ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DE SOFTWARE

CUARTA PROMOCIÓN

TESIS PRESENTADA PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGISTER EN INGENIERÍA DE SOFTWARE

TEMA: "DESARROLLO DE UN MARCO DE TRABAJO, IMPLEMENTADO CON UNA HERRAMIENTA ALM, PARA MEJORAR LA GESTIÓN DEL PROCESO DE FABRICACIÓN DE SOFTWARE DEL ÁREA DE ANÁLISIS Y DESARROLLO DEL DEPARTAMENTO DE SISTEMAS DE LA EMPRESA FARMAENLACE CÍA. LTDA.".

AUTOR: ING. JOSÉ ANTONIO QUIÑA MERA

DIRECTOR: ING. JORGE GEOVANNY RAURA RUIZ. MSc.

LATACUNGA

2015

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DE SOFTWARE

CUARTA PROMOCIÓN

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"DESARROLLO DE UN MARCO DE TRABAJO, IMPLEMENTADO CON UNA HERRAMIENTA ALM, PARA MEJORAR LA GESTIÓN DEL PROCESO DE FABRICACIÓN DE SOFTWARE DEL ÁREA DE ANÁLISIS Y DESARROLLO DEL DEPARTAMENTO DE SISTEMAS DE LA EMPRESA FARMAENLACE CÍA. LTDA."** realizado por el señor **NOMBRE DEL AUTOR**, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo 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 **JOSÉ ANTONIO QUIÑA MERA** para que lo sustente públicamente.

Latacunga, 30 de noviembre de 2015

Ing. Jorge Geovanny Raura Ruiz. MSc.
DIRECTOR

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DE SOFTWARE
CUARTA PROMOCIÓN

AUTORÍA DE RESPONSABILIDAD

Yo, ING. **JOSÉ ANTONIO QUIÑA MERA**, con cédula de identidad 1002322384 declaro que este trabajo de titulación **"DESARROLLO DE UN MARCO DE TRABAJO, IMPLEMENTADO CON UNA HERRAMIENTA ALM, PARA MEJORAR LA GESTIÓN DEL PROCESO DE FABRICACIÓN DE SOFTWARE DEL ÁREA DE ANÁLISIS Y DESARROLLO DEL DEPARTAMENTO DE SISTEMAS DE LA EMPRESA FARMAENLACE CÍA. 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, 30 de noviembre de 2015

Ing. José Antonio Quiña Mera.
C.C. 1002322384

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y

TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DE SOFTWARE

CUARTA PROMOCIÓN

AUTORIZACIÓN

Yo, **JOSÉ ANTONIO QUIÑA MERA**, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación **"DESARROLLO DE UN MARCO DE TRABAJO, IMPLEMENTADO CON UNA HERRAMIENTA ALM, PARA MEJORAR LA GESTIÓN DEL PROCESO DE FABRICACIÓN DE SOFTWARE DEL ÁREA DE ANÁLISIS Y DESARROLLO DEL DEPARTAMENTO DE SISTEMAS DE LA EMPRESA FARMAENLACE CÍA. LTDA."** cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Latacunga, 30 de noviembre de 2015

Ing. José Antonio Quiña Mera.
C.C. 1002322384

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DE SOFTWARE
CUARTA PROMOCIÓN

DEDICATORIA

Esta tesis la dedico a DIOS, que es el dador de todo en la vida, a mi madre que siempre me apoya durante todas las etapas de mi vida con sus enseñanzas de vida, educación, apoyo y consejos.

A mi amada esposa Cathy y mi cuñado Alex que son con quienes compartí todo el proceso en este proyecto de aprendizaje.

A mis compañeros de estudio que se convirtieron en buenos amigos, los que hicieron que todo el programa de estudio se haga entretenido y llevadero a pesar de las largas jornadas de trabajo.

Antonio.

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DE SOFTWARE
CUARTA PROMOCIÓN

AGRADECIMIENTO

En primer lugar doy gracias a Dios por darme fuerzas, valor y sus bendiciones para culminar esta etapa de mi vida.

A mi amada esposa, mi querida madre por estar siempre junto a mí durante todo el proceso del programa de estudios y de la elaboración de este trabajo de tesis.

Agradezco de manera especial al Ing. Dennis Criollo y la empresa Farmaenlace Cía. Ltda., por la apertura y ayuda para culminar estos estudios de cuarto nivel.

A mis compañeros de trabajo, el equipo de desarrollo de Software de la empresa Farmaenlace Cía. Ltda., en donde se desarrolló e implementó el presente trabajo.

A mis compañeros maestrantes, al coordinador del programa el Ing. Lucas Garcés, a mi director de tesis Ing. Geovanny Raura, a todos los maestros que con su nivel de conocimientos realzaron el prestigio de la maestría.

Gracias a todos ustedes.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En la actualidad la empresa privada a nivel nacional ha logrado un crecimiento importante en el proceso de desarrollo de software, mediante la capacitación del talento humano y el mejoramiento de los procesos administrativos y de gestión, lo que ha permitido automatizar procesos empresariales de una manera ágil dando como resultado el aumento de la productividad en el entorno comercial. El objetivo de este proyecto es aportar a dicho crecimiento mediante el desarrollo de un marco de trabajo para la fabricación de software el cual se implementó y automatizó mediante una herramienta de administración de ciclo de vida de las aplicaciones (ALM). Mejorando la gestión del proceso de desarrollo realizado por el Área de Análisis y Desarrollo de Software de la empresa Farmaenlace Cía. Ltda. Para realizar lo antes mencionado se inició haciendo un estudio para establecer las principales problemáticas del proceso de fabricación de software en el área de análisis y desarrollo de la empresa, luego se hizo un estudio comparativo de las metodologías tradicionales y ágiles de desarrollo de software, en donde se determinó que las metodologías ágiles se adaptan para solucionar las problemáticas establecidas en el primer estudio. Posteriormente se hizo un nuevo estudio comparativo entre las principales metodologías ágiles para establecer la metodología base que fue utilizada para el desarrollo, tomando en cuenta la realidad empresarial. Se concluyó con la implementación del marco de trabajo, en donde se identificó aspectos medibles en las diferentes etapas de desarrollo de software que permitieron validar los indicadores propuestos en el presente proyecto.

PALABRAS CLAVE:

- **DESARROLLO DE SOFTWARE**
- **HERRAMIENTA ALM**
- **INGENIERÍA DE SOFTWARE**
- **EMPRESA FARMAENLACE CÍA. LTDA.**

ABSTRACT

Nowadays, the private sector at the national level has achieved significant growth in the software development process, through training of human talent and improving administrative and management processes, allowing us to automate business processes in an agile manner resulting in increased productivity in the commercial environment. The objective of this project is to add to mentioned growth through the development of a work frame to the software manufacturing which was implemented and automated through a management tool of lifecycle applications (ALM), improving the management of the development process done by the Department of Analysis and Software Development of the company Farmaenlace Limited Company. To do what was mentioned before, a study to stablish the main problems of the manufacturing of software in the Department of Analysis and Company development was started, then a comparative study of the traditional methodology and agile of the development of software was done, where was determined that the agile methodology adapt to solve the problems stablished in the first study. Later, a new comparative study was done between the main agile methodologies to stablish the base methodology that was used for the development, taking into account the company reality. It was concluded with the implementation of the work frame where was identified measurable aspects in the different stages of the software development that allowed to valid the indicators proposed in the current project.

KEY WORDS:

- **SOFTWARE DEVELOPMENT,**
- **AML TOOL,**
- **SOFTWARE ENGINEERING,**
- **FARMAENLACE LIMITED COMPANY**

CAPÍTULO I

GENERALIDADES

1. Introducción

Farmaenlace Cía. Ltda. es una empresa ecuatoriana con sede en Quito, dedicada a la distribución y comercialización de productos farmacéuticos y artículos de primera necesidad. Nace en el año 2005 a través de una alianza estratégica entre dos empresas importantes de distribución farmacéutica: Representante Ortiz Cevallos y farmacéuticas Espinosa.

Es importante destacar el crecimiento que ha tenido la empresa desde su inscripción, tanto así que en el año 2013 según la revista Vistazo publica en su ranking anual de empresas a nivel nacional, que Farmaenlace se sitúa dentro de las 100 empresas más importantes del país.

Actualmente Farmaenlace Cía. Ltda., es propietaria de las marcas: Farmacias Económicas, Farmacias Medicitys, Farmacias El Descuento, distribuidora farmacéutica Difarmes y asociados. Cuenta con 396 puntos de venta a nivel nacional entre farmacias propias, franquicias y Socios estratégicos. El talento humano de la empresa se conforma de 2.371 colaboradores.

1.1 Planteamiento del problema

El crecimiento de la empresa Farmaenlace Cía. Ltda. en los últimos años ha sido una constante y por aquello las necesidades de todas las áreas internas también se han multiplicado, lo que refleja un problema en el cumplimiento eficiente de los objetivos empresariales. Al momento se mantienen un número considerable de procesos administrativos que no

están sistematizados ni automatizados lo que dificulta la productividad de la empresa.

El área de Análisis y desarrollo de Software del departamento de sistemas de la Empresa Farmaenlace Cía. Ltda. se encarga en desarrollar sistemas a la medida para cubrir la demanda y necesidades internas. El área actualmente se conforma de 16 personas de los cuales hay 2 coordinadores y 14 desarrolladores. Entre los desarrolladores se destacan 3 líderes de proyectos que manejan un promedio de 2 desarrolladores cada uno. Además los líderes dan el soporte y mantenimiento de tres líneas de producción que son la comercialización, la distribución y compras. El resto de segmentos empresariales se distribuyen entre los coordinadores. Por lo que el área de desarrollo se organiza en 5 equipos de trabajo.

El manejo de la demanda interna de desarrollo de software y la coordinación de trabajos entre los equipos del área, ha sido bastante fuerte y complejos, por lo cual surgen varias necesidades que se pueden identificar a continuación:

- Falta de una metodología o marco de trabajo claro para el desarrollo de software.
- Dificultad para establecer los roles que desempeñan cada uno de los actores dentro del ámbito del proceso de desarrollo de software.
- Dificultad para fortalecer la gestión de la Planificación de trabajos de desarrollo de software.
- Falta de integración del control de versiones de las soluciones informáticas con la planificación de trabajo.
- Necesidad de un sistema de gestión para el ciclo de vida de las aplicaciones.
- Limitado acceso a la información de los proyectos en desarrollo y desarrollados.
- Dificultad en la integración y socialización del estado del desarrollo de los proyectos con otras áreas empresariales.

- Falta de reportes gerenciales y operativos de los desarrollos.

Por tal motivo, a continuación se identifica el siguiente problema.

1.2 Formulación del problema

¿Cómo mejorar la gestión del proceso de fabricación de software del área de análisis y desarrollo del departamento de Sistemas de la empresa Farmaenlace Cía. Ltda. ?

Tabla 1
Problemas identificados, Solución, Enfoque y Evidencias de la Solución

PROBLEMA	SOLUCIÓN	ENFOQUE	EVIDENCIA
Falta de una metodología o marco de trabajo claro para el desarrollo de software.	Desarrollo de un marco de trabajo (metodología) para mejorar el proceso del desarrollo de software.	Estrategia: Mejora de Procesos	Mejora en la gestión del desarrollo de software e incremento de la productividad del equipo de desarrollo.
Se debe establecer los roles que desempeñan cada uno de los actores dentro del ámbito del proceso de desarrollo de software.	Establecer y Capacitar acerca de los roles del equipo de trabajo en el desarrollo de software.	Estrategia: Competencias	Equipo de trabajo capacitado en roles específicos.
Necesidad de un sistema de automatización de la gestión del ciclo de vida de las aplicaciones.	Implementar un sistema para automatizar la gestión en el Ciclo de vida de las aplicaciones	Estrategia: Seguimiento y Control.	Seguimiento y Control automatizados del ciclo de vida de las aplicaciones.

CONTINÚA

	basado en el marco de trabajo desarrollado.		
Dificultad en la Integración y socialización del estado del desarrollo de los proyectos con otras áreas empresariales.	Reportes operativos y gerenciales mediante la herramienta que automatiza la gestión del desarrollo de software.	Estrategia: Resultado del marco de trabajo propuesto.	Modelo Propuesto

1.2.1 Preguntas de Investigación

¿Cómo conceptualizar la gestión del proceso de fabricación de software del área de análisis y desarrollo del departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.?

¿Cómo automatizar la gestión del proceso de fabricación de software del área de análisis y desarrollo del departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.?

¿Cómo aplicar y poner en funcionamiento la automatización de la gestión del proceso de fabricación de software del área de análisis y desarrollo del departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.?

¿Cómo verificar el resultado de la automatización en la gestión del proceso de fabricación de software del área de análisis y desarrollo del departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.?

1.2.2 Limitaciones y supuestos

Este proyecto pretende llegar al desarrollo, aplicación y validación de un marco de trabajo, implementado con una herramienta ALM, para mejorar la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda. No es posible aplicar los métodos de Aseguramiento de Calidad dentro del desarrollo de software por la limitación del tiempo y recursos.

1.3 Objetivo general

Desarrollar un marco de trabajo, implementado con una herramienta ALM, para mejorar la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.

1.3.1 Objetivos específicos

Determinar el marco teórico vinculado a la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.

Diseñar y desarrollar un marco de trabajo, implementado con una herramienta ALM, para mejorar la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.

Implementar el marco de trabajo con una herramienta ALM, para mejorar la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.

Validar los resultados obtenidos del desarrollo de un marco de trabajo, implementado con una herramienta ALM, para mejorar la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.

1.3.2 Meta

Desarrollo de un marco de trabajo, implementado con una herramienta ALM, para mejorar la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda., de la ciudad de Ibarra, durante el período agosto 2014 – noviembre 2015.

1.4 Hipótesis

¿El desarrollo de un marco de trabajo, implementado con una herramienta ALM, mejorará la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.?

1.4.1 Variables de la investigación

Variable dependiente:

Se mejora la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.

Indicadores

- Porcentaje de cumplimiento de las actividades en la planificación del trabajo, dentro de las iteraciones del proyecto de software, en el área de

análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.

- Porcentaje de desarrollo de trabajos nuevos y de corrección de errores, en la planificación del trabajo, dentro de las iteraciones del proyecto de software, en el área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.

- Porcentaje de tareas no planificadas, en la planificación del trabajo, dentro de las iteraciones del proyecto de software, en el área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.

- Porcentaje de tareas en las fases del desarrollo de software, en la planificación del trabajo, dentro de las iteraciones del proyecto de software, en el área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.

Variable independiente:

Se desarrolla un marco de trabajo, implementado con una herramienta ALM.

Conceptualización de la Variable Independiente:

El **marco de trabajo** es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos que puede servir de base para la organización y desarrollo de software.

La **Herramienta ALM** (Application Lifecycle Management / Administración del Ciclo de Vida de las Aplicaciones) es la herramienta que gestiona las actividades del ciclo de vida del desarrollo software, además facilita e integra la gestión de requerimientos, arquitectura, codificación, pruebas, seguimiento y versiones.

1.5 Justificación e importancia

Debido a la evolución tecnológica y de mercado, las empresas deben estar inmersas en un cambio permanente de sus sistemas informáticos, de ahí nace la necesidad de tener la capacidad de cambiar o fabricar software empresarial de una manera oportuna de acuerdo a las necesidades.

Para tener la capacidad de fabricación o cambio de las aplicaciones informáticas, primero se debe tener un marco de trabajo claro y práctico para gestionar el proceso del desarrollo de software eficaz, eficiente y que esté dentro de las normas de calidad necesarias para satisfacer los requerimientos y demanda empresarial.

Es importante técnicamente, porque la fabricación de software mediante un marco de trabajo se lo hace de una manera organizada, estructurada y con reglas claras en todo el ciclo de vida de las aplicaciones, el cual puede ser gestionado y automatizado con una herramienta ALM; además que aporta mucho a la integración de las aplicaciones dentro de la empresa. Adicionalmente se puede rescatar las experiencias de la gestión de proyectos anteriores, lo cual sirve para tener una mejora continua, dar mantenimiento y ofrecer actualizaciones de acuerdo a diferentes contextos y entornos planteados.

Económicamente el proyecto es importante porque ayuda a reducir costos, dentro de la optimización de procesos y recursos en la fabricación de software de la empresa Farmaenlace Cía. Ltda. Además la aplicación del marco de trabajo, métricas, indicadores, procedimientos claros y prácticos es un aporte significativo al problema de operatividad y socialización no sólo en el Departamento de Sistemas, sino también en los demás departamentos y áreas internas de la empresa que hacen uso de los sistemas informáticos desarrollados.

Socialmente es importante debido a que el proyecto incentivará a los miembros del equipo del área de análisis y desarrollo del Departamento de Sistemas, a trabajar de una manera ordenada y armónica en todas las fases de desarrollo mediante una administración de trabajos y recursos, y así poder evitar el incumplimiento de trabajos planificados, o en su defecto justificar de manera adecuada y consiente las demoras de las entregas. Esta manera organizada de trabajo mejora los procesos en el área de desarrollo y aumenta los conocimientos e imagen profesional y personal dentro y fuera de la empresa a los miembros del equipo de trabajo.

CAPÍTULO II

MARCO TEÓRICO

2. Introducción

Para una mejor apreciación de la organización del presente trabajo, este capítulo se inicia mostrando un mapa conceptual en donde se define dentro del estado del arte: la evolución histórica de las metodologías en el desarrollo de software, la definición del proceso del desarrollo de software, la evaluación comparativa de las metodologías más importantes y herramientas ALM, con lo cual se determinará el marco de trabajo propio enfocado a mejorar la gestión del desarrollo de software.

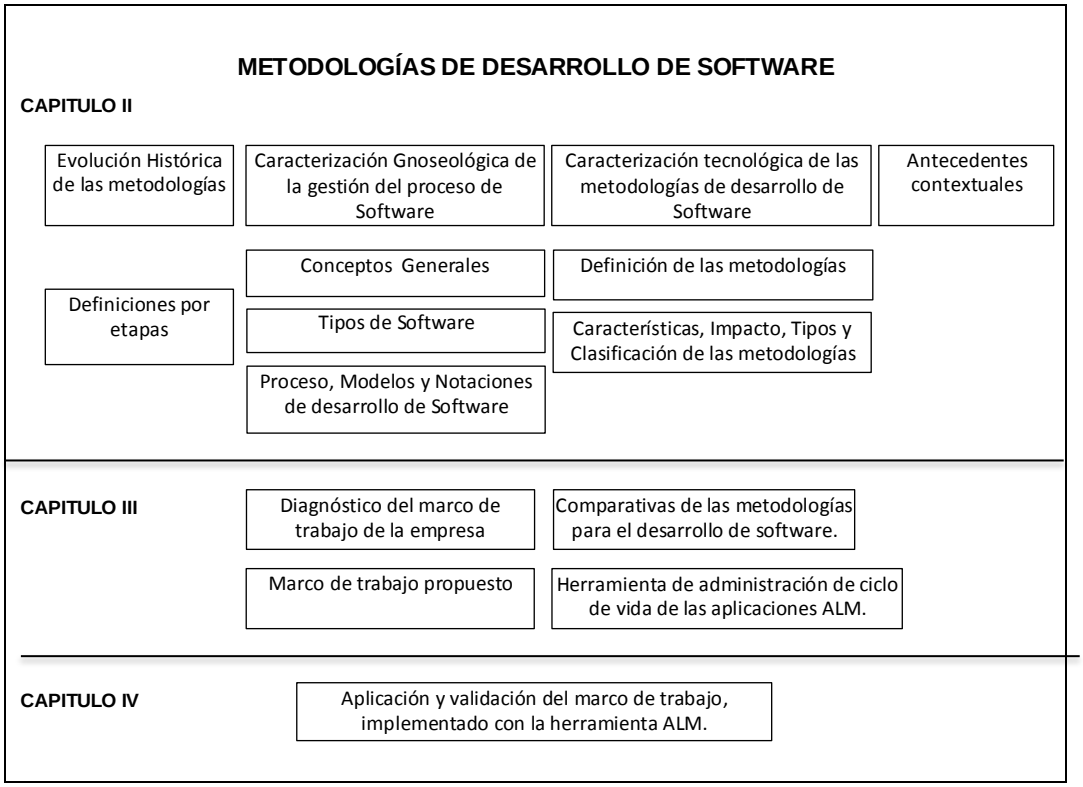


Figura 1 Mapa Conceptual sobre la investigación realizada

2.1 Estado del arte

El estado del arte del proyecto se presenta en base al mapa conceptual expuesto en la sección anterior, que permitió desarrollar el tema de tesis propuesto.

2.2 Evolución histórica de las metodologías en la gestión del proceso de fabricación de software

En este apartado se presenta la evolución en el tiempo de las metodologías, es decir la evolución del conjunto de métodos, técnicas y buenas prácticas utilizadas para el desarrollo de software.

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

2.2.1 Etapa 1: Década de 1970 – Lenguajes de programación

El comienzo de las metodologías de cierta manera se les puede atribuir a la aparición de los lenguajes de programación y con ellos la manera de programar. En 1969 aparece la programación estructurada SOL por sus siglas en Inglés (Structured Oriented Languaje), el objetivo principal es brindar al programador un lenguaje para reflejar una metodología de concepción de ideas, más que un lenguaje de programación estándar. Este modelo separa la estructura de control de un programa con el resto de las instrucciones del lenguaje de programación, la cual se puede modelar por el

comportamiento de una jerarquía de sentencias y objetos de computación (Torre, Brueghel, & Commons, 2008).

Realizar programas con este lenguaje o metodología resulta de utilidad como un medio unificado de programación, ya que la solución de un problema puede plantearse con un grado de independencia del lenguaje a utilizar (Fontao, Delrieux, Kalocai, Goñi, & Ramoscelli, 2002).

En 1975 el científico inglés Michael A. Jackson publicó su libro "Principios de Diseño del Programa" (Principles of Program Design), el cual fue un precedente ya que desde su lanzamiento no se ha vuelto a publicar otro método general de Diseño Estructurado de Programas, el motivo es porque teóricamente no hay mucho que aportar porque los principios solventan de una manera completa y muy clara el diseño de programas estructurados. Describe la solución del procedimiento formal de diseño, otorgando claridad y lógica, además tiene la característica de que a los programas diseñados se lo pueda codificar en cualquier lenguaje, los más importantes y significativos de la época eran Cobol, Fortran y PL/1. Lo más relevante del método es que todo el diseño de la estructura del programa se basa en los datos en términos de secuencias, iteraciones y selecciones que luego se aplican al diseñar la estructura detallada, conocida y fija del control de programas, los cuales seguirán pasos concretos y definidos que terminan en codificación de pseudocódigo, o directamente en cualquier lenguaje de programación (Macluskey, 2009).

2.2.2 Etapa 2: Década de 1980 – Sistemas estructurados y métodos de diseño

A inicios de 1980 se marca el paso del **Análisis de Sistemas Estructurados y Métodos de Diseño** ("SSADM" por sus siglas en inglés de "Structured Systems Analysis and Design Method"), se trata de un conjunto de normas para el análisis de sistemas y aplicaciones con un diseño

ampliamente utilizado para proyectos de computación, en Europa tuvo gran aceptación por el gobierno del Reino Unido.

SSADM es una combinación de texto y diagramas a lo largo de todo el ciclo de vida de un sistema, desde la idea inicial hasta el diseño físico de la aplicación. Utiliza la combinación de tres técnicas de identificación, modelado y documentación: “El modelado de datos lógicos” se encarga de los requisitos de datos del sistema. El “Flujo de datos de modelado” se ocupa del modelado de cómo los datos se mueve alrededor de un sistema de información. Y la “Entidad Comportamiento Modelado”, se encarga de los eventos que afectan a cada entidad y la secuencia en que se producen estos eventos. Estas tres técnicas proporcionan modelos con un punto de vista diferente del diseño del mismo sistema, y se necesita cada punto de vista para formar un modelo completo, con esta referencia cruzada se puede garantizar la integridad y exactitud en las aplicaciones (ITC Infotech, 1988).

Análisis Estructurado y Técnicas de Diseño (“SADT” por sus siglas en inglés de Structured Analysis and Design Technique), ha sido desarrollado y probado en campo durante 1969 a 1973 por Douglas T. Ross y SofTech, y el punto de partida para un uso extensivo fue en 1973 por la fuerza aérea de los Estados Unidos, y además lo formalizó y publicó como IDEF0 en 1981. Es una metodología de Ingeniería de Software para describir sistemas con una jerarquía de funciones mediante el análisis estructurado del lenguaje de modelado que utiliza dos tipos de diagramas: modelos de actividad y modelos de datos. El objetivo es ayudar a describir y comprender los sistemas mediante bloques de construcción; las entidades son representadas por cajas y las actividades representadas por una variedad de flechas para relacionar estas cajas (Diagrams, 2004).

Ingeniería de la información (“IE” por sus siglas en inglés de “Intelligence Engennier”) o metodología de ingeniería de la información (IEM), desde 1981 es la aplicación de una sucesión de técnicas integradas para el planeamiento, análisis, y diseño de las cuales se construyen modelos

de empresas, datos y procesos, estas se basan en un amplio conocimiento y son usadas para crear y mantener los sistemas de procesamientos de datos de sistemas de información. Las técnicas de la Ingeniería de la Información contienen las técnicas de la Ingeniería del Software de una manera modificada para aplicar técnicas estructuradas a un proyecto, las que se aplican a la empresa o a un amplio sector de la empresa, tomando como un todo al sector aplicado (Martínez Méndez, 1996).

2.2.3 Etapa 3: Década de 1990 – Desarrollo rápido de aplicaciones

El **Desarrollo Rápido de Aplicaciones** (“RAD” por sus siglas en inglés de Rapid Application Development), comenzó con las ideas de Barry Boehm y Scott Shultz, luego Martin desarrolló RAD durante los años 1980 en IBM y finalmente lo formalizó publicando un libro en 1990. Es un método que comprende el desarrollo iterativo, la construcción de prototipos de las aplicaciones y el uso de herramientas y utilidades CASE (siglas en inglés de Computer Aided Software Engineering), y por lo general también suele abarcar la usabilidad, utilidad y la rapidez de ejecución (Reece, 2006).

RAD es una adaptación a “Alta velocidad” que utiliza un enfoque de construcción mediante el desarrollo lineal y secuencial de software basado en componentes que enfatiza en un ciclo de desarrollo considerablemente corto, en donde el equipo de desarrollo tiene claros los requerimientos y se limita al ámbito del proyecto, para crear sistemas completamente funcionales en periodos cortos de tiempo, principalmente para aplicaciones de sistemas de información (Reece, 2006).

Programación orientada a objetos (“OOP” por sus siglas en inglés de Object Oriented Programming), su utilización fue popularizado en la década de los años 1990, el término de Programación Orientada a Objetos indica más que modelos de diseño, significa una metodología de desarrollo de software o un lenguaje de programación, ya que es todo un paradigma que

usa objetos en sus interacciones para diseñar aplicaciones y programas informáticos en cualquier lenguaje de programación que soporte el orientado a objetos.

La diferencia básica con los lenguajes estructurados es que los lenguajes estructurados son una lista de acciones de una determinada secuencia sobre un conjunto de datos, por tal motivo si en el transcurso del desarrollo se modifica la estructura de los datos o las acciones realizadas, el programa también cambia, caso contrario la OOP aporta un enfoque diferente convirtiendo la estructura de datos en el centro sobre el que pivotan las operaciones, de esta forma las modificaciones de la estructura de datos tiene efecto inmediato sobre las acciones que se realizan sobre ellas (Aguilar, 1996).

La OOP está basada en varias técnicas como la herencia, la cohesión, la abstracción, el polimorfismo, el acoplamiento y el encapsulamiento que utilizan objetos como elementos fundamentales en la construcción de la solución. Un objeto se considera a la abstracción de algún hecho o ente del mundo real, estos contienen dos partes fundamentales: los “atributos” que representan sus características o propiedades, y los “métodos” que emulan su comportamiento o actividad; todas las propiedades y métodos comunes a los objetos se pueden encapsular o agrupar en clases, estas se consideran como una plantilla, en donde se puede decir que cada objeto es una instancia o ejemplar de una clase (Aguilar, 1996).

La **máquina de estado finito virtual** (“VFSM” por sus siglas en inglés de “Virtual finite state machine”), desde 1990s se refiere a una máquina de estados finitos (FSM) definida en un entorno virtual, proporciona un método de especificación de software para describir el comportamiento de un sistema de control, por lo que su aplicación y utilización principalmente se ha visto centrada en software de control de máquinas, de instrumentación y telecomunicaciones complejas. Su funcionamiento y estructura básicamente

se conforma por Propiedades de Control, Acciones, el Entorno Virtual y Ejecución del Modelo (Wagner, Wagner, Wolstenholme, & Group, 2006).

Método de desarrollo de sistemas dinámicos ("*DSDM"* por sus siglas en inglés de "*Dynamic Systems Development Method"*), fue desarrollado en el Reino Unido en los años 90 por un consorcio de proveedores y de expertos en la materia del desarrollo de sistemas de información (IS), su primera versión fue terminada en enero de 1995 y posteriormente publicada en febrero del mismo año, el método provee un framework (marco de trabajo) para el desarrollo ágil de software de manera interactiva e incremental, es una extensión del RAD, que básicamente se enfoca en los proyectos de sistemas de información. El método reconoce que los proyectos son limitados por tiempos y recursos, por lo que se dirige al manejo de problemas más frecuentes que ocurren en el desarrollo de los sistemas de información, como es sobrepasar tiempos de entrega de desarrollos, presupuestos, falta de participación del usuario en el proyecto y el apoyo de comisiones por la gerencia (Coleman & Verbruggen, 1998)**.**

La metodología **Scrum** es un modelo de desarrollo ágil sus orígenes lo tiene en la década de los 80, pero no es hasta 1993 que se realizó el primer Scrum para desarrollo de software, y en 1995 Ken Schwaber formalizó el proceso para la industria de desarrollo de software. Scrum es una metodología ágil y flexible para gestionar el desarrollo de software, basándose en un conjunto de buenas prácticas para trabajar colaborativamente en equipo, y obtener el mejor resultado posible de un proyecto cuyo principal objetivo es maximizar el retorno de la inversión para su empresa (ROI) (Johnston, 2006).

Estas buenas prácticas interactúan y se apoyan entre sí y su selección y aplicación se origina de un estudio de la manera de trabajo de equipos altamente productivos, la estrategia para aplicar el método es construir primero la funcionalidad de mayor valor para el cliente apoyado de una inspección continua, adaptación, innovación y auto-gestión, luego se

continúa con entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto, por esta manera de trabajo se pude decir que Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos (Johnston, 2006).

Scrum está caracterizado por adoptar una estrategia de desarrollo incremental en lugar de una planificación y ejecución completa del producto, la calidad del resultado se basa más en el conocimiento tácito de los equipos de trabajo auto organizados, que en la calidad de los procesos empleados (ScrumInc, 2012). En el siguiente apartado se expondrá los conceptos de la metodología Scrum de una manera más detallada (Johnston, 2006).

El **Proceso Racional Unificado** (“RUP” por sus siglas en inglés de “Rational Unified Process”), su publicación fue en 1999 por Ivar Jacobson, Grady Booch y James Rumbaugh. es un proceso de desarrollo de software desarrollado por la empresa Rational Software, actualmente propiedad de IBM, consiste en un conjunto de metodologías adaptables al contexto y necesidades de las empresas que permite el desarrollo de software a gran escala, mediante la información entrelazada de diversos artefactos (documentación), estrategias, objetivos, actividades, en un proceso continuo de pruebas y retroalimentación a lo largo de las fases del desarrollo lo que garantiza el cumplimiento de ciertos estándares de calidad, el esfuerzo del recurso humano se lo enfoca en términos de habilidades, competencias y capacidades para asumir roles específicos con responsabilidades bien definidas. Por su robusta documentación suele tener inconveniente por generar complejidad al generar los controles de administración, sin embargo hay autores que afirman que los beneficios obtenidos recompensan el esfuerzo invertido en este aspecto (Rumbaugh, Jacobson, & Booch, 1999).

La **Programación Extrema** (“XP” por sus siglas en inglés de eXtreme Programming) es una metodología de desarrollo de la Ingeniería de Software formulada por Kent Beck, autor del primer libro sobre la materia en 1999, la

programación extrema se diferencia de las metodologías tradicionales principalmente porque se basa en la adaptabilidad de los cambios de los requisitos más que en la previsibilidad de los mismos, su éxito se atribuye porque enfatiza la satisfacción del cliente brindándole el software que necesita en lugar de entregar todo el software que desea en una fecha lejana.

XP promueve el trabajo de equipo de manera horizontal ya que los gerentes, clientes y desarrolladores son miembros iguales que se desenvuelven en un entorno simple pero eficaz dentro de un marco de colaboración que se auto-organiza alrededor del problema, por este motivo les permite ser altamente productivos. En la actualidad XP es una de las metodologías ágiles más populares y ha demostrado ser muy exitosa en muchas empresas de todos los tamaños e industrias en todo el mundo (Wells, 2003). En el siguiente apartado se expondrá los conceptos de la metodología XP de una manera más detallada

2.2.4 Etapa 4: Desde 2000 – Procesos de desarrollo de software

El **Proceso Unificado Empresarial** (“EUP” por sus siglas en inglés de “Enterprise Unified Process”), es una variante extendida del Proceso Unificado “RUP” que fue desarrollado por Scott W. Ambler y Larry Constantino en el año 2000, y luego reelaborado en 2005 por Ambler, John Nalbone y Michael Vizdos, es una metodología de desarrollo para sistemas de información orientada a la empresa, originalmente su introducción al mercado es justamente para superar algunas carencias de RUP, para lo cual añadieron dos fases y varias disciplinas nuevas (Scott & Constantine, 2000).

