket en la misma herramienta de help desk, las calificaciones disponibles se muestran en la siguiente tabla.

Tabla 9

Puntuación de la calificación de tickets

Descripción	Puntuación
Muy satisfactorio	10
Satisfactorio	9
Poco satisfactorio	8
Nada satisfactorio	7

Esta calificación ayudará al departamento de tecnología a realizar planes de mejora para cada vez brindar un mejor servicio.

Es importante en cada planificación del sprint asignar un tiempo para la resolución de los incidentes reportados por los usuarios internos, este tiempo depende del número de tickets que se creen al mes y el tiempo que se demora en atender el requerimiento, en la Cooperativa de Ahorro y Crédito El Sagrario se asigna a cada miembro del departamento de tecnología de 20 a 25 horas para atender estos requerimientos.

En caso que existan incidentes que no se puedan resolver en el sprint que se está trabajando, se debe colocar en la planificación del siguiente sprint para que dicho incidente sea atendido de acuerdo al SLA establecido.

3.4.2. Procesos de apoyo del ciclo de vida

A. Documentación

El proceso de desarrollo y mantenimiento, posee registros que avalan cada actividad que se va a realizar y permiten tener un mejor control en el proceso de desarrollo y mantenimiento de software. Los registros resultantes del proceso son los siguientes:

- Solicitud de desarrollo de software.
- Acta de comité tecnológico.

- Especificación de requerimientos.
- Registro de estimación de tiempos.
- Registro de planificación diaria.
- Registro de plan de pruebas de desarrollo y control de calidad.
- Acta de cierre de sprint.
- Detalle de funcionalidades para el paso a producción.
- Registro de fallas en producción.

B. Gestión de la configuración

El departamento de tecnología de la Cooperativa de Ahorro y Crédito El Sagrario Ltda. desarrolla sus productos de software en Visual Studio .Net 2010 y utiliza un repositorio de código denominado Team Foundation Server 2010 (TFS). Este repositorio de código permite automatizar y optimizar el proceso de desarrollo de software cuando se tiene más de un programador, este software permite revisar todas las modificaciones que ha tenido un determinado archivo en el tiempo. La siguiente figura muestra la manera como se maneja el versionamiento del aplicativo.

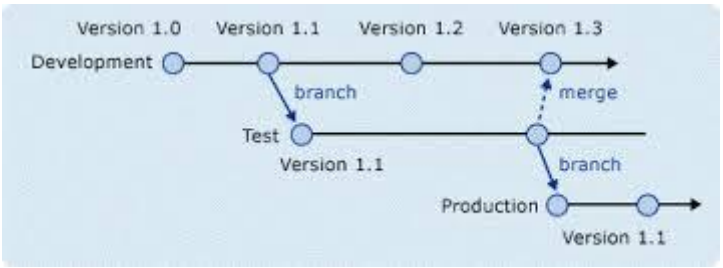


Figura 28 Gestión de la configuración con TFS

Fuente: Microsoft

Una vez que se ha terminado de programar todas las actividades propuestas en el sprint, los analistas programadores deben subir sus cambios al gestor de código TFS, a esta actividad se la conoce como Check In, por cada Check In que el analista programador realice, el TFS genera un número único denominado set change. Los analistas programadores deben pasar estos cambios al área de control de calidad en el siguiente registro.


COOPERATIVA DE AHORRO Y CREDITO EL SAGRARIO

CAMBIOS VERSIÓN 2.0.0.X

Desarrollador	Descripción	Responsabe del proceso	Origen	Conjunto de cambios

Figura 29 Registro para la gestión de la configuración

Desarrollador.- Se especifica la persona que programó los cambios en el aplicativo informático.

Descripción.- Se especifica una descripción corta del cambio realizado.

Responsable del proceso.- Se especifica el nombre del dueño del proceso.

Origen.- Se especifica el origen del cual proviene el cambio realizado este puede provenir de una especificación de requerimiento o de la generación de un ticket.

Conjunto de cambios.- Se especifica el o los números identificadores únicos proporcionado por el TFS (set change). Que se encuentran asociados al cambio realizado en el aplicativo.

Una vez que el área de desarrollo proporciona este registro al área de control de calidad, el área de control de calidad procede a descargarse solo

los conjuntos de cambios descritos en el registro de control de cambios para posteriormente realizar un demo con el cual se realizan las pruebas de control de calidad.

Luego que el comité tecnológico aprueba la puesta en producción del aplicativo informático con los nuevos cambios el ingeniero de producción y base de datos procede a realizar un branch con la versión que se va a liberar en producción.

C. Aseguramiento de la calidad

La institución cuenta con un área de control de calidad que se encarga de realizar la verificación del producto de software antes de ser liberado en producción. Los registros resultantes son los siguientes.

- Plan de pruebas del área de desarrollo.
- Plan de pruebas del área de control de calidad.

D. Verificación

El producto de software es verificado en el área de desarrollo tanto por el analista programador como por el jefe de desarrollo, posteriormente en el área de control de calidad y una vez liberado en producción por el ingeniero de producción y base de datos.

En este proceso se definen indicadores de calidad, los cuales proveen información que nos permite medir de manera objetiva la calidad del proceso de software. Tomando en consideración la información que se genera del proceso de desarrollo de software propuesto, se plantea los siguientes indicadores de calidad que nos permiten medir la calidad tanto del proceso como del producto de software.

Tabla 10

Indicadores de calidad

No	Descripción del indicador	Fórmula del indicador
1	Cumplimiento del sprint	$\frac{\sum horas\ cumplidas}{\sum horas\ estimadas} * 100$
2	Errores detectados en QA	$\frac{\sum escenarios\ fallidos}{\sum escenarios\ probados} * 100$
3	Errores detectados en producción	$\sum errores\ detectados\ por\ vesrión$
4	Cumplimiento de tickets	$\frac{\sum tickets\ atendidos\ cumpliendo\ el\ SLA}{\sum tickets\ creados\ en\ el\ mes} * 100$
5	Satisfacción del cliente	Promedio de las calificaciones de los tickets en el mes

Los registros resultantes de este proceso son los siguientes:

- Plan de pruebas del área de desarrollo.
- Plan de pruebas del área de control de calidad.
- Indicadores de calidad

E. Validación

La validación del producto de software se la realiza con la reunión de validación con el dueño del proceso, con la finalidad que el producto de software desarrollado cumpla con las expectativas del dueño del producto. Los registros resultantes de esta actividad son:

- Actas de revisión en la solicitud de desarrollo de software.
- Actas de revisión en la solicitud de incidentes.

F. Revisiones conjuntas

Estas revisiones se las realiza constantemente con el dueño del proceso, tanto para definir los requerimientos como para validar el producto de software. Los registros resultantes de esta actividad son:

- Especificación de requerimientos de software.
- Solicitud de incidentes.
- Actas de revisión en la solicitud de desarrollo de software.
- Actas de revisión en la solicitud de incidentes.

G. Auditoria

La institución cuenta con un departamento de auditoría interna, el cual se encarga de realizar al menos una auditoría interna y una auditoría externa de forma anual, con la finalidad de verificar que la metodología propuesta se esté efectuando correctamente. Los registros resultantes de esta actividad son:

- Informes de auditoría.

H. Solución de problemas

La mayor ventaja de utilizar metodologías ágiles, es que al estar en constante dialogo con el dueño del proceso los problemas que se puedan tener en el producto de software se los soluciona de manera inmediata.

3.4.3. Procesos organizativos del ciclo de vida

A. Gestión

En la institución este proceso lo realiza el comité tecnológico en conjunto con el gerente de tecnología, los mismos que analizan la viabilidad de los proyectos y destinan los recursos necesarios para los proyectos. Las actividades resultantes de este proceso son las siguientes:

- Aprobación o negación de la solicitud de desarrollo.

- Reunión de planificación del Sprint.

- Autorización para la liberación en producción.

B. Mejora

En este proceso se mantienen reuniones con todo el equipo que interviene en el proceso de software, con la finalidad de mejorar el proceso de desarrollo de software. En esta etapa se realiza la reunión de retrospectiva al final de cada sprint con el objetivo de analizar los indicadores de calidad para poder tomar las acciones correctivas que permita mejorar el proceso y con ello mejorar el producto de software.

