UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

MAESTRÍA Y DOCTORADO EN CIENCIAS E INGENIERÍA

"MÉTODO PARA ESTABLECER LA RELACIÓN ENTRE LA CALIDAD DEL PRODUCTO Y LA CALIDAD DEL PROCESO DE SOFTWARE UTILIZANDO MÉTODOS CUANTITATIVOS"

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS

PRESENTA:

LUZ ADRIANA CÁRDENAS MARTÍNEZ

TIJUANA, BAJA CALIFORNIA **MARZO 2011**

Universidad Autónoma de Baja California

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

COORDINACIÓN DE POSGRADO E INVESTIGACIÓN

FOLIO No. 056

Tijuana, B. C., a 01 de marzo de 2011

C. LUZ ADRIANA CÁRDENAS MARTÍNEZ
Pasante de: Maestro en Ciencias
Presente

El tema de trabajo y/o tesis para su examen profesional, en la Opción TESIS

Es propuesto, por el C. Dr. J. Reyes Juárez Ramírez

quien será responsable de la calidad de trabajo que usted presente, referido al tema "Método para establecer la relación entre la calidad del producto y la calidad del proceso de software utilizando métodos cuantitativos".

el cual deberá usted desarrollar, de acuerdo con el siguiente orden:

I.- CONTENIDO
II.- CONCEPTOS BÁSICOS Y ESTADO DEL ARTE
III.- PROPUESTA DE METAMODELO PARA LA RELACIÓN PROCESO-PRODUCTO
IV.- EXPERIMENTOS REALIZADOS
V.- DISCUSIÓN DE RESULTADOS
VI.- CONCLUSIONES Y TRABAJO FUTURO
VII.- REFERENCIAS BIBLIOGRAFICAS
VIII.- APENDICE

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Q. Noemí Hernández Hernández
Sub-Director Secretario

Dr. J. Reyes Juárez Ramírez
Asesor

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

Dr. Luis Enrique Palafox Maestre
Director

Dedicatorias

Me gustaría dedicar esta Tesis a mis padres Artemio y Luz María que siempre han estado ahí, apoyándome incondicionalmente en cualquier meta que me proponga.

Mamá, gracias por escucharme, por tus consejos, tus valores, por la motivación constante que siempre me has dado, lo que me ha permitido ser quien soy.

Papá, gracias porque siempre me impulsa a seguir adelante sin importar los obstáculos que se interpongan, por tu ejemplo de constancia y perseverancia que te caracterizan y eso ha influido en mí.

Muchas Gracias, los quiero!

Agradecimientos

A Dios por permitirme llegar a este momento tan especial en mi vida. Por los triunfos y los momentos difíciles que me han enseñado a valorarlo y amarlo cada día más.

Todo mi agradecimiento a mis padres que siempre me han apoyado en todo lo que me he propuesto, por sus consejos, observaciones y su amor. Gracias por todo, ustedes han hecho de mí la mujer que soy.

Gracias a mis hermanas Carla y Julieta por que siempre han sido un gran soporte para mí, créanme que sin su apoyo y comprensión no hubiera podido culminar con este trabajo.

A mi novio Daniel Hernández, por su apoyo y consejos que siempre me brinda, ya que los dos nos pusimos la meta de terminar un posgrado y lo logramos y vamos por más…

A mi director de tesis, tutor, profesor, consejero y amigo, Dr. J. Reyes Juárez, por su esfuerzo, dedicación y paciencia que siempre me brindo para el desarrollo de este trabajo. Él ha inculcado en mí un sentido de seriedad, responsabilidad y dedicación sin los cuales no podría tener una formación completa como profesional e investigador. Muchas Gracias Profe por todo!!

A mi alma mater Universidad Autónoma de Baja California, en especial a la Facultad de Ciencias Químicas e Ingeniería, por todo el apoyo que me brindaron para realizar mi trabajo de maestría, así como el apoyo en la realización de mi estancia de investigación y las participaciones en congresos relacionados con el tema de tesis.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT), por el apoyo económico brindado durante la realización de este proyecto.