EUP trata al desarrollo de software desde un punto de vista como el cliente lo ve, una práctica para construir, mejorar o sustituir sistemas informáticos, integrados al ciclo de vida de la TI (tecnología informática) de la empresa/negocio o del ciclo de vida de la empresa misma. En 2013 se

comenzó a trabajar para evolucionar el EUP basándose en la Entrega Ágil Disciplinada en lugar del proceso unificado.

Metodología de diseño Construccionista (“CDM” por sus siglas en inglés de Constructionist design methodology), fue desarrollado por el Investigador de IA (Inteligencia Artificial) Kristinn R. Thorisson y sus estudiantes de la Universidad de Columbia y la Universidad de Reikiavik desde 2004. La metodología se usa para el desarrollo de robótica cognitiva, comunicativa de humanoides y sistemas de Inteligencia Artificial, la creación de este tipo de sistemas requiere la Integración de un gran número de funcionalidades que deben coordinarse cuidadosamente para lograr un comportamiento de un sistema coherente, se basa en el proceso de diseños iterativos que conduce a la creación de una red de módulos de nombres interactivos a través de corrientes explícitas y mensajes discretos (Thórisson & Nivel, 2009).

CDM se ha utilizado en la creación de muchos sistemas que incluyen la robótica, animación facial, simulación a gran escala y humanos virtuales, uno de los primeros sistemas de la época es Mirage, un humano simulado en un entorno de realidad aumentada que podría interactuar con la gente a través del habla y de gestos (Thórisson & Nivel, 2009).

El Proceso Unificado Ágil (“AUP” por sus siglas en inglés de “Agile Unified Process”), Scott Ambler desde 2001 intenta imprimir aspectos ágiles en la metodología RUP hasta que en 2005 libera la primera versión de AUP que termina siendo una versión simplificada del Proceso Unificado de Rational (RUP) de IBM. Esta metodología describe de manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas y conceptos ágiles, que aun así se mantienen válidos en RUP, dentro de sus técnicas ágiles incluye Desarrollo Dirigido por Pruebas (test driven development - TDD), Modelado Ágil AMDD (Agile Model Driven Development), Gestión de Cambios Ágil, y Refactorización de Base de Datos para mejorar la productividad, que se aplican en un ciclo de vida serial en lo

grande, e iterativo en lo pequeño, liberando entregables incrementales en el tiempo (Johnston, 2006).

Tomando en cuenta la evolución historia de las metodologías, se puede tener una visión más clara acerca de las funcionalidades y tendencias en las que se puede basar el estudio para desarrollar el marco de trabajo.

2.3 Caracterización gnoseológica de la gestión del proceso de fabricación de software

La gestión del proceso de fabricación de software, se clasifica dentro de la etapa de planificación en la gestión de proyectos de software.

Es este segmento para entender de mejor manera el concepto y la clasificación en la que se encuentra la gestión en el desarrollo o fabricación de software, es necesario entender algunos conceptos claves para tener una visión general del proceso, y poder cumplir los objetivos básicos que son estimar costes, tener conocimiento de los recursos necesarios, obtención y presentación de métricas. Y así con estos aspectos poder establecer un plan efectivo para los proyectos.

2.3.1 Conceptos generales en el proceso de fabricación de software

Entre los conceptos para comprender y definir mejor el proceso de fabricación de software se cita los siguientes:

a) Ingeniería de software

De acuerdo al IEEE, dice que la Ingeniería de Software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación, y mantenimiento del software.

Y según Bohem en 1976, la Ingeniería de Software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también con los términos de desarrollo, producción o fabricación de software.

b) Proyectos de software

Un proyecto es temporal con un principio y fin, normalmente limitado en tiempo, costos y/o entregables, diseñado para producir un producto, servicio o resultado, que dará lugar a un cambio o valor agregado.

c) Gestión de proyectos de software

La Gestión de Proyectos se puede definir como una disciplina que se encarga del proceso de planificación, ejecución y control de proyectos, desde su comienzo hasta su implementación, el propósito es alcanzar un objetivo final en un plazo de tiempo, costo y nivel de calidad determinados, a través de la administración de recursos técnicos, financieros y humanos (Garzás, 2013) .

Las etapas de la Gestión de Proyectos de Software son las siguientes.

- Planificación de Proyectos de Software.
- Proceso de Desarrollo de Software.
- Estructura Orgánica en Proyectos de Software.
- Mediciones en Producto y Proceso de Software.
- Estimación en Proyectos de Software.
- Control en proyectos de software.
- Aseguramiento de la Calidad del Software- SQA.
- Gestión de Configuración del Software – SCM.
- Testing en Productos de Software.

d) Producto de software

El Software según el estándar 729 de la IEEE, es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación. En el medio existe una polémica como tratar al software, si como producto o como servicio, ya que tiene características de ambos términos, en este apartado se lo va a referir como producto software. Los productos se pueden clasificar en:

- Productos genéricos: Son producidos por organizaciones y comercializados en el mercado de manera masiva.
- Productos hechos a medida: Son sistemas desarrollados a la medida, el objetivo es adaptarse de la mejor manera para satisfacer las necesidades del usuario.

Estos productos deben cumplir varias características al ser entregados, estas son:

- Mantenibles: La facilidad con que un sistema software o componente puede ser modificado para corregir faltas, mejorar rendimiento u otros atributos, o adaptar a un entorno cambiante.
- Confiabilidad: probabilidad de una operación de software de estar libre de fallos por un periodo específico de tiempo en un entorno específico.
- Eficiencia: El software debe administrar de la mejor manera los recursos.
- Utilización adecuada: Debe tener Interfaz de usuario y documentación adecuada (Montilva, 2006).

2.3.2 Tipos de software

Para especificar los tipos de software se presenta la siguiente clasificación:

Tabla 2

Tipos de software

CLASIFICACIÓN	TIPO	DEFINICIÓN
Por su estructura	Funcional	Son un conjunto de funciones puramente matemáticas que definen un objeto específico de aplicación.
	Orientado a listas	Hereda la potencia y versatilidad en el manejo de listas.
	Orientado a objetos	El software se organiza como una colección de objetos discretos que contiene tanto estructura de datos como también un comportamiento.
	Orientado a componentes	El componente, es un pedazo de software capaz de realizar acciones y comunicarse con otros componentes.
Por su función	Software de Gestión. (Producto)	Permite gestionar y administrar procesos de negocio de manera integrada.
	Software de Sistemas.	Programas escritos para servir a otros programas, tienen una fuerte interacción con el hardware, soporta múltiples usuarios que operan concurrente compartiendo recursos y estructuras de datos complejas.
Por su plataforma de ejecución	Sistemas embebidos	Diseñado para realizar una o pocas funciones dedicadas, frecuentemente es utilizado sistema de tiempo real.
	Sistemas de computación distribuida	Se define como una colección de computadoras separadas físicamente y conectadas entre sí por una red de comunicaciones.
	Sistemas de computación paralela	Utiliza simultáneamente múltiples elementos de procesamiento para resolver un problema.
	Sistemas de tiempo real	Sistema informático que interacciona con su entorno físico y responde a los estímulos del entorno dentro de un plazo de tiempo determinado.
	Sistemas basados en Chips	Tecnologías de fabricación que integran todos o gran parte de los módulos o componentes de un computador o cualquier otro sistema informático o electrónico en un único circuito

CONTINÚA

		integrado o chip.
	Sistemas de cómputo ubicuos	Computación ubicua (ubicomp) es entendida como la integración de la informática en el entorno de la persona, de forma que los ordenadores no se perciban como objetos diferenciados.
	Wearable computing systems (computadora corporal o vestible)	Dispositivo electrónico que lleva una persona y que lo puede portar debajo, junto o por encima de su vestimenta.

2.3.3 Proceso de desarrollo de software

El Proceso de Desarrollo de Software forma parte de la planificación dentro la gestión de proyectos de software, el objetivo de este proceso es la transformación de necesidades del usuario en un producto de software aprobado y certificado para su operación. No existe una manera universal de manejar este proceso para abastecer los diferentes contextos en los que se aplica, pero a pesar de esto existe un conjunto de actividades comunes y fundamentales que son:

- **Especificación de Software:** Se define las funcionalidades operativas y restricciones funcionales que debe cumplir el software.
- **Diseño e Implementación:** Se diseña y construye el software de acuerdo a la especificación.
- **Validación:** El software construido se debe validar, para verificar que cumpla con los requerimientos del cliente.
- **Evolución:** El software debe evolucionar, para adaptarse a las necesidades del cliente.

Según Pressman, adicional a las actividades antes expuestas, menciona un conjunto de “actividades protectoras”, que se aplican a lo largo del proceso de desarrollo del software y son las siguientes:

- Seguimiento y control de proyecto de software.
- Revisiones técnicas formales.
- Garantía de calidad del software.
- Gestión de configuración del software.
- Preparación y producción de documentos.
- Gestión de reutilización.
- Mediciones.
- Gestión de riesgos.

Pressman caracteriza un proceso de desarrollo de software como se muestra en la ilustración, con los elementos que se describen a continuación:

Figura 2 Elementos del proceso de software

Fuente: (BARINIA, 2010)

- Un marco común del proceso, definiendo un pequeño número de actividades del marco de trabajo que son aplicables a todos los proyectos de software, con independencia del tamaño o complejidad.

- Un conjunto de tareas, cada uno es una colección de tareas de ingeniería del software, hitos de proyectos, entregas y productos de trabajo del software, y puntos de garantía de calidad, que permiten que las actividades del marco de trabajo se adapten a las características del proyecto de software y los requisitos del equipo del proyecto.
- Las actividades de protección, tales como garantía de calidad del software, gestión de configuración del software y medición, abarcan el modelo del proceso. Las actividades de protección son independientes de cualquier actividad del marco de trabajo y aparecen durante todo el proceso (Pressman, 2001).

2.3.4 Modelos en el desarrollo de software

Entre los modelos más relevantes en el desarrollo de software se describen los siguientes:

a) Modelo secuencial lineal

Se le conoce también como modelo en Cascada, Modelo Clásico o Modelo Tradicional, es uno de los más antiguos y el más usado, se caracteriza porque su estructura está basado en un enfoque lineal, en donde la característica más importante es que los requerimientos deben estar bien definidos y estables para que cuando ocurra algún cambio o mejoría en su ciclo de vida, el software siga su curso (Fl, 2010).

Fases: Análisis de Requisitos (definición de necesidades y requerimientos del usuario, se debe hacer un documento de especificación de requisitos), Diseño del Sistema (diseño arquitectónico y detallado de los que va hacer el sistema, se debe realizar un documento de Diseño de Software), Diseño del Programa (desarrollo de los algoritmos y análisis de las herramientas a utilizar), Codificación (se implementa el código fuente del sistema), Pruebas (se comprueba los elementos ya programados en el

sistema para revisar su funcionalidad), Verificación (el usuario final ejecuta el sistema, se somete a las pruebas finales para dejar entregando y validando), Mantenimiento (es una de las etapas más críticas debido a que durante el tiempo de uso el sistema debe continuar su vida útil a través de actualizaciones).

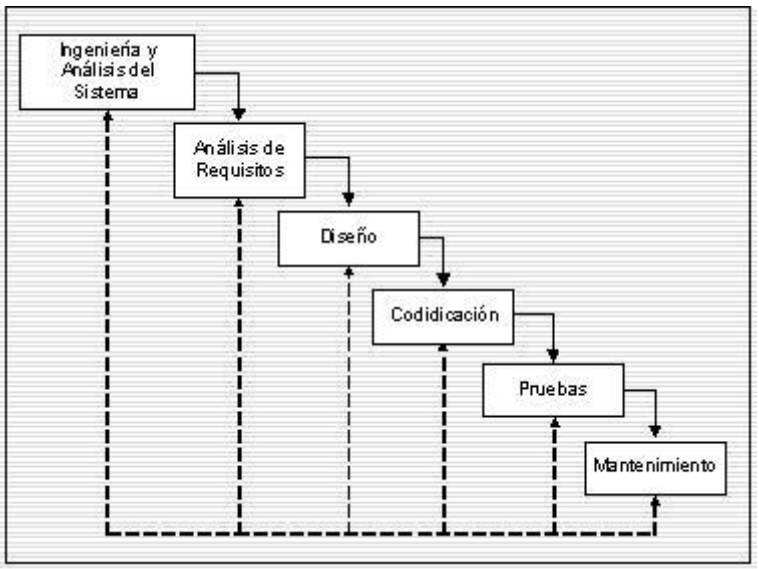


Figura 3 Modelo Secuencial Lineal

Fuente: (Calero, 2010)

b) Modelo de construcción de prototipos

Este modelo permite al usuario final visualizar y utilizar un prototipo del sistema, el mismo que es construido de manera rápida para que luego sea retroalimentado por los mismos usuarios, puede considerarse en la fase de requerimientos como una manera eficiente de verificar la funcionalidad del sistema por parte del usuario final y dar continuidad a su desarrollo. Es recomendable utilizar este modelo cuando los requerimientos aún no están definidos (Fl, 2010).

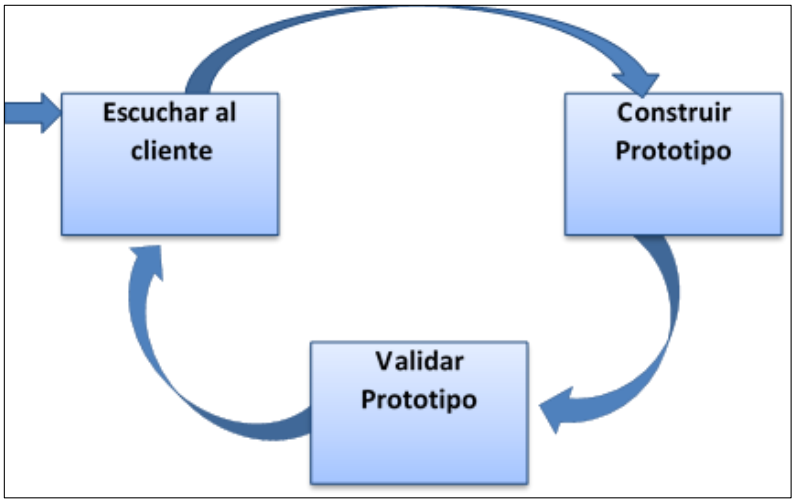


Figura 4 Modelo de construcción de prototipos

Fuente: (Angelfire, 2010)

c) Modelo evolutivo

Es un modelo que se adapta a los cambios que van realizándose a lo largo del desarrollo del sistema, mantienen fases interactivas en donde se obtienen versiones más completas del sistema. Existen algunos ejemplos de modelos evolutivos como son:

Modelo incremental

Combina elementos del modelo en cascada aplicado en forma interactiva, mantienen secuencias lineales que se convierten en incrementos en función del tiempo, es aplicable cuando el personal necesario para una implementación completa de software no está disponible, así, si se desea incorporar recurso de cualquier tipo se lo puede hacer sin entorpecer el desarrollo del sistema (Fl, 2010).

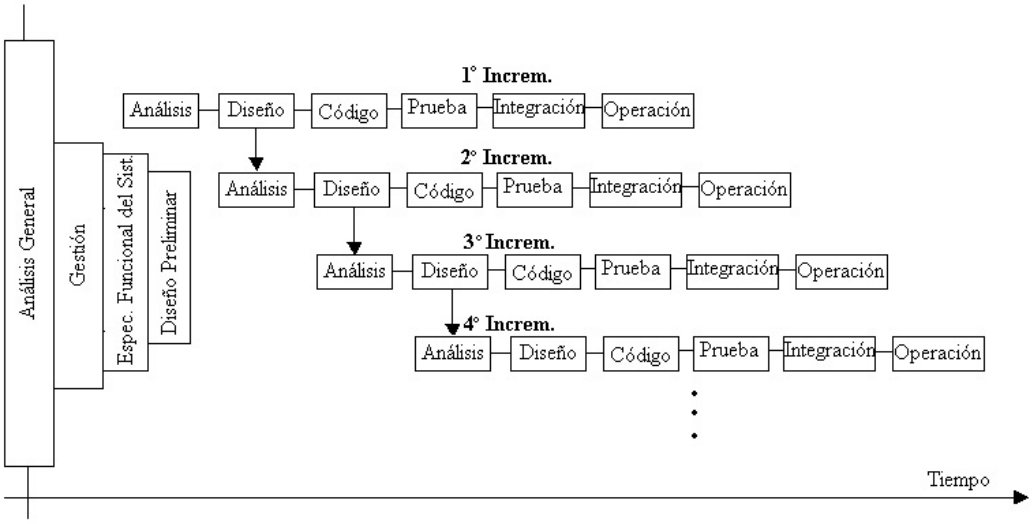


Figura 5 Modelo Incremental

Fuente: (Wikipedia, Software, 2015)

Modelo espiral

Según su creador Boehm “es un generador de modelo de proceso guiado por el riesgo que se emplea para conducir sistemas intensivos y concurrentes con muchos usuarios”, sus fases están definidas por: la Determinación de los Objetivos, Análisis de Riesgo, Desarrollar y Probar y la Planificación. Está representado por un modelo espiral como caracola y mantienen dos dimensiones la radial (que determina el aumento del coste del proyecto de software) y la angular (determina el avance del proyecto). Este modelo requiere de mucha experiencia y habilidad para evaluación de los riesgos que sin duda es uno de los requisitos para el éxito del sistema de software (Fl, 2010).

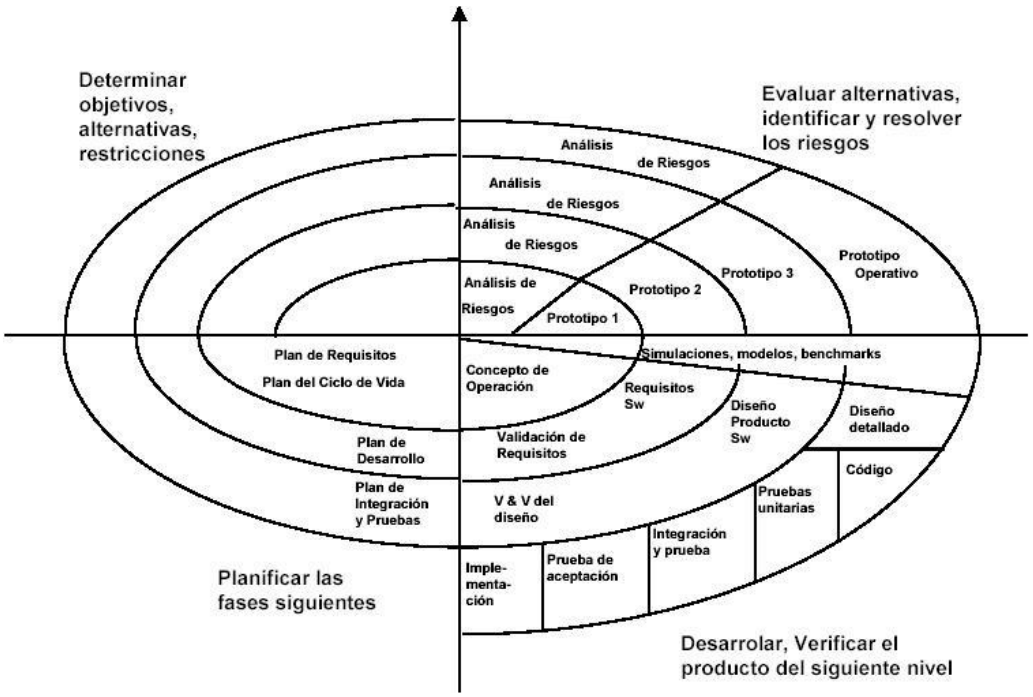


Figura 6 Modelo Espiral

Fuente: (Ardanaz, s.f.)

Modelo de desarrollo basado en componentes

Este modelo incorpora algunas características del modelo espiral, como es la interactividad y la evolución, está basado en componentes preparados de software (clases) que permiten la reutilización de software y simplificación de las pruebas. Este modelo se caracteriza por utilizar componentes de software (unidad de composición de aplicaciones de software), para la mejora de la calidad, la reducción de ciclo de desarrollo y un mejor entorno de la inversión (Gutierrez, 2011).

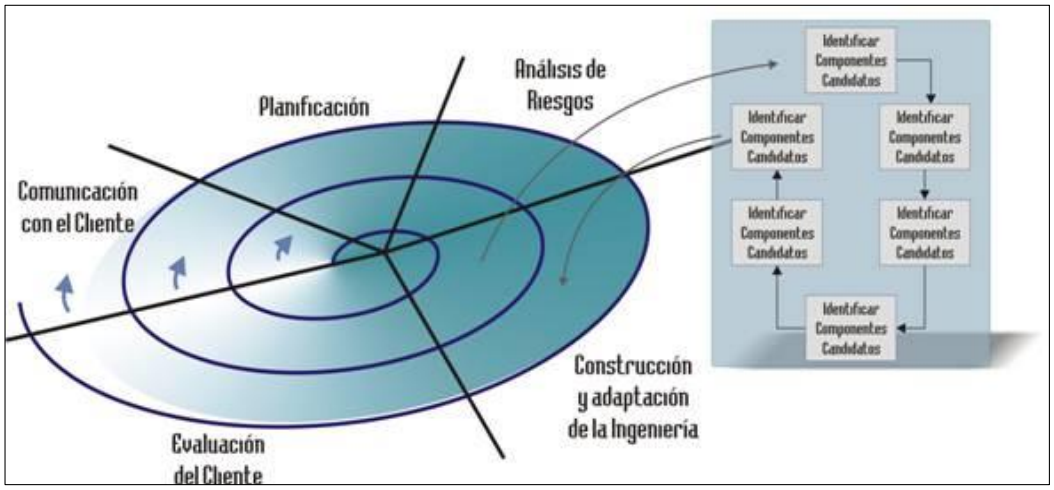


Figura 7 Modelo desarrollo basado en componentes

Fuente: (Nacional, 2006)

Modelo de desarrollo concurrente

Este modelo se utiliza cuando las actividades están ocurriendo simultáneamente, donde permite definir el conocimiento del estado real en el que se encuentra el proyecto, los requerimientos en este modelo son la línea base del proyecto, sin embargo los cambios se dan en el transcurso del diseño del sistema por lo que no es aconsejable parar el diseño en este punto en todo caso existe la necesidad de cambiar la línea base del proyecto y continuar con el proceso del sistema, pero siempre cuidando que estos cambios no afecten el desarrollo del proyecto (Gutierrez, 2011).

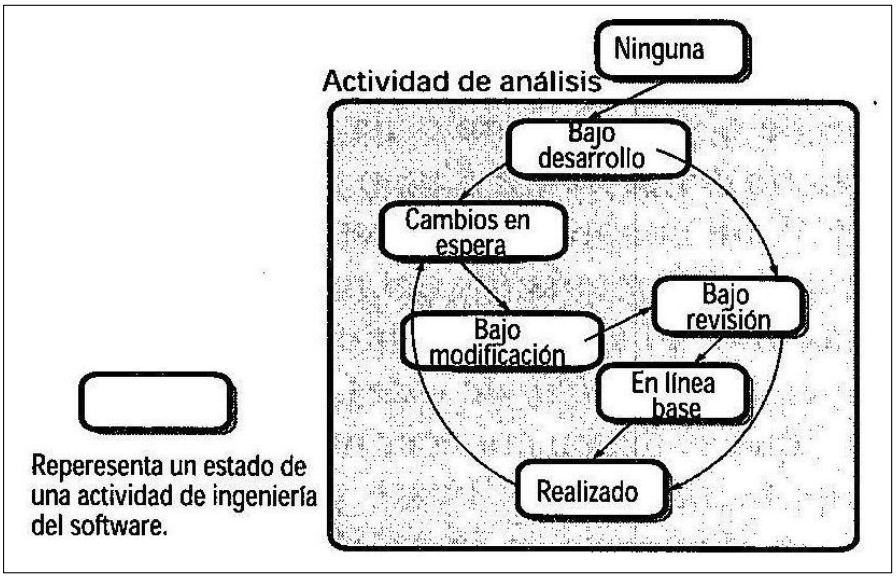


Figura 8 Modelo desarrollo concurrente

Fuente: (ejchcias, 2010)

Modelo de desarrollo rápido de aplicaciones (DRA)

Mantienen ciclos extremadamente cortos, trabaja a través de módulos con equipos de desarrollo paralelos y con gran número de personas, el tiempo juega un papel decisivo en este tipo de modelo, está enfocado en las siguientes fases:

- Modelo de Gestión: que identifica a donde va, quien lo genera, que información se genera y se conduce.
- Modelo de Datos: identifica objetos y relaciones.
- Modelo de Procesos: describe el proceso de negocio.
- Generación de Aplicaciones: reusabilidad de componentes.
- Pruebas y Entregas: prueba de componentes nuevos e interfaces (Pressman, 2001).

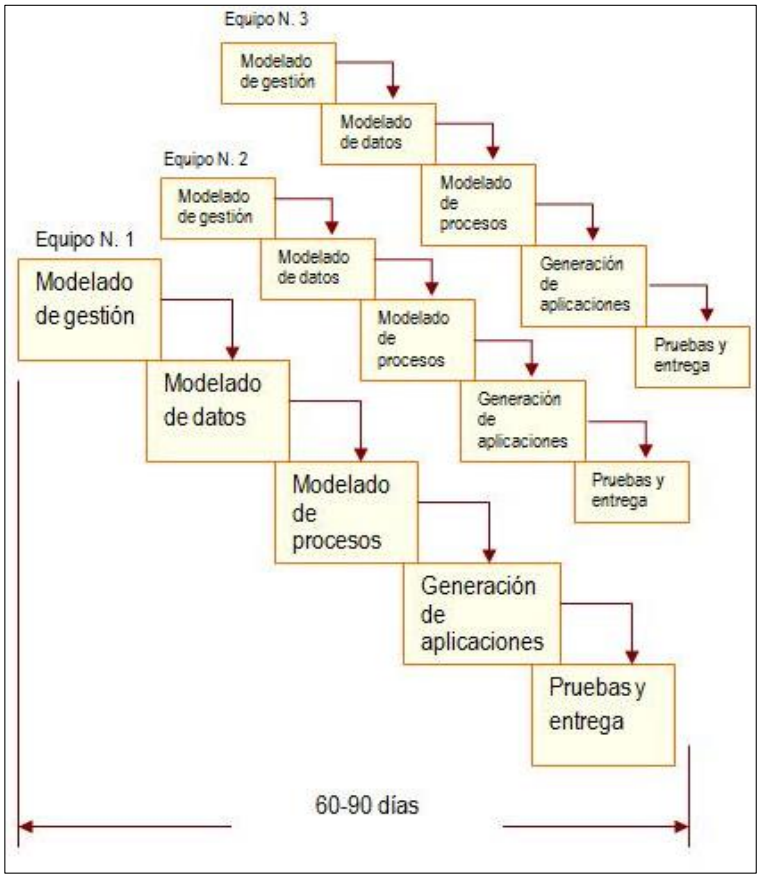


Figura 9 Modelo desarrollo DRA

Fuente: (Distancia, s.f.)

2.3.5 Notaciones gráficas y textuales en el desarrollo de software

Las notaciones son un sistema de signos convencionales que se adoptan para expresar estructuras, comportamientos, diseños arquitecturales, diseños detallados, y algunas notaciones se emplean principalmente en el contexto de métodos específicos. Dentro de la ingeniería de Software se tiene las siguientes notaciones:

Lenguaje Unificado de Modelado (LUM o UML)

(UML, por sus siglas en inglés de “Unified Modeling Language”), es el lenguaje gráfico más conocido para el modelado de sistemas software y describir/especificar métodos o procesos.

Con UML se puede visualizar, especificar, construir y documentar un sistema informático, en los que se incluyen aspectos conceptuales como procesos de negocio, funciones del sistema; y también se incluyen aspectos concretos como expresiones de programación, esquemas de bases de datos y compuestos reciclados.

RUP (Proceso Unificado Racional) es la metodología de desarrollo de software que más se soporta o aplica el lenguaje UML, pero no quiere decir que es en la única que se puede utilizar (Fowler, Seott, V, & Peake, 1999).

BPMN (notación para el modelado de procesos de negocios)

(BPMN por sus siglas en inglés de “Business Process Model and Notation”), es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de trabajo (workflow).

El objetivo principal de BPMN es proporcionar una notación estándar, legible y fácil de entender por los interesados (stakeholders) del modelo de procesos de negocios.

La notación coordina mediante un conjunto de actividades relacionadas la secuencia de los procesos de negocio y los mensajes que fluyen entre los participantes. Con esto se pretende mejorar la comunicación que frecuentemente se presenta entre el diseño de los procesos y su implementación (Garimella, Michael Lees, & Bruce Williams, 2012).

Características básicas de los elementos de BPMN

- Objetos de flujo.
- Objetos de conexión.
- Swimlanes. (carriles de piscina)
- Artefactos.

Diagrama de flujo de datos (DFD)

El diagrama de flujo de datos es una representación gráfica del flujo o procesamiento de datos a través de un sistema de información.

Mediante los DFD's se puede presentar de una manera visual al usuario, la funcionalidad, el alcance y cómo un sistema se pondrá en marcha. También se puede representar una idea física del resultado de los datos, y su incidencia en la estructura del sistema (Pressman, 2001).

Los niveles del modelo de datos son los siguientes:

- Nivel 0: Diagrama de contexto.
- Nivel 1: Diagrama de nivel superior.
- Nivel 2: Diagrama de detalle o expansión.

a) Estándares en la Ingeniería de Software

Los estándares son el conjunto de criterios aprobados, documentados y disponibles para determinar la adecuación de una acción (estándar de proceso) o de un objeto (estándar de producto).

Utilidad de los estándares

Según Sommerville, la utilidad o beneficios de los estándares se fundamentan en que agrupan lo mejor de las buenas prácticas en los diferentes ámbitos de la ingeniería del desarrollo y abarca los conocimientos de las organizaciones que representan dichos estándares. Los mismos proporcionan un marco para implementar procedimientos de aseguramiento de la calidad y continuidad entre el trabajo de los interesados del negocio.

Tipos de estándares.

Tipos de estándares en ingeniería del software son los siguientes:

- Estándares para datos.
- Estándares de codificación.
- Estándares estructurales.
- Estándares de documentación.
- Estándares de proceso software.

b) Estándares relacionados con el proceso software

Modelos de Capacidad y Madurez (SEI's CMM)

El enfoque del Instituto de Ingenieros de Software (SEI por sus siglas en inglés de "Software Engineering Institute"), es un modelo de evaluación de los procesos de una empresa, proporciona una medida de la eficacia global de las prácticas de ingeniería del software y establece para ello cinco niveles de madurez del proceso.

SEI define los niveles de madures de la empresa mediante la evaluación de un cuestionario basado en los modelos de capacidad y madurez (CMM por sus siglas en inglés de "Capability Maturity Model").

Los niveles de madures de CMM son:

- Nivel 1: Inicial.
- Nivel 2: Repetible.
- Nivel 3: Definido.
- Nivel 4: Gestionado.
- Nivel 5: Optimizado.

Estándares en el proceso del ciclo de vida

Los procesos estándar son:

- Familia ISO 9000 – Calidad del Software.
- IEEE 1074-1998 – Estándares IEEE para el desarrollo y los procesos del ciclo de vida del Software.
- ISO/IEC 12207:1995 (E) Tecnologías de la Información – Los procesos del ciclo de vida del software (posteriormente adoptado Por IEEE / EIA).
- IEEE – Institute of Electrical and Electronics Engineers (Instituto de ingenieros de Electricidad y Electrónica).
- ISO – International Organization for Standardization (Organización Internacional de Estándares).
- IEC – International Electrotechnical Commission (Comisión Internacional de Electrónica).

c) Organismos y asociaciones de la ingeniería de software

Las asociaciones y organismos de investigación más reconocidos y representativos dentro de la ingeniería de software son los siguientes:

IEEE Computer Society (IEEE-CS)

Cabe aclarar que el Instituto de Ingenieros en Electricidad y Electrónica IEEE (siglas en inglés de “The Institute of Electrical and Electronics Engineers”), desarrolla sus estándares a través de una de sus entidades, la IEEE Standards Association (IEEE-SA). En el caso de los estándares de Ingeniería del Software se potencia de entidades técnicas como la IEEE Computer Society (IEEE-CS) y el IEEE Technical Council on Software Engineering (TCSE), las cuales participan mediante el comité Software & Systems Engineering Standards Committee (S2ESC) (Kumar, 2011).

Asociación de los Sistemas Informáticos (ACM)

(ACM por sus siglas en inglés de “Association for Computing Machinery”), fue fundada en 1947 como la primera sociedad de la informática educativa y científica del mundo, proporciona recursos que aporta y permite avanzar a la informática como ciencia y como profesión.

ACM ofrece una Biblioteca Digital de computación que brinda a sus miembros y los profesionales de la informática de publicaciones de vanguardia, proporciona recursos de la carrera, patrocina conferencias en varias áreas del campo principalmente en el SIGGRAPH (Grupo de interés en infografía o computación gráfica) y SIGCOMM (Grupo de interés en la comunicación y redes informáticas), y también patrocina eventos relacionados con otras ciencias de la computación. En la actualidad la mayor sociedad de informática educativa y científica que tiene presencia en más de 100 países (The Joint Task Force on Computing Curricula (Association for Computing Machinery IEEE-Computer Society), 2013)

Instituto de Ingeniería de Software (SEI)

(SEI por sus siglas en inglés de “Software Engineering Institute”), fundado por el Congreso de los Estados Unidos en 1984 es un instituto federal de investigación y desarrollo de modelos de evaluación y mejora del desarrollo de software, con el objetivo de dar respuesta a los problemas que generaba al ejército estadounidense en la programación e integración de los sub-sistemas de software en la construcción de complejos sistemas militares. Es un referente en la Ingeniería de Software por realizar el desarrollo del modelo SW-CMM (1991) que ha sido el punto de arranque para los modelos que se han desarrollado sobre el concepto de capacidad y madurez, hasta llegar al actual CMMI (The Joint Task Force on Computing

Curricula (Association for Computing Machinery IEEE-Computer Society), 2013).