3.4.4. Configuración de ambientes

Dado que la Cooperativa de Ahorro y Crédito El Sagrario Ltda. pertenece a un sector económico crítico, tanto por la información como por los activos de dinero que maneja, la institución divide al área de desarrollo de software en tres ambientes: desarrollo, control de calidad o QA y producción. A continuación se muestra un gráfico de cómo interactúan estos tres ambientes dentro del sprint para el desarrollo de software.

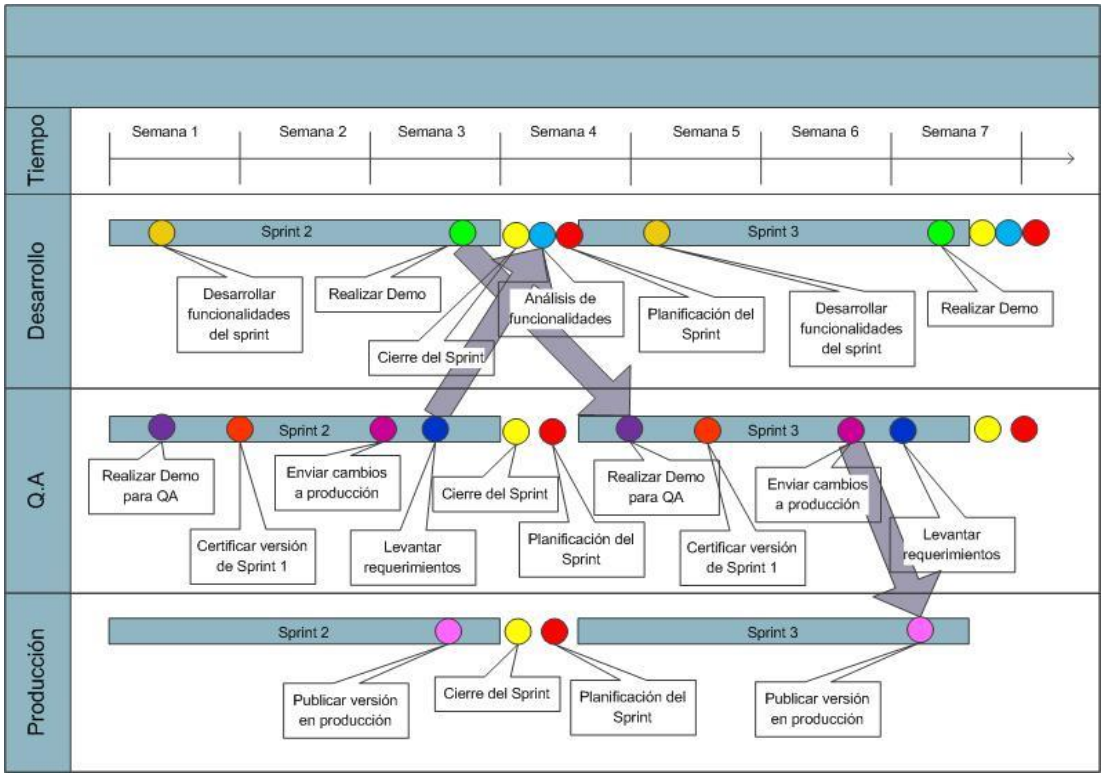


Figura 30 Actividades a desarrollarse con Scrum

En este gráfico se puede observar cómo interactúan los tres ambientes para crear un producto de software, el proceso es el siguiente:

El área de desarrollo de software en el sprint en curso, se encarga de desarrollar las funcionalidades asignadas en el sprint y realizar un demo para poder validar con el dueño del proceso. Una vez finalizado el sprint se tiene 3 días en los cuales se debe realizar la reunión de finalización del sprint y retrospectiva, la reunión de planificación del sprint, la reunión de análisis y estimación de tiempos y la revisión del sprint.

El área de control de calidad debe generar un demo de las funcionalidades pasadas por el área de desarrollo de software del sprint anterior, con la finalidad de verificar que todas las funcionalidades estén correctas. Una vez certificadas todas las funcionalidades, se envía a comité tecnológico para su aprobación y posteriormente al área de producción para la liberación del producto. Como el área de control de calidad se encarga de elaborar el

documento de especificación de requerimientos en conjunto con el dueño del proceso, esta actividad se la realiza en el sprint actual y le permite al área de desarrollo de software analizar y estimar los tiempos para que sea desarrollado en el próximo o próximos sprints.

3.5. Caso de estudio

En esta sección se va a implementar el marco de trabajo elaborado, para lo cual se realiza las actividades de un sprint. De esta manera en el cual explicaremos la manera de implementar el marco de trabajo elaborado.

3.5.1. Desarrollo y mantenimiento

Con la finalidad de explicar de mejora manera el marco de trabajo elaborado, se va a tomar como ejemplo la implementación de un sistema de recaudación en línea de recaudaciones patronales del IESS.

3.5.2. Elaboración de la solicitud de desarrollo de software

El primer paso para comenzar con el desarrollo de software dentro de la institución es elaborar la solicitud de desarrollo de software por parte de los dueños de proceso, en el ejemplo que se va analizar, el gerente de operaciones elabora la solicitud de desarrollo de software. El anexo C muestra la solicitud llena.

3.5.3. Aprobación o negación de la solicitud de desarrollo de software.

El Gerente de tecnología debe citar a una reunión a todos los miembros del comité tecnológico con la finalidad de tratar temas referentes al área de tecnología de la institución, uno de los puntos a tratar en esta reunión es el análisis de las solicitudes de desarrollo que se tenga pendiente. Una vez analizado estas solicitudes los miembros del comité tecnológico aprueban o niegan esta solicitud. Como resultado de esta reunión se tiene el acta de comité tecnológico, el mismo que se muestra en el anexo D.

En esta reunión, los integrantes del comité tecnológico establecen las prioridades de los requerimientos que estén en progreso. Es necesario llenar

la solicitud de desarrollo de software con el número de acta de comité tecnológico con el cual fue aprobado o negado el mismo, en nuestro caso debe ir SAG-CT-2015-001.

3.5.4. Reunión de planificación del sprint

Una vez que tenemos priorizados los proyectos en los cuales vamos a trabajar, realizamos reuniones con el dueño de procesos, jefe de desarrollo, ingeniero de control de calidad y analistas programadores, con la finalidad de analizar los requerimientos solicitados. En nuestro caso se realiza una reunión con el gerente de operaciones para analizar cómo debe funcionar este producto, resultado de estas definiciones tenemos el documento de especificación de requerimientos que se encuentran en el anexo E.

3.5.5. Análisis y estimación de tiempos de la especificación de requerimientos.

Al tener la especificación de requerimientos y saber con exactitud las especificaciones del dueño de proceso, se procede a desglosar los requerimientos en actividades no mayores a ocho horas, como se muestra en el anexo F.

Una vez que el analista programador obtiene todas las actividades que se deben realizar con sus respectivos tiempos, el jefe de desarrollo revisa estas actividades y realiza su estimación de tiempos. La estimación de tiempos de cada actividad es el resultado del promedio entre los tiempos del analista programador y el jefe de desarrollo de software.

Como se puede observar en el anexo F, se estima que el desarrollo de este proyecto sea de 223 horas aproximadamente.

3.5.6. Definición y planificación de actividades del sprint.

Una vez que estimamos los proyectos en los cuales vamos a trabajar en el siguiente sprint, analizamos las actividades prioritarias y distribuimos el trabajo entre todos los miembros del equipo de desarrollo, tal como se muestra en el anexo G.

Las actividades que no se realicen en el sprint, se las realizará en el próximo sprint. En este documento se encuentran las actividades que van a realizar los analistas programadores en el Sprint, se trata de distribuir de tal manera que todas las horas que tienen disponibles en el sprint sean cubiertas, las demás horas faltantes para cubrir las 120 horas se las programa asignando un tiempo para reuniones, atención de tickets y tickets que tengan pendientes de resolver.

3.5.7. Desarrollo del sprint

Cada día se mantiene una reunión entre el equipo de desarrollo para revisar los avances y las actividades que cada miembro se compromete a realizar en el día, resultado de esta reunión se obtiene el anexo H. Este anexo indica en que se va a trabajar en el día y al concluir cada actividad se anota el tiempo real que llevó realizar la misma. Esta información nos permite evaluar la diferencia de tiempo entre el estimado y el real, con la finalidad de afinar la estimación de tiempos de actividades similares.