Sociedad de Computación Británica (BCS)

(BCS por sus siglas en inglés de “British Computer Society”), fue fundado en Londres en 1957, es un Instituto Colegiado de profesionales de las tecnologías de la información (TI).

Sus bases de interés son las siguientes:

- El intercambio de conocimientos de TI y el conocimiento, promueve la interrelación de expertos de la industria de software, la academia y la empresa.
- Profesionales de apoyo, proporciona a sus miembros, comunidades y voluntarios de todo el mundo apoyo y servicios de información prácticos a través de un desarrollo profesional continúo basados en una serie de requisitos de TI, lo que busca es promover la práctica profesional en sintonía con las demandas de negocio.
- El establecimiento de estándares y marcos, el Instituto colabora con los órganos de gobierno y la industria a establecer buenas prácticas de trabajo, códigos de conducta, marcos de competencias y normas comunes. También ofrece servicios de consultoría (Colussi, 2010).

Asociación de Desarrollo de Software de Rusia (RUSSOFT)

RUSSOFT fundada el 9 de septiembre de 1999, es una asociación de empresas de software de Rusia con sede central en San Petersburgo. Se conforma de más de 70 empresas con más de 25000 ingenieros de software

altamente calificados y con grados avanzados de posgrado en Tecnología y Ciencias de la Computación.

RUSSOFT colabora con las principales universidades de Rusia, en muchas veces ganadores de concursos de programación ACM. Como proveedores de software RUSSOFT a menudo clasificados en los Cuadrantes Mágicos de Gartner.

Mantienen un alto nivel de los estándares internacionales en materia de formación profesional, gestión de proyectos, control tradicional y ágil de la calidad, las empresas son líderes europeos absolutos en la certificación CMMI (Nivel 4 y 5) (Gonzalez & Gasco, 2006).

La Sociedad de Ingenieros de Software (SSE)

(SSE por sus siglas en inglés de “Society of Software Engineers”), fue fundada en 2000 como el Club de Ingeniería de Software y en el 2002 cambió su nombre por el de la Sociedad de Ingenieros de Software es una organización profesional para ayudar a los estudiantes de ingeniería informática y en una variedad de temas académicos. Este grupo se considera parte de la organización en el Colegio Golisano de Informática y Ciencias de la Información (Yu, 2010).

2.4 Caracterización tecnológica de las metodologías en el desarrollo de software

En este apartado se revisa los aspectos relevantes de las metodologías para el desarrollo de software, sus clasificaciones, tipos, las principales metodologías del mercado, como también las herramientas tecnológicas que ayudan a las metodologías para automatizar el control o gestión del ciclo de vida de las aplicaciones.

2.4.1 Definición de metodologías en el desarrollo de software

Una metodología de desarrollo de software se refiere a una manera o forma de trabajo, conocido también como marco de trabajo o framework, es un conjunto de procedimientos, técnicas, herramientas y un soporte documental usado para estructurar, planear y controlar el proceso de desarrollo en sistemas de información.

El marco de trabajo, para la metodología de desarrollo de software es una filosofía de desarrollo de programas informáticos con el enfoque del proceso de desarrollo de software. Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto, y en el caso de la metodología indica cómo hay que hacerlo para obtener los productos parciales y finales (Gutierrez, 2011).

Como conclusión se puede decir que la metodología representa el camino a seguir para planificar y controlar el proyecto del desarrollo software de manera sistemática, y tiene como objetivo establecer un proceso estándar para la fabricación de mejores aplicaciones.

2.4.2 Principales características de las metodologías

Tomando en cuenta las características más destacadas de las metodologías se citan las siguientes:

- Existencia de reglas predefinidas.
- Cobertura total del ciclo de desarrollo.
- Verificaciones intermedias.
- Planificación y control.
- Comunicación efectiva.
- Utilización sobre un abanico amplio de proyectos.

- Fácil formación.
- Herramientas CASE.
- Actividades que mejoren el proceso de desarrollo.
- Soporte al mantenimiento.
- Soporte de la reutilización de software.
- Claridad y facilidad de comprensión.
- Capacidad de soportar la evolución de los sistemas (mantenimiento).
- Facilitar la portabilidad.
- Versatilidad respecto a los tipos de aplicaciones.
- Flexibilidad / Escalabilidad (independencia respecto de la dimensión de los proyectos).
- Rigurosidad.
- Adopción de estándares

2.4.3 Impacto de las metodologías en el entorno de desarrollo

En esta sección se indica el impacto que implica la implementación de una metodología en las organizaciones y cuál es su manera de funcionamiento.

Para implementar una Metodología en un Entorno de Desarrollo se debe escoger de las siguientes opciones:

- Se puede crear una metodología o marco de trabajo a medida que se adapte de la mejor manera a las condiciones del entorno del desarrollo de software de la organización; para lo cual se debe investigar, recopilar, seleccionar y combinar de entre las opciones que brindan los métodos de gestión, técnicas de desarrollo y soporte automatizado (Letelier & Penadés, 2006).
- Otra opción es investigar, analizar y evaluar las metodologías de desarrollo de software existentes y seleccionar la que más se adapte a las necesidades (Letelier & Penadés, 2006).

Los factores que influyen y se debe tener en cuenta para escoger o crear una metodología es:

- Tamaño y estructura de la organización.
- Tipo de aplicaciones a desarrollar.

En el siguiente gráfico se puede observar como interactúa los diferentes elementos en un entorno del desarrollo de software (Ruiz, 2010).

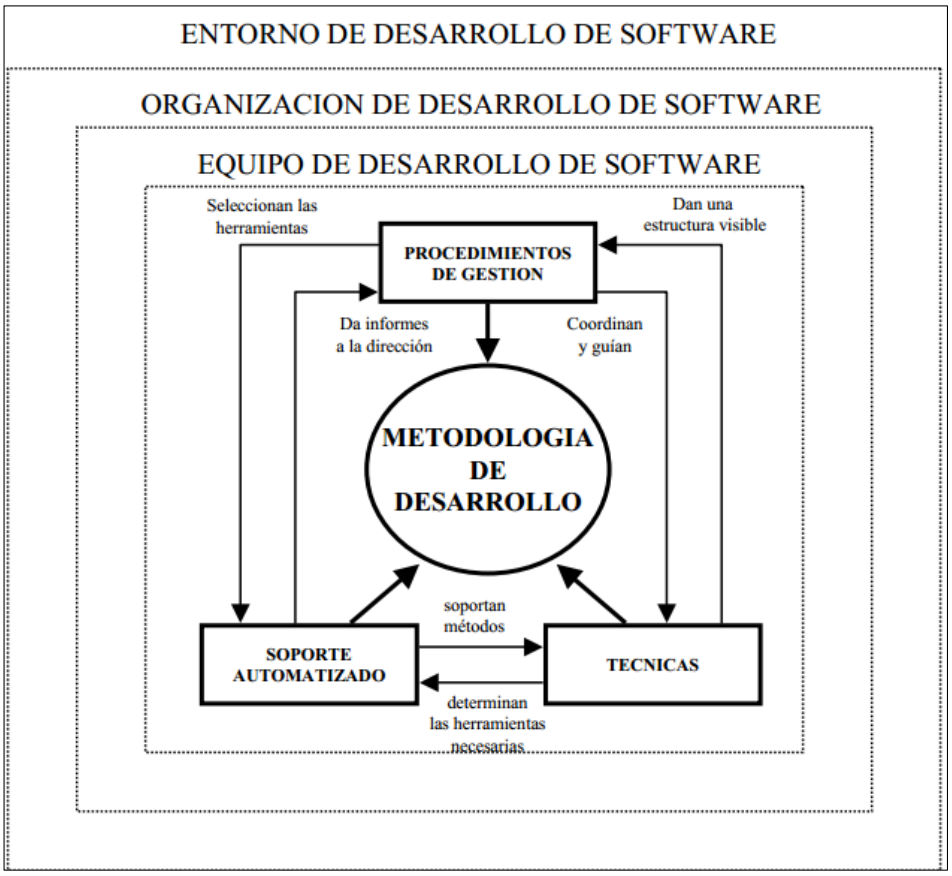


Figura 10 Entorno desarrollo de software

Fuente: (Francisco, s.f.)

2.4.4 Tipos y clasificación de las metodologías de desarrollo de software

La definición de los tipos de metodologías es un poco compleja, debido a la diversidad de propuestas y las diferencias en sus detalles y alcances. Los

tipos de metodologías, tomando como criterio las notaciones utilizadas para especificar artefactos producidos en actividades de análisis y diseño, se pueden clasificar en los siguientes grupos: Metodologías Estructuradas, Metodologías Orientadas a Objetos y para Sistemas de Tiempo Real (Virrueta Mendez, 2010).

Por otra parte, la clasificación de las metodologías de desarrollo de software considerando su filosofía, se clasifican en metodologías tradicionales y ágiles. Cabe mencionar que las metodologías tradicionales y ágiles contienen a las metodologías clasificadas por su notación, las mismas que se muestran en el siguiente gráfico.

Tabla 3

Tipos y clasificación de las metodologías de Software

ENFOQUE	TIPOS	POR SU FILOSOFÍA
Estructuradas	Orientadas a Procesos	TRADICIONALES
	Orientadas a datos ▪ Jerárquicas ▪ No Jerárquicas	
Orientadas a Objetos	Revolucionarios o Puros Sintetistas o Evolutivos	
Para Sistemas de Tiempo Real		ÁGILES

a) Metodologías estructuradas

Entre las metodologías que se utilizan en el desarrollo de software se describen las siguientes.

Metodologías Orientadas a Procesos

La metodología estructurada orientada a procesos, se centra en descomponer de la funcionalidad para obtener la especificación estructurada

del sistema, para lograr esto se apoya en técnicas gráficas como son los modelos gráficos, particionados, descendentes y jerárquicos en función de los procesos del sistema y de los datos utilizados (Polo, 2010).

Los componentes que utiliza esta metodología son: los Diagramas de Flujo de Datos, los Diccionario de Datos y las Especificaciones de Procesos.

Metodologías Orientadas a Datos Jerárquicas

En las metodologías estructurales orientadas a datos jerárquicos, la estructura de control del programa debe ser jerárquica y se deriva de la estructura de datos del programa. El diseño consiste primero en definir las estructuras de los datos de entrada y salida, luego se debe combinarlas en una estructura jerárquica del programa y por último ordenar detalladamente la lógica procedimental para que se ajuste al diseño lógico, que debe preceder y estar separado del diseño físico, los métodos existentes son:

- JSP (Jackson Structured Programming) y JSD (Jackson Structured Design) de Jackson (1975).
- LCP (Logical Construction Program) de Warnier (1974).
- LCS (Logical Construction Systems) de Warnier y Orr (1981).

Metodologías Orientadas a Datos No Jerárquicas

En esta metodología se considera a los datos la parte esencial del sistema porque son más estables que los procesos, este modelo representa los datos de la organización formado por un conjunto de entidades de datos básicos y sus relaciones. Los procesos derivan de una definición inicial de los datos. El método existente es la Metodología Ingeniería de la Información (Information Engineering - IE) de J. Martin y C. Finkelstein (Martin, 1986), la cual se basa en las siguientes fases:

- Planificación: Construcción de la arquitectura de la información y la estrategia que soporte los objetivos de la organización.
- Análisis: Comprende las áreas del negocio y se determinan los requerimientos del sistema.
- Diseño: Se establece el comportamiento del sistema.
- Construcción: En esta fase se construye el sistema conforme a las fases anteriores.

b) Metodologías orientadas a objetos

En las metodologías orientadas a objetos cambian los principios de las metodologías estructurales ya que las estructurales examinan los sistemas desde las funciones y tareas, a diferencia de las orientadas a objetos que modelan el sistema examinando el dominio del problema como un conjunto de objetos que interactuante entre sí, en donde los objetos encapsulan o contienen las funciones y los datos (Ibarz, Carlos, & Juan, 2015).

Existen dos enfoques que defienden las metodologías orientadas a objetos y estas son:

Revolucionarios o Puros

En donde se entiende que las metodologías Orientadas a Objetos son un cambio profundo de las metodologías estructuradas a las cuales las consideran como obsoletas y tienen como principales representante a la metodología de Bosh (Grady Bosh 1993), y la metodología CRC/RDD (Responsibility-driven design) Diseño de responsabilidad impulsada (Mckean, 2002).

Sintetistas o Evolutivos

Estas metodologías consideran al análisis y diseño estructurado como la base del desarrollo orientado a objetos, sus principales representantes son

la metodología de modelado de objetos OMT (Object Modeling Technique) descrito por (Rumbaugh 1991), y la popular metodología RUP (Rational Unified Process) Proceso Unificado de Rational por (Rational Software IBM 1999)

c) Metodologías para sistemas de tiempo real

Las metodologías para sistemas de tiempo real, deben cumplir con varias demandas como cumplir las necesidades de los clientes que son los requerimientos, respetar los plazos y costos y alcanzar determinados niveles de calidad. Para lo cual tiene las siguientes características: manejo de interrupciones, comunicación y sincronización entre tareas, gestión de procesos concurrentes, respuesta oportuna ante eventos externos, datos continuos o discretos.

d) Metodologías Tradicionales

Las metodologías Tradicionales o pesadas son aquellas que enfocan el mayor énfasis y esfuerzo en la planificación y control del proyecto, en especificación precisa de requisitos y modelado.

A continuación se citan 3 de las metodologías más populares consideradas como tradicionales:

Microsoft Solution Framework (MSF)

MSF es un marco de trabajo que se compone de un conjunto de las mejores prácticas, principios, modelos, disciplinas, conceptos y directrices para la administración de proyectos de desarrollo de software, pero también es aplicable a cualquier proyecto de TI como proyectos de implementación, redes o infraestructura (Alexander, Allen, & Bindoff, 2013)

Las fases de los proyectos son:

- Visión y Alcances.
- Planificación.
- Desarrollo.
- Estabilización.
- Implantación.

Figura 11 Metodología MSF

Fuente: (Microsoft, 2006)

Métrica 3

MÉTRICA es una metodología del Gobierno de España para la planificación, desarrollo y mantenimiento de sistemas de información, para la sistematización de actividades del ciclo de vida de los proyectos software en el ámbito de las administraciones públicas.

Esta metodología se basa en el modelo de procesos del ciclo de vida de desarrollo ISO/IEC 12207 (Information Technology - Software Life Cycle Processes), como también en la norma ISO/IEC 15504 SPICE (Software Process Improvement And Assurance Standards Capability Determination).

Sus Procesos principales son:

- Planificación de Sistemas de Información (PSI).

- Desarrollo de Sistemas de Información (DSI).
  - Estudio de Viabilidad del Sistema (EVS).
  - Análisis del Sistema de Información (ASI).
  - Diseño del Sistema de Información (DSI).
  - Construcción del Sistema de Información (CSI).
  - Implantación y Aceptación del Sistema (IAS).
  - Mantenimiento de Sistemas de Información (MSI) (Europa, 1988)

El siguiente gráfico muestra la manera de gestionar de la metodología:

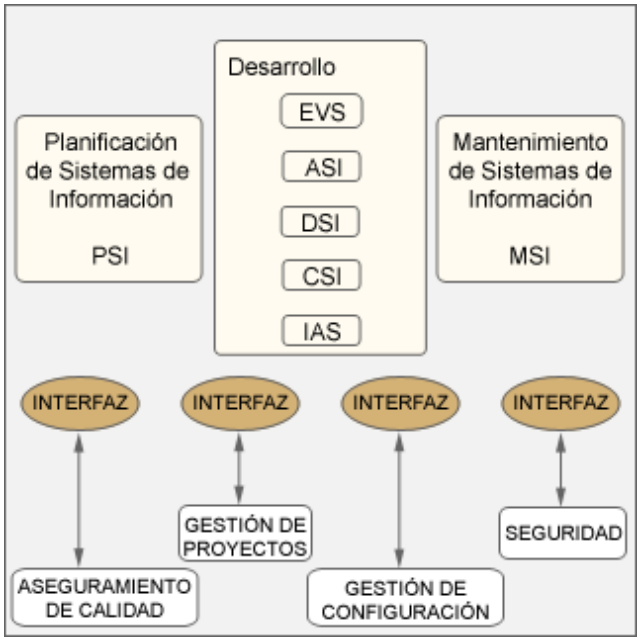


Figura 12 Metodología Métrica 3

Fuente: (EMMANUEL, 2009)

Proceso Unificado de Desarrollo (RUP)

RUP es un proceso formal, que provee un acercamiento disciplinado para asignar tareas y responsabilidades a los interesados del negocio de la organización de desarrollo. Su objetivo principal es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales los cuales deben respetar los tiempos establecidos y el presupuesto.

Es guiado por casos de uso y centrado en la arquitectura, y utiliza UML como lenguaje de notación y divide el proceso en cuatro fases, en las cuales se realizan en un número variable de iteraciones según el proyecto y se basa en las distintas actividades que se reflejan en el siguiente gráfico (Teams, 2004)

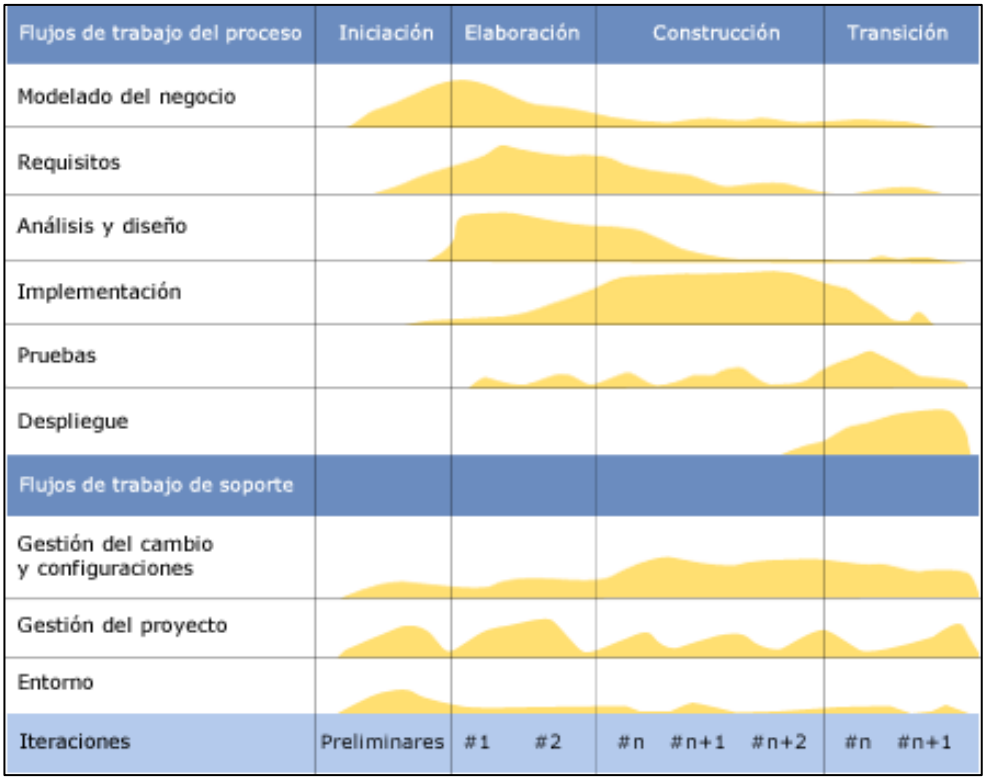


Figura 13 Metodología RUP

Fuente: (Wikipedia, Proceso Unificado de Rational, 2014)

Principales características de la Metodología RUP:

- Forma disciplinada de asignar tareas y responsabilidades.
- Implementar las mejores prácticas de la Ingeniería de Software.
- Desarrollo iterativo.
- Administración de requisitos
- Uso de arquitectura basada en componentes.
- Control de cambios.
- Modelado visual del software.
- Verificación de la calidad del software.

e) Metodologías Ágiles

Las Metodologías Ágiles, están más orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen gran referencia al trabajo en equipo e involucran activamente al cliente en el proceso.

Dentro de las metodologías ágiles se citas a 3 de las más representativas:

Scrum

La metodología ágil Scrum se caracteriza por la estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto, la calidad se basa más en el conocimiento tácito de las personas que trabajan en equipos auto organizados, que en la calidad de los procesos empleados. Solapamiento de las fases del desarrollo, en lugar de realizarlas en modo secuencial o cascada (Trigas, 2013).

Las características de la metodología Scrum son las siguientes:

- Enfatiza los valores y prácticas de gestión, antes que los temas técnicos como requerimientos, prácticas de desarrollo, implementación.
- Hace uso de prácticas para armar equipos de trabajo auto dirigidos y auto organizados.
- Puede ser aplicado a cualquier contexto en donde un grupo de gente necesita trabajar en conjunto para lograr una meta común, esto quiere decir que no solamente se aplica en proyecto de desarrollo de software.
- Desarrollo de software iterativo e incremental.
- Recomienda iteraciones de trabajo de 2 a 4 semanas, varía dependiendo el contexto del proyecto, llamadas también como Sprint.

- La persona que se encarga de la gestión del Sprint se denomina Scrum Master.
- Se practica reuniones diarias de no más de 15 minutos denominadas “Scrum Daily Meeting”, en las cuales se presenta al equipo de trabajo el avance diario, para obtener realimentación sobre las tareas y los obstáculos que se presentan en el proceso.

En el siguiente gráfico se presente las fases de la metodología:

Figura 14 Metodología Scrum

Fuente: (Solution, s.f.)

Extreme Programming XP

La programación extrema XP es una de las metodologías ágiles más destacada para el desarrollo de software, se diferencia de las metodologías tradicionales porque su énfasis es la adaptabilidad antes que la previsibilidad. En la metodología se considera los cambios de requisitos en el proceso del desarrollo es un aspecto natural e inevitable en cualquier punto de la vida del proyecto.

Las características fundamentales de la metodología son:

- Desarrollo iterativo e incremental.
- Pruebas unitarias continuas.

- Programación en parejas.
- Frecuente integración del equipo de programación con el cliente o usuario.
- Corrección de todos los errores antes de añadir nueva funcionalidad.
- Refactorización del código.
- Propiedad del código compartida.

En XP se cree que la simplicidad en el código se complementa con la comunicación del equipo de trabajo para identificar que se debe o no hacer en el proceso del desarrollo (Wells, 2003).

En el siguiente gráfico se muestra las etapas de la metodología XP.

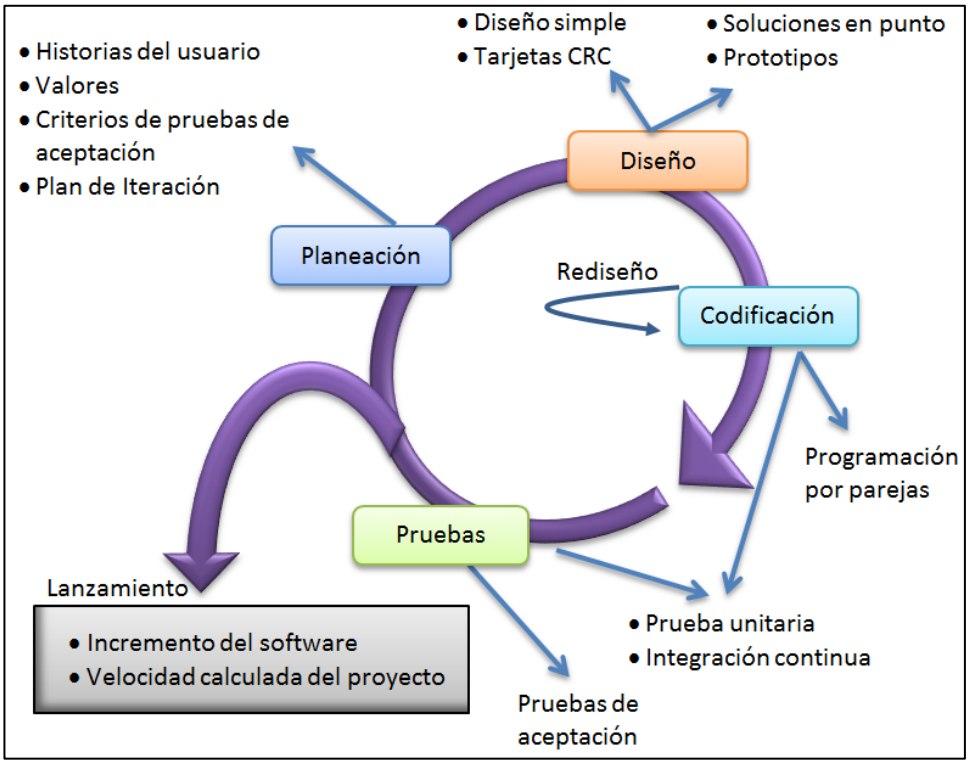


Figura 15 Metodología XP

Fuente: (González, 2012)

Kanban

Kanban es un método para gestionar el trabajo intelectual, se maneja el proceso desde la definición de una tarea hasta la entrega al cliente. Kanban se puede dividir en dos partes, la primera es un sistema de gestión de proceso visual que le indica qué, cuánto y cuándo producir, y la otra parte es el manejo como método, que es una aproximación a la mejora del proceso evolutivo e incremental para las organizaciones.

Kanban utiliza para el desarrollo de software un sistema virtual para limitar el trabajo en curso. Su origen en japonés se traduce como "tarjeta de señal", pero no funcionan como señales para realizar más trabajo, sino que las tarjetas representan los elementos de trabajo, por esta razón se las denomina "sistema virtual" porque no existe una tarjeta física. Kanban utiliza un sistema de extracción limitada del trabajo en curso, para exponer los problemas del funcionamiento del sistema o proceso, y así estimular a los miembros del equipo la colaboración para una mejora continua del sistema (Skarin & Kniberg, 2010).

Los principios del método de Kanban son:

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

Las prácticas que utiliza Kanban son las siguientes:

- Visualizar.
- Limitar el trabajo en curso.
- Dirigir y gestionar el flujo.
- Hacer las Políticas Explícitas del Proceso.

- Utilizar modelos para reconocer oportunidades de mejora.

En el siguiente gráfico se muestra un tablero de tareas del proceso de Kanban.

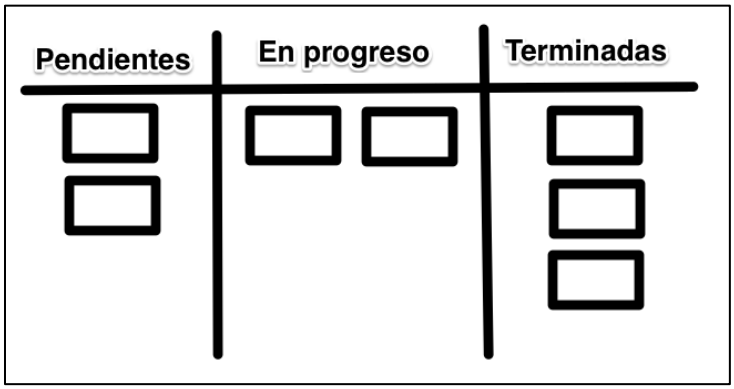


Figura 16 Método Kanban

Fuente: (Infante, 2014)

2.4.5 Herramientas de administración del ciclo de vida de las aplicaciones (ALM)

(ALM por sus siglas en inglés de “Application Lifecycle Management”), o Administración del Ciclo de Vida de las Aplicaciones. Estas herramientas se basan en la automatización y sobre todo en la integración de las herramientas que gestionan cada una de las fases (Análisis, Diseño, Desarrollo, Pruebas y Despliegue) del ciclo de vida de las aplicaciones (Chappell, 2008).

El objetivo principal es crear un repositorio central y brindar una metodología pragmática con un conjunto de funcionalidades que permitan automatizar y simplificar el proceso del desarrollo de software, y de esta manera facilitar la colaboración entre los diferentes roles del equipo de trabajo durante todas las etapas del desarrollo, adicional ALM tiene los siguientes objetivos:

- Mayor comunicación entre recursos.

- Aumento de la productividad.
- Mejora de la calidad del producto.
- Integración de herramientas.
- Mayor velocidad de desarrollo.
- Disminución de costos.
- Transparencia en el negocio.

a) Funcionalidades de las herramientas ALM

Dentro de las funcionalidades que brindan las herramientas ALM se citan a las siguientes:

Administración de requerimientos: sirve para registrar los requerimientos de funcionalidades nuevas, mejoras o depuraciones.

Administración de tareas: sirve para administrar las tareas con sus diferentes características como por ejemplo asignar a los integrantes del equipo, asignar prioridades, tipo de tarea, tiempo de duración, y demás particularidades.

Control de Código Fuente: Es el manejo de los cambios que se realizan sobre los elementos de algún producto o configuración. Una versión, edición o revisión es el estado del producto en un momento determinado de su desarrollo o modificación.

Control de defectos/bugs: sirve para administrar el registro de los defectos encontrados, removidos y pendientes de las aplicaciones.

Administración de casos de pruebas: sirve para organizar, documentar y administrar los casos de pruebas que aplicarán a las aplicaciones en desarrollo.

Ejecución de Pruebas y reporte de Defectos: sirve para administrar y controlar la ejecución de los casos de pruebas y reportar los resultados de los mismos.

Herramientas de Integración de la aplicación: sirve para administrar la integración continua de las aplicaciones.

Administración de Versiones: sirve para administrar y controlar de manera ordenada la liberación de versiones del producto final.

Pruebas automatizadas: sirve para robotizar pruebas y generar escenarios en las que se realizan pruebas de regresión, carga y estrés de las aplicaciones.

Monitoreo y reportes: sirve para construir informes y cuadros de control del proceso de desarrollo de las aplicaciones, para obtener indicadores y herramientas prácticas para llevar el control y la auditoria de los proyectos.

b) Herramientas ALM y las metodologías

Las herramientas ALM tienen la capacidad de gestionar los diferentes tipos de metodologías ya sean tradicionales o ágiles, es decir podemos escoger modelos robustos como RUP (Rational Unified Process), o también las aplicar las populares metodologías ágiles Scrum o XP.

Varias herramientas ALM incluyen plantillas de procesos, en las cuales vienen cargadas con las estructuras, términos, artefactos que contienen los modelos de las metodologías, también en algunas herramientas suelen tener la opción para poder modificar las plantillas existentes, con esto abre la posibilidad de construir plantillas propias que se adapten mejor a las necesidades de la organización.

Tomando en cuenta lo antes expuesto, se puede decir que las herramientas ALM debido a sus características de adaptabilidad y administración de plantillas de metodologías, ayudan al equipo de trabajo a llevar un control particular y dinámico en cada etapa, así como en la transición de una etapa a otra en el desarrollo de software (Silclir, 2013)

c) Herramientas ALM disponibles

Existe una gran variedad de herramientas ALM en el mercado, tanto open source como comerciales, las mismas que tienen particularidades y maneras de funcionamiento. Para saber cuáles son las principales herramientas del mercado se tomará los resultados que brinda el informe del cuadrante mágico de Gartner. La empresa Gartner se encarga de hacer estudios de clasificación de herramientas basándose en dos variables, la visión del negocio y la capacidad de ejecución. Además se enfoca en los siguientes parámetros significativos: las estrategias del mercado, estrategia de ventas, la capacidad de ejecución, capacidad de venta, conocimiento de clientes, productos principales. En el informe del año 2013 muestra como líderes de las herramientas ALM a los fabricantes Microsoft con su producto TFS (Team Foundation Server), seguido por IBM con sus productos CLM (Collaborative Lifecycle Management) y Atlassian con sus productos de JIRA (Figuerola, 2014).

Tomando en cuenta las características gnoseológicas expuestas en este apartado se puede establecer la base conceptual y alcance del conocimiento que fundamente el trabajo de investigación para desarrollar el marco de trabajo implementado con la herramienta ALM, en la empresa Farmaenlace Cía. Ltda.

2.5 Antecedentes contextuales

La empresa Farmaenlace Cía. Ltda., es una empresa ecuatoriana dedicada a la comercialización y distribución de productos farmacéuticos y productos de primera necesidad, su inscripción se la hizo en 2005 resultado de la alianza de dos importantes empresas de distribución farmacéutica, la representación de Ortiz Cevallos y farmacéuticas Espinosa, y son dueños de las marcas de farmacias Económicas, farmacias Medicitys, farmacias El Descuento, Distribuidoras farmacéuticas Difarmes y farmacias de Socios estratégicos.

La empresa se conforma por 2371 empleados repartidas en 396 farmacias y oficinas administrativas, que se distribuyen de la siguiente manera: 292 farmacias Económicas que trabajan 1187 personas, 79 farmacias Medicitys en las que trabajan 419 empleados y 25 farmacias de socios estratégicos que constan por 49 empleados, las farmacias están ubicadas a nivel nacional, también existe 3 distribuidoras farmacéuticas Difarmes ubicadas en Quito y Ambato en las que trabajan 15 empleados. La parte administrativa se conforma por las oficinas en la matriz ubicada en Quito en la Av. Rafael Ramos e2-210 y Castelli en las que trabajan 659 personas, y también oficina sucursal en Ibarra ubicadas en las calles Abelardo Morán Muñoz 2-20 y Juan José Páez, en las que trabajan 42 empleados administrativos.

El presente proyecto se desarrollará e implementará en el área de Análisis y Desarrollo de Software del departamento de Sistemas ubicado en las oficinas administrativas de la ciudad de Ibarra, en donde se encuentra el equipo de desarrollo que se conforma de 15 personas que se distribuyen en los siguientes cargos: 2 coordinadores, 5 líderes de proyectos y 8 desarrolladores, los cuales conforman el equipo de desarrollo de software, los que utilizarán y ejecutarán el actual proyecto, y los resultados se entregará a la Gerencia de Sistemas que está ubicado en las oficinas de la matriz de la empresa en la ciudad de Quito.