Al concluir las actividades asignadas, cada analista programador elabora un plan de pruebas para evaluar el funcionamiento del producto de software. El plan de pruebas se muestra en el anexo I.

Una vez probado el producto de software, se mantiene una reunión tanto con el jefe de desarrollo como con el dueño del producto para validar que la funcionalidad desarrollada cumpla con los requerimientos especificados. Una vez realizada la revisión el jefe de desarrollo da como cumplida la actividad y se procede a evidenciar en la solicitud de desarrollo la revisión realizada

conjuntamente con las observaciones o sugerencias realizadas por las personas que revisaron el producto de software.

3.5.8. Finalización del sprint y reunión de retrospectiva

Al día siguiente de terminar el sprint, se realiza la reunión de finalización de sprint y retrospectiva. En esta reunión se da a conocer el cumplimiento del Sprint y se analizan tanto los inconvenientes presentados como los aspectos positivos de la metodología, con la finalidad de mejorar la metodología y corregir los errores suscitados. El anexo J muestra el acta de finalización del sprint.

3.5.9. Paso de funcionalidades del área de desarrollo al área de control de calidad.

La primera actividad que debe cumplir el área de control de calidad en el siguiente sprint es recibir del área de desarrollo los cambios que se han realizado al sistema informático. El ingeniero de control de calidad con esos cambios elabora un demo conjuntamente con un plan de pruebas y se procede a verificar el producto de software. El anexo k muestra el plan de pruebas ejecutado por el ingeniero de calidad.

3.5.10. Autorización para liberación a producción.

Verificado que el producto de software cumpla con todos los requerimientos tanto técnicos como operativos, se convoca a una reunión de comité tecnológico para autorizar la liberación de las funcionalidades desarrolladas a producción, esta reunión nos sirve también para receptar nuevas solicitudes de desarrollo y priorizarlas. El anexo L muestra la solicitud para cambio de versión.

Una vez autorizado el paso a producción, se planifica la capacitación a los usuarios que van a utilizar el sistema, la fecha en que va a ser liberado el producto de software en producción y la difusión a través de un mensaje de correo a todos los colaboradores de la institución, para que tengan conocimiento de las actualizaciones realizadas en el sistema informático.

3.5.11. Seguimiento de la versión en producción.

Una vez liberado el producto de software tanto el dueño del proceso como el ingeniero de control de calidad verifican que todas las funcionalidades creadas funcionen correctamente. En el caso planteado se detectó un fallo en el sistema por lo cual se procede a llenar el formulario que se encuentra en el anexo M.

Al final del desarrollo de un requerimiento se tiene la solicitud de desarrollo de software completamente llena. Esta acta nos permite realizar el seguimiento de un requerimiento desde la concepción del requerimiento hasta la liberación en producción. El anexo N muestra la solicitud de desarrollo llena.

3.6. Conclusiones del capítulo

La fusión entre la norma ISO/IEC 12207:2008 y Scrum, nos da como resultado una metodología ágil consistente ya que mutuamente se complementan, la norma ISO/IEC 12207:2008 nos proporciona una serie de lineamientos y buenas prácticas que debemos utilizar en el ciclo de vida del software, pero no nos dice cómo hacerlo ni tampoco nos proporciona las herramientas para llevar a cabo el proceso de desarrollo, por otro lado Scrum nos ayuda con procesos para la gestión del proyecto pero no proporciona lineamientos para ser utilizados en el ciclo de vida de desarrollo de software. Al combinar las fortalezas tanto de la norma como de la metodología obtenemos un marco de trabajo que se adapta a las necesidades de la institución y proporciona lineamientos, registros y buenas prácticas que permiten obtener productos de software de buena calidad, satisfaciendo las

necesidades del cliente interno que va a utilizar el producto y en los tiempos acordados.

CAPÍTULO IV

VALIDACIÓN DEL MARCO DE TRABAJO

4. Introducción

En este capítulo se va a realizar el análisis de los resultados obtenidos al aplicar el marco de trabajo implementado en la Cooperativa de Ahorro y Crédito El Sagrario Ltda. Para lo cual se van analizar los siguientes indicadores:

a. Cumplimiento de las actividades encomendadas al área de desarrollo de software en el sprint.

b. Errores detectados en el área de aseguramiento de la calidad.

c. Errores detectados en producción.

d. Efectividad del marco de trabajo implementado de acuerdo a entrevistas realizadas.

Los indicadores a, b y c se va a valuar de manera cuantitativa, tomando los sprints desarrollados en el primer semestre del año 2015. El indicador (d) se va a evaluar de manera cualitativa a través de encuestas elaboradas al equipo de desarrollo de software y a los usuarios que solicitan el producto de software respectivamente, para lo cual se va aplicar el modelo de calidad propuesto por (Orna, 2013).

4.1. Cumplimiento de las actividades encomendadas al área de desarrollo de software en el sprint

En el primer semestre de este año se realizaron seis sprint, con el siguiente cumplimiento:

Tabla 11

Datos del cumplimiento del sprint

Sprint	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Prom.
% cump.	90.50	100.00	98.25	100.00	98.00	100.00	97.79%

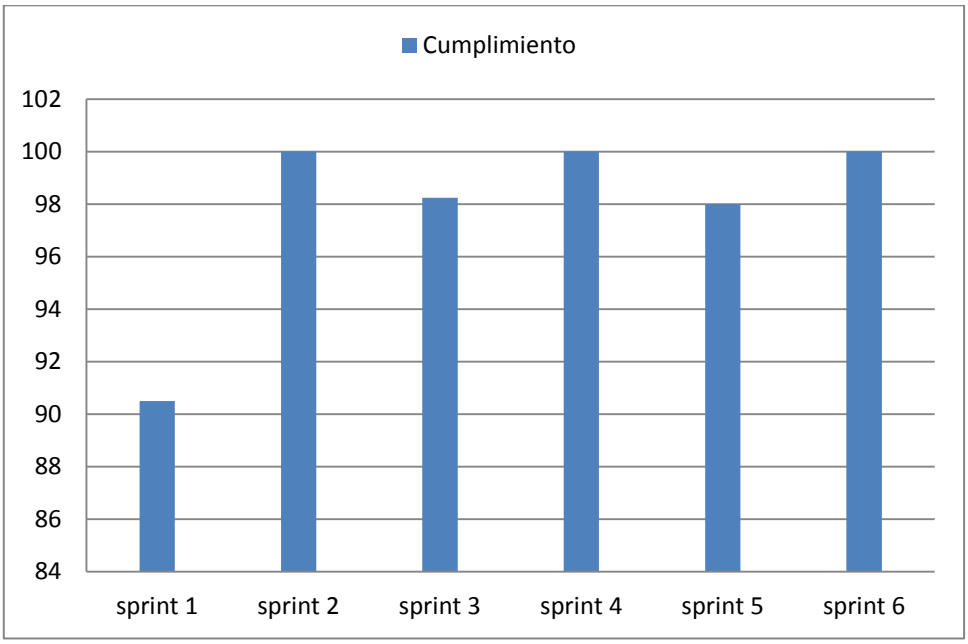


Figura 31 Cumplimiento del sprint

La figura 31 muestra el porcentaje de cumplimiento de cada sprint, se puede observar que el primer sprint fue el de menor cumplimiento, esto se debe a la adaptación del equipo de desarrollo al nuevo marco de trabajo. En este sprint existieron aspectos a mejorar; entre los cuales se puede resaltar la estimación de tiempos para actividades como el soporte a usuarios y reuniones.

A partir de estas correcciones los siguientes sprints tienen un cumplimiento de casi el 100%, los sprints que no cumplen el 100% se debe básicamente a incumplimientos por parte de proveedores externos, este aspecto se lo debe tomar muy en cuenta ya que puede poner en riesgo el avance de un proyecto.

En términos generales se puede evidenciar que la implementación del marco de trabajo ha permitido que el equipo de desarrollo cumpla con las actividades planificadas en el sprint.

4.2. Errores encontrados en el área de aseguramiento de la calidad

Todas las actividades realizadas en un sprint son validadas por el área de control de calidad, por lo que se obtuvo 6 informes.

Para realizar el cálculo del indicador se aplica la siguiente fórmula:

$$\frac{\sum escenarios\ fallidos}{\sum escenarios\ probados} * 100$$

Los resultados obtenidos son los siguientes:

Tabla 12

Datos de errores encontrados en control de calidad

Sprint	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Promedio
Escenarios fallidos	3	5	15	5	3	2	33
Total de escenarios	29	92	154	73	56	88	492
Indicador	10%	5%	10%	6.80%	5.3%	2.27%	6.71%

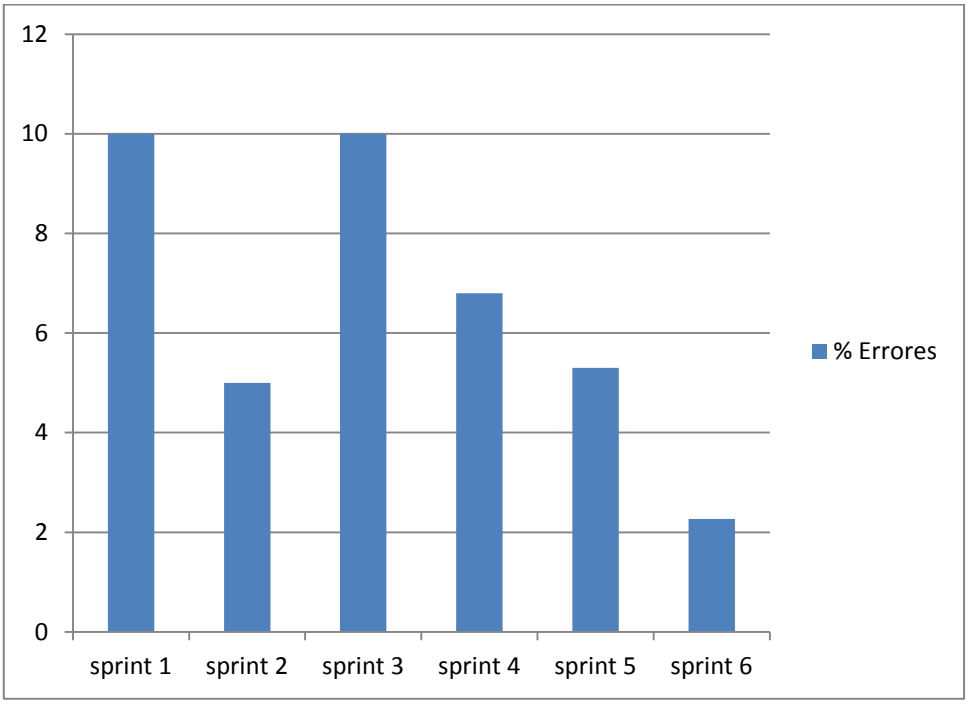


Figura 32 Errores detectados en control de calidad

La figura 32 muestra los errores detectados en el área de control de calidad, los mismos que indican un valor aceptable en la calidad de software del área de desarrollo, entre menor sea este indicador mayor será la calidad de software que se está produciendo, ya que se evita re procesos en el producto de software por parte del área de desarrollo de software y se optimiza el tiempo de área de control de calidad, el mismo que puede ser utilizado para verificar más escenarios con la finalidad de liberar en producción un producto con la menor cantidad de fallos.

4.3. Errores encontrados en producción

En base al anexo M registro de fallas en producción podemos obtener la siguiente información de las funcionalidades liberadas en producción de los seis primeros sprint.

Tabla 13

Datos de errores encontrados en producción

Sprint / Criticidad	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Total
Alta	1	0	1	0	0	1	3
Media	2	0	2	0	0	0	4
Baja	1	2	1	0	2	3	9
Total	4	2	4	0	2	4	16

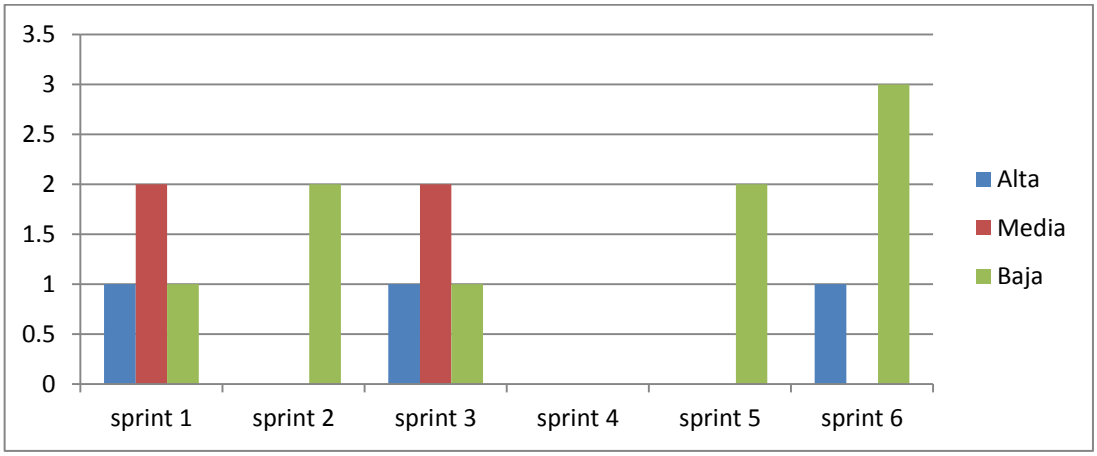


Figura 33 Errores encontrados en producción

En la figura 33 se puede apreciar el número de errores detectados en producción de acuerdo a la criticidad, como se puede observar dichos errores está dentro del rango tolerable, tomando en cuenta que eliminar totalmente los errores en un producto de software es casi imposible.

4.4. Efectividad del marco de trabajo implementado

Para medir la efectividad del marco de trabajo implementado en la Cooperativo de Ahorro y Crédito El Sagrario Ltda. se propone una encuesta tanto al personal de tecnología de la institución, como a los usuarios que utilizan los productos de software desarrollados. Para lo cual la investigación se orientará de la siguiente manera:

- Población (Personal de la Cooperativa de Ahorro y Crédito El Sagrario Ltda.)
- Muestra
    - Encuesta orientada al personal de tecnología de la institución.
    - Encuesta orientada a los dueños de proceso.

4.4.1. Procesamiento de datos y corroboración de los resultados

A continuación se presenta la estructura sobre la cual se va a evaluar los resultados obtenidos en las encuestas (Orna, 2013):

- Valoración de resultado entre 1 y 5 para cada pregunta.
    - 1.- Ocasionalmente (en escasas ocasiones)
    - 2.-Ordinariamente (en ocasiones puntuales)
    - 3.- Frecuentemente (en la mayoría parte)
    - 4.- Muy Frecuentemente (en casi todas las ocasiones)
    - 5.- Siempre

- Se deberá marcar con una (X) el valor seleccionado.

PREGUNTA	**Valor**				
Pregunta Nº					

- Se deberá sumar las (X) por cada columna respectivamente.
- Este resultado se deberá multiplicar por el valor que se indica para columna, obteniendo así el total del puntaje de la misma.
- Se deberá sumar los totales de cada columna.

Para la interpretación, se ha dividido a los resultados en cuatro grupos de puntajes, que abarcan un 100% de los datos, de esta manera se conocerá en

que grupo se califica al modelo propuesto, como se muestra a continuación. (Orna, 2013)

Hasta de 39 por ciento: El marco de trabajo propuesto no se cumple, se cumple en aspectos parciales o tiene una fidelidad muy baja con las actividades realmente realizadas, y deben tomarse medidas correctoras urgentes y globales para implantar un sistema de calidad eficaz.

Entre 40 y 59 por ciento: El marco de trabajo propuesto se cumple, pero con deficiencias en cuanto a procesos o a la continuidad y sistemática de su cumplimiento, o tiene una fidelidad deficiente con las actividades realmente realizadas. Se deberán solucionar las deficiencias urgentemente, para que el sistema sea eficaz.

Entre 60 y 85 por ciento: El marco de trabajo propuesto se cumple, pero con leves inconvenientes. Se deberán solucionar las deficiencias a corto plazo, para que el modelo no deje de ser eficaz. Su tendencia hacia la Gestión de la Calidad es muy positiva. Les sugerimos analicen sus puntos sobresalientes y apliquen medidas similares a los temas con más baja puntuación.