Para la justificación científica del problema se elaboró un instrumento de investigación (Encuesta) (Anexo A), el mismo que se enfocó en dos aspectos: el primero dirigido a lo relacionado con el problema del proyecto, y el segundo relacionado con la propuesta del proyecto de investigación, en donde fue aplicado a todo el equipo de desarrollo y el gerente de sistemas.

2.5.1 Instrumento de investigación – Encuesta

A continuación se presenta el resumen de la encuesta realizada a los 15 miembros del equipo de desarrollo de la empresa, adicional se pone el resultado porcentual de cada una de las preguntas realizadas.

ENCUESTA

1) ¿Se ha presentado problemas en los proyectos de desarrollo de software en la empresa Farmaenlace Cía. Ltda. anteriormente?

SI: 8 (53%)	NO:	Ocasionalmente: 7

RESULTADO: Se evidencia que existe la presencia de problemas en los proyectos de software en un 53% y ocasionalmente un 47%, teniendo como principales problemas: Los requerimientos incompletos, ambiguos o poco funcionales con 73%, seguido por escasa información acerca del proceso 67%, La desorganización de la información 40%, y que el dueño del proceso con poco conocimiento, y dificultad de la comunicación con los usuarios en 33%.

2) ¿Se ha presentado retrasos en la entrega de proyectos de desarrollo de software?

SI: 6 (40%)	NO:	Ocasionalmente: 9

RESULTADO: Según las respuestas se puede evidenciar que existe retrasos en las entregas de proyectos de software ocasionalmente en un

60%, y un si con 40%. Las principales causas son las siguientes; Tareas imprevistas y urgentes que con se consideran en planificaciones con 93%, La mala estimación de tareas con 80%, Falta de metodología para desarrollo de software 47%, y la dificultad de comunicarse con los usuarios en un 20%.

3) ¿Existe la falta de una metodología (marco de trabajo) para el desarrollo de software?

SI: 14 (93%)	NO: 1 (7%)	Ocasionalmente:

RESULTADO: Se evidencia que existe la falta de una metodología de desarrollo de software en un 93% y con un negativa del 7%.

4) ¿Cree usted que el usuario dueño del proceso debe involucrarse más, durante todo el proceso de desarrollo de software?

SI: 12 (80%)	NO:	Ocasionalmente:3 (20%)

RESULTADO: Con las respuestas se evidencia que el usuario dueño del proceso debería involucrarse más durante el desarrollo de software en un 80% y ocasionalmente en un 20%.

5) ¿Ha existido dificultad en la ejecución de las planificaciones de trabajos de los proyectos de desarrollo de software?

SI: 5 (33%)	NO:	Ocasionalmente: 10

RESULTADO: Según las respuestas se evidencia que ha existido dificultad en la ejecución de las planificaciones de trabajos de proyectos de software ocasionalmente en 67% y en si con 33% siendo las principales problemas los siguientes: No se planifica adecuadamente el tiempo de soporte, mantenimiento y seguimiento de los sistemas nuevos con 80%, No se planifica adecuadamente las tareas no programadas con 60%, no se

analizan a profundidad las actividades a realizarse con 53%, y el desconocimiento de la información con respecto a la solución con el 27%.

6) ¿Cree usted que existe la necesidad de establecer indicadores de productividad del equipo de desarrollo de software dentro de la empresa?

SI: 11 (73%)	NO: 1 (7%)	Ocasionalmente: 3

RESULTADO: Conforme a las respuestas se evidencia la aceptación de utilizar indicadores de productividad del equipo de desarrollo de software con un 73% y un 20% ocasionalmente y una negativa del 7%.

7) ¿Cree que sería necesario que se desarrolle y establezca una metodología (marco de trabajo) para el proceso de desarrollo de software acorde con la realidad de la empresa?

SI: 15 (100%)	NO:	Ocasionalmente:

RESULTADO: Según las respuestas hay una aceptación unánime para que se desarrolle y establezca una metodología para el proceso de desarrollo de software de la empresa.

8) ¿Cree que sería necesario aplicar una herramienta tecnológica para facilitar la gestión en el ciclo de vida en el proceso del desarrollo de software?

SI: 15 (100%)	NO:	Ocasionalmente:

RESULTADO: Según las respuesta si existe una necesidad del 100% de las respuestas con la aceptación de aplicar una herramienta tecnológica para facilitar la gestión en el ciclo de vida de las aplicaciones en el desarrollo de software de la empresa.

9) ¿Cree que sería necesario aplicar una herramienta tecnológica para

implementar reportes gerenciales y operativos de los proyectos de desarrollo de software?

SI: 11 (73%)	NO:	Ocasionalmente: 4

RESULTADO: La aceptación para implementar reportes gerenciales y operativos de los proyectos de software es de un 73%, y un 27% que si requieren ocasionalmente.

10) ¿Estaría dispuesto a utilizar una herramienta tecnológica para gestionar el ciclo de vida de las aplicaciones, basándose en una metodología desarrollada acorde a las necesidades de la empresa?

SI: 15 (100%)	NO:	Ocasionalmente:

RESULTADO: La aceptación del 100% de si estarían dispuestos a utilizar una herramienta tecnológica para gestionar el ciclo de vida de las aplicaciones, basándose en una metodología desarrollada acorde a las necesidades de la empresa. Quiere decir según la encuesta que el presente proyecto tiene la aceptación por parte del equipo de desarrollo de software de la empresa.

Con lo expuesto en este capítulo se pudo establecer la base histórica-conceptual de las metodologías y los antecedentes contextuales de la necesidad del desarrollo de un marco de trabajo implementado con una herramienta ALM en la empresa Farmaenlace Cía. Ltda.

CAPÍTULO III

DESARROLLO DE UN MARCO DE TRABAJO, IMPLEMENTADO CON UNA HERRAMIENTA ALM, PARA MEJORAR LA GESTIÓN DEL PROCESO DE FABRICACIÓN DE SOFTWARE.

3. Introducción

El desarrollo del marco de trabajo o metodología para la fabricación de software, es parte fundamental para la mejora de la gestión del proceso de desarrollo de software. Al aplicar metodologías que existen en el mercado o proponer nuevas metodologías apegadas a la realidad de las empresas da lugar a optimizar el proceso de desarrollo, como por ejemplo mejorar la comunicación y coordinación de los equipos de trabajo, cumplir oportunamente con los objetivos, con los tiempos de entrega y la calidad de software ofrecidos al usuario final. Lo antes expuesto hace referencia y se desprende del estudio realizado por Ramos (Ramos, 2010). El presente estudio, tomará como base este marco de trabajo, para ser aplicado en un caso de estudio en la empresa.

En este apartado se desarrolla un marco de trabajo implementado con una herramienta ALM, para mejorar la gestión del proceso de fabricación de software, se comenzará haciendo un estudio comparativo entre los tipos de metodologías tradicionales y ágiles, seguidamente se realizará un segundo estudio en referencia al tipo de metodología que se escogió, con la finalidad de adoptar las mejores prácticas y que servirá de base en el desarrollo del nuevo marco de trabajo propuesto.

Luego de haber establecido el marco de trabajo de desarrollo de software de la empresa, se podrá automatizar mediante una herramienta

ALM en la que se administra las fases del ciclo de vida en el desarrollo de las aplicaciones. Para lo cual se hará un nuevo estudio comparativo de las principales herramientas ALM, para escoger la que más convenga en dicha automatización. Culminado las fases de estudio se dará continuidad al desarrollo del proyecto propuesto.

A continuación se resume la estructura del desarrollo de este marco de trabajo.

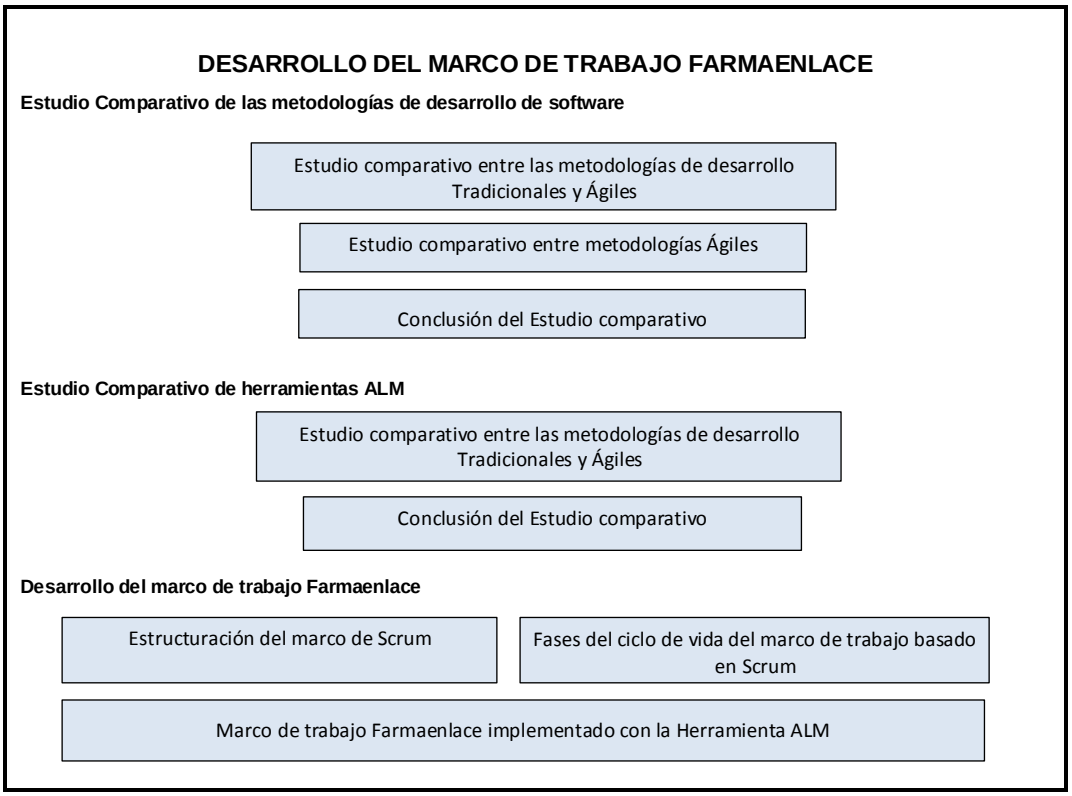


Figura 17 Estructura del desarrollo del marco de trabajo Farmaenlace

3.1 Estudio comparativo de metodologías de desarrollo de software

Para el estudio comparativo de las metodologías, primero se hará una comparativa entre las metodologías tradicionales y ágiles, de la cual se evaluará y escogerá el tipo de metodología más conveniente. La segunda parte del estudio será la comparativa de los modelos del tipo de metodología escogida en el punto anterior, y así poder obtener los criterios necesarios para fundamentar la elaboración de un marco de trabajo institucional

conveniente y adaptado a las necesidades del caso de la empresa Farmaenlace Cía. Ltda.

3.1.1 Estudio comparativo entre metodologías tradicionales y ágiles

El propósito de este estudio es hacer una comparación entre las metodologías de desarrollo de software tradicionales y ágiles. Se considera como variable de decisión el costo beneficio empresarial, dentro de este criterio se toma en cuenta la situación actual del software en producción, los proyectos en desarrollo y un listado de proyectos de software por desarrollar con un tiempo aproximado de 2 años, utilizando el mismo número de integrantes del equipo de desarrollo de la empresa Farmaenlace Cía. Ltda.

La comparativa se hará de la siguiente manera: se tomará en cuenta varios aspectos principales del desarrollo de los proyectos de software, en las cuales se expondrá las principales características, prácticas o técnicas de cada tipo de metodología, a las mismas que se les asignará un peso desde el punto vista del costo beneficio empresarial, dicho peso será establecido por el gerente de Sistemas de la empresa. La escala de peso es de 0 a 4 siendo: 0 no necesario, 1 poco necesario, 2 necesario intermedio, 3 necesario, 4 muy necesario.

Tabla 4
Cuadro comparativo de las metodologías tradicionales vs ágiles

Aspectos de desarrollo	Tradicional	Ágiles	Peso tradicional	Peso ágil
Requisitos	Los requisitos no pueden cambiar.	Los requisitos son muy cambiantes. Se requiere de un feedback sobre un resultado obtenido para determinar si es lo requerido o no.	1	4

CONTINÚA

Requisitos (funcionalidades innecesarios)	Debido a la recolección inicial de requisitos es frecuente que se soliciten funcionalidades innecesarias.	No permite que se incluyan funcionalidades innecesarias.	1	4
Cambios	Hacer un cambio al alcance requiere de un proceso formal de control de cambios.	El cambio es bienvenido en cualquier momento del proyecto.	1	4
Tiempo	Existe un compromiso respecto al tiempo de entrega del proyecto. (No siempre se cumple esta meta).	Existe incertidumbre respecto al tiempo de entrega de todo el producto. Máximo cada 2 meses y en el caso de Scrum1 mes, se hace la entrega de producto de valor para el cliente.	3	3
Costo	Es definido para el proyecto.	Existe incertidumbre respecto al costo del proyecto. Se invierte en las funcionalidades que más valor le dan al cliente y cíclicamente se avanza hasta que se logre, ya sea: • El producto deseado. • Se acabe el presupuesto.	3	3
Documentación	Atención exhaustiva a la documentación.	Solo se genera la documentación que genera valor al cliente y al proyecto.	3	4
El cliente	El cliente apoya el desarrollo del producto mediante la participación en reuniones.	Involucración directa del cliente en el desarrollo del producto. El cliente es parte de equipo.	3	4
Iteraciones	Pocas iteraciones que generan gran volumen de información y software para construcción del producto.	Utilización de múltiples iteraciones de desarrollo para aprender y evolucionar el producto.	2	4

CONTINÚA

Riesgos	Los riesgos son asumidos por el proveedor.	Es voluntad del cliente para compartir la responsabilidad en las decisiones y riesgos.	2	4
Se valora más	El proceso.	El individuo y las interacciones de los mismos.	4	3
La planeación	Requieren un plan detallado desde el inicio del proyecto.	Se va planeando a medida que se avanza en el proyecto. Planeación gradual y constante.	3	3
El éxito del proyecto	Es dado por el seguimiento del plan.	Es dado por la entrega continua de valor y funcionalidad al cliente.	4	4
Elaboración de entregables	Se generan entregables que requieren mucho tiempo de elaboración.	Se centran en hacer entregables en tiempos cortos con alta calidad inmersa.	1	4
La retroalimentación del cliente	Es conocida al final, pudiendo generar insatisfacción.	Es constante a lo largo del proyecto.	1	4
Participación del equipo	Empodera al Gerente del proyecto para el éxito del mismo, es el que toma las decisiones.	Empodera al equipo para trabajar de forma creativa e innovadora.	2	4
Proceso de desarrollo de software (Plantillas)	Innumerables plantillas y artefactos para cumplir con el proceso.	Pocas plantillas y artefactos.	2	4
Roles	Algunos roles para ejecutar el proyecto	Pocos roles.	2	4
Arquitectura	Es un ejercicio que se realiza al inicio o en una etapa del proyecto.	Es un ejercicio constante durante el proyecto	2	4
Puntos			**40**	**68**
Porcentaje			**55%**	**94%**

Considerando los 18 aspectos principales del proceso de desarrollo de software recopiladas por (Abad Hernán, 2014), se ha valorado a cada

pregunta de acuerdo a los pesos establecidos anteriormente por el gerente de sistemas. Por lo que se obtiene 68 puntos para los pesos de las ágiles y 40 para las tradicionales, a estos puntajes se establece un cálculo de porcentajes para establecer los resultados que corresponden a la comparativa de dichas metodologías.

El resultado del estudio comparativo muestra que las características, técnicas y prácticas de las metodologías ágiles cubren el 94% de las funcionales esperadas, a comparación del 55% que brindan las metodologías tradicionales, por lo cual se puede concluir que las metodologías ágiles son las más convenientes y cumplen con las expectativas esperadas para el proyecto de desarrollo de un marco de trabajo para la empresa Farmaenlace Cía. Ltda.

3.1.2 Estudio comparativo entre metodologías ágiles

Una vez establecido que los tipos de metodologías que se usará en el proyecto son las ágiles, se debe seleccionar una, tomando en cuenta la que más convenga a la empresa. Para aquello se apoyará en el estudio realizado por Iacovelli, que ofrece un framework para la clasificación y evaluación de metodologías ágiles para el desarrollo de software, el cual se fundamenta en cuatro puntos de vista fundamentales de los aspectos de las metodologías (Uso, Capacidad de agilidad, Aplicabilidad, Procesos y productos), cabe aclarar que cada aspecto tiene un conjunto de atributos a ser evaluados.

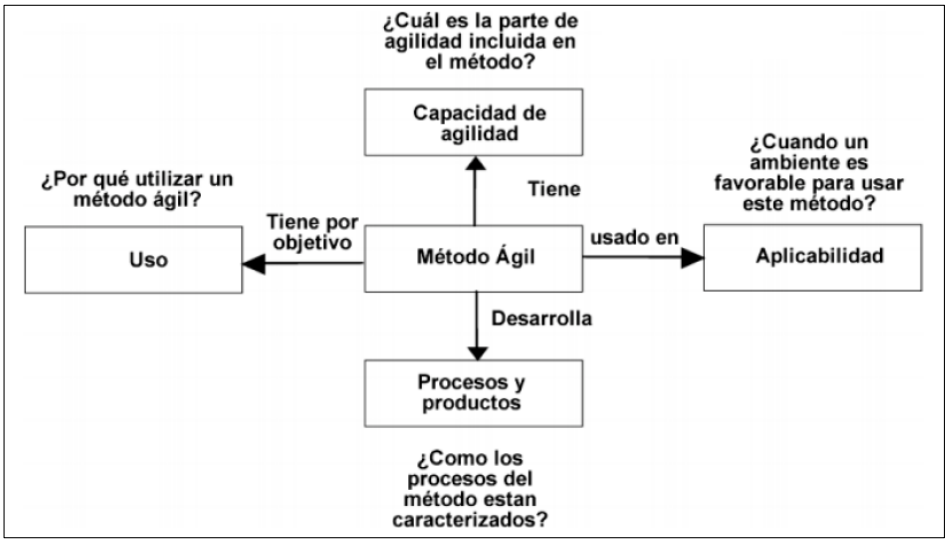


Figura 18 Puntos de vista de metodología ágiles (Lacovelli)

En este segmento se da una breve explicación acerca del propósito de cada uno de los puntos de vista de las metodologías ágiles según el método de Lacovelli:

- El punto de vista de “Uso”, refleja para que utilizar una metodología ágil en el desarrollo de software, los beneficios que pueden obtener el equipo de trabajo y el cliente.
- El punto de vista de “Capacidad de agilidad”, representa los conceptos de agilidad que brinda las metodologías dentro del ciclo de vida del desarrollo de las aplicaciones.
- El punto de vista de “Aplicabilidad”, su objetivo es mostrar el impacto de los aspectos ambientales, los aspectos del entorno y representar si es favorable la aplicación de metodologías ágiles.
- El punto de vista de “Procesos y productos”, representa cómo se caracterizan los procesos ágiles de la metodología en dos dimensiones: la primera son las actividades de desarrollo de software; y la segunda representa el nivel de abstracción de sus directrices y reglas (Egas & Játiva, 2008).

La manera de aplicar el método del framework de Lacovelli es la siguiente:

1. Llenar un formulario prestablecido de “Elección de una metodología ágil”, que contiene como preguntas los atributos de los 4 puntos de vista de los aspectos de las metodologías, las respuestas a este formulario se debe fundamentar en las necesidades empresariales.
2. Extraer los mismos atributos del formulario anterior pero ahora aplicado a las metodologías ágiles a estudiar.
3. Comparar los formularios del punto 1 y el punto 2, en los atributos que coinciden las respuestas poner el valor de uno y en los que no coinciden poner el valor cero.
4. Al final se debe sumar por metodología los puntos que coincidieron con las necesidades empresariales, y la metodología que mayor puntaje tenga es la que más conviene aplicar para satisfacer las necesidades empresariales.

La asignación de los valores en los formularios se hizo mediante el método de decisión por consenso tomando en cuenta los criterios de la gerencia de sistemas y los 2 coordinadores de desarrollo del área de análisis y desarrollo de la empresa.

A continuación se aplica el método de Lacovelli, para el desarrollo del marco de trabajo de la empresa Farmaenlace Cía. Ltda.

1. Formulario: “Elección de una metodología ágil”,

Vista: USO

Verdadero (V) o Falso (F) en cada una de las premisas.

Tabla 5
Valoración del Uso

Nro.	PREMISA	RESPUESTA
1	Respeto de las fechas de entrega	Verdadero
2	Cumplimiento de los requisitos	Verdadero

CONTINÚA

3	Respeto al nivel de calidad	Verdadero
4	Satisfacción del usuario final	Verdadero
5	Entornos turbulentos	Falso
6	Favorable al Off shoring	Verdadero
7	Aumento de la productividad	Verdadero

Vista: CAPACIDAD DE AGILIDAD

Verdadero (V) o Falso (F) en cada una de las premisas.

Tabla 6
Valoración de la capacidad de agilidad

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

Vista: APLICACIÓN

Tabla 7
Valoración de la aplicación

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

Vista: PROCESOS Y PRODUCTOS

Verdadero (V) o Falso (F) en cada una de las premisas.

- Nivel de abstracción de las normas y directrices:

Tabla 8
Valoración de normas y directrices de la metodología ágil

Nro.	PREMISA	RESPUESTA
1	Gestión de proyectos	Verdadero
2	Descripción de procesos	Verdadero
3	Normas y orientaciones concretas sobre las actividades y productos	Falso

- Las actividades cubiertas por el método ágil:

Tabla 9
Valoración de las actividades cubiertas por la metodología ágil

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

- Productos de las actividades del método:

Tabla 10
Valoración de los productos de las actividades de la metodología ágil

Nro.	PREMISA	RESPUESTA
1	Modelos de diseño	Falso
2	Comentario del código fuente	Verdadero
3	Ejecutable	Verdadero
4	Pruebas unitarias	Falso
5	Pruebas de integración	Falso
6	Pruebas del sistema	Verdadero
7	Pruebas de aceptación	Verdadero
8	Informes de calidad	Falso
9	Documentación de usuario	Verdadero

2. Extraer los mismos atributos del formulario anterior pero ahora aplicado a las metodologías ágiles a estudiar.

Tabla 11
Formulario de Vistas de las metodologías

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

CONTINÚA

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

3. Comparar los formularios del punto 1 y el punto 2, en los atributos que coinciden las respuestas poner el valor de uno, caso contrario poner el valor cero.

Tabla 12
Formulario de Vistas de las metodologías

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

CONTINÚA

	Modelado	0	0	1	0
	Código	1	1	1	1
	Pruebas unitarias	0	0	0	0
	Pruebas de integración	0	0	0	0
	Prueba del sistema	1	1	1	1
	Prueba de aceptación	0	0	0	0
	Control de calidad	1	1	1	1
	Sistema de uso	0	0	0	0
	Productos de las actividades del método ágil				
	Modelos de diseño	1	0	1	0
	Comentario del código fuente	1	1	1	1
	Ejecutable	1	1	1	1
	Pruebas unitarias	0	0	0	0
	Pruebas de integración	0	0	0	0
	Pruebas del sistema	1	0	1	1
	Pruebas de aceptación	0	0	0	0
	Informes de calidad	1	1	1	1
	Documentación de usuario	0	0	0	0
	TOTAL EVALUACIÓN	**27**	**33**	**24**	**30**

4. La sumatoria de los puntos por metodología nos indica que SCRUM con 33 puntos y SCRUMBAN con 30 puntos son los que más se adaptan a las necesidades expuestas en el estudio.

3.1.3 Conclusión del estudio comparativo de las metodologías

De acuerdo al estudio comparativo de la sección anterior se puede ver que Scrum tiene el puntaje más alto, lo que quiere decir que es la metodología ágil que mejor se adapta y cumple con las expectativas expuestas por la empresa, y por lo tanto se puede concluir y escoger como la metodología base para el presente estudio; pero eso no quiere decir que las características, técnicas o buenas prácticas de las otras metodologías no se puedan aplicar para el desarrollo del marco de trabajo de la fabricación de software de la empresa Farmaenlace Cía. Ltda.

3.2 Estudio comparativo de herramientas ALM

El estudio comparativo de las herramientas para la administración del ciclo de vida de las aplicaciones ALM, se lo realizará tomando como punto de partida los estudios de los cuadrantes mágicos de Gartner, en donde

resumen en una gráfica la situación del mercado de un producto tecnológico en un momento determinado. Las compañías representantes se las distribuyen en función de su tipología y de sus productos, por lo que se las cataloga de la siguiente manera:

- Líderes (leaders): la mayor puntuación es resultado de la combinación del (vender y ofrecer soporte a sus productos y servicios a nivel global), con el alcance de visión que se refiere a su potencial.
- Aspirantes (challengers): se caracteriza por ofrecer buenas funcionalidades y un número considerable de instalaciones del producto.
- Visionarios (visionaries): tienen capacidades de ofrecer la gestión de contenido empresarial de forma nativa o mediante alianzas con otros socios, lo cual es un fuerte impulso a la integración de programas y plataformas, como también una habilidad para anticiparse a las necesidades del mercado.
- Nichos específicos (niche players): enfocados a determinadas áreas tecnológicas de la gestión de contenido empresarial, pero sin disponer de una suite completa.

A continuación se muestra el cuadrante mágico de Gartner del estudio de las herramientas ALM del año 2013:

Figura 19 Cuadrante mágico de herramientas ALM

Fuente: (Oviedo, 2014)

Como se puede apreciar en la gráfica del estudio de Gartner, los líderes son Microsoft, IBM y Atlassian, de las cuales se considera las dos primeras suites de herramientas que son Microsoft TFS y la suite IBM Rational Jazz, para hacer una comparativa de las principales características.

La manera de evaluar la comparativa será la siguiente: en una tabla se listará las etapas con las principales características tomadas del reporte del estudio de Gartner realizada en noviembre 2013 de las herramientas ALM. En la columna “PESO EMPRESARIAL”, se asignará un valor de 0 a 4 que representa el costo beneficio empresarial.

La asignación de los valores lo hará el gerente de Sistemas y los 2 coordinadores del área de análisis y desarrollo de la empresa Farmaenlace Cía. Ltda., mediante el método de decisión por consenso, con el objetivo de saber cuál herramienta ALM satisface las necesidades de la empresa.

A continuación se verifica las características que poseen las soluciones de Microsoft TFS e IBM Rational Jazz, donde se compara con las características mencionadas del estudio de Garnet, en el caso de tenerlas se registra el valor “SI” en la columna “TIENE” y se asigna el valor del “PESO EMPRESARIAL” a la columna “PESO”, en el caso de no tener la característica se asigna el valor de “NO” en la columna “TIENE” y se asigna el valor de cero en la columna “PESO”.

Los valores de “PESO EMPRESARIAL”, se los cataloga de la siguiente manera: 0 no necesario, 1 poco necesario, 2 necesario intermedio, 3 necesario, 4 muy necesario.

Tabla 13
Tabla comparativa de Herramientas ALM

Etapas	Características	PESO EMPRESARIAL	Microsoft TFS		IBM Rational Jazz	
			TIENE	PESO	TIENE	PESO
Requerimientos	Procesador de textos o una hoja de cálculo a los usuarios.	3	NO	0	SI	3
	Sistemas y equipos de ingeniería de software para mejorar la colaboración y agilidad proyecto a través de la edición concurrente, versionado automático y priorización eficaz.	4	NO	0	SI	4
	Gestión de requisitos de soluciones de usuarios.	4	NO	0	SI	4
	Único servidor Web de requisitos para una colaboración más estrecha entre los equipos de desarrollo de productos multi-disciplinarios.	4	NO	0	SI	4
Codificación	Control de versiones.	4	SI	4	SI	4
Trabajo	Herramientas de planificación.	4	SI	4	SI	4
	Trabajo con plantillas ágiles y formales.	4	SI	4	SI	4
	Colaborar con recursos del equipo.	3	SI	3	SI	3
	Enlace de trabajos para apoyar la	3	SI	3	SI	3

CONTINÚA

	trazabilidad.					
	Crear, personalizar y administrar informes.	3	SI	3	SI	3
	Personalizar la experiencia de seguimiento de trabajo.	3	SI	3	SI	3
	Sincronizar TFS con Project Server.	2	SI	2	NO	0
	Cuadros de mandos de varios niveles.	3	SI	3	SI	3
Generar /construcción de aplicaciones (Build)	Definir y personalización de proceso de construcción.	3	SI	3	SI	3
	Ejecutar, controlar y gestionar la construcción de las aplicaciones.	2	SI	2	SI	2
	Diagnosticar problemas de la construcción.	3	SI	3	NO	0
Pruebas	Planificación de pruebas.	3	SI	3	SI	3
	Crear, ejecutar pruebas manuales.	3	SI	3	SI	3
	Seguimiento de resultados.	3	SI	3	SI	3
	Compartir pasos entre los casos de prueba.	3	SI	3	NO	0
	Repetir pruebas con diferentes datos.	3	SI	3	SI	3
Automatizar implementaciones (Release)	Administrar usuarios, grupos y permisos.	3	SI	3	NO	0
	Administrar Implementaciones.	3	SI	3	NO	0
	Lanzamiento sin agentes de implementación.	2	SI	2	NO	0
	Lanzamiento con agentes de implementación.	2	SI	2	NO	0
	Desencadenar una implementación de una generación de la aplicación.	3	SI	3	NO	0
	Implementación continuamente para Azure.	2	SI	2	NO	0
	Acciones para implementar una aplicación	3	SI	3	NO	0
	Las variables de configuración y variables de sistema	2	SI	2	NO	0
Modelado de aplicación.	Cambio del Diseño usando la Visualización y Modelado.	2	SI	2	NO	0

	Visualizar código	2	SI	2	NO	0
	El desarrollo de modelos de Diseño de Software.	2	SI	2	NO	0
	El uso de modelos dentro del Proceso de Desarrollo.	2	SI	2	NO	0
	La validación de su sistema durante el desarrollo.	3	SI	3	NO	0
	Extendiendo Modelos y diagramas UML.	2	SI	2	NO	0
	Extendiendo Diagramas de capa.	2	SI	2	NO	0
	SDK de Modelado para Visual Studio - lenguajes específicos de dominio.	3	SI	3	NO	0
TOTALES		**116**		**100**		**59**
PORCENTAJES				**86%**		**51%**

CONTINÚA

3.2.1 Conclusión del Estudio comparativo de herramientas ALM

Luego de hacer el estudio comparativo de las dos principales herramientas del mercado para la administración del ciclo de vida de las aplicaciones ALM, Microsoft TFS y la plataforma de tecnología IBM Rational Jazz, se puede analizar lo siguiente: En la primera comparativa de las características de las herramientas se obtuvo los siguientes resultados, la herramienta TFS de Microsoft obtuvo 100 puntos de los 116 posibles, lo que representa un 86% de satisfacción de las necesidades empresariales, a diferencia de la plataforma de herramientas IBM Rational Jazz que obtuvo 59 puntos lo que representa un 51% de cumplimiento del total de características esperadas. Lo que quiere decir que las herramientas de Microsoft son las que más aportan con características para apoyar en la automatización de la administración del ciclo de vida de las aplicaciones en el desarrollo de software para la empresa Farmaenlace Cía. Ltda.

3.3 Desarrollo del marco de trabajo Farmaenlace

Luego de haber realizado el análisis del estudio de las metodologías para el desarrollo del software se concluyó que Scrum es el marco de trabajo que más se adapta a las necesidades de la empresa Farmaenlace Cía. Ltda., por lo cual se tomará como modelo base para el desarrollo de la metodología propia de la empresa.

La manera que se realizará el desarrollo del marco de trabajo es el siguiente: se considerará como base los conceptos, técnicas, roles, eventos, reglas y mejores prácticas de Scrum. Se escogerá, modificará o suprimirá características a cada una de las fases de Scrum, tomando en cuenta las que mejor se adapten y convengan al equipo de trabajo de la empresa. De esta manera se estructurará y se desarrollará el marco de trabajo para el desarrollo de software del Departamento de Sistema de la Empresa Farmaenlace. Cía. Ltda.

El criterio para escoger las características del desarrollo del marco de trabajo propuesto, se realizará mediante el método de decisión por consenso entre el gerente de Sistemas y 2 coordinadores de análisis y desarrollo de la empresa.

3.3.1 Estructuración del marco de trabajo basado en Scrum

Scrum se basa en el control de procesos empíricos, de esta manera se asegura que las decisiones se las toma en base a la experiencia. Su implementación se fundamenta en tres valores que son la *transparencia*, *inspección* y *adaptación*, desde un enfoque interactivo e incremental para optimizar la predictibilidad y control de riesgos.

Transparencia: Los aspectos significativos del proceso deben ser visibles y transparentes para todas las personas que son responsables del resultado del producto desarrollado, por lo cual se debe definir un estándar común

para que los involucrados y observadores tengan un mismo entendimiento del proceso.

Inspección: Los usuarios de Scrum deben inspeccionar frecuentemente los artefactos de Scrum y el progreso del trabajo hacia el objetivo, para detectar variaciones. Las inspecciones son beneficiosas cuando lo hacen inspectores expertos en el lugar de trabajo, pero no deben ser tan frecuentes para no interferir en el proceso de desarrollo.

Adaptación: Si el inspector determina que algún aspecto del proceso o que el producto resultante presenta desviación de los límites aceptables, estos deben ser ajustados cuanto antes para minimizar desviaciones mayores.

Para estructurar las características del marco de trabajo se tomó como base "La Guía Definitiva de Scrum: Las Reglas del Juego" que se conforma de eventos, artefactos, técnicas, y roles del equipo de trabajo; que interactúan a lo largo del ciclo de vida del desarrollo de las aplicaciones.

a) Roles del equipo de trabajo basados en Scrum

A continuación se detallan las características de las personas que conforman los roles del equipo de trabajo.