Más de 85 por ciento: El marco de trabajo propuesto optimizara el proceso de desarrollo de software agregando factores de calidad a las tareas que se ejecuten en la empresa aplicable.

Los resultados de este método empírico se recogen en tablas y en cada caso se valoran aspectos mencionados a fin de documentar las evidencias que demuestran que calidad en los productos de software ha mejorado con la implementación del marco de trabajo propuesto en este tema de investigación.

4.4.2. Análisis de resultados de la encuesta aplicada a los dueños de proceso.

A continuación se presenta los resultados de la encuesta aplicado a los 10 dueños de proceso que laboran en la Cooperativa de Ahorro y Crédito El Sagrario Ltda.

Para el cálculo de estos valores se dividió la suma total de puntos obtenidos (ST) para el valor que resulta de multiplicar el número total de preguntas cerradas de la encuesta por el valor más alto de la misma, y multiplicar por 100%. Se debe tomar en cuenta que el instrumento se aplica a diez personas.

Tabla 14

Resultados de la efectividad del marco de trabajo según los dueños de proceso

Valor	**1**	**2**	**3**	**4**	**5**
Total de (X)	**0**	**0**	**0**	**13**	**77**
Multiplicación	*1	*2	*3	*4	*5
Resultado parcial	**0**	**0**	**0**	**52**	**385**
Suma total de puntos obtenidos (ST)					**437**
RESULTADO [(ST/(26*5))*100%]	**97,11 % de aceptación**				

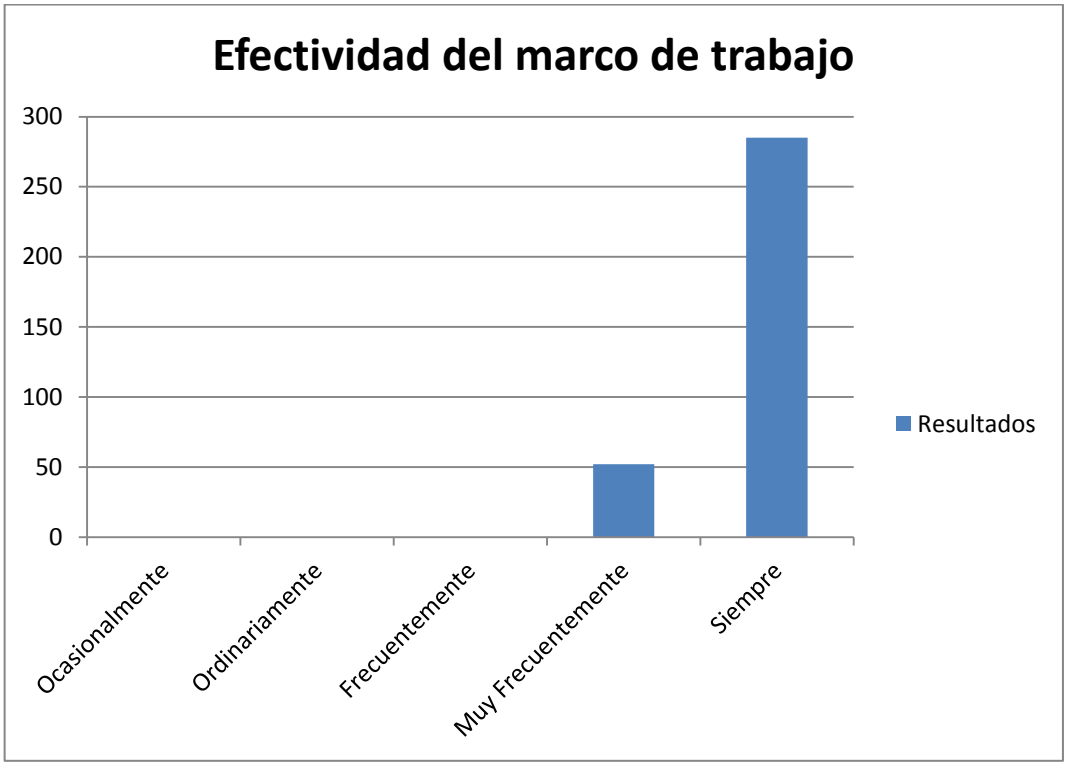


Figura 34 Encuesta de efectividad del marco de trabajo a los dueños de proceso

Como se puede observar los dueños de proceso opinan que la implementación del marco de trabajo, ha servido para mejorar la calidad de los productos de software en la institución. Al aplicar la fórmula del modelo, el resultado nos arroja que existe un 97.11% de aceptación del marco de trabajo para los dueños de proceso.

4.4.3. Análisis de resultados de la encuesta aplicada al personal de tecnología.

A continuación se presenta los resultados de la encuesta aplicado a los 7 miembros de tecnología que laboran en la Cooperativa de Ahorro y Crédito El Sagrario Ltda.

Tabla 15

Resultados de la efectividad del marco de trabajo según los miembros del departamento de tecnología

Valor	1	2	3	4	5
Total de (X)	**0**	**0**	**0**	**13**	**57**
Multiplicación	*1	*2	*3	*4	*5
Resultado parcial	**0**	**0**	**0**	**52**	**285**
Suma total de puntos obtenidos (ST)					**337**
RESULTADO [(ST/(26*5))*100%]	**96,28 % de aceptación**				

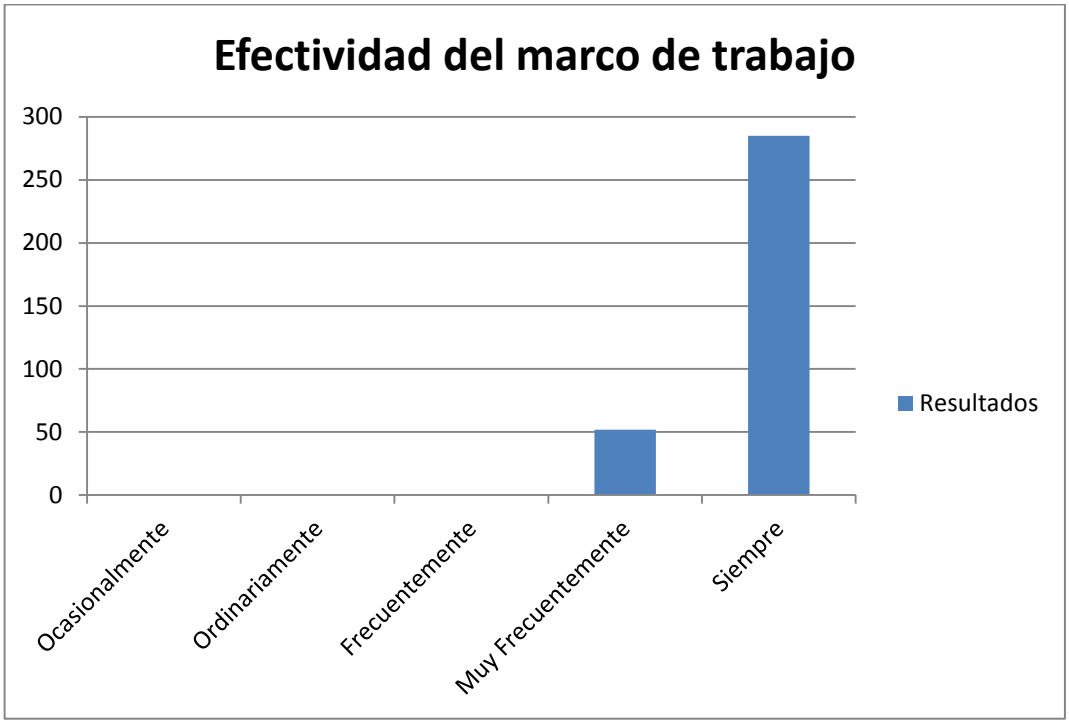


Figura 35 Encuesta de efectividad del marco de trabajo a los miembros del departamento de tecnología

Como se puede observar los miembros del departamento de tecnología opinan que la implementación del marco de trabajo, ha servido para mejorar la calidad de los productos de software en la institución y hacer más eficiente

sus labores. Al aplicar la fórmula del modelo, el resultado nos arroja que existe un 96.28% de aceptación del marco de trabajo dentro de los miembros del departamento de tecnología.

4.4.4. Prueba de la hipótesis con Chi cuadrado.

A. Planteamiento de la hipótesis.

A través de la prueba de hipótesis conocida como chi cuadrado, se realizará el análisis del grado de asociación entre la variable dependiente e independiente de esta investigación. Para lo cual definimos las variables de esta investigación.

a) **Hipótesis de la investigación:** Si se implementa un marco de trabajo basada en la norma ISO/IEC 12207 y Scrum, entonces se mejorará la calidad del producto de software en la Cooperativa de Ahorro y Crédito ‘El Sagrario’ Ltda.

b) **Variable independiente:** Implementación de un marco de trabajo basada en la norma ISO/IEC 12207 y Scrum.

c) **Variable dependiente:** Mejoramiento de la calidad de productos de software en la Cooperativa de Ahorro y Crédito ‘El Sagrario’ Ltda.

- **Hipótesis nula (Ho):** La implementación de un marco de trabajo basado en la norma ISO/IEC 12207 y Scrum y el mejoramiento de la calidad de los productos de software, son independientes.

- **Hipótesis alternativa (Ha):** La implementación de un marco de trabajo basado en la norma ISO/IEC 12207 y Scrum y el mejoramiento de la calidad de los productos de software, son dependientes.

B. Cálculos de frecuencias esperadas, correspondientes a cada frecuencia observada.

a) Frecuencia observada

A continuación se muestra los resultados obtenidos de las encuestas realizadas al grupo de dueños de proceso y al grupo de tecnología, los mismos que se encuentran en el ANEXO O.

En la tabla 4.6 se encuentra los resultados de la variable independiente, mientras que en la tabla 4.7 se encuentra los resultados de la variable dependiente.

Tabla 16

Resultados de la variable independiente (marco de trabajo)

Valoración	Dueños de proceso	Miembros del departamento de tecnología	Total
1.- Ocasionalmente	0	0	0
2.- Ordinariamente	0	0	0
3.- Frecuentemente	0	0	0
4.- Muy frecuentemente	12	7	19
5.- Siempre	38	28	66

Tabla 17

Resultados de la variable dependiente (mejoramiento de la calidad)

Valoración	Dueños de proceso	Miembros del departamento de tecnología	Total
1.- Ocasionalmente	0	0	0
2.- Ordinariamente	0	0	0

CONTINÚA

Valoración	Dueños de proceso	Miembros del departamento de tecnología	Total
3.- Frecuentemente	0	0	0
4.- Muy frecuentemente	8	4	12
5.- Siempre	42	31	73

b) Frecuencia esperada

La tabla 17 muestra la interacción entre los datos de la variable independiente (marco de trabajo) y la variable dependiente (mejoramiento de la calidad).

Tabla 18

Frecuencia esperada entre la variable independiente y dependiente

		Elaboración del marco de trabajo					
Mejoramiento de la calidad en de productos	**Valoración**	**1**	**2**	**3**	**4**	**5**	**Totales**
	1	0	0	0	19	66	85
	2	0	0	0	19	66	85
	3	0	0	0	19	66	85
	4	12	12	12	31	78	145
	5	73	73	73	92	139	450
	Totales	**85**	**85**	**85**	**180**	**415**	**850**

A continuación se muestra la ecuación Ec. 5.1 para el cálculo de los valores para las Variables Elaboración del marco de trabajo frente al mejoramiento de la calidad en productos de software.

$$E_{i,j} = \frac{\Sigma_{i=1}^{m} O_{i,j} * \Sigma_{j=1}^{n} O_{i,j}}{\Sigma_{i=1}^{m} {}_{j=1}^{n} O_{i,j}}$$

Ec. 4.1

Dónde:

m: número de columnas **j**: posición de filas

n: número de filas **O**: frecuencia observada

i: posición de columnas **E**: frecuencia esperada

Aplicando la ecuación Ec. 4.1, en la Tabla 4.5 se representa la Frecuencia esperada se obtiene los siguientes valores.

Tabla 19

Frecuencia esperada entre la variable independiente y dependiente aplicando Ec 4.1

		Elaboración del marco de trabajo					
Mejoramiento de la calidad en productos de software	**Valoración**	**1**	**2**	**3**	**4**	**5**	**Totales**
	1	8.5	8.5	8.5	18	41.5	85
	2	8.5	8.5	8.5	18	41,5	85
	3	8.5	8.5	8.5	18	41.5	85
	4	14.5	14.5	14.5	30.705	70.79	145
	5	45	45	45	95.295	219.71	450
	Totales	85	85	85	180	415	850

C. Cálculo del valor de Chi Cuadrado.

Una vez calculado la frecuencia esperada, se procede a realizar el cálculo de Chi cuadrado el mismo que nos permite verificar la dependencia entre las variables propuestas. A continuación se muestra la fórmula para el cálculo de Chi cuadrado.

$$x^2 = \Sigma_{i=1}^{m} \Sigma_{j=1}^{n} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}$$

Ec. 4.2

Dónde:

m: número de columnas **j**: posición de filas

n: número de filas **O**: frecuencia observada

i: posición de columnas **E**: frecuencia esperada

x: valor de Chi Cuadrado

Tabla 20

Cálculo de Chi cuadrado entre la variable independiente y dependiente aplicando Ec 4.2

		Elaboración del marco de trabajo					
	Valoración	**1**	**2**	**3**	**4**	**5**	**Totales**
Mejoramiento de la calidad en productos de software	**1**	8.5	8.5	8.5	0.05555	14.4638	40.01941
	2	8.5	8.5	8.5	0.05555	14.4638	40.01941
	3	8.5	8.5	8.5	0.05555	14.4638	40.01941
	4	0.43103	0.43103	0.43103	0.00281	0.73346	2.029381
	5	17.4222	17.4222	17.4222	0.11387	29.6461	82.02671
	Totales	43.3532	43.3532	43.3532	0.28335	73.7712	**204.1143**

Realizando los cálculos correspondientes en relación a la Tabla 4.7, se puede observar que la sumatoria para hallar el valor de Chi Cuadrado es:

$$\boldsymbol{x^2_{observado} = 204,1143}$$

D. Cálculo del valor crítico de Chi Cuadrado.

Para el cálculo del Valor Crítico de Chi Cuadrado, inicialmente se procede a calcular los grados de Libertad para Chi Crítico tomando el nivel de significancia supuesto de 0.05, que indica que hay una probabilidad del 0.95 de que la hipótesis nula sea verdadera.

Nivel de Significancia (α):

$$\boldsymbol{\alpha = 0.05}$$

Grados de Libertad (ν):

$$\nu = (número\ de\ filas - 1) * (número\ de\ olumnas - 1)$$

$$\nu = (5 - 1) * (5 - 1)$$

$$\nu = (4) * (4)$$

$$\boldsymbol{\nu = 16}$$

Con la sumatoria de los datos obtenidos en la Tabla 20 se procede a la búsqueda del Valor Crítico de Chi Cuadrado según el Nivel de Significancia (filas) y Grados de Libertad (columnas) en la tabla 4.8.