Tabla 14
Roles del equipo de trabajo basados en Scrum

ROLES	DEFINICIÓN
ROLES PRINCIPALES	
El Dueño del Producto - (Product Owner).	Es el responsable de maximizar el trabajo del equipo y del producto desarrollado desde una perspectiva del negocio. Representa la voz del cliente y su principal tarea es gestionar la Lista del Producto (Product Backlog).
Maestro Scrum – (Scrum Master).	Su trabajo es eliminar los obstáculos que impidan al equipo Scrum alcanzar el objetivo del sprint. Actúa como una protección entre el equipo y cualquier influencia que les distraiga. Asegura de que el proceso y las reglas Scrum se

	cumpla de manera adecuada.
El Equipo De Desarrollo - (Development Team).	Se conforma de profesionales multidisciplinarios y auto-organizados que son capaces de elegir la mejor manera de llevar a cabo su trabajo para realizar entregas iterativas e incrementales del producto "Terminado", que potencialmente se puedan poner en producción al final de cada Sprint.
	ROLES AUXILIARES
Interesados - (Stakeholders).	Son personas que hacen posible el proyecto y quienes se beneficiarán con el desarrollo del producto.
Administradores - (Managers).	Son las personas que establecen el ambiente para el desarrollo del producto.

En esta sección se estructurará una plantilla detallada con características, responsabilidades y reglas de los roles principales del equipo de trabajo Scrum, de las cuales se escogerá las que más se apeguen y convengan a la realidad del entorno de desarrollo de la empresa, esto permitirá construir el marco de trabajo propuesto. A continuación se presenta las plantillas elaboradas en este estudio tomando en cuenta la Guía Definitiva de Scrum para cada uno de los roles principales del equipo de trabajo Scrum.

- **Dueño del Producto - (Product Owner)**

Tabla 15
Dueño del Producto - (Product Owner)

Responsabilidades	Aplica
Deberá expresar claramente los elementos de la Lista del Producto.	Sí
Ordenará los elementos en la Lista del Producto para alcanzar los objetivos de la mejor manera posible.	Sí
Optimizará el valor del trabajo que desempeña el Equipo de Desarrollo.	Sí
Asegurará que la Lista del Producto muestre el trabajo que realizará el equipo de una manera transparente y clara.	Sí
Asegurará que el Equipo de Desarrollo entienda los elementos de la Lista	Sí

CONTINÚA

del Producto.	
Puedrá delegar el trabajo al Equipo de Desarrollo, pero él sigue siendo el responsable de lo antes expuesto.	Sí
Reglas	
El Dueño de Producto es una persona, no un comité.	Sí
Podrá representar los deseos de un comité en la Lista del Producto, pero si quieren cambiar la prioridad de un elemento de la Lista deben hacerlo a través del Dueño de Producto.	Sí
La organización deberá respetar sus decisiones, para que pueda hacer bien su trabajo.	Sí
Las decisiones tomadas se reflejan en el contenido y en la priorización de la Lista del Producto.	Sí
Nadie podrá solicitar al Equipo de Desarrollo que trabaje con un conjunto diferente de requerimientos.	Sí
El Equipo de Desarrollo no deberá actuar con base a lo que diga cualquier otra persona.	Sí

- **Equipo de desarrollo - (Development Team)**

Tabla 16
Equipo de desarrollo - (Development Team)

Reglas	Aplica
Son los únicos que deberán participar en la creación del incremento.	Sí
La organización los deberá empoderar para que puedan organizarse y gestionar su propio trabajo.	Sí
Características	
Deberán ser auto-organizados, nadie (ni siquiera el "Scrum Master") debe indicar cómo convertir los elementos de la "Lista del Producto" en	Sí

Incrementos funcionales de software.	
Deberán ser multifuncionales, capaces de tener todas las habilidades necesarias para crear el Incremento funcional del producto.	Sí
Scrum no reconocerá títulos entre los miembros del Equipo de Desarrollo, todos son Desarrolladores independientemente del trabajo que realicen. **No hay excepciones a esta regla.**	Sí
No se reconocerá sub-equipos dentro de los equipos de desarrollo, no importa las características o funciones particulares que tengan. **No hay excepciones a esta regla.**	Sí
Aunque los miembros o áreas tengan tareas especializadas, la responsabilidad recaerá en el Equipo de Desarrollo como un todo.	Sí

- **Tamaño del Equipo de Desarrollo.**

El tamaño óptimo deberá ser lo suficientemente pequeño como para permanecer ágil y lo suficientemente grande para completar una cantidad de trabajo significativo.

Tabla 17
Tamaño del equipo de desarrollo

Características	Aplica
Tener menos de tres miembros reduce la interacción y las ganancias de productividad. **"Los Equipos de Desarrollo" más pequeños podrían tener limitaciones en cuanto a las habilidades necesarias durante un Sprint, y como consecuencia no entregar el Incremento del producto que potencialmente se pueda poner en producción.**	Sí
En los equipo de desarrollo al tener más de nueve miembros, requiere demasiada coordinación, y generan demasiada complejidad como para que se pueda gestionar mediante un proceso empírico.	Sí
El Dueño de Producto y el Scrum Master no se contarán en el cálculo del	Sí

tamaño del equipo de desarrollo a menos que estén contribuyendo a trabajar en la Lista de Pendientes del Sprint (Sprint Backlog).	

- **Maestro Scrum - (Scrum Master)**

Tabla 18
Maestro Scrum - (Scrum Master)

Responsabilidades	Aplica
Asegurará que Scrum sea entendido y adoptado por todos los involucrados en el proyecto.	Sí
Asegurará que el Equipo Scrum trabaje ajustándose a la teoría, prácticas y reglas de Scrum.	Sí
Deberá ser un líder que está al servicio del Equipo Scrum.	Sí
Deberá ayudar a las personas externas del equipo a entender que interacciones ayudan o no al Equipo Scrum.	Sí
Deberá ayudar a todos a modificar las interacciones para maximizar el valor creado por el Equipo Scrum.	Sí

- **Servicio del Scrum Master al Dueño de Producto**.

El Scrum Master prestará servicio al Dueño de Producto de las siguientes maneras:

Tabla 19
Servicio del Scrum Master al Dueño de Producto

Características	Aplica
Encontrará técnicas para gestionar la Lista de Producto de manera efectiva.	Sí
Ayudará para que el Equipo Scrum entienda la necesidad de contar con elementos claros y concisos en la Lista de Producto.	Sí

CONTINÚA

Entenderá la planificación del producto en un entorno empírico.	Sí
Asegurará que el Dueño de Producto conozca cómo ordenar la Lista de Producto para maximizar el valor.	Sí
Entenderá y practicará la agilidad.	Sí
Facilitará los eventos de Scrum según se requiera o necesite.	Sí

- **Servicio del Scrum Master al Equipo de Desarrollo**

El Scrum Master prestará servicio al Equipo de Desarrollo de la siguiente manera:

Tabla 20
Servicio del Scrum Master al Equipo de Desarrollo

Características	Aplica
Guiará al Equipo de Desarrollo para que pueda ser auto-organizado y multifuncional.	Sí
Ayudará al Equipo de Desarrollo a crear productos de alto valor.	Sí
Eliminará impedimentos para el progreso del Equipo de Desarrollo.	Sí
Facilitará los eventos de Scrum según se requiera o necesite.	Sí
Guiará al Equipo de Desarrollo y a la organización en los entornos que Scrum aún no ha sido adoptado o entendido por completo.	Sí

- **Servicio del Scrum Master a la Organización**

El Scrum Master prestará servicio a la organización de varias formas:

Tabla 21
Servicio del Scrum Master a la Organización

Características	Aplica

Liderará y guiará a la organización en la adopción de Scrum.	Sí
Planificará las implementaciones de Scrum en la organización.	Sí
Ayudará a los empleados e interesados a entender y llevar a cabo Scrum en el proceso de desarrollo empírico de productos.	Sí
Motivará cambios que incrementen la productividad del Equipo Scrum.	Sí
Trabajará con otros Scrum Masters para incrementar la efectividad de la aplicación de Scrum en la organización	Sí

b) Eventos basados en Scrum

Los eventos son intervalos de tiempo que tendrán una duración fija y máxima apropiada para no permitir desperdicios en el proceso, son diseñados para crear regularidad y minimizar la necesidad de reuniones no definidas y así habilitar la transparencia y adaptación en el proceso, a continuación se listan los eventos con sus respectivas definiciones:

Tabla 22
Eventos basados en Scrum

EVENTOS	DEFINICIÓN
Sprint.	Es considerado como el corazón de Scrum, es un intervalo de tiempo durante el cual se crea un incremento de producto "Terminado", utilizable y potencialmente entregable al cliente. Es el contenedor del resto de eventos.
Reunión de Planificación de Sprint - (Sprint Planning Meeting).	En este evento se revisa y planifica el trabajo que se realizará durante el Sprint.
Scrum Diario - (Daily Scrum).	Es una reunión diaria para inspeccionar el avance y la proyección del trabajo que se va a hacer en el lapso de un día.
Revisión de Sprint - (Sprint Review).	Es una reunión que se realiza al final del Sprint en donde se hace la presentación del trabajo realizado.
Objetivo del Sprint - (Sprint Goal).	Es la meta que se establece, y que se cumplirá con el trabajo realizado del Sprint.

Retrospectiva de Sprint - (Sprint Retrospective).	Es una reunión en donde el equipo de trabajo puede realizarse una auto-inspección y crear un plan de mejoras que sean abordadas durante el siguiente Sprint.

A continuación se detallarán cada uno de los eventos, y de los cuales se debe escoger, modificar o eliminar las características que más convengan al equipo de trabajo de la empresa.

- **Sprint.**

Tabla 23
Sprint

Características	Aplica
El tiempo de los Sprints puede durar entre 1 a 4 semanas.	Sí
En implementaciones de Scrum, se puede comenzar con Sprints de 2 o 3 semanas, y luego ir ajustando de acuerdo al ritmo del equipo, aunque sin relajarlo demasiado.	Sí
Es recomendable que la duración de los Sprints sea definida por el equipo con base a su experiencia.	Sí
Es conveniente que la duración de los Sprints sea constante a lo largo del esfuerzo del desarrollo.	Sí
Cada nuevo Sprint comienza inmediatamente después de la finalización del Sprint anterior.	Sí
Al final de cada sprint, el equipo debe presentar los avances logrados, que debería ser el incremento del producto potencialmente entregable al cliente.	Sí
No es recomendable agregar objetivos al sprint o "Sprint Backlog". **Solo se puede agregar objetivos, en caso que la falta de estos amenace al éxito del proyecto.**	Sí
La constancia permite la concentración del equipo y mejora la productividad del trabajo.	Sí
Reglas	
No se debe realizar cambios que afecten al Objetivo del Sprint (Sprint Goal).	Sí
Los objetivos de calidad no deben disminuir.	Sí
El Dueño de Producto y el Equipo de Desarrollo pueden clarificar y renegociar el alcance, esto se lo realiza conforme se vaya aprendiendo más acerca del contexto.	Sí

Cada Sprint puede considerarse un proyecto, pero con un horizonte no mayor de un mes.	Sí
Al igual que los proyectos, los Sprints se usan para lograr algo.	Sí
Cada Sprint tiene una definición, diseño y un plan flexible de construcción, que guiará el trabajo y el producto resultante.	Sí
Cuando el tiempo del Sprint es muy largo, la definición o complejidad del trabajo planificado por lo general cambia y aumenta el riesgo del proyecto.	Sí
Al finalizar los Sprints se habilita la predictibilidad, la inspección y adaptación del progreso del proyecto.	Sí
La duración de los Sprints limita hasta un mes calendario el riesgo del costo.	Sí
Cancelación de Sprint	
Debido a la corta duración del Sprint, la cancelación es poco común y rara vez tiene sentido.	Sí
Podrán ser cancelados solo por el Dueño de Producto.	Sí
Se podrá cancelar cuando el objetivo del Sprint llega a quedar obsoleto.	Sí
Se podrá cancelar cuando las condiciones del mercado, dirección de la empresa o la tecnología cambian.	Sí
Se podrá cancelar cuando no hay sentido seguir con él trabajo dadas las circunstancias.	Sí
Al cancelar el Sprint, se debe revisar los Elementos de la Lista de Producto que se hayan completado.	Sí
Al cancelar el Sprint, si una parte del trabajo es potencialmente entregable, el Dueño de Producto normalmente lo deberá aceptar.	Sí
Al cancelar el Sprint, los elementos de la "Lista de Producto" no completados se deben volver a estimar e introducir a la Lista de Producto.	Sí
Como consecuencia, el trabajo finalizado perderá valor con rapidez y frecuentemente se debe volver a estimar.	Sí
Como consecuencia, se consumirá recursos porque el trabajo se deberá reagrupar en otra Reunión de Planificación de Sprint para empezar nuevamente.	Sí
Como consecuencia, podrían ser traumáticas para el equipo Scrum.	Sí

- **Reunión de Planificación de Sprint**

Tabla 24
Reunión de Planificación de Sprint

Características	Aplica
El Scrum Master, debe asegurar que el evento se lleve a cabo.	Sí
El Scrum Master, debe asegurar que los asistentes entiendan su propósito.	Sí
El Scrum Master, debe enseñar al Equipo Scrum a mantenerse dentro del bloque de tiempo asignado para la reunión.	Sí
La reunión deberá responder las siguientes preguntas: ○ ¿Qué puede ser terminado en el Sprint? ○ ¿Cómo se conseguirá completar el trabajo seleccionado?	Sí
¿Qué puede ser terminado en el Sprint?	
Entrada de la reunión: La Lista de Producto.	Sí
Entrada de la reunión: El último Incremento de producto.	Sí
Entrada de la reunión: La capacidad proyectada por el Equipo de Desarrollo para el Sprint. (Planificación)	Sí
Entrada de la reunión: El rendimiento del Equipo de Desarrollo en el Sprint pasado.	No
El Equipo de Desarrollo, proyectará la funcionalidad que se desarrollará en el Sprint.	Sí
El Dueño de Producto, discute el objetivo y los Elementos de la Lista de Producto que se deberán lograr en el Sprint.	Sí
El Equipo Scrum completo, colaborará para que se entienda el trabajo del Sprint.	Sí
El equipo de Desarrollo, definirá el número de elementos de la Lista de Producto que se desarrollarán en el Sprint.	Sí
Solo el Equipo de Desarrollo podrá evaluar qué es capaz de lograr durante el Sprint.	Sí
Luego que el Equipo de Desarrollo proyecte que elementos de la Lista de Producto entregará en el Sprint, el Equipo Scrum elaborará el Objetivo del	Sí

Sprint (Sprint Goal).	
El "Objetivo del Sprint" deberá lograrse a través de la implementación de la Lista de Producto escogida para el Sprint.	Sí
¿Cómo se conseguirá completar el trabajo seleccionado?	
Los elementos de la Lista de Producto seleccionados, más el plan de como terminarlos, recibe el nombre de "Lista de Pendientes del Sprint" (Sprint Backlog).	Sí
El Equipo de Desarrollo, comenzará por el diseño del sistema y definirá el trabajo a realizar.	Sí
La estimación del trabajo se lo hará en esfuerzo o tamaño de producto y este puede ser variable dependiendo de los equipos Scrum.	Sí
El Equipo de Desarrollo planificará suficiente trabajo para hacer una proyección de lo que necesita realizar durante el Sprint.	Sí
Al finalizar la reunión, el trabajo planificado por el Equipo de Desarrollo se deberá descomponer en unidades de un día o menos.	Sí
El Equipo de desarrollo se auto-organizará para asumir el trabajo de la Lista de Pendientes del Sprint. Esto lo deberá hacer durante la reunión de Planificación como a lo largo del Sprint.	Sí
El Dueño de Producto podrá ayudar a aclarar los elementos de la Lista de Producto seleccionados y hacer concesiones.	Sí
Cuando el trabajo seleccionado para el Sprint es demasiado o muy poco, el Equipo de Desarrollo podrá renegociarlo con el Dueño de Producto.	Sí
El Equipo de Desarrollo podrá invitar a personas externas al equipo Scrum, con el fin que proporcionen asesoría técnica o relacionada con el dominio del negocio.	Sí
Para finalizar la Reunión de Planificación de Sprint, el Equipo de Desarrollo deberá ser capaz de explicar al Dueño de Producto y al Scrum Master cómo pretende trabajar como un equipo auto-organizado para lograr el Objetivo del Sprint y crear el Incremento del producto esperado.	Sí

- **Objetivo del Sprint**

Tabla 25
Objetivo del Sprint

Características	Aplica
Se creará durante la reunión de Planificación del Sprint.	Sí
Ofrecerá al equipo de desarrollo cierta flexibilidad con respecto a la funcionalidad que se implementará en el Sprint.	Sí

Los elementos seleccionados de la Lista del Producto deberán ofrecer una función coherente para convertirse en el objetivo del Sprint.	Sí
Representarán un nexo para que el Equipo de Desarrollo trabaje en conjunto y no en iniciativas separadas.	Sí
El objetivo se deberá mantener presente en el transcurso del trabajo del Equipo de Desarrollo.	Sí
Proporcionará una guía al Equipo de Desarrollo del porqué está construyendo el incremento.	Sí
Se deberá implementar la funcionalidad y tecnología necesaria para satisfacer el objetivo del Sprint.	Sí
Si el trabajo resulta diferente a lo que el Equipo de Desarrollo esperaba, pueden negociar con el Dueño del Producto el alcance de la Lista de pendientes del Sprint (Sprint Backlog).	Sí

- **Scrum Diario**

Tabla 26
Scrum Diario

Características	Aplica
La reunión deberá tener una duración máxima 15 minutos.	Sí
Se inspeccionará el trabajo avanzado desde el último Scrum Diario.	Sí
Se deberá hacer una proyección del trabajo que podrá completarse antes del siguiente Scrum Diario.	Sí
El Equipo de Desarrollo sincronizará sus actividades y creará un plan para las siguientes 24 horas.	Sí
La reunión se realizará en el mismo lugar y a la misma hora.	Sí
Durante la reunión, cada miembro del Equipo de Desarrollo explica: ○ **¿Qué hice desde la última reunión que ayudó al Equipo de Desarrollo a lograr el Objetivo del Sprint?** ○ **¿Qué haré hoy para ayudar al Equipo de Desarrollo a lograr el Objetivo del Sprint?** ○ **¿Hay algún impedimento que evite que el Equipo de Desarrollo o yo logremos el Objetivo del Sprint?**	Sí
El Equipo de Desarrollo evaluará el progreso y la tendencia hacia la finalización del trabajo para lograr el Objetivo del Sprint.	Sí
La reunión optimizará las posibilidades para que el Equipo de Desarrollo cumpla con el Objetivo del Sprint.	Sí

El Equipo de Desarrollo deberá entender y buscar la manera de trabajar como un equipo auto-organizado para lograr el Objetivo del Sprint y crear el Incremento funcional del producto.	Sí

Revisión del Sprint

Tabla 27
Revisión del Sprint

Características	Aplica
Durante la Revisión de Sprint, el Equipo Scrum y los interesados informarán acerca de lo que se hizo durante el Sprint.	Sí
Cualquier cambio que haya habido en la Lista de Producto durante el Sprint, los asistentes deberán colaborar para determinar los pasos a seguir para optimizar el valor.	Sí
Será una reunión informal en donde se fomenta la colaboración del equipo Scrum.	Sí
El tiempo de duración de la reunión será equivalente a una hora por cada semana que dure el Sprint. (Ejemplo: Reunión de cuatro horas para un Sprint de cuatro semanas).	Sí
El Scrum Master asegurará que el evento se lleve a cabo y que los asistentes entiendan su propósito.	Sí
El Scrum Master enseñará a todos a mantener el evento dentro del tiempo establecido.	Sí
Los asistentes a la reunión serán el Equipo Scrum, y los interesados clave que son invitados por el Dueño de Producto.	Sí
El Dueño de Producto explicará los elementos de la Lista de Producto que se terminaron y los que no se terminaron.	Sí
El Equipo de Desarrollo explicará que estuvo bien durante el Sprint, también explicará los problemas que se presentaron y la manera de cómo fueron resueltos.	Sí
El Equipo de Desarrollo mostrará el trabajo “Terminado” y responderá preguntas acerca del Incremento.	Sí
El Dueño de Producto hablará acerca del estado actual de la Lista de Producto.	Sí
El Dueño de Producto, podrá proyectar probables fechas de finalización	Sí

basándose en el progreso obtenido hasta el momento (si es necesario).	
Todo el equipo Scrum podrá colaborar sugiriendo acerca de qué trabajo hacer a continuación, de modo que la Revisión del Sprint proporcione información de entrada valiosa para Reuniones de Planificación de Sprints subsiguientes.	Sí
Se revisará el cambio del mercado o el uso potencial del producto, esto aportará como valor a trabajos futuros.	Sí
Se revisará el tiempo del proyecto, presupuesto, capacidades potenciales y mercado para la próxima entrega prevista del producto.	Sí
Se definirá posibles elementos de la Lista de Producto para el siguiente Sprint.	Sí
Se definirá posibles ajustes en la Lista de Producto para enfocarse en nuevas oportunidades.	Sí
El resultado de la reunión será la Lista de Producto revisada.	Sí

- **Retrospectiva del Sprint**

Tabla 28
Retrospectiva del Sprint

Características	Aplica
Se realizará después de la Revisión de Sprint y antes de la siguiente Reunión de Planificación de Sprint.	Sí
El tiempo de duración será proporcional a tres horas para un Sprint de cuatro semanas.	Sí
El Scrum Master asegurará de que el evento se lleve a cabo y que los asistentes entiendan su propósito.	Sí
El Scrum Master enseñará a todos a que la reunión se lleve a cabo en el tiempo establecido.	Sí
El Scrum Master participará en la reunión como un miembro del equipo ya que es el responsable del proceso Scrum.	Sí
Se inspeccionará las relaciones entre las personas, los procesos y herramientas del último Sprint.	Sí
Se identificará y ordenará los elementos más importantes que salieron bien y sus posibles mejoras.	Sí
Se creará un plan para implementar las mejoras a la manera de cómo realizar el trabajo del Equipo Scrum.	Sí
El Scrum Master alentará al equipo a mejorar su desarrollo y prácticas	Sí

dentro del marco de proceso Scrum, para hacerlos más efectivos en los siguientes Sprints.	
El Equipo Scrum planificará formas de aumentar la calidad del producto según crea conveniente, mediante la adaptación de la Definición de "Terminado" (Definition of "Done").	Sí
El Equipo Scrum podrá identificar mejoras que se pueda implementar en el siguiente Sprint.	Sí

c) Artefactos basados en Scrum

Representan el trabajo o el valor del producto en diferentes formas, se transforman en herramientas utilizadas para la inspección y adaptación del producto. Son diseñados para presentar la información necesaria y asegurar que todos los involucrados tengan el mismo entendimiento en el proceso y progreso del trabajo.

Tabla 29
Artefactos basados en Scrum

ARTEFACTOS	DEFINICIÓN
Lista de Producto - (Product Backlog).	Es una lista ordenada de todos los elementos que podrían ser necesarios para desarrollar un producto. Se considera la única fuente de requisitos.
Lista de Producto del Sprint - (Sprint Backlog).	Es un conjunto de elementos seleccionados de la Lista de Producto, que se desarrollarán en el Sprint. Se complementa con una planificación para la entrega del Incremento del producto "Terminado" y de esta manera conseguir el Objetivo del Sprint
Incremento del producto terminado.	Se considera al grupo de elementos completados de la Lista de Producto durante el desarrollo de un Sprint.

A continuación se detallará los artefactos de Scrum, de los cuales se escogerá los que convengan al equipo de trabajo empresarial.

- **Lista de Producto (Backlog)**

Tabla 30
Lista de Producto (Backlog)

Características	Aplica
El Dueño de Producto será el responsable de la Lista de Producto, se encargará de administrar el contenido, disponibilidad y orden de prioridad.	Sí
La Lista de Producto nunca estará completa.	Sí
Al principio reflejará los requisitos conocidos y mejor entendidos.	Sí
Evolucionará a medida de que el producto o su entorno también lo hagan.	Sí
Será dinámica, cambia constantemente conforme se identifica necesidades para que el producto sea adecuado, competitivo y útil.	Sí
La Lista de Producto existirá, mientras el producto exista.	Sí
Enumerará todas las características, funcionalidades, requisitos, correcciones y mejoras que constituye para la creación o cambios que se harán en el producto.	Sí
Los elementos de la Lista de Producto tienen como atributos la descripción, el orden de prioridad, la estimación y el valor.	Sí
Se incrementará conforme el producto sea utilizado y el mercado proporciona retroalimentación.	Sí
Será un artefacto vivo ya que los requisitos nunca dejan de cambiar.	Sí
Podrá cambiar porque los requisitos de negocio, las condiciones del mercado o la tecnología cambian.	Sí
Varios Equipos Scrum podrán trabajar juntos en el mismo producto.	Sí
La Lista de Producto será única para todos los equipos de trabajo Scrum del mismo producto.	Sí
El Dueño del Producto podrá actualizar en cualquier momento los elementos de la Lista de Producto.	Sí
Los elementos de la Lista de Producto con mayor orden generalmente serán los más claros y detallados.	Sí
La claridad y detalle de los Elementos producirán estimaciones más precisas.	Sí
Los elementos de la Lista de Producto que el Equipo de Desarrollo ocupará en el siguiente Sprint deberán tener una granularidad mayor, de tal manera que cualquier elemento pueda ser “Terminado” dentro de la duración del Sprint.	Sí
Los elementos de la Lista de Producto que puedan ser “Terminados” por el Equipo de Desarrollo en un Sprint serán considerados como “preparados” o “accionables”, y podrán ser seleccionados en una reunión de Planificación de Sprint.	Sí

El Equipo de Desarrollo será el responsable de proporcionar todas las estimaciones.	Sí
El Dueño de Producto podrá influenciar al Equipo ayudándoles a entender y seleccionar soluciones de compromiso, pero las personas que harán el trabajo son las que hacen la estimación final.	Sí
El Refinamiento (Refinement).	
El Dueño de Producto y el Equipo de Desarrollo colaborarán acerca del refinamiento de los detalles de los elementos de la Lista de Producto.	Sí
Durante el refinamiento, se examinará y revisarán sus elementos.	Sí
El Equipo Scrum decidirá cómo y cuándo se hace el refinamiento.	Sí
El refinamiento usualmente no deberá consumir más del 10% de la capacidad del Equipo de Desarrollo.	Sí
Los elementos de la Lista de Producto adquirirán un mejor detalle y claridad luego de las actividades de refinamiento.	Sí
Seguimiento del trabajo	
Se deberá poder sumar a cualquier momento el trabajo total restante para alcanzar el objetivo.	Sí
El Dueño de Producto deberá comparar y hacer el seguimiento de la cantidad del trabajo total restante, al menos en cada Revisión de Sprint.	Sí
El Dueño del Producto deberá evaluar el progreso hacia la finalización del trabajo proyectado.	Sí
El Dueño del Producto deberá mostrar la información del progreso y proyección del trabajo de forma transparente a todos los interesados.	Sí

- **Lista de Producto del Sprint (Sprint Backlog)**

Tabla 31

Lista de Producto del Sprint (Sprint Backlog)

Características	Aplica
El Equipo de Desarrollo realizará una predicción del trabajo necesario para crear las funcionalidades que serán parte del Incremento "Terminado" del producto.	Sí
Se hará visible todo el trabajo que el Equipo de Desarrollo identifica como necesario para alcanzar el Objetivo del Sprint.	Sí
Se deberá realizar un plan con un detalle suficiente como para entenderlo durante el progreso de los Scrum Diarios.	Sí

El Equipo de Desarrollo podrá modificar la Lista de Pendientes del Sprint durante el Sprint.	Sí
El Equipo de Desarrollo podrá añadir nuevo trabajo a la Lista de Pendientes del Sprint, según lo requiera.	Sí
El Equipo de Desarrollo actualizará la estimación del trabajo restante a medida que se ejecuta el Sprint.	Sí
Cuando algún elemento del plan pasa a ser considerado innecesario, será eliminado.	Sí
La Lista de Pendientes del Sprint será la representación visible del tiempo real de trabajo que el Equipo de Desarrollo planea llevar a cabo durante el Sprint.	Sí
Seguimiento del Progreso del Sprint	
En cualquier momento durante un Sprint, se podrá calcular el trabajo restante de los elementos de la Lista de Pendientes del Sprint.	Sí
El Equipo de Desarrollo deberá hacer el seguimiento del trabajo restante al menos en cada Scrum Diario, para poder proyectar la factibilidad de conseguir el Objetivo del Sprint.	Sí
Con el seguimiento del trabajo restante a lo largo del Sprint, el Equipo de Desarrollo podrá gestionar su progreso.	Sí

- **Incremento del producto terminado**

Tabla 32
Incremento del producto terminado

Características	Aplica
Al final del Sprint el nuevo Incremento deberá estar "Terminado".	Sí
El incremento deberá cumplir con la "Definición de Terminado" (Definition of Done) por parte del Equipo Scrum.	Sí
El Dueño de Producto deberá decidir si se libera o no el incremento de producto terminado.	Sí
Definición de Terminado (Definition of "Done")	
Si la definición de "Terminado" para un incremento del producto es parte de convenciones, estándares o guías de la organización, entonces todos los Equipos Scrum deberán seguirla. Caso contrario el Equipo de Desarrollo del Equipo Scrum deberá establecer una definición de "Terminado" apropiada para el incremento del producto.	Sí
Cada Incremento se integrará con los Incrementos anteriores y debe ser probado exhaustivamente para asegurar que todos los Incrementos	Sí

funcionan en conjunto.	
A medida que los Equipos Scrum maduran, se esperará que su definición de "Terminado" se amplíe para incluir criterios más rigurosos y así aumentar el rango de calidad de los productos.	Sí
Cualquier producto deberá tener una definición de "Terminado", y se convertirá en un estándar para cualquier trabajo realizado en él.	Sí

d) Técnicas de transparencia, inspección y adaptación

Son métodos gráficos que se utilizan para cumplir los valores del marco de trabajo Scrum:

Tabla 33
Técnicas de transparencia, inspección y adaptación

TÉCNICAS	DEFINICIÓN	Aplica
(Burndown) trabajo consumido	Es un diagrama que muestra de una manera gráfica el esfuerzo restante durante un Sprint, y sirve para dar seguimiento al progreso del trabajo.	Sí
(Burnup) trabajo avanzado	Es un gráfico que muestra el trabajo total que se va a realizar, indica el progreso hacia la finalización de los proyectos.	Sí
(Cumulative flow) flujo acumulado	Es un diagrama que muestra procesos importantes como la velocidad del trabajo, WIP (trabajo en proceso), y tiempos de ciclos. Sirve para hallar cuellos de botella y problemas en el proceso de desarrollo.	No

- **Trabajo consumido (Burndown)**

El gráfico debe contener lo siguiente:

- Eje X muestra los días laborables.
- Eje Y muestra el esfuerzo restante.
- Esfuerzo Ideal (-----).
- El verdadero progreso del esfuerzo (-----).

Se puede utilizar diferentes atributos en el eje "Y" como: números de elementos, unidad de tiempo (horas), puntos de historia (Story Points), esto

varía dependiendo de la experiencia de los equipos de trabajo. Se recomienda utilizar las dos primeras cuando los equipos de trabajo son jóvenes. A continuación se muestra un ejemplo del gráfico con número de elementos como unidad de medida del eje Y.

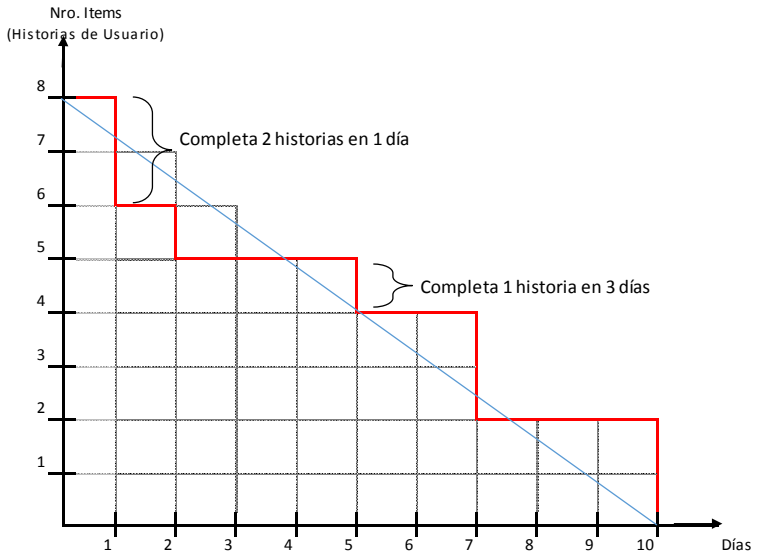


Figura 20 Gráfico Burndown

- **Trabajo avanzado (BurnUp)**

El gráfico debe contener lo siguiente:

- Eje X muestra los días laborables.
- Eje Y muestra el esfuerzo restante.
- Trabajo total (-----).
- Trabajo completado (-----).

El eje vertical es la cantidad de trabajo medida en unidades propias del proyecto que suelen ser el número de elementos, horas estimadas o puntos de la historia y en el eje horizontal es el tiempo que normalmente se mide en días. La distancia entre las dos líneas es la cantidad de trabajo restante y el proyecto se completa cuando se encuentren las dos líneas. A continuación se presenta un gráfico ejemplo.

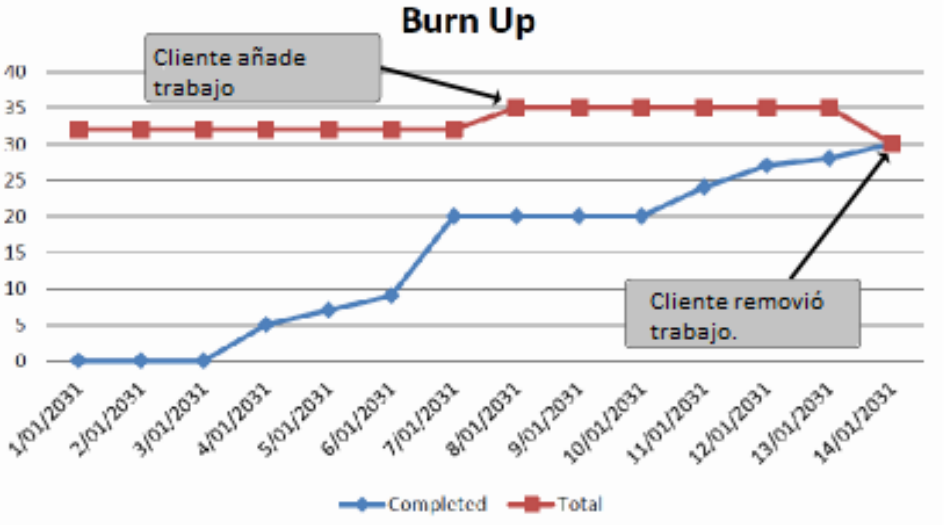


Figura 21 Gráfico Burnup

- **Flujo acumulado (Cumulative flow)**

Es un diagrama en donde se muestra procesos importantes como la velocidad del trabajo, WIP (trabajo en proceso) y los tiempos de ciclos, sirve para localizar cuellos de botella y problemas en el proceso de desarrollo, son herramientas muy útiles para realizar seguimientos y previsiones en proyectos ágiles. A continuación se presenta un gráfico ejemplo.

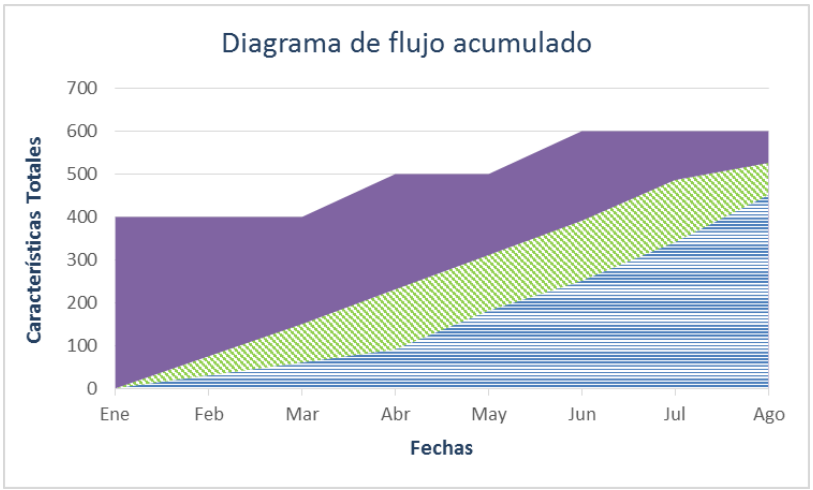


Figura 22 Gráfico Diagrama de Flujo Acumulado

3.3.2 Fases del ciclo de vida del marco de trabajo basado en Scrum

El ciclo de vida del desarrollo de Scrum se puede ver de manera clara en el siguiente gráfico, muestra los eventos, artefactos e interacción del equipo Scrum en las 3 fases del proceso.

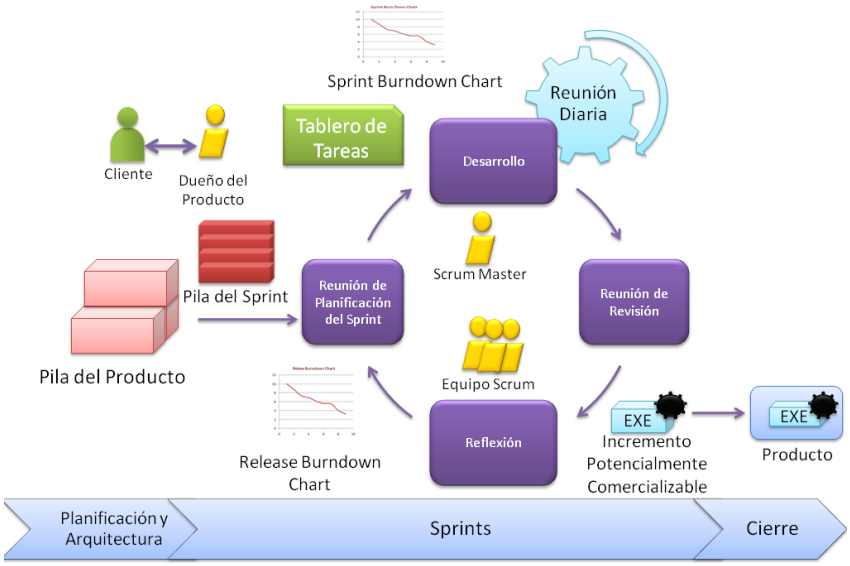


Figura 23 Fases del ciclo de vida del marco de trabajo basado en Scrum

a) Planificación y Arquitectura

En esta fase se transforma el documento de requisitos de software en una lista de elementos de producto (Product Backlog) complementado con una arquitectura tecnológica a utilizar.

- **Definición de requerimientos**

Aquí se determina los requerimientos que se desarrollarán en el proyecto, y se lo transformará en una lista de elementos del producto (Product Backlog). Los elementos son actividades que describen acciones o funcionalidades que se desea que contenga el producto de software o también podrían realizar en formato de las denominadas historias de usuario.

Tabla 34
Definición de requerimientos

Entradas de la fase			
ENTRADAS	**TIPO**	**RESPONSABLES (Roles)**	**DOCUMENTO**
Documento de especificación de requisitos de Software.	Artefacto Externo.	Dueño del producto.	(ERS_CasoPráctico).doc
Salidas de la fase			
SALIDAS	**TIPO**	**RESPONSABLES (Roles)**	**DOCUMENTO**
Lista de producto.	Artefacto Scrum.	Dueño del producto.	ProductBacklog (Herramienta TFS)

b) Implementación de trabajos - Sprints

Esta fase contiene sub fases que son la planificación, el desarrollo, la revisión y la reflexión del trabajo que se realizará en un Sprint. El objetivo es realizar todo el proceso de desarrollo de software pero solo para un incremento del producto terminado. A continuación se detalla las entradas y salidas de las sub fases.

- **Planificación de trabajos**

En esta sub fase se debe establecer el listado de elementos de requerimientos para desarrollar el proyecto, se debe escoger los elementos de lista de producto que se van a realizar en el Sprint.

Tabla 35
Planificación de trabajos

Entradas de la fase			
NOMBRE	**TIPO**	**RESPONSABLES (Roles)**	**DOCUMENTO**
Reunión de Planificación de Sprint.	Evento Scrum.	Todos los roles.	NA.

CONTINÚA

Salidas de la fase			
NOMBRE	**TIPO**	**RESPONSABLES (Roles)**	**DOCUMENTO**
Lista de producto del Sprint.	Artefacto Scrum.	Todos los roles.	Sprint Backlog. (Herramienta TFS)
Planificación del trabajo.	Artefacto propio.	Todos los roles.	Sprint Backlog. (Herramienta TFS)

- **Desarrollo de software y seguimiento de trabajos**

En esta sub fase se realiza todas las tareas que corresponden al desarrollo de software en función de la lista de productos del Sprint.

Tabla 36
Desarrollo de software y seguimiento de trabajos

Entradas de la fase			
NOMBRE	**TIPO**	**RESPONSABLES (Roles)**	**DOCUMENTO**
Reunión de Scrum Diario.	Evento Scrum.	Scrum Master. Equipo de desarrollo.	NA.
Codificación de Software.	NA.	Equipo de desarrollo.	NA.
Control de versiones.	NA.	Equipo de desarrollo.	NA.
Pruebas de funcionalidad.	NA.	Equipo de desarrollo.	NA.
Salidas de la fase			
NOMBRE	**TIPO**	**RESPONSABLES (Roles)**	**DOCUMENTO**
Incremento del producto terminado.	Artefacto Scrum.	Equipo de desarrollo.	NA.
Burndown.	Técnica Scrum	Scrum Master. Equipo de desarrollo.	Gráfica Burndown.
BurnUp.	Técnica Scrum	Scrum Master. Equipo de desarrollo.	Gráfica BurnUp.

c) Revisión y retrospectiva de trabajos

- **Retrospectiva de trabajos**

Se lo realiza al final del Sprint con la finalidad de presentar y revisar las funcionalidades del incremento de producto terminado por parte del equipo Scrum.

Tabla 37
Revisión de trabajos

Entradas de la fase			
NOMBRE	**TIPO**	**RESPONSABLES (Roles)**	**DOCUMENTO**
Reunión de Revisión del Sprint.	Evento Scrum	Todos los roles.	NA.
Salidas de la fase			
NOMBRE	**TIPO**	**RESPONSABLES (Roles)**	**DOCUMENTO**
Modificaciones de la Lista de Producto.	Artefacto Scrum	Dueño del producto.	NA.

- **Retrospectiva de trabajos**

Esta sub fase presenta la oportunidad para que el Equipo Scrum pueda inspeccionarse a sí mismo y crear un plan de mejoras que sean abordadas durante el siguiente Sprint.

Tabla 38
Retrospectiva de trabajos

Entradas de la fase			
NOMBRE	**TIPO**	**RESPONSABLES (Roles)**	**DOCUMENTO**
Reunión de Retrospectiva de Sprint.	Evento Scrum	Todos los roles.	NA.
Salidas de la fase			

CONTINÚA

NOMBRE	TIPO	RESPONSABLES (Roles)	DOCUMENTO
Plan de mejoras.	Artefacto Propio	Scrum Master.	PlanMejoras.docx (Elaboración propia)

d) Despliegue del incremento del producto.

Esta fase es donde el incremento del producto ha sido probado y tiene la aprobación para su liberación en el entorno de producción y pueda ser utilizado por los usuarios finales, previó a esto se debe hacer una capacitación de uso, si fuese necesario.

Despliegue Total del producto terminado: Es la liberación total del producto terminado en el ambiente producción, se debe complementar con la entrega formal del producto.

Tabla 39
Despliegue del incremento del producto

Entradas de la fase			
NOMBRE	**TIPO**	**RESPONSABLES (Roles)**	**DOCUMENTO**
Incremento del producto terminado. (En entorno producción)	Artefacto Scrum	Equipo de desarrollo.	NA.
Salidas de la fase			
NOMBRE	**TIPO**	**RESPONSABLES (Roles)**	**DOCUMENTO**
Plan de capacitaciones.	Artefacto Propio	Scrum Master.	PlanCapacitaciones.docx (Elaboración propia)
Acta entrega recepción. (Cuando se entregue el total del producto terminado).	Artefacto Propio	Scrum Master.	ActaEntregaRecepción.docx (Elaboración propia)

Una vez que se haya realizado la elección de todas las características que formarán parte del marco de trabajo de desarrollo de la empresa Farmaenlace Cía. Ltda., se elaborará el documento anexo marco_trabajo_farmaenlace.docx, el cual se socializará a los involucrados del negocio y principalmente al equipo de desarrollo para que lo utilicen como una guía que regirá la manera de trabajar en los proyectos de desarrollo de software.

3.4 Marco de trabajo Farmaenlace implementado con la herramienta ALM

El objetivo de este apartado es estructurar y definir como se implementa y automatiza la metodología empresarial definida en el capítulo anterior bajo la herramienta “ALM”, que permitirá administrar el ciclo de vida de las aplicaciones de software.

Luego de haber realizado el estudio comparativo entre las herramientas ALM que se desarrolló en el apartado 3.2, se determinó que la suite de herramientas Team Foundation Server (TFS), es la más recomendable para implementarse en este proyecto. TFS y el conjunto de herramientas con las que se integra, gestionan la planificación del trabajo y su seguimiento, el control de versiones, modelamiento, compilaciones, pruebas y reportes en las diferentes fases del ciclo de vida de las aplicaciones de software. A continuación se establecen las etapas de implementación del marco de trabajo FARMAENLACE bajo las herramientas ALM - TFS.

3.4.1 Arquitectura de la herramienta ALM - Team Foundation Server (TFS)

En esta etapa se establece la arquitectura donde muestra la integración de las tecnologías que cubren las necesidades planteadas en el marco de trabajo establecido. Se verifica las herramientas que deben ser instaladas y ejecutadas considerando tres factores importantes: infraestructura

tecnológica actual, necesidades del marco de trabajo y la orientación técnica de la herramienta planteada. Las actividades que se realizan dentro de la configuración de la arquitectura TFS son las siguientes:

- Diseño de la arquitectura tecnológica basada en los requerimientos del marco de trabajo y el ambiente tecnológico actual.
- Alineación con la tecnología definida en la estrategia corporativa.
- Análisis de la tecnología que mejor se adapte a las necesidades de los clientes.

Documentos entregables.

- Diseño de la arquitectura tecnológica de la herramienta ALM.

3.4.2 Configuración de la herramienta Team Foundation Server (TFS)

En esta etapa se detalla la configuración de la herramienta antes de ser utilizada, el objetivo es cumplir con los estándares de calidad para el buen funcionamiento de la gestión de la fabricación de software en TFS. Las actividades que se realizan en esta etapa son las siguientes:

- Instalación de la herramienta TFS.
- Creación de un Proyecto de Equipo (Team Project) en TFS.
- Agregación de Miembros al Proyecto de Equipo.
- Agregación de Código Fuente al Control de Versiones.
- Configuración de Integración Continua.

Documentos entregables

- Manual técnico de configuración de TFS.

3.4.3 Planificación y seguimiento de proyectos de Equipo con TFS

En esta etapa a través de TFS se proporciona una herramienta de planeación centralizada, en la cual se realiza el seguimiento de la información, organización del código fuente, de las compilaciones y de las

pruebas en las aplicaciones. De igual manera los proyectos de equipo (Team Project) proporcionan un punto central donde el equipo comparte las actividades necesarias para desarrollar software o un producto concreto. A continuación se detallan los pasos a seguir para cumplir con dicha etapa:

- Definición de las herramientas para administrar el ciclo de vida de las aplicaciones de software.
- Definición de los proyectos de equipo, equipos y guía de procesos.
- Definición de los clientes de TFS.
- Definición de los elementos de trabajo, consultas y alertas.
- Estructuración de los documentos e informes.

Documentos entregables.

- Manual de planificación y seguimiento de proyectos con TFS.

3.4.4 Modelamiento de aplicaciones con TFS

En esta etapa se define el modelamiento de las aplicaciones de software, el objetivo es asegurar el desarrollo de las aplicaciones conforme a los requerimientos del usuario. Una de las herramientas que se puede utilizar y se integra con TFS, es Visual Studio en su versión Ultimate la cual proporciona las funcionalidades de entender el código y modelar aplicaciones. Además con la utilidad de mostrar el código, se puede visualizar de una manera más fácil para entender su estructura, relaciones y comportamiento. Para la empresa Farmaenlace Cía Ltda., se requiere modelar únicamente el diagrama de clases para reflejar las reglas de negocio que tendrá la aplicación a desarrollarse. Las actividades que se realizan en esta etapa son las siguientes:

- Desarrollo de modelos para el diseño de software.
- Creación de modelos de los requisitos de los usuarios.
- Modelamiento de la arquitectura de una aplicación de Software.

- Visualización y comprensión del código.
- Validación del proyecto durante el desarrollo.
- Administración de modelos y gráficos con control de versiones.
- Generación y configuración de aplicaciones a partir de modelos.
- Ampliación de modelos y diagramas UML.
- SDK de modelado y virtualización - Lenguajes específicos de dominio.

Documentos entregables

- Manual del modelado del diagrama de clases de las aplicaciones.

3.4.5 Control de versiones con TFS

En esta etapa se maneja los cambios que se realizan sobre los elementos de algún producto o configuración ya sea una versión, edición o revisión del estado del producto en un momento determinado de su desarrollo o modificación. Dichos cambios son realizados bajo la herramienta de Control de Versiones TFS que está diseñado para hacer este trabajo de manera simple y silenciosa, que mejora la productividad personal, la recuperación de información cuando hay problemas y facilita la colaboración y transparencia del trabajo en equipo. Las actividades que se realizarán en esta etapa son las siguientes:

- Definición del equipo de desarrollo.
- Desarrollo de la aplicación con código sometido a control de versiones.
- Aporte del trabajo al equipo.
- Visualización y administración de versiones pasadas.
- Comparación de carpetas y archivos.
- Resolver conflictos de archivos.
- Trabajar con bloqueos del control de versiones.
- Suspender el trabajo.

Documentos entregables.

- Manual de control de versiones con TFS.

3.4.6 Compilación de aplicaciones con TFS.

En esta etapa se transforma el código funcional a código de máquina para su ejecución. Para este proceso se construye las aplicaciones de software mediante un sistema de compilación integrado, en donde el equipo de trabajo crea y administra procesos de compilación automática y establece una estrategia de integración continua o comprobaciones rigurosas de calidad durante el desarrollo de software. Las actividades que se realizarán en esta etapa son las siguientes:

- Implementar un sistema de compilación.
- Definir el proceso de compilación.
- Ejecutar, supervisar y administrar las compilaciones.
- Diagnóstico de problemas de compilación.

Documentos entregables.

- Manual de compilación de aplicaciones con TFS.

3.4.7 Pruebas de aplicaciones con TFS.

En esta etapa se define las pruebas en función de los trabajos de los proyectos de equipo, se proporciona información objetiva e independiente sobre la calidad del producto, se accede a las funcionalidades de ejecutar las pruebas de rendimiento o esfuerzo de manera automática y se administra pruebas sistemáticamente para que el equipo conozca la calidad del software en cualquier momento, con lo cual mejora la productividad en la fase de pruebas del ciclo de vida de desarrollo de las aplicaciones, este proceso en el medio también se lo conoce con el nombre de Testing. Las actividades que se pueden realizar en esta sección son las siguientes:

- Ejecución de trabajos de pruebas de actualización de versiones anteriores de Visual Studio.
- Administración de pruebas de Microsoft.
- Administración de herramientas de pruebas en Visual Studio.
- Ejecución de pruebas o recopilación de datos de forma remota.

Documentos entregables.

- Manual de pruebas de aplicación con TFS.

CAPÍTULO IV

IMPLEMENTACIÓN DE UN MARCO DE TRABAJO, CON UNA HERRAMIENTA ALM, PARA MEJORAR LA GESTIÓN DEL PROCESO DE FABRICACIÓN DE SOFTWARE

4. Introducción

En este capítulo se explica la implementación del marco de trabajo con la herramienta ALM (Team Foundation Server), que permitió mejorar la gestión de la fabricación de software en el área de análisis y desarrollo de la empresa Farmaenlace Cía. Ltda., la implementación se la hizo bajo los lineamientos desarrollados y establecidos en el capítulo 3. Se comenzará explicando la configuración y arquitectura de la herramienta TFS, luego se implementará las fases del ciclo de vida del marco de trabajo con la herramienta TFS, luego el despliegue del incremento del producto. En la siguiente ilustración se muestra la estructura establecida para este capítulo.

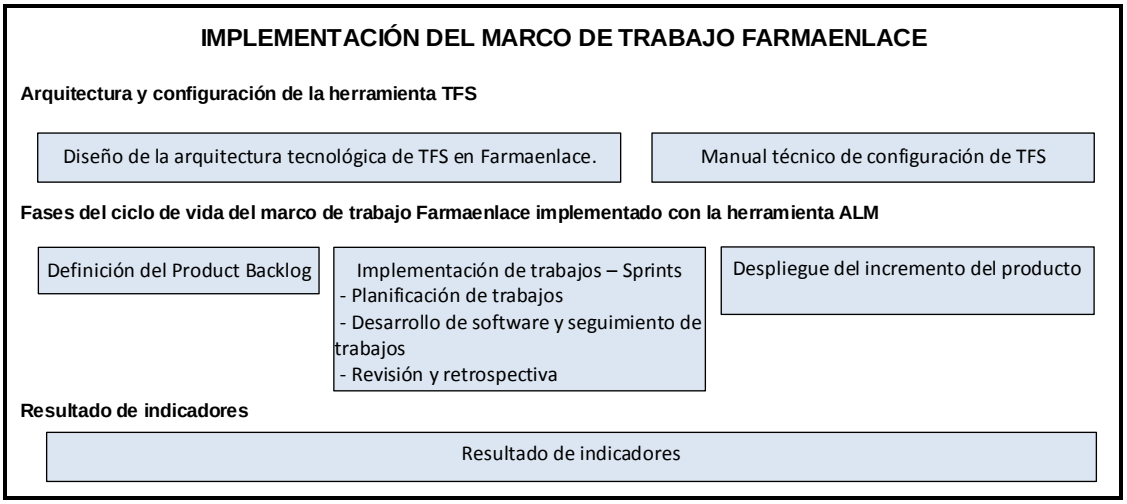


Figura 24 Estructura de la Implementación del marco de trabajo Farmaenlace

Para demostrar la implementación se consideró como caso práctico el desarrollo del sistema informático “Gestión de Comisiones de Franquiciados”, realizado por 2 miembros del equipo de desarrollo.

4.1 Arquitectura y configuración de la herramienta TFS

A continuación se muestra la arquitectura y la configuración de la herramienta de administración del ciclo de vida de las aplicaciones TFS que se utilizó en la gestión de fabricación de software, de acuerdo a lo establecido en la sección 3.4.1 (Arquitectura de la herramienta ALM – Team Foundation Server) y 3.4.2 (Configuración de la herramienta Team Foundation Server).

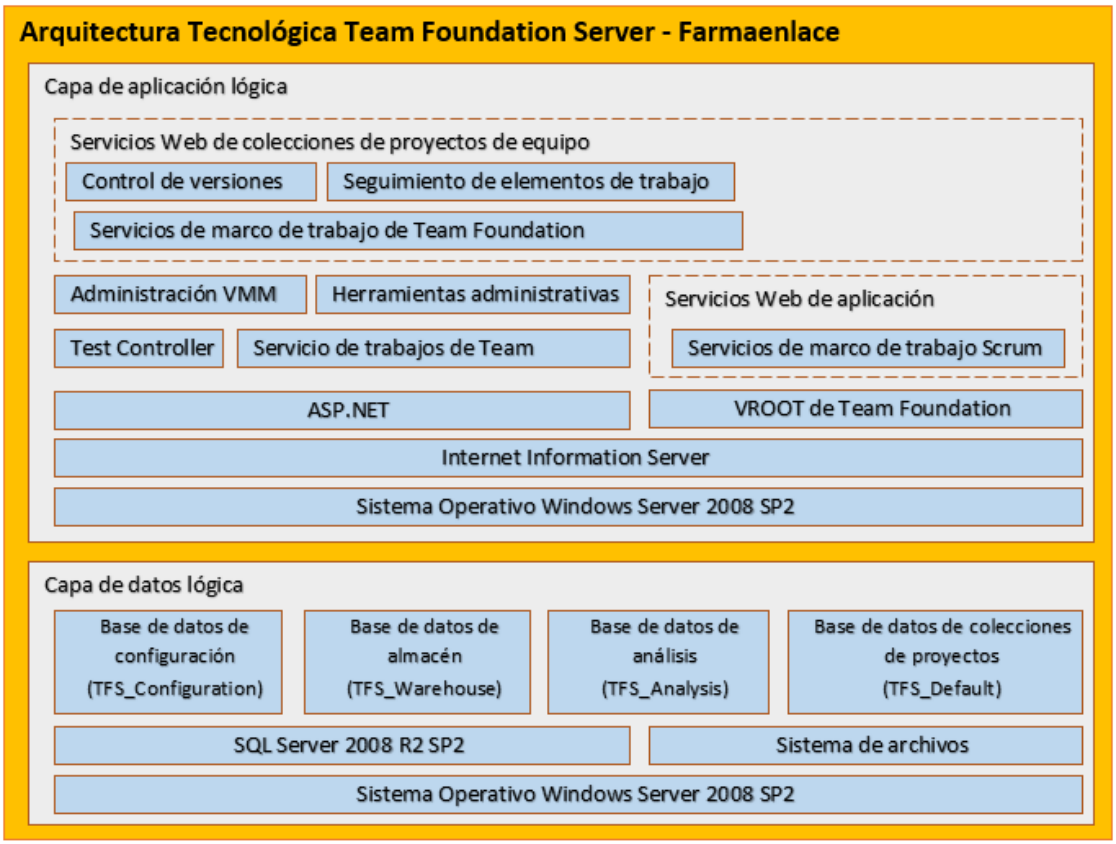


Figura 25 Diseño de la arquitectura tecnológica TFS en Farmaenlace Cía. Ltda.

- **Configuración de la herramienta TFS**

La configuración inició con la instalación de la herramienta TFS 2012, y con ello todos los pre-requisitos necesarios para su funcionamiento, además la configuración base para la creación de un proyecto de equipo (Team Project). El manual técnico de configuración de TFS está disponible en el Anexo B, el mismo que contiene los siguientes puntos:

- Instalación de la herramienta TFS.
- Creación de un Proyecto de Equipo (Team Project) en TFS.
- Agregación de Miembros al Proyecto de Equipo.
- Agregación de Código Fuente al Control de Versiones.
- Configuración de Integración Continua.

4.2 Fases del ciclo de vida del marco de trabajo Farmaenlace implementado con la herramienta ALM

En esta sección se explicará cómo se implementó los eventos, artefactos, técnicas del marco de trabajo en cada fase del ciclo vida del desarrollo de las aplicaciones mediante la herramienta TFS.

4.2.1 Definición del Product Backlog

Aquí se determinó los requerimientos con una planificación y estimación general de lo que se va desarrollar en el proyecto, mediante la pila de elementos del producto (Backlog), que es una lista de elementos o actividades que describen funcionalidades que se desea que el sistema contenga.

- **Definición de requerimientos**

Para definir los requisitos se transformó el documento ERS (Especificación de Requisitos de Software) (Anexo C) en elementos que abarcan funcionalidades más pequeñas las cuales se denominaron actividades o elementos de la lista de producto del proyecto (Backlog).

De acuerdo a la sección 3.3.2.1 (Planificación y arquitectura), la entrada de esta fase es el documento de especificación de requisitos de software, para lo cual se utilizó el estándar de la IEEE830, en el documento se detalla el propósito, alcance, personal involucrado, descripción del proyecto, requerimientos funcionales y no funcionales del caso práctico.

Para obtener la salida de esta fase se realizó la revisión del documento de requerimientos para convertirlo en una lista del producto "Product Backlog", la cual se registró en la herramienta TFS. La manera de registrar en TFS se muestra en el Anexo D (Manual de planificación y seguimiento de proyectos con TFS), a continuación se muestra la lista del producto del caso práctico realizada en la herramienta TFS.

Backlog items

New | Create query | Column options

Type: Product Backlog Item

Title: Add

Order	Work Item Type	Title	State	Effort	Value Area
1	Product Backlog Item	Análisis del Documento de requerimientos y de las bases de datos a utilizar	New	24	Business
2	Product Backlog Item	Estructuración del Proyecto	New	16	Business
3	Product Backlog Item	Diseño, contrucción de la Base de Datos	New	24	Business
4	Product Backlog Item	Corrección y rediseño de la Base de Datos	New	18	Business
5	Product Backlog Item	Administración de Maestros - Estructura de aplicación (menú).	New	20	Business
6	Product Backlog Item	Administración de Maestros - Administración de Períodos	New	20	Business
7	Product Backlog Item	Administración de Maestros - Administración de Farmacias	New	22	Business
8	Product Backlog Item	Administración de Maestros - Administración de Franquiciados y Asociados	New	80	Business
9	Product Backlog Item	Administración de Rubros - Creación y Parametrización de Rubros	New	64	Business
10	Product Backlog Item	Administración de Rubros - Aplicación de Rubros a Farmacias	New	56	Business
11	Product Backlog Item	Administración de Rubros - Prioridad de Rubros	New	32	Business
12	Product Backlog Item	Administración de Rubros - Aplicación de filtros en vistas para Rubros a nivel de ...	New	48	Business
13	Product Backlog Item	Administración de Rubros - Traspaso de Conocimientos.	New	32	Business
14	Product Backlog Item	Administración de Rubros - Validaciones de administración de rubros.	New	24	Business
15	Product Backlog Item	Administración de Rubros - Parametrización de Farmacias, fechas y porcentajes ...	New	40	Business
16	Product Backlog Item	Administración de Rubros - Almacenamiento de datos temporal de Rubros para ...	New	16	Business
17	Product Backlog Item	Administración de Rubros - Cálculos de Rubros	New	56	Business
18	Product Backlog Item	Administración de Rubros - Almacenamiento de resultado de cálculos	New	16	Business

CONTINÚA

19	Product Backlog Item	Administración de Rubros - Validaciones con fechas globales para almacenamie...	New	16	Business
20	Product Backlog Item	Administración de Rubros - Almacenamiento de datos temporal de Rubros para ...	New	32	Business
21	Product Backlog Item	Cálculos - Cálculos de Rubros	New	56	Business
22	Product Backlog Item	Cálculos - Almacenamiento de resultado del cálculo de Rubros	New	8	Business
23	Product Backlog Item	Cálculos - Cálculo y almacenamiento de ventas globles	New	16	Business
24	Product Backlog Item	Cálculos - Reunión para revisión de cálculos con Auditoria y Operaciones.	New	2	Business
25	Product Backlog Item	Reportes - Reporte de Cálculos por farmacia (Comisiones Globales y Rubros)	New	16	Business
26	Product Backlog Item	Reportes - Reporte de Base General (Resultado Globales y Rubros)	New	8	Business
27	Product Backlog Item	Reportes - Corrección en cálculos	New	24	Business
28	Product Backlog Item	Comisiones de PAFs - Administración de farmacias, aumentar los porcentajes de ...	New	16	Business
29	Product Backlog Item	Comisiones de PAFs - Administración de Rubros con listas de precios	New	8	Business
30	Product Backlog Item	Comisiones de PAFs - Administración de Rubros x farmacias de listas de precios.	New	8	Business
31	Product Backlog Item	Comisiones de PAFs - Carga de información para cálulos con listas de precios.	New	8	Business
32	Product Backlog Item	Comisiones de PAFs - Cálculos de Rubros por listas de precios.	New	48	Business
33	Product Backlog Item	Descuentos de facturas y Pagos - Tipos de Descuentos	New	24	Business
34	Product Backlog Item	Descuentos de facturas y Pagos - Administración de descuentos programados	New	16	Business

Figura 26 Lista del producto (Product Backlog)

4.2.2 Implementación de trabajos - Sprints

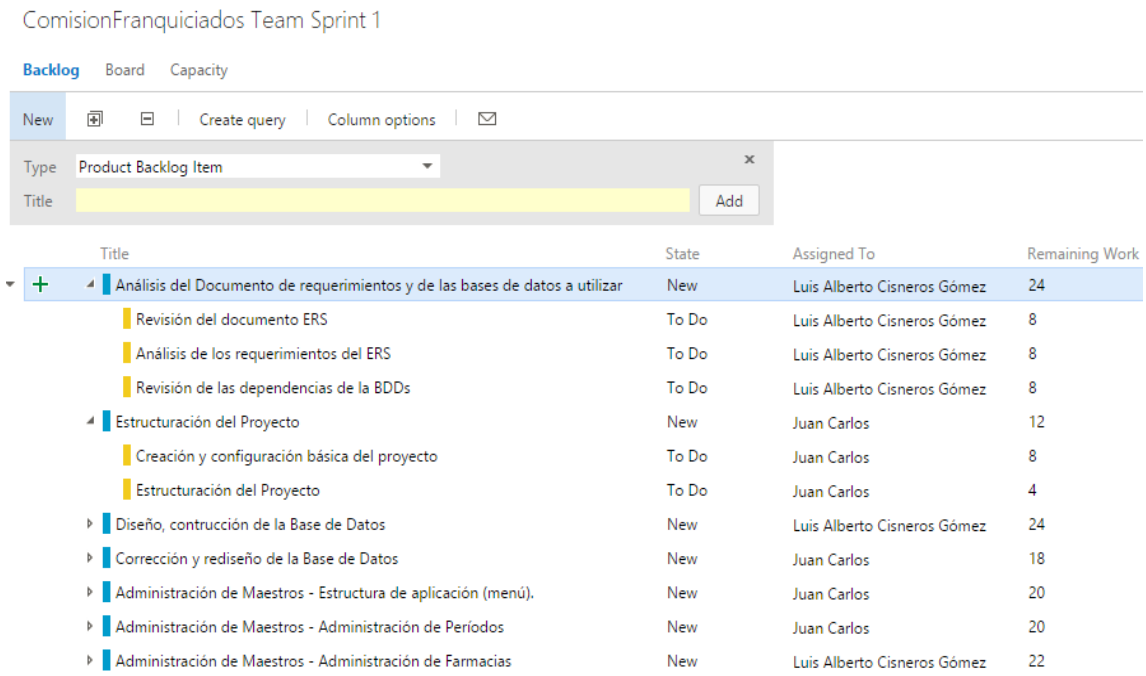
Esta fase fue muy importante dentro del ciclo de la vida de las aplicaciones ya que aquí es donde se realizó todo el proceso de desarrollo para obtener el incremento de producto de software. Esta fase se repitió de manera iterativa e incremental mediante los Sprints, hasta completar el producto terminado de software. A continuación se detalla cada una de las sub fases.

a) Planificación de trabajos

En la planificación se estableció el trabajo que realizó el equipo de desarrollo durante el Sprint, para lo cual se hace referencia a la sección 3.3.2.2 (Implementación de trabajos - Sprints), que detalla las entradas y salidas que tiene esta sub fase.

La entrada es la Reunión de Planificación del Sprint, es donde el equipo de trabajo escogió los elementos de la lista de producto que realizará durante el Sprint, y también definió la manera de cómo realizar los trabajos, esto se lo hizo bajo los lineamientos del marco de trabajo especificados en la sección 3.3.1.2 (Eventos basados en Scrum).