Tabla 21

Chi cuadrado crítico

Grados de Libertad (ν)	Áreas de Extremos Superior (α)					
	0.25	**0.10**	**0.05**	**0.025**	**0.01**	**0.005**
1	1.323	2.706	3.841	5.024	6.635	7.879
2	2.773	4.605	5.991	7.378	9.210	10.597
3	4.108	6.251	7.815	9.348	11.345	12.838
4	5.385	7.779	9.488	11.143	13.277	14.860
5	6.662	9.236	11.071	12.883	15.086	16.750
6	7.841	10.645	12.592	14.449	16.812	18.548
7	9.037	12.017	14.067	16.013	18.475	20.278
8	10.129	13.362	15.507	17.535	20.090	21.955
9	11.389	14.684	16.919	19.023	21.666	23.589
10	12.549	15.987	18.307	20.483	23.209	25.188
11	13.701	17.275	19.675	21.920	24.725	26.757
12	14.845	18.549	21.026	23.337	26.217	28.299
13	15.984	19.812	22.362	24.736	27.688	29.819
14	17.117	21.064	23.685	26.119	29.141	31.319
15	18.245	22.307	24.996	27.488	30.578	32.801
16	19.369	23.542	**26.296**	28.845	32.000	34.267
17	20.489	24.769	27.587	30.191	33.409	35.718
18	21.605	25.989	28.869	31.526	34.805	37.156

Por tanto, el Valor Crítico de Chi Cuadrado es:

$$x^2_{crítico} = 26,296$$

E. Comparación entre el valor observado y el valor crítico.

De acuerdo a la figura 36, se puede observar que el valor de Chi cuadrado se encuentra en la zona de rechazo, por lo que se acepta la hipótesis alternativa Ha

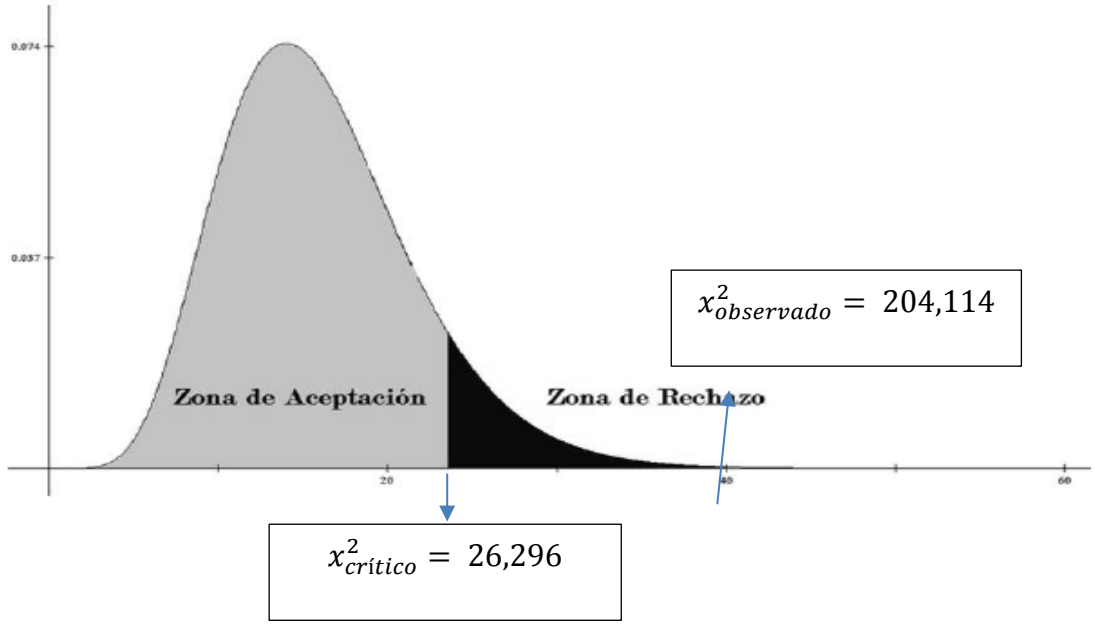


Figura 36 Comparación de Chi Cuadrado

Para dar cumplimiento a la hipótesis planteada en este proyecto de investigación se procede a aplicar la siguiente regla de decisión entre el valor de Chi crítico y el valor calculado de Chi cuadrado.

Regla de decisión:

- Se acepta la Hipótesis Nula (Ho), si: $x^2_{observado} < x^2_{crítico}$
- Se acepta la Hipótesis Alternativa (Ha), si: $x^2_{observado} > x^2_{crítico}$

Siendo: $\boldsymbol{x^2_{observado} = 204,114}$ y $\boldsymbol{x^2_{crítico} = 26,296}$

$$\boldsymbol{204,114 > 26,296}$$

Entonces: $\boldsymbol{x^2_{observado} > x^2_{crítico}}$

Por lo tanto**: "Se acepta la Hipótesis Alternativa (Ha)"**

4.5. Conclusiones del capítulo

Al obtener los valores de los indicadores cuantitativos planteados para demostrar la hipótesis de este tema de investigación, se puedo observar que la calidad de software de la institución ha mejorado, ya que al implementar el marco de trabajo desarrollado se ha establecido procedimientos y buenas prácticas sobre los cuales se puede desarrollar software de calidad cumpliendo los requisitos de los usuarios en los tiempos ofrecidos y con un porcentaje aceptable de errores en producción.

Además al aplicar las encuestas para medir la satisfacción, tanto de los dueños de proceso como de los miembros del departamento de tecnología; se puedo demostrar matemáticamente a través de Chi cuadrado la aceptación del marco de trabajo implementado.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5. Introducción

En este capítulo se establecerá las conclusiones y recomendaciones que se ha generado al implementar el marco de trabajo en la Cooperativa de Ahorro y Crédito El Sagrario Ltda.

5.1. Conclusiones

- La investigación bibliográfica, nos muestra el constante esfuerzo que cada día realizan todas las personas involucradas en el desarrollo de software por encontrar mecanismos que ayuden a mejorar la calidad del software a nivel mundial. Sin embargo todas las dificultades que conlleva elaborar un producto de software no permite que existe un mecanismo único para elaborar software de calidad, por lo que es necesario analizar todas las propuestas metodológicas que existen, para posteriormente adoptar y adaptar dicha metodología a nuestro entorno.

- La norma ISO/IEC 12207 está compuesta de varias actividades las cuales contienes buenas prácticas, las mismas que nos permite establecer procedimientos al proceso de desarrollo de software para obtener productos de software de calidad. Para poder implementar la norma ISO/IEC 12207 dentro de una organización, es importante establecer la base del modelo propuesto para posteriormente adaptarlo a los procesos que maneja la organización que desea adoptar esta norma y de esta manera obtener el mayor provecho de esta norma.

- Al combinar la norma ISO/IEC 12207 y Scrum, se obtiene una metodología robusta, que permite establecer lineamientos sobre los cuales se puede desarrollar software de calidad, aprovechando todos los recursos con los que cuenta la organización. La norma ISO/IEC 12207 aprovecha la ventaja que posee Scrum para administrar proyectos y recursos; mientras que Scrum se beneficia de la forma como administra el proceso de desarrollo

de software la norma ISO/IEC 12207, por este motivo al fusionar estos dos mecanismos para el desarrollo de software se obtiene un marco de trabajo sólido que establece lineamientos, buenas prácticas para el desarrollo de software y una manera ágil de administrar tanto el proyecto de software como los recursos con los que cuenta la organización.

- Al implementar este marco de trabajo nos podemos dar cuenta, que es posible desarrollar software de calidad en empresas cuya área de tecnología es limitada, a través de la implementación de estándares de calidad y metodologías que se adapten a la organización y permita aprovechar todos los recursos con los que cuenta la misma. El marco de trabajo desarrollado le ha permitido a la institución contar con estándares y procedimientos necesarios para elaborar productos de software de calidad.

- El marco de trabajo desarrollado es posible extrapolar a otras instituciones financieras, cuya estructura tecnológica sea parecida al objeto de estudio de esta investigación y de igual manera a organizaciones que realicen desarrollo in house y posean un número limitado en el área de tecnología con la finalidad de mejorar la organización del área y establecer procedimientos que permita realizar desarrollo de software de manera más eficiente.

5.2. Recomendaciones

- En sectores económicos críticos como lo es el sector financiero, es necesario tener al menos tres ambientes al desarrollar software: ambiente de desarrollo, ambiente de control de calidad y ambiente de producción. Esta segmentación permite minimizar el riesgo de fraudes dentro de una organización financiera.

- La creación de un organismo externo al área de tecnología, como lo es el comité tecnológico en nuestro contexto, es de vital importancia para sostener el marco de trabajo elaborado. Este organismo permite filtrar los

requerimientos realmente necesarios para la organización y los prioriza de acuerdo a la importancia que el negocio requiere.

- El área de control de calidad es un área crítica dentro del proceso de desarrollo de software, por lo que es recomendable que la persona que ocupe ese cargo, sea una persona que conozca los procesos del negocio. De esta manera la organización asegura la liberación a producción de productos de software de buena calidad.

- Las auditorías informáticas son de suma importancia dentro del proceso de desarrollo de software, ya que permite mantener una disciplina al momento de desarrollar software de acuerdo a los procedimientos elaborados. Las auditorías internas al menos se las debe realizar dos veces al año y auditorías externas al menos una vez al año, con la finalidad de evaluar el cumplimiento de la misma y realizar recomendaciones para mejorarla.

- Se recomienda al especificar las actividades que se realizarán en el sprint, considerar el tiempo destinado para reuniones y soporte de usuario, lo que permitirá obtener un documento que describa las actividades reales a desarrollarse.

- Una vez afianzada el marco de trabajo se recomienda, validarla contra la norma ISO/IEC 15507 con la finalidad de tener una metodología más robusta.

BIBLIOGRAFÍA

- (2010). Ingeniería del software, un enfoque práctico. En R. Pressman, Ingeniería del software, un enfoque práctico (pág. 12). México, D.F.: McGraw-Hill Interamericana.
- Pressman, R. (2010). Ingeniería del software, un enfoque práctico. México, D.F : McGraw-Hill Interamericana.
- Schach, S. (2006). Ingeniería del software clásica y orientada a objetos. En S. Schach, Ingeniería del software clásica y orientada a objetos (págs. 5-6). México: McGraw-Hill Interamericana.
- Sommerville, I. (2002). Ingeniería del software. México D.F: Pearson Educación.
- Sommerville, I. (2002). Ingeniería del software. México D.F: Pearson Educación.
- Sommerville, I. (2011). En I. Sommerville, Ingeniería del software (págs. 5-6). México D.F: Pearson.
- Sommerville, I. (2011). En I. Sommerville, Ingeniería del software (págs. 5,6,8). México D.F: Pearson.
- Albariz, B. (2006). Resumen Objetivos, (Cmm), 1–50.

LINKOGRAFÍA

- boletines_mensuales - SEPS. (2015). Retrieved May 26, 2015, from http://www.seps.gob.ec/web/guest/boletines_mensuales
- Cadavid, A. N., Daniel, J., Martínez, F., & Vélez, J. M. (2013). Revisión de metodologías ágiles para el desarrollo de software A review of agile methodologies for software development, 30–39. Retrieved May 26, 2015, from http:// dialnet.unirioja.es/descarga/articulo/4752083.pdf.
- Definición.de. (no registra de no registra de no registra año). Definición.de. Recuperado el 07 de 05 de 2015, de Definición.de: http://definicion.de/planeacion/
- Sommerville, I. (2005). Ingeniería del software. Danielr.Obolog.Es. Retrieved from 07 de 05 de 2015 http://danielr.obolog.es/ingenieria-software-355416\nhttp://fondoeditorial.uneg.edu.ve/citeg/numeros/c02/c02_art10.pdf
- Vega Zepeda, V., Gasca Hurtado, G., & Echeverry Arias, J. (2012). Análisis Comparativo de Modelos de Calidad, 7. Retrieved 07 de 05 de 2015 from http://www.infonorchile2012.uta.cl/download.php?file=infonor2012_1.pdf
- VersionOne. (2013). 8th Annual State of Agile Survey - 2013-state-of-agile-survey.pdf. Retrieved November 18, 2015, from https://www.versionone.com/pdf/2013-state-of-agile-survey.pdf
- Wirth, N. (2008). A Brief History of Software Engineering. IEEE Annals of the History of Computing, 30(3), 32–39. Retrieved 01 de 01 de 2015 from http://doi.org/10.1109/MAHC.2008.33
- Capability Maturity Model Integration - Wikipedia, la enciclopedia libre. (2015). Retrieved August 13, 2015, from https://es.wikipedia.org/wiki/Capability_Maturity_Model_Integration
- Cataldi, Z. (2000). Calidad en la Industria del Software . La Norma ISO-9126. Retrieved 01 de 02 de 2015 from http://

repositorio.utp.edu.co/dspace/bitstream/11059/.../0053L864e_anexo.pdf.

- Deemer, Por Pete, Gabrielle Benefield, Craig Larman, B. V. (2009). Básica De Scrum (the Scrum Primer). Scrum Training Institute, 1.1, 1–20. Retrieved 04 de 04 de 2015 from http://cs.union.edu/~striegnk/courses/csc497/scrumprimer.pdf.
- Desarrollo en espiral - Wikipedia, la enciclopedia libre. (2015). Retrieved July 29, 2015, from https://es.wikipedia.org/wiki/Desarrollo_en_espiral
- e Standish Group International, I. (2013). ChaosManifesto2013.pdf. Retrieved November 18, 2015, from https://larlet.fr/static/david/stream/ChaosManifesto2013.pdf
- HERRAMIENTAS CASE: 3. CLASIFICACION. (2008). Retrieved August 14, 2015, from http://tpsis324.blogspot.com/2008/09/3-clasificacion.html
- IBM - Rational Rose Enterprise. (2015, July 28). IBM Corporation. Retrieved from http://www-03.ibm.com/software/products/es/enterprise
- IEEE 2009 Annual Report. (2009). IEEE.
- ISO/IEC 12207 - Wikipedia, la enciclopedia libre. (2015). Retrieved August 13, 2015, from https://es.wikipedia.org/wiki/ISO/IEC_12207
- ISO/IEC 12207:2008 - Systems and software engineering -- Software life cycle processes. (2008). Retrieved August 13, 2015, from http://www.iso.org/iso/catalogue_detail?csnumber=43447
- JDeveloper - Wikipedia, la enciclopedia libre. (2015). Retrieved July 29, 2015, from https://es.wikipedia.org/wiki/JDeveloper
- Martin, J. (1990). Rapid Application Development.
- Orna, C. (2013). Escuela politécnica del ejército extensión latacunga.
- Palacio, J. (2006). SW-CMM (CMM for Software) Historia y evolución Otros modelos CMM, 1–6.
- PMOinformatica. (2013). Una breve historia de las metodologías ágiles - La Oficina de Proyectos de Informática. Retrieved August 17,

2015, from http://www.pmoinformatica.com/2013/06/una-breve-historia-de-las-metodologias.html

- Rational Unified Process, E. (2012). Metodología RUP: Algo de Historia... Retrieved August 17, 2015, from http://rupequipo1.blogspot.com/2012/12/algo-de-historia.html
- Real Academia Española. Diccionario Usual. (n.d.-a). Retrieved August 13, 2015, from http://buscon.rae.es/drae/srv/search?id=IfgQnomA7DXX2Fvf5p2i
- Real Academia Española. Diccionario Usual. (n.d.-b). Retrieved August 13, 2015, from http://buscon.rae.es/drae/srv/search?id=1LrLYKMX6DXX2xXUseC5
- Sáez Martínez, P. (2013). Identificación y valoración de técnicas ágiles de gestión de proyectos software, 1, 120.
- Software Engineering Institute - Wikipedia, la enciclopedia libre. (2015). Retrieved July 29, 2015, from https://es.wikipedia.org/wiki/Software_Engineering_Institute
- Universidad de Murcia. (05 de 04 de 2014). Universidad de Murcia. Recuperado el 24 de 05 de 2015, de Universidad de Murcia: http://www.um.es/docencia/barzana/IAGP/IAGP2-Ingenieria-software-introduccion.html
- Universidad Politécnica de Valencia. (4 de 1 de 2011). Blog, Historia de la informática. Recuperado el 25 de 5 de 2015, de http://histinf.blogs.upv.es/2011/01/04/la-crisis-del-software/
- Villa, R. (27 de 08 de 2013). Prezi. Recuperado el 24 de 5 de 2015, de Historia de la ingeniería del software: https://prezi.com/kegowmp-4rwy/historia-de-la-ingenieria-de-software/
- Wikipedia. (20 de marzo de 2014). Crisis del software. Recuperado el 25 de 05 de 2015, de http://es.wikipedia.org/wiki/Crisis_del_software
- Wikipedia. (9 de 4 de 2015). Wikipedia. Recuperado el 29 de 5 de 2015, de Wikipedia: http://es.wikipedia.org/wiki/Desarrollo_en_cascada

ANEXOS

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

MAESTRÍA EN INGENIERÍA DE SOFTWARE

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por el Sr. Ing. Jairo Andrés Bejarano Montesdeoca, bajo nuestra supervisión.

Ing. Lucas Garcés Ms.C

DIRECTOR

Ing. Lucas Garcés Ms.C.

DIRECTOR DEL PROGRAMA

Dr. Rodrigo Vaca

SECRETARIO ACADEMICO