Las salidas de la fase son los elementos escogidos de la lista de producto que se van a realizar en un Sprint, definido como “Sprint Backlog”, la otra salida es la planificación de cómo hacer estos elementos, para esto cada elemento de la lista se lo debe detallar o descomponer en tareas de no más de 8 horas, este tiempo lo establece el equipo de desarrollo basándose en la sección 3.3.1.3 (Artefactos basados en Scrum). A continuación se presenta como ejemplo los elementos del Sprint Backlog del Sprint1 del caso práctico.



Title	State	Assigned To	Remaining Work
Análisis del Documento de requerimientos y de las bases de datos a utilizar	New	Luis Alberto Cisneros Gómez	24
Revisión del documento ERS	To Do	Luis Alberto Cisneros Gómez	8
Análisis de los requerimientos del ERS	To Do	Luis Alberto Cisneros Gómez	8
Revisión de las dependencias de la BDDs	To Do	Luis Alberto Cisneros Gómez	8
Estructuración del Proyecto	New	Juan Carlos	12
Creación y configuración básica del proyecto	To Do	Juan Carlos	8
Estructuración del Proyecto	To Do	Juan Carlos	4
Diseño, contrucción de la Base de Datos	New	Luis Alberto Cisneros Gómez	24
Corrección y rediseño de la Base de Datos	New	Juan Carlos	18
Administración de Maestros - Estructura de aplicación (menú).	New	Juan Carlos	20
Administración de Maestros - Administración de Períodos	New	Juan Carlos	20
Administración de Maestros - Administración de Farmacias	New	Luis Alberto Cisneros Gómez	22

Figura 27 Sprint Backlog del caso práctico

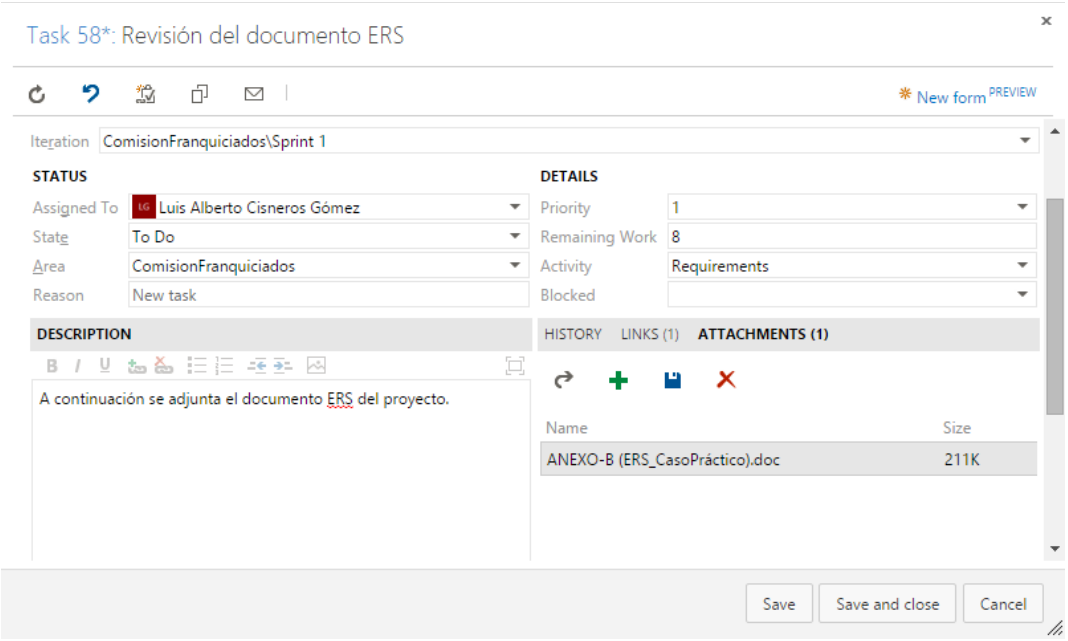


Figura 28 Ejemplo de una Tarea del Sprint1

b) Desarrollo de software y seguimiento de trabajos

En esta sección se estableció los eventos, técnicas y fases necesarias para el desarrollo del software, de acuerdo la sección 3.3.2.2 (Implementación de trabajos – Sprints), a continuación se especifica las entradas y salidas que generó esta sub fase.

- **Entradas**

El Scrum Diario, este evento proporcionó una reunión diaria para inspeccionar el avance y la proyección del trabajo que se realizó en el lapso de un día. Los lineamientos que se siguieron fueron del marco de trabajo especificado en la sección 3.3.1.2 (Eventos basados en Scrum). La aplicación del evento en el caso práctico ayudó mucho para localizar problemas funcionales, lógicos, técnicos, y humanos, los cuales se pudieron solucionar de una manera ágil con la ayuda de la gestión realizada por el Scrum Master.

La codificación del software, consistió en realizar las tareas necesarias de programación para llevar a código fuente, todo lo planificado y diseñado en la fase anterior. Esto lo realizó el equipo de programación, siguiendo las indicaciones de los requisitos funcionales y no funcionales del ERS, el diseño y la planificación del trabajo del Sprint. Para el caso práctico la codificación se la realizó en el lenguaje de programación C# mediante la herramienta Microsoft Visual Studio Ultimate 2012 y se conectó al proyecto TFS "Comisión de franquiciados" siguiendo las indicaciones del ANEXO B, el código fuente del sistema del caso práctico se lo deja en el CD de anexos, en la sección /Anexos/CasoPrático/CódigoFuente/appFranquiciado.

Control de versiones, en esta etapa se manejó los cambios codificados de los elementos del Sprint Backlog, los cuales fueron realizados en la herramienta de Control de Versiones TFS. Lo sobresaliente de esta etapa es que se integra la codificación del proyecto, la cual mediante el control de versiones da de baja las tareas de la planificación de una manera integrada, para verificar de una manera detallada el proceso se puede revisar el Manual de control de versiones con TFS (Anexo E). A continuación para el caso práctico se muestra de una manera resumida como ocurre la integración de lo antes mencionado.

Pasos:

- Realizar cambios en el código fuente. (Microsoft Visual Studio)
- Escoger cambios pendientes en el Team Explorer.
- Proteger los cambios.
    - Poner un comentario.
    - Escoger una tarea del Sprint Backlog. (Aquí se integra con la planificación).
    - Presionar Proteger.

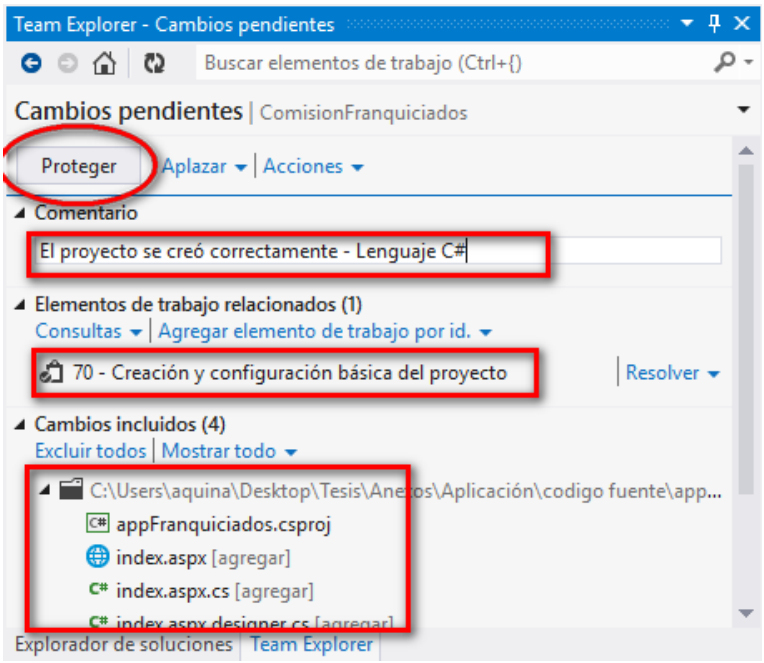


Figura 29 Proteger cambios en el Control de Versiones

Para verificar el efecto del control de versiones, se muestra en el TFS las siguientes pantallas que muestra el movimiento de las tareas del Sprint Backlog.

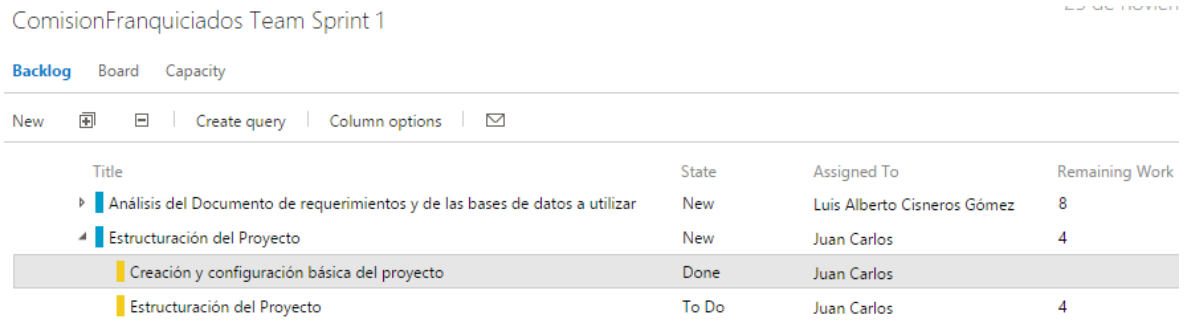


Figura 30 Proteger cambios en el Control de Versiones

El código fuente se lo puede revisar en la siguiente pantalla de TFS.

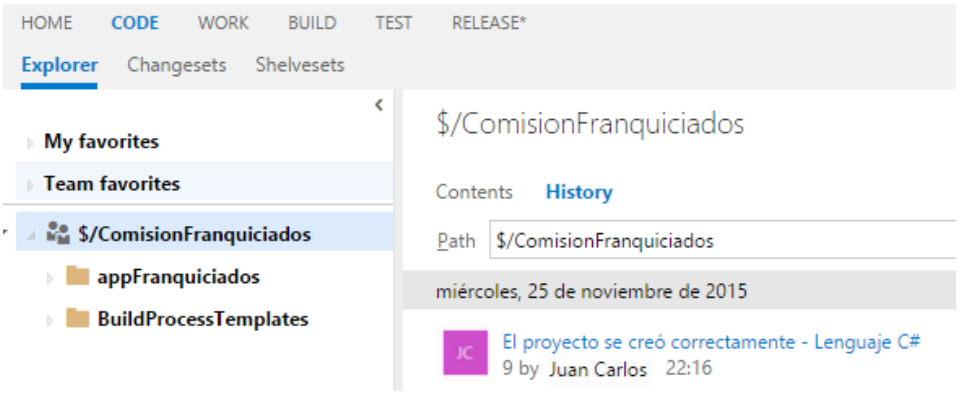


Figura 31 Historial de Control de Versiones

Las pruebas de funcionalidad, lo realizó cada miembro del equipo de desarrollo en función a la codificación de los elementos desarrollos del Sprint Backlog. Al ejecutar las pruebas se pudo revisar el estado de la codificación en cualquier momento, lo que mejoró la productividad y la transferencia en esta fase y proporcionó información objetiva e independiente sobre la calidad del producto. Dicho proceso se encuentra establecido en el Manual de administración de pruebas con Microsoft Test Manager de Visual Studio (Anexo F). A continuación para el caso práctico se muestra un ejemplo de cómo se crea y ejecuta una prueba manual al trabajo realizado código desarrollado.

Pasos: (En Microsoft Test Manager)

- Crear el plan de pruebas. (Anexo F)
- Crear casos de prueba, asignado al responsable y Sprint correspondiente.
    - Crear pasos con la acción y respuesta esperada.
    - Asignar a un elemento del backlog.
- Ejecutar casos de prueba.
- Resumen de casos de prueba.

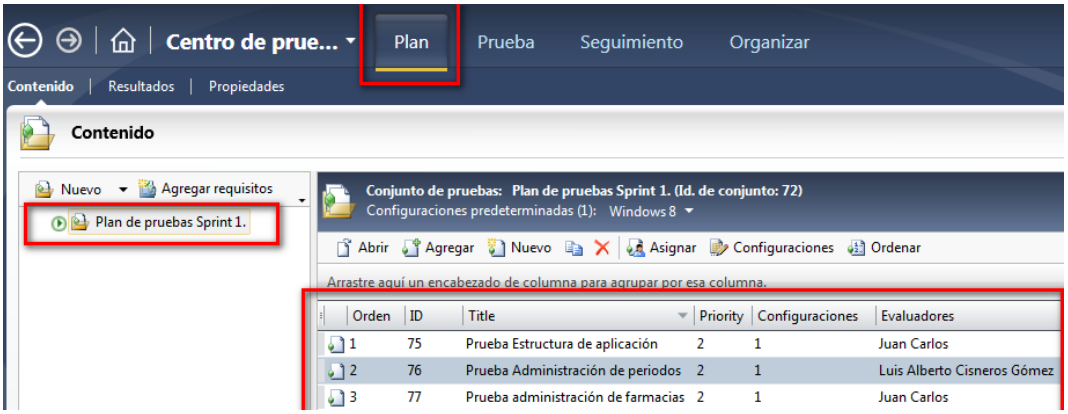


Figura 32 Plan de pruebas – Sprint 1

Figura 33 Crear Caso de prueba – Pasos

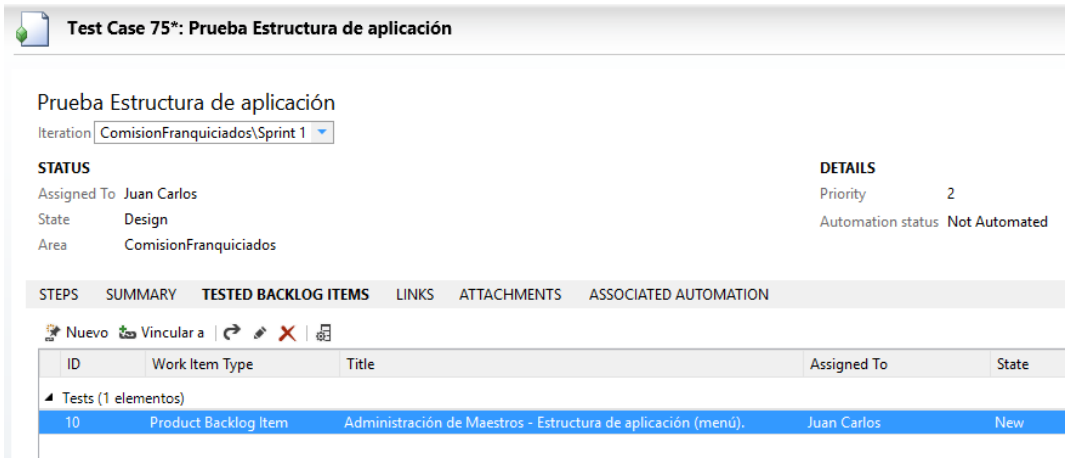


Figura 34 Caso de prueba – referencia a elemento del Backlog

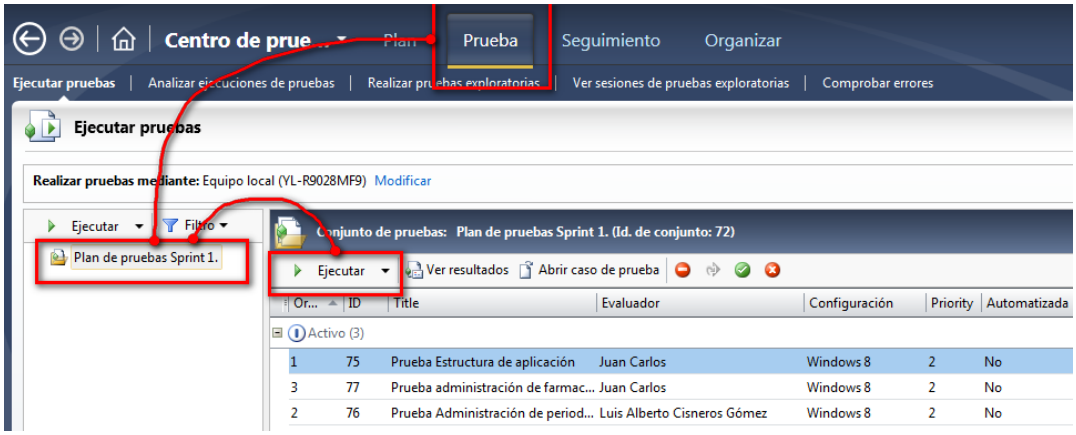


Figura 35 Caso de prueba – Ejecución

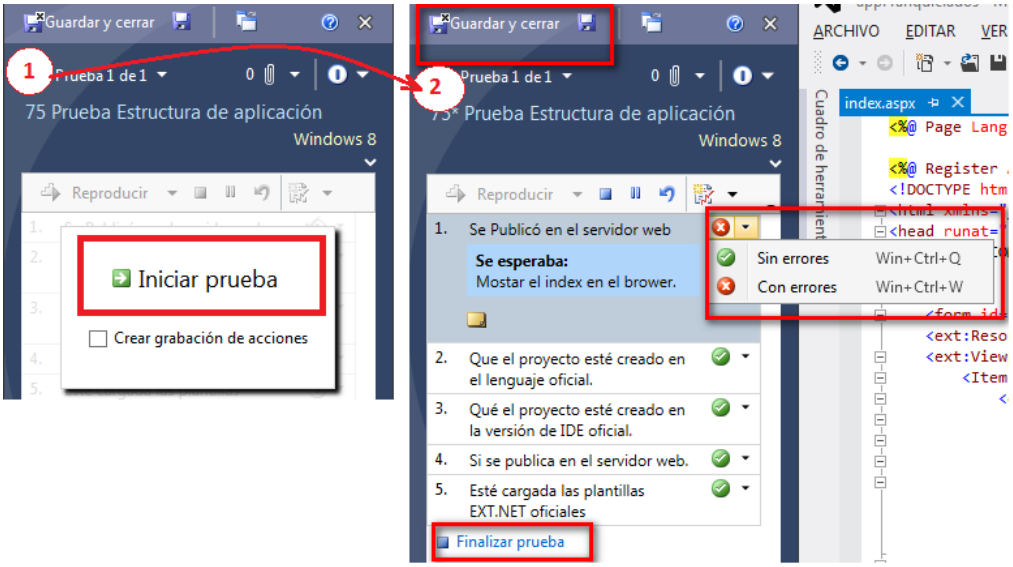


Figura 36 Caso de prueba – Ejecución

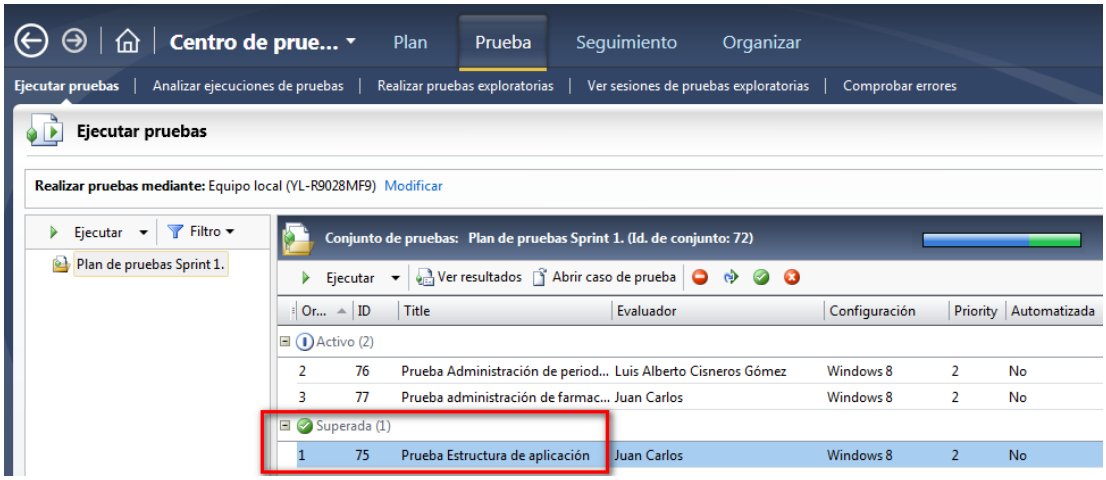


Figura 37 Caso de prueba – Estado

- **Salidas**

Incremento del producto terminado, se consideró al grupo de elementos completados y probados de la Lista de Producto de cada Sprint. Estos incrementos se los juntó a los anteriores hasta que formó el producto terminado para el ejemplo el caso práctico.

Burndown (Trabajo consumido), se lo realizó utilizando el valor de días laborables en el eje X y el valor de horas estimadas en el eje Y, de esta manera fue una técnica muy útil para el seguimiento del trabajo, la cual ayudó a identificar problemas y retrasos en el transcurso del Sprint, y así se pudo aplicar correctivos de una manera eficiente, la manera de armar el gráfico se lo detalla en la sección 3.3.1.4 (Técnicas de transparencia, inspección y adaptación). A continuación para el caso práctico se muestra como ejemplo el gráfico del Burndown en la mitad del desarrollo del Sprint 1.

Figura 38 BurnDown del Sprint 1

Burnup (Trabajo avanzado), este gráfico se realizó asignando al eje Y la cantidad de horas estimadas del proyecto y en el eje X el tiempo medido en días. Esta técnica fue muy útil para la proyección, seguimiento y transparencia del trabajo, la cual ayudó mucho a la toma de decisiones de parte de la gerencia en la asignación de recursos, administración y priorización de nuevos proyectos, la manera de armar el gráfico se lo detalla

en la sección 3.3.1.4 (Técnicas de transparencia, inspección y adaptación). A continuación se muestra el gráfico Burnup del caso práctico.

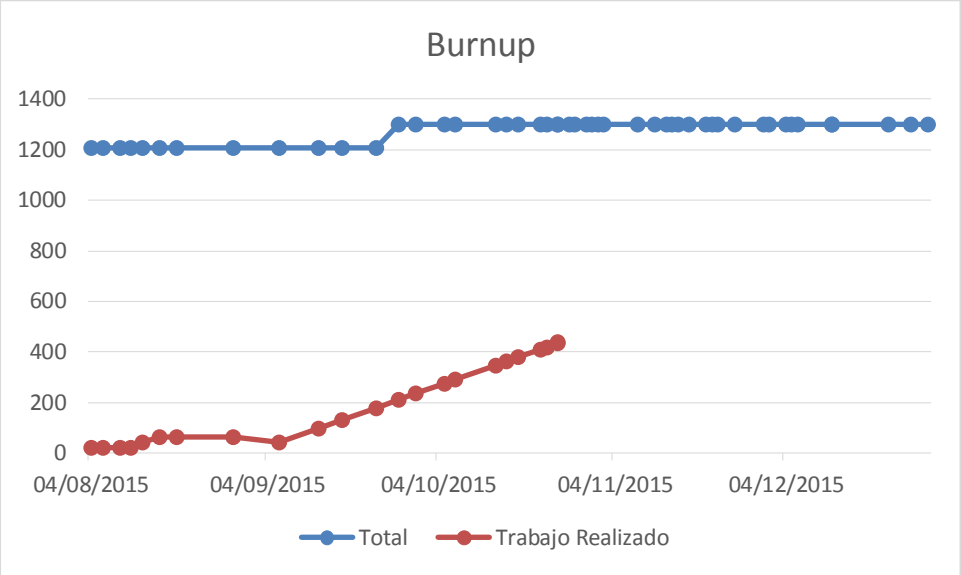


Figura 39 Burnup del caso práctico

c) Revisión y retrospectiva de trabajos

- **Revisión de trabajos**

Como entrada de esta fase se utilizó a la "Reunión de Revisión del Sprint" que se realizó al final de cada Sprint en donde se presentó y revisó las funcionalidades del incremento de producto; por lo general se lo hizo con el dueño del producto y el equipo de desarrollo. Este evento fue muy importante ya que es donde se pudo receptar la retroalimentación del dueño del producto, y así poder validar si el trabajo realizado estaba conforme a lo esperado, lo cual dio como resultado la salida de la fase, que fue el establecer modificaciones o nuevas funcionalidades en los elementos del Backlog. En la sección 3.3.1.2 (Eventos basados en Scrum) se puede observar a detalle las características de la reunión de revisión de Sprint. A continuación para el caso práctico se muestra un ejemplo de modificación de un elemento del Product Backlog, y se describe porqué se lo hizo.

Tabla 40
Ejemplo de modificación del Backlog

Elementos del Backlog antes de la reunión de revisión		
ELEMENTOS	**Esfuerzo**	**MOTIVOS**
Administración de Maestros – Administración de Franquiciados y Asociados.	80	
Elementos del Backlog después de la reunión de revisión		
ELEMENTOS	**Esfuerzo**	**MOTIVOS**
Administración de Maestros – Administración de Franquiciados y Asociados.	100	El motivo de subir el esfuerzo en 20, es porque la tecnología de manipular los datos está más clara y se necesita un poco más de tiempo para terminar el elemento del Product Backlog.

- **Retrospectiva de trabajos**

Como entrada de esta fase se utilizó a la “Reunión de Retrospectiva de trabajos del Sprint”, la cual proporcionó al Equipo Scrum una oportunidad para expresar cosas que no le gustó de los trabajos del Sprint anterior, como también pudieron aportar con ideas para mejorar la manera de realizar los trabajos en futuros Sprints, con estos aportes se pudo obtener la salida de la fase que fue un plan de mejoras, cabe mencionar que no en todos los Sprints hubieron aportes de mejoras, en la sección 3.3.1.2 (Eventos basados en Scrum) se puede observar a detalle las características de la reunión de retrospectiva de trabajos del Sprint. A continuación para el caso práctico se muestra un plan de mejoras que se registró en el Sprint 1, las cuales ya se pudieron implementar en la planificación del Sprint 2.

Tabla 41
Plan de mejoras

Plan de mejoras	
MEJORAS	**RESPONSABLES (Roles)**
En la planificación, tratar en lo posible enfocarse en un solo tema durante el Sprint, y no en varios paralelamente, para no perder la concentración.	Todos los roles.
Hacer capacitaciones de temas comunes entre compañeros, que los que más sepan en temas puntuales enseñe a los demás.	Equipo de desarrollo.

4.2.3 Despliegue del incremento del producto

Una vez que el incremento del producto ya se verificó y se aprobó para su liberación en el entorno de producción por parte del dueño del producto, se hizo las capacitaciones necesarias a los usuarios finales, esto se lo hizo bajo los lineamientos detallados en la sección 3.3.1.3 (Artefactos basados en Scrum). Para el ejemplo práctico se presenta un resumen de las capacitaciones que se realizó a los usuarios finales.

Tabla 42
Plan de capacitaciones

Capacitaciones		
MÓDULOS	**USUARIOS**	**DURACIÓN**
Administración de Periodos, Farmacias, Franquiciados y Rubros.	2 Administradores de Operaciones.	4 horas
Parametrización de fechas, porcentajes de comisiones, cálculos rubros.	2 Administradores de Operaciones.	3 horas
Reportes de cálculo de comisiones, cierre de periodo, envío de resultados.	2 Administradores de Operaciones.	4 horas

Despliegue Total del producto terminado: Luego de haber terminado, probado e integrado todos los incrementos del producto se obtuvo el producto total terminado, el mismo que se liberó al ambiente producción. Y se complementó con la entrega formal del producto mediante el “Acta

Entrega Recepción" de la aplicación desarrollada. Para el ejemplo práctico se detalla en el Anexo G de este documento.

4.3 Resultado de indicadores

Para presentar el cálculo de indicadores de la gestión del desarrollo de software, se tomó como referencia una muestra de las planificaciones de trabajos de 4 desarrolladores que realizaron diferentes proyectos en 2 Sprints, el detalle de las planificaciones se muestra en el Anexo H. A continuación se muestra el resultado de los indicadores establecidos en la sección 1.4.1 (Variables de la investigación).

Indicador 1:

Porcentaje de cumplimiento de las actividades en la planificación del trabajo, dentro de las iteraciones del proyecto de software.

Estado de Tareas	Tiempo planificado	Porcentaje por Estado	
Finalizado	665,5	87%	
Pendiente	3	0%	
Producción	38	5%	
Suspendido	61	8%	
Total Horas	**767,5**	100%	
	Porcentaje de cumplimiento:		**92%**
	Porcentaje de Pendientes y suspendidos:		**8%**

Nota: Se considera como cumplido la planificación a los estados Finalizado y Producción.

Figura 40 Indicador 1: Porcentaje de cumplimiento de las actividades de la planificación

Indicador 2:				
Porcentaje de desarrollo de trabajos nuevos y de corrección de errores, en la planificación del trabajo, dentro de las iteraciones del proyecto de				
Tareas	**Tiempo planificado**	**Tiempo real consumido**	PORC. PLANIFICADO	**PORC. REAL**
Bug	89,5	95,84	12%	13%
Nuevo	678	653,93	88%	**87%**
Total (horas)	**767,5**	**749,77**	100%	100%
	Porcentaje de nuevos desarrollos:		**87%**	
	Porcentaje de corrección de errores:		**13%**	

Figura 41 Indicador 2: Porcentaje de desarrollos nuevos y corrección de errores

Indicador 3:				
Porcentaje de tareas no planificadas, en la planificación del trabajo, dentro de las iteraciones del proyecto de software.				
Tareas	**Tiempo planificado**	**Tiempo real consumido**	**PORC. Tiempo real**	
No Planificado	26	71,58	**10%**	
Tareas planifica	741,5	678,19	90%	
Total (horas)	**767,5**	**749,77**	100%	
	Porcentaje de tareas no planificadas:		**10%**	
	Porcentaje tareas planificadas:		**90%**	

Figura 42 Indicador 3: Porcentaje de tareas no planificadas

Indicador 4:			
Porcentaje de tareas en las fases del desarrollo de software, en la planificación del trabajo, dentro de las iteraciones del proyecto de software.			
Fases	**Tiempo planificado**	**Tiempo real consumido**	**PORCENTAJE**
Análisis	11,5	12,5	2%
Desarrollo	333	330,4	44%
Diseño	16	14,3	2%
Documentación	15	16	2%
Implementación	41	42	6%
Investigar	35	24,4	3%
Planificación	26	26	3%
Pruebas	128	98	13%
Revisión	101	77	10%
Seguimiento	10	9	1%
Soporte	51	100,17	13%
Total general	**767,5**	**749,77**	100%

Figura 43 Indicador 4: Porcentaje de tipos de tareas

CAPÍTULO V

PRESENTACIÓN DE RESULTADOS

5 Introducción

En este capítulo se desarrolla el análisis de los resultados obtenidos al aplicar un marco de trabajo implementado con una herramienta ALM, para mejorar la gestión del proceso de fabricación de software en la empresa Farmaenlace Cía. Ltda.

Se expuso a los usuarios finales el desarrollo del marco de trabajo implementado con la herramienta ALM con el objetivo de que validen el proyecto y emitan sus conclusiones. Se hizo una capacitación técnica de 24 horas al equipo de desarrollo de la empresa y un seguimiento de dos meses durante la ejecución del marco de trabajo y de la herramienta ALM en el proceso de desarrollo de software; se distribuyó manuales de usuario para facilitar el uso de la herramienta.

Para recabar datos de la valoración de resultados se utilizó una encuesta como instrumento de medición (Anexo I), que pretende evaluar las variables de investigación planteadas inicialmente, los cuales implican mejorar la gestión del desarrollo de software del área de desarrollo de la empresa.

Para la obtención de resultados se trabajó con el Gerente de Sistemas, 2 Coordinadores y los 14 desarrolladores del área de desarrollo de software, que están definidos en el plan del proyecto y son el universo de la población en la encuesta.

5.1 Instrumento de medición

Para el procesamiento se implementó una metodología dirigida a los resultados del instrumento de medición (Encuesta) realizada, se determinó

20 preguntas cerradas, que se dividieron en 10 preguntas para la variable independiente y 10 preguntas para la variable dependiente de la investigación, también se consideró la etapa de pruebas del proyecto (Cabanilla, 2011)

La encuesta se estructuró de la siguiente manera:

- La encuesta tiene 20 preguntas cerradas las cuales dieron a conocer el criterio de los encuestados en base a la propuesta desarrollada.
- Valoración de resultados entre 1 y 5 para cada pregunta.
    1. Nunca
    2. Rara vez
    3. A veces
    4. Con frecuencia
    5. Siempre
- En cada pregunta se deberá marcar con una (X) la respuesta deseada.

1	2	3	4	5

- Se sumará las (X) por cada columna.
- Se multiplicará la suma de las (X) por el valor de la columna, obteniendo así un puntaje total por columna.
- Se sumará los totales de cada columna y dividirá para el número de encuestados (17).
- El resultado representa el porcentaje de aceptación de la propuesta.

Para la interpretación del resultado, se dividió en cuatro grupos que abarcan un 100% del grado de aceptación, de esta manera se conocerá en que grupo se calificó a la propuesta y que significa el resultado.

De 1 hasta el 39 por ciento: El marco de trabajo propuesto para el desarrollo de software implementado con una herramienta ALM no cumple con los indicadores y tiene una fidelidad muy baja con las actividades realizadas, por lo que deberá tomar acciones correctivas para mejorar el marco de trabajo.

Entre 40 y 59 por ciento: La propuesta cumple con los indicadores pero existen un número considerable de observaciones, las cuales se deberán corregir de manera urgente para dar cumplimiento a la eficiencia del marco de trabajo.

Entre 60 y 85 por ciento: La propuesta cumple con los indicadores y observaciones leves que deberán solucionarse a corto plazo. La tendencia a mejorar el proceso en el desarrollo de software en el área de desarrollo de la empresa es positiva, por lo que se sugiere mantener y fortalecer los puntos favorables y aplicar correctivos a los puntos con baja puntuación.

Más de 85 por ciento: La propuesta cumple con los indicadores, además mejora el proceso en el desarrollo de software en el área de desarrollo de la empresa Farmaenlace Cía. Ltda., por lo que se considera que la propuesta es aplicable.

Los resultados se mostrarán en tablas que valoran la propuesta, y de esta manera se documentará las evidencias del cumplimiento del presente estudio.

5.1.1 Resultado y análisis del instrumento de medición

Se realizó la encuesta propuesta a los 17 involucrados en el proyecto y los resultados se presentan en la siguiente tabla conforme a la estructura del instrumento de medición explicado en la sección 5.1. (El detalle de la tabulación de los resultados de la encuesta se encuentra en el Anexo J).

Tabla 43

Resultados de aceptación de la propuesta
Nro. Encuestados = 17

Respuestas	Nunca	Rara vez	A veces	Con frecuencia	Siempre
(1) Valoración	1	2	3	4	5
(2) Total de respuestas (X)	0	0	15	105	220

(3) Resultado parcial = (1) * (2)	0	0	45	420	1100
(4) Suma total de resultados parciales = ∑(3)			1,565		
Aceptación = (4) / Nro. encuestados			**92,06% de aceptación**		

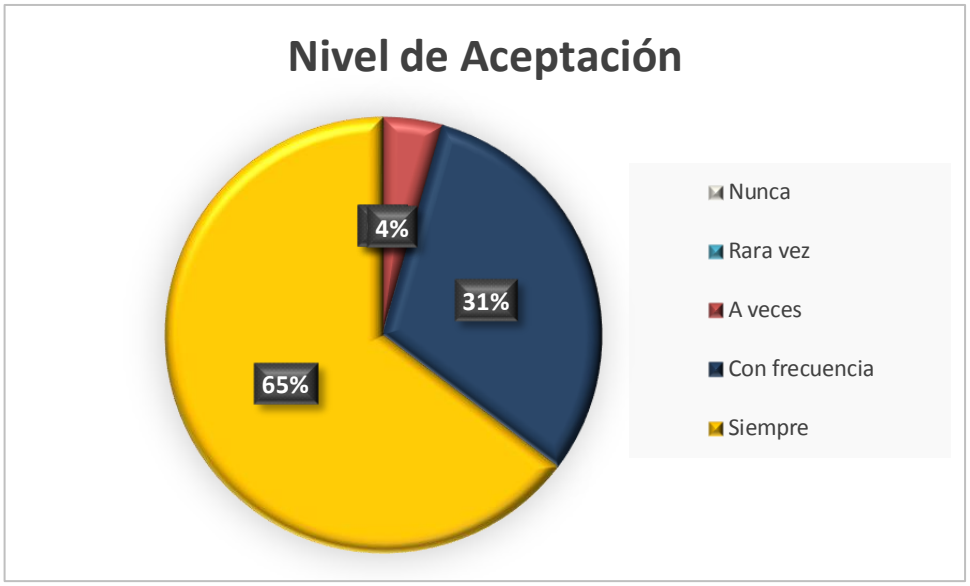


Figura 44 Aceptación de la propuesta

En la Tabla 5.1.1.1 al igual que en la Ilustración 5.1.1.1 de los resultados obtenidos con respecto al porcentaje de aceptación, se puede observar la valoración que proporcionó las personas encuestadas, dando como resultado que el marco de trabajo implementado con la herramienta ALM, consiguió un 92,06% de aceptación en el cumplimiento de los indicadores y objetivos propuestos, por tal motivo se puede concluir que el marco de trabajo implementado con la herramienta ALM mejora la gestión de fabricación de software del área de análisis y desarrollo de la empresa Farmaenlace Cía. Ltda.

5.2 Prueba de Hipótesis

Mediante el instrumento de medición (encuesta) se pudo transformar los resultados cualitativos en cuantitativos de las dos variables de investigación. Con los valores del resultado se puede probar la hipótesis de manera

matemática, para lo cual se escogió el método de Chi Cuadrado con la que se demostrará la dependencia y correlación entre las variables independiente y dependiente (Monge & Pérez, 2010)

5.2.1 Planteamiento de la hipótesis

a) **Hipótesis de Investigación:** ¿El desarrollo de un marco de trabajo, implementado con una herramienta ALM, mejorará la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.?

b) **Variable Independiente:** Se desarrolla un marco de trabajo, implementado con una herramienta ALM.

c) **Variable Dependiente** Se mejora la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda.

- **Hipótesis Nula ($\mathbf{H_0}$):** El marco de trabajo, implementado con una herramienta ALM y la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda. son independientes.

- **Hipótesis Alternativa ($\mathbf{H_A}$):** El marco de trabajo, implementado con una herramienta ALM y la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda. son dependientes.

a) Cálculo de las frecuencias observada y esperada

Frecuencia Observada

La frecuencia observada es el resultado cuantitativo de las encuestas aplicadas a los involucrados a cerca de la aceptación de las variables de investigación, el detalle de los resultados se puede observar en el Anexo J. A continuación se muestra las tablas de resumen de resultados.

Tabla 44

Valoración de la variable Dependiente

Respuesta	Valor de la respuesta	Número de respuestas
Nunca	1	0
Rara vez	2	0
A veces	3	14
Con frecuencia	4	62
Siempre	5	94

Tabla 45

Valoración de la variable Independiente

Respuesta	Valor de la respuesta	Número de respuestas
Nunca	1	0
Rara vez	2	0
A veces	3	1
Con frecuencia	4	43
Siempre	5	126

Tabla 46

Frecuencia Observada de las variables de investigación

		Variable independiente (Marco de trabajo, implementado con la herramienta ALM)					
Variable dependiente (Mejora la gestión del proceso de fabricación de software)	**Valoración**	**1**	**2**	**3**	**4**	**5**	**Totales**
	1	0	0	14	62	94	170
	2	0	0	14	62	94	170
	3	1	1	15	63	95	175
	4	43	43	57	105	137	385
	5	126	126	140	188	220	800
	Totales	170	170	240	480	640	1700

Frecuencia Esperada

La frecuencia esperada o teórica se calcula de acuerdo a las leyes de la probabilidad, a continuación se muestra la fórmula para el cálculo de los

valores enfocado en las variables dependiente e independiente de la investigación.

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

Figura 45 Ecuación EC.1 para calcular la frecuencia esperada (Guevara, 2015)

En dónde:

- **m** : número de columnas
- **j** : posición filas
- **n** : número de filas
- **O** : frecuencia observada
- **I** : posición columnas
- **E** : frecuencia esperada

A continuación se aplica la ecuación EC.1 a la tabla de valores de la Frecuencia Observada, y así obtener la tabla de valores de la frecuencia esperada.

Tabla 47

Frecuencia Esperada de las variables de la investigación

		Variable independiente (Marco de trabajo, implementado con la herramienta ALM)					
Variable dependiente (Mejora la gestión del proceso de fabricación de software)	**Valoración**	**1**	**2**	**3**	**4**	**5**	**Totales**
	1	17	17	24	48	64	170
	2	17	17	24	48	64	170
	3	17.5	17.5	24.71	49.41	65.88	175
	4	38.5	38.5	54.35	108.71	144.94	385
	5	80	80	112.94	225.88	301.18	800
	Totales	170	170	240	480	640	1700

5.2.2 Cálculo del valor de Chi Cuadrado

Para calcular el valor del Chi Cuadrado se aplica la ecuación EC.2 a las tablas de valores de las frecuencias observada y esperada de las variables de investigación, y de esta manera se podrá corroborar matemáticamente la dependencia de la variable dependiente frente a la variable independiente, y así poder comprobar la hipótesis.

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

Figura 46 Ecuación EC.2 cálculo del Chi cuadrado (Guevara, 2015)

En dónde:

- **x** : valor Chi Cuadrado
- **m** : número de columnas
- **j** : posición filas
- **n** : número de filas
- **O** : frecuencia observada
- **I** : posición columnas
- **E** : frecuencia esperada

Tabla 48

Cálculo de Chi Cuadrado para las variables de la investigación

		Variable independiente (Marco de trabajo, implementado con la herramienta ALM)					
Variable dependiente (Mejora la gestión del proceso de fabricación de software)	**Valoración**	**1**	**2**	**3**	**4**	**5**	**Totales**
	1	17.0000	17.0000	4.1667	4.0833	14.0625	56.31
	2	17.0000	17.0000	4.1667	4.0833	14.0625	56.31
	3	15.5571	15.5571	3.8130	3.7368	12.8690	51.53
	4	0.5260	0.5260	0.1289	0.1263	0.4351	1.74
	5	26.4500	26.4500	6.4828	6.3532	21.8796	87.62
	Totales	76.5331	76.5331	18.7581	18.3829	63.3086	**253.5159**

En la tabla del cálculo del Chi Cuadrado el resultado de la sumatoria de los totales es el valor de Chi Cuadrado.

$$x^2_{observado} = \mathbf{253,5159}$$

5.2.3 Cálculo del valor crítico de Chi Cuadrado

El valor crítico de Chi Cuadrado es un determinado porcentaje del área de la cola en la distribución de aceptación de la hipótesis, para obtener este cálculo se debe considerar un nivel de significancia supuesto, en este caso se tomará un grado de significancia moderado que será 0,05. Luego calcula los grados de libertad que representa el tamaño de la tabla del problema que se está resolviendo, a continuación se muestra los cálculos de Chi crítico.

Nivel de significancia: **0,05**

Grados de libertad = $(número\ filas - 1)(número\ columnas - 1)$

$= (5 - 1)(5 - 1)$

$= 4 * 4$

$= \mathbf{16}$

A continuación se escogerá el valor crítico en una tabla constante de valores de la distribución de Chi cuadrado crítico, se tomará como fila el número de los grados de libertad y de columna el nivel de significancia.

Tabla 49

Tabla de distribución de Chi Cuadrado Crítico

Grados de Libertad	NIVEL DE SIGNIFICANCIA					
	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	6.626	9.236	11.071	12.833	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	10.219	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
19	22.718	27.204	30.144	32.852	36.191	38.582
20	23.828	28.412	31.410	34.170	37.566	39.997

Valor de Chi Cuadrado Crítico: $x^2_{crítico} = 26,296$

5.2.4 Comparación de los valores de Chi Cuadrado

En este apartado se tomará en cuenta el valor de Chi Cuadrado calculado en la sección 5.2.3, y el valor crítico de Chi Cuadrado calculado en la sección 5.2.3, y se someterán a una comparativa mediante reglas de decisión para saber si la hipótesis nula se aplica o si la hipótesis alternativa es viable.

Regla de decisión:

- Se aceptará la hipótesis nula (H_0), sí: $x^2_{observado} < x^2_{crítico}$

- Se aceptará la hipótesis alternativa (H_A), sí: $x^2_{observado} > x^2_{crítico}$

Valores de Chi Cuadrado:

- $x^2_{observado} = 253{,}5159$
- $x^2_{crítico} = 26{,}296$

Resultado de prueba de Chi Cuadrado:

$$x^2_{observado} > x^2_{crítico}$$

253,5159 > 26,296

De acuerdo a las reglas de decisión y a los valores de Chi Cuadrado, se puede observar y comprobar que la hipótesis alternativa (H_A) es aceptada.

5.2.5 Conclusiones de los resultados de la prueba

El aplicar la prueba estadística de Chi Cuadrado da un respaldo matemático para comprobar y aceptar la hipótesis planteada, y en función de los resultados presentados se concluye que para mejorar la gestión del proceso de fabricación de software del área de análisis y desarrollo del departamento de sistemas de la empresa Farmaenlace Cía. Ltda. depende del marco de trabajo implementado con la herramienta ALM.

Cabe mencionar que con los resultados de aceptación obtenidos de la encuesta se puedo observar que la variable independiente de la investigación “marco de trabajo, implementado con una herramienta ALM” es un aporte importante e idóneo para el área de análisis y desarrollo de la empresa. Y de esta manera también se puede corroborar la tesis doctoral “Un Marco Metodológico para la Mejora en la Gestión de los Equipos de Desarrollo Software Global” planteada por el D. Javier Saldaña Ramos” de la Universidad Carlos III de Madrid citada en el capítulo 3.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

6 Introducción

En este capítulo se describe las conclusiones y las recomendaciones que dieron como resultado al final de este proyecto. Se analizará el trabajo realizado para el cumplimiento de la hipótesis planteada, y los logros obtenidos correspondientes al objetivo general y a los objetivos específicos establecidos al inicio del proyecto.

6.1 Conclusiones

Se determinó un marco teórico vinculado a la gestión del proceso de fabricación de software del área de análisis y desarrollo del Departamento de Sistemas de la empresa Farmaenlace Cía. Ltda., lo que dio como resultado una base histórica-conceptual de las metodologías y antecedentes contextuales de la necesidad para el desarrollo de un marco de trabajo implementado con una herramienta ALM en la empresa.

El desarrollo del marco de trabajo dio como resultado una solución apegada a las necesidades funcionales de la empresa, el cual se pudo socializar con los involucrados del negocio y principalmente al equipo de desarrollo para que lo utilicen como una guía que regirá la manera de trabajar en los proyectos de desarrollo de software.

Se implementó el marco de trabajo Farmaenlace con la herramienta ALM mediante un caso práctico de estudio. Se ejecutó cada una de las fases establecidas y se generaron los correspondientes entregables, evidenciando la mejora continua durante el ciclo de desarrollo del proyecto.

La validación de los resultados obtenidos del desarrollo del marco de trabajo implementado con una herramienta ALM, se lo realizó mediante los la implementación de la propuesta, y recabando los resultados de los indicadores establecidos en la sección 1.4.1 (Variables de la investigación), los cuales aportaron para la toma de decisiones en la ejecución de acciones correctivas o de mejora en la gestión del proceso de fabricación de software.

Para el indicador Porcentaje de cumplimiento de las actividades en la planificación del trabajo, de las iteraciones del proyecto de software, se pudo establecer que el 92% de las actividades se terminaron, y el 8% restante no se cumplieron por dependencias externas, y por tareas no planificadas.

El Porcentaje de nuevos desarrollos o de corrección de errores en la planificación de trabajos en los Sprints, fue del 87% de desarrollos nuevos, y 13% de corrección de errores de los sistemas que ya están en producción.

El Porcentaje de tareas no planificadas dentro de las iteraciones del proyecto de software, fue del 10% en el cumplimiento de una planificación inicial de la iteración, lo que permitió implementar acciones correctivas en tareas repetitivas de soporte a los sistemas que están en producción.

Los porcentajes de las fases del desarrollo de software, en la ejecución de la planificación de trabajo fueron: 2% Análisis, 2% Diseño, 3% Planificación, 3% Investigar soluciones, 44% Desarrollo, 13% Pruebas funcionales, 13% Revisión desarrollos, 6% Implementación, 2% Documentación, 1% Seguimiento, 13% Soporte. Por lo cual se desprende que existen varias fases que se pueden potenciar como el análisis, diseño, planificación, documentación y seguimiento, y por otra parte buscar mecanismos que permitan bajar el 13% de soporte.

El porcentaje de 92,06% de aceptación de la propuesta evidencia que la hipótesis se cumplió y aportó de una manera conveniente a la empresa

6.2 Recomendaciones

Para definir un marco de trabajo de desarrollo de software, el primer paso importante es realizar un diagnóstico estructural actualizado del área en el que se aplicará la solución y estar conscientes de las habilidades que tienen los miembros del equipo de trabajo. Otro paso es indagar y conocer el ambiente laboral real de la organización, el estado de las relaciones entre las personas que interactuarán en el proyecto. Estos elementos ayudarán mucho al momento de seleccionar de manera acertada la definición del marco de trabajo, como también la manera en que se llegará a los miembros del equipo para que la solución sea aceptada, viable y factible.

Para contextos similares a los expuestos en este estudio, se sugiere aplicar la metodología ágil Scrum como base para definir un marco de trabajo empresarial propio, debido a la factibilidad de adaptarse a equipos pequeños entre 5 y 9 integrantes de un mismo proyecto, también se puede adaptar varios grupos de trabajo bajo la misma metodología y luego integrarlos si fuese el caso, y así obtener un método de gestionar los proyecto de desarrollo de software de una manera ordena, ágil, medible y productiva.

Para implementar marcos de trabajos o metodologías ágiles es muy recomendable utilizar una herramienta de administración de ciclo de vida de las aplicaciones (ALM), ya que integra la información de todas las fases del desarrollo que se genera y muestra de una manera ordena y automática en artefactos de medición y seguimiento del proyecto, ayudando así a una mejor gestión de los proyectos.

Para contextos similares a los expuestos en este estudio, se recomienda utilizar Microsoft Team Foundation Server (TFS) como herramienta de administración de ciclo de vida de las aplicaciones (ALM), por su integración con una variedad importante de herramientas tanto Microsoft y externas para

gestionar la fabricación de software y por ofrecer más funcionalidades que las soluciones similares del mercado.

Se sugiere considerar futuras investigaciones como el aseguramiento de la calidad de Software, para mejorar la calidad de puesta a producción y minimizar tiempos de soporte de las aplicaciones desarrolladas. También el estudio de la usabilidad del software para que los usuarios finales mediante las aplicaciones desarrolladas alcancen objetivos específicos con efectividad, eficiencia y satisfacción. Con los estudios antes recomendados se puede seguir en el camino para complementar y reforzar el proceso de la Ingeniería de Software dentro del área de análisis y desarrollo de la empresa Farmaenlace Cía. Ltda.

LINKOGRAFÍA

Aguilar, L. J. (1996). Programacion orientada a objetos, 653;Creación:1996;Recuperado:9 mayo 2015. Retrieved from https://unefazuliasistemas.files.wordpress.com/2011/04/programacion-orientada-a-objetos-luis-joyanes-aguilar.pdf

Alexander, L., Allen, S., & Bindoff, N. L. (2013). Summary for Policymakers. Journal of Chemical Information and Modeling, 53(September 2013), 1–36;Creación:2013;Recuperado:8 mayo 2013. http://doi.org/10.1017/CBO9781107415324.004

Cabanilla, E. (2011). METODOLOGÍA PARA ELABORAR UN ESTUDIO POR ENCUESTAS DE LA SATISFACCIÓN DEL TURISTA :, 8–26;Creación:2011;Recuperado:8 mayo 2015. Retrieved from www.dialnet.com

Chappell, D. (2008). What Is Application Lifecycle Management ?, (December), Creación:2008;Recuperado:8 mayo 2015. Retrieved from http://www.davidchappell.com/WhatIsALM--Chappell.pdf

Coleman, G., & Verbruggen, R. (1998). A quality software process for rapid application development. Software Quality Journal, 7, 107–122;Creación:1998;Recuperado:8 mayo 2015. http://doi.org/10.1023/A:1008856624790

Colussi, N. (2010). Taller de Historia de las Ciencias de la Computacion Algunos Pioneros de la Computacion, 1–11;Creación:2010;Recuperación:13 junio 2015. Retrieved from http://www.fceia.unr.edu.ar/~iilcc/material/apuntes/taller-cs-historia.pdf

Diagrams, D. F. (2004). III . Structured Analysis and Design Technique (SADT) SADT : Structured Analysis and Design Technique, Creación:2004;Recuperado:9 abril 2015. Retrieved from http://www.cs.toronto.edu/~jm/2507S/Notes04/SADT.pdf

Egas, L., & Játiva, J. (2008). Evolución de las Metodologías de Desarrollo de la Ingeniería de Software en el Proceso la Ingeniería de Sistemas Software y Determinación de una metodología adaptable orientada a una organización pequeña, Creación:2008;Recuperado:9 mayo 2015. Retrieved from http://repositorio.espe.edu.ec/bitstream/21000/8771/1/AC-ESPEL-SOF-0004.pdf

Europa, M. de A. P. (1988). 4 D. Botanical Magazine Tokyo, 167, 161–167;Creación:1988;Recuperado:18 mayo 2015. http://doi.org/10.1038/nn.2703

Figuerola, N. (2014). ALM versus SDLC, Creación:2014;Recuperado:9 mayo 2015. Retrieved from https://articulosit.files.wordpress.com/2014/03/alm-versus-sdlc.pdf

Fl, A. (2010). Procesos de ingeniería de software, 26–39;Creación:2010;Recuperado:5 mayo 2015. Retrieved from http://revistavinculos.udistrital.edu.co/files/2012/12/procesosdeingenieriade.pdf

Fontao, R., Delrieux, C., Kalocai, G., Goñi, G., & Ramoscelli, G. (2002). SOL: Un lenguaje para el programador. In IV Workshop de Investigadores en Ciencias de la Computación (p. 285;Creación:2002;Recuperado:6 mayo 2015). Buenos Aires. Retrieved from http://sedici.unlp.edu.ar/handle/10915/21894

Fowler, M., Seott, K., V, J. G., & Peake, D. M. (1999). UML gota a gota, 224;Creación:1999;Recuperado:12 mayo 2015. Retrieved from http://www.face.ubiobio.cl/~cvidal/modelamiento/libros/UMLgotaagota.pdf

Garimella, K., Michael Lees, & Bruce Williams. (2012). BPM (GERENCIA DE PROCESOS DE NEGOCIO) Tomado del Libro BPM, 39;Creación:2012;Recuperado:9mayo 2015. Retrieved from http://www.konradlorenz.edu.co/images/publicaciones/suma_digital_sistemas/bpm.pdf

Garzás, J. (2013). Gestión ágil de Proyectos de Software, 226;Creación:2013;Recuperado:9 mayo 2015. Retrieved from http://www.inf.utfsm.cl/~guerra/publicaciones/Gestion de Proyectos de Software.pdf

Gonzalez, R., & Gasco, J. (2006). Information systems offshore outsourcing: A descriptive analysis. Industrial Management & Data Systems, 106(9), 1233–1248;Creación:2006;Recuperado:8 mayo 2015. http://doi.org/10.1108/02635570610712555

Guevara, C. (2015). “DESARROLLO DE UNA PLATAFORMA DE BUSINESS INTELLIGENCE PARA FACILITAR EL ANÁLISIS DE DATOS DE LAS COMPETENCIAS GENERALES DE FORMACIÓN APLICADAS EN EL DESEMPEÑO LABORAL DE LOS EGRESADOS DE LA UNIVERSIDAD TÉCNICA DEL NORTE, Creación:2015;Recuerpado:10 junio 2015. Retrieved from http://repositorio.espe.edu.ec/bitstream/21000/10160/1/T-ESPEL-MAS-0021.pdf

Gutierrez, D. (2011). Métodos de Desarrollo de Software. Universidad de Los Andes, Creación:2011;Recuperado:26 mayo 2015. Retrieved from

http://www.codecompiling.net/files/slides/IS_clase_13_metodos_y_procesos.pdf

Ibarz, C., Carlos, J., & Juan, F. De. (2015). 230010 - MOO - Metodología y Programación Orientada a Objetos 230010 - MOO - Metodología y Programación Orientada a Objetos, 1–8;Creación:2015;Recuperado:6 mayo 2015. Retrieved from http://www.upc.edu/estudispdf/guia_docent.php?codi=230010&lang=esp

ITC Infotech. (1988). Structured systems analysis and design methodology, Creación:2012;Recuperado:4 abril 2015. Retrieved from http://www.google.co.uk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCwQFjAA&url=http://vijaysamyal.tripod.com/SSAD.pdf&ei=3lV1U8qvCcmFO8G7gZAJ&usg=AFQjCNFEzcgWbncQUi_2Kf49dr31bobsmw&sig2=02zSz3_47gSU9ZskOIewwA&bvm=bv.66917471,d.ZWU

Johnston, A. K. (2006). Agile Architecture, 1–32;Creación:2006;Recuperado:9 mayo 2015. Retrieved from http://rua.ua.es/dspace/bitstream/10045/1287/4/Offshore_outsourcing_imds_(August_2006).pdf

Kumar, S. (2011). IEEE Computer Society / Software Engineering Institute Software Process Achievement (SPA) Award 2009. Engineering, (March), Creación:2011;Recuperado:12 junio 2015. Retrieved from http://www.sei.cmu.edu/reports/11tr008.pdf

Letelier, P., & Penadés, M. C. (2006). Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Técnica Administrativa, 5(26), 17;Creación:2006;Recuperado:8 mayo 2015. http://doi.org/1666-1680

Macluskey. (2009). Historia de un Viejo Informático. La Programación Estructurada. | El Cedazo, Creación:marzo 2009;Recuperado:3 abril 2015. Retrieved from http://eltamiz.com/elcedazo/2009/03/16/historia-de-un-viejo-informatico-la-programacion-estructurada/

Martínez Méndez, F. J. (1996). Qué es Ingeniería de la Información ? Jornadas de Información Y Documentación Empresarial, 3, Creación:2011;Recuperado:11 abril 2015. Retrieved from http://www.um.es/~gtiweb/fjmm/ingenieria.htm

Mckean, R. (2002). Object Design : Roles , Responsibilities , and Collaborations. Creación:2002;Recuperado:9 mayo 2015. Retrieved from http://buhoz.net/public/books/computacion/desarrollo/analisis_y_disenio/Object.Design.Roles.Responsibilities.and.Collaborations.Rebecca.Wirfs-Brock.Alan.McKean.2002.pdf

Monge, I., & Pérez, J. (2010). Estadística no paramétrica prueba chi-cuadrado χ2, 1–20;Creación:2010;Recuperado:10 mayo 2015. Retrieved from http://www.uoc.edu/in3/emath/docs/Chi_cuadrado.pdf

Montilva, a. (2006). Desarrollo de Software Basado en Líneas de Productos de Software Contenidos La idea básica :, 1–34;Creación:2006;Recuperado:20 mayo 2015. Retrieved from http://www.ieee.org.ar/downloads/2006-montilva-productos.pdf

Peña, A. (2006). Ingeniería de Software : Una Guía para Crear Sistemas de Información A l e j a n d r o P e ñ a A y a l a, 120;Creación:2006;Recuperado:13 mayo 2015. Retrieved from http://www.wolnm.org/apa/articulos/ingenieria_software.pdf

Polo, M. (2010). - Metodología - Técnica - Herramienta - Tarea - Procedimiento - Producto, Creación:2010;Recuperado:5 mayo 2015. Retrieved from http://www.inf-cr.uclm.es/www/mpolo/asig/is4/tema04.pdf

Pressman, R. S. (2001). User Interface Design. Software Engineering: A Practitioner's Approach, 401–418:Creación:2001;Recuperado:24 mayo 2015. Retrieved from http://iit.qau.edu.pk/books/Software Eng, Roger Presmen.pdf

Reece, J. (2006). Prototyping and Rapid Application Development (RAD) RAD - Background What is RAD RAD - Goals RAD - Quality RAD - Properties RAD - Cost, Creación:2010;Recuperado:16 abril 2015. Retrieved from file:///C:/Users/Usuario/Downloads/RapidApplicationDevelopment.pdf

Ruiz, F. (2010). Procesos de Ingeniría del Software, 1–49;Creación:2010;Recuperado:9 mayo 2015. Retrieved from http://www.ctr.unican.es/asignaturas/is1/is1-t02-trans.pdf

Rumbaugh, J., Jacobson, I., & Booch, G. (1999). The UML reference manual. New York: Addison-Wesley, 1, 999;Creación:1999;Recuperado:17 junio 2015. Retrieved from http://www.dcc.fc.up.pt/~zp/aulas/1011/asw/geral/bibliografia/Addison Wesley - UML Reference Manual.pdf

Scott, A., & Constantine, L. (2000). The Unified Process Inception Phase: Best Practices in Implementing the UP, 308;Creación:2000;Recuperado:9 junio 2015. Retrieved from file:///C:/Users/Usuario/Downloads/55003e790cf260c99e8f8a20.pdf

ScrumInc. (2012). The Scrum Papers, Creación:2012;Recuperado:7 mayo 2015. Retrieved from http://www.hbagency.com/scrum/tl_files/scrum_inc/documents/ScrumPapers.pdf

Silclir, J. M. (2013). Universidad Tecnológica Nacional Software con Herramientas ALM, Creación:2013;Recuperado:9 junio 2015. Retrieved from http://www.institucional.frc.utn.edu.ar/sistemas/lidicalso/pub/file/Tesis/Gestión de Procesos de Desarrollo de Software con Herramientas ALM(1).pdf

Skarin, M., & Kniberg, H. (2010). Kanban y Scrum – obteniendo lo mejor de ambos, 123;Creación:2010;Recuperado:8 mayo 2015. Retrieved from http://www.proyectalis.com/documentos/KanbanVsScrum_Castellano_FINAL-printed.pdf

Teams, D. (2004). Rational Unified Process Best Practices for Software. Development, 1–21;Creación:2004;Recuperado:10 mayo 2015. http://doi.org/10.1.1.27.4399

The Joint Task Force on Computing Curricula (Association for Computing Machinery IEEE-Computer Society). (2013). Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. Practice, pp. 1–172;Creación:2013;Recuperado:19 mayo 2015. http://doi.org/10.1145/2534860

Thórisson, K. R., & Nivel, E. (2009). Achieving Artificial General Intelligence Through Peewee Granularity. Proceedings of the 2nd Conference on Artificial General Intelligence, 222–223;Creación:2009;Recuperado:9 mayo 2015. http://doi.org/10.2991/agi.2009.42

Torre, L., Brueghel, P., & Commons, C. (2008). Lenguajes de programación, (1563), 4;Creación:2008;Recuperado:8 mayo 2015. Retrieved from http://lsi.ugr.es/curena/doce/lp/tr-11-12/lp-c01-impr.pdf

Trigas, M. (2013). Metodología Scrum, 56;Creación:2013;Recuperado:9 mayo 2015. Retrieved from http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf

Virrueta Mendez, A. (2010). Instituto tecnológico superior de apatzingán, (453), Creación:2010;Recuperado:9 mayo 2015. Retrieved from http://www.monografias.com/trabajos-pdf4/metodologias-de-desarrollo-software/metodologias-de-desarrollo-software.pdf

Wagner, F., Wagner, T., Wolstenholme, P., & Group, F. (2006). Modeling Software with Finite State Machines A Practical Approach. Group, 302;Creación:2006;Recuperado: 12 junio 2015. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Modeling+software+with+finite+state+machines:+a+practical+approach#0

Wells, D. (2003). Extreme Programming: A gentle introduction, Creación:2003;Recuperado:2 mayo 2015. Retrieved from http://public.callutheran.edu/~reinhart/CSC335ADEP/Week9/extreme_programming.pdf

Yu, Y. (2010). Software engineering and knowledge engineering : theory and practice. Retrieved from https://books.google.com.ec/books?id=5iDnlBiMGVYC&pg=PA504&lpg=PA504&dq=Engineers,+society+of+S.+(n.d.).+About+The+SSE+?+Society+of+Software+Engineers&source=bl&ots=Z6QSvMQR2K&sig=un7tvebU701hxw-eKsynpvHZXvk&hl=es-419&sa=X&redir_esc=y#v=onepag

Angelfire. (24 de Agosto de 2010). Software Procesos. Obtenido de http://www.angelfire.com/my/jimena/ingsoft/guia2.html

Ardanaz, S. (s.f.). procesosoftware. Obtenido de procesosoftware: http://procesosoftware.wikispaces.com/Modelo+Espiral

BARINIA, A. P. (Mayo de 2010). Modelo de proceso de Software. Obtenido de ingenieria de software: https://portal-ingenieriadesoftware.wikispaces.com/Modelo+de+proceso+de+Software

Calero, W. (8 de Octubre de 2010). Ingeniería de Software. Obtenido de http://ingenieraupoliana.blogspot.com/2010/10/modelo-lineal-secuencial.html

Distancia, U. N. (s.f.). Ingeniería de Software. Obtenido de http://datateca.unad.edu.co/contenidos/301404/301404_ContenidoEnLinea/leccin_13__el_modelo_dra_desarrollo_rpido_de_aplicaciones.html

ejchcias, B. (20 de Septiembre de 2010). Modelos Evolutivos de Software. Obtenido de http://gproyectos-s4b.blogspot.com/2010/09/modelo-concurrente.html

EMMANUEL, C. O. (Junio de 2009). Diseño 3 Capas. Obtenido de http://ecodisenouml.blogspot.com/

Francisco, R. (s.f.). Procesos de Ingeniería de Software. Obtenido de http://www.ctr.unican.es/asignaturas/is1/is1-t02-trans.pdf

González, C. (2012). Metodologias Agiles. Obtenido de http://slideplayer.es/slide/2273638/

Infante, S. (20 de Enero de 2014). El poderoso tablero Kanban en la gestión de tareas. Obtenido de https://sergioinfantemontero.wordpress.com/2014/01/20/el-poderoso-tablero-kanban-en-la-gestion-de-tareas/

Microsoft. (30 de Noviembre de 2006). Desarrollo del proceso de pruebas. Obtenido de https://technet.microsoft.com/es-es/library/bb490187.aspx

Nacional, I. P. (2006). Enlazando el conocimiento. Obtenido de http://www.paginasprodigy.com/escomino/Sinapsys/sinapsysMetodologia.htm

Oviedo, E. (14 de Octubre de 2014). Cuadrante de Gartner para Herramientas de ALM. Obtenido de Cuadrante de Gartner para Herramientas de ALM

Solution, I. (s.f.). Metodología. Obtenido de http://www.reserv.com.ar/metodologia.php

Wikipedia. (Julio de 2014). Proceso Unificado de Rational. Obtenido de https://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational

Wikipedia. (23 de Octubre de 2015). Software. Obtenido de https://es.wikipedia.org/wiki/Software

ANEXOS

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DE SOFTWARE
CUARTA PROMOCIÓN

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por el señor Ing. José Antonio Quiña Mera.

En la ciudad de Latacunga, a los 21 días del mes de diciembre del 2015.

Ing. Jorge Geovanny Raura Ruiz. MSc
DIRECTOR DEL PROYECTO

Aprobado por:

Ing. Lucas R. Garces G. MSc
COORDINADOR DE LA MAESTRÍA DE INGENIERÍA DE SOFTWARE

Ing. Gonzalo Patricio Espinel Mena. MSc
OPONENTE

Dr. Rodrigo Vaca
SECRETARIO ACADÉMICO