A los sinodales de mi comité de tesis, Dr. Licea Sandoval, Dr. Antonio Rodríguez Díaz y Dr. Manuel Castañón Puga, gracias por sus

valiosas observaciones y comentarios sobre este documento y por el seguimiento que siempre le dieron a mi trabajo de investigación.

Un especial agradecimiento a las Maestras Ma. Guadalupe Ibargüengoitia por su tiempo brindado y sus consejos profesional y de vida, Maestra muchas gracias. A la maestra Cecilia Pérez por su ayuda en hacerme comprender uno de los temas centrales de esta tesis y por su tiempo brindado. A Blanca Gil por su amistad y tiempo brindado en mi estancia en el DF, y a mis compañeros Esmeralda, Evelia y Omar por su apoyo en mi estancia en la UNAM y hacerme sentir parte de su grupo. Gracias por todo.

A Ana y Rey por brindarme un espacio en su casa durante mi estancia en el D.F, y hacerme sentir siempre en casa.

A mi tía Beatriz por haberme ayudado a revisar el documento de tesis y hacerme sus observaciones de forma y fondo. También por sus consejos y apoyo que siempre nos brinda a mí y a mis hermanas.

A Israel Villalobos por haberme apoyado en darle formato al documento de tesis y aguantar la carilla del trabajo.

Muchas Gracias!

I CONTENIDO

II LISTA DE FIGURAS

III LISTA DE TABLAS

IV LISTA DE ACRONIMOS

Acrónimo	Significado
ANT	Artefactos No Tangibles.
APE	Administración de Proyectos Específicos.
AT	Artefactos Tangibles.
ATCD	Artefactos Tangibles de Contribuidor Directo.
ATCI	Artefactos Tangibles de Contribuidor Indirecto.
CMM	Capability Maturity Model (Modelo de Maduración de Capacidades).
CMMI	Capability Maturity Model Integration (Modelo de Maduración de Capacidades Integrado).
DEICU	Desviación de Esfuerzo del total de las tareas completada para la Implementación de cada Caso de Uso.
DEPC	Desviación del Esfuerzo (tiempo) de cada tarea completada del Plan de Capacitación.
DEPT	Desviación del Esfuerzo (tiempo) de cada tarea completada del Plan de Trabajo.
DMS	Desarrollo y Mantenimiento de Software.
EECU	Número de Elementos de la especificación de Casos de Uso según el estándar.
EP	Eficiencia en la ejecución de Pruebas.
ER	Eficiencia de las Revisiones de validación de casos de uso.
FCQI	Facultad de Ciencias Químicas e Ingeniería.
FMEA	Failure Mode Effect Analysis (Análisis de Modos y Efectos de Fallos).
IC	Ingeniería en Computación.
IEC	International Electrotechnical Commission (Comisión Electrotécnica Internacional).
ISO	International Organization for Standardization (Organización Internacional de Estándares).
MOF	Meta Object Facility.
MoProSoft	Modelo de Procesos para la Industria de Software.
MOSCA	Modelo Sistémico de Calidad de software.
NANA	Número de Actividades No Actualizadas registradas en el plan de trabajo.
NAP	Número de Actividades Planeadas.
NBE	Numero de Bitácoras Establecidas en el proceso.
NBR	Numero de Bitácoras Realizadas.
NCUID	Número de Casos de Uso Identificados en la fase de Análisis.
NCUIM	Número de Casos de Uso Implementados.
NCUNV	Número de Casos de Uso que No definen Validaciones y excepciones.
NDER	Número de Defectos Encontrados en Revisiones de validaciones.
NDEP	Número de Defectos Encontrados en Pruebas.
NII	Número de Interfaces Implementadas según las especificaciones.
NINC	Número de Interfaces que sus datos No son Correctos según las especificaciones.

NMR	Número de Minutas Realizadas.
NRT	Número de Reuniones de Trabajo realizadas.
NPP	Número de casos de Pruebas diseñados y/o Planeadas.
NPR	Número de Pruebas Requeridas para obtener una cobertura adecuada.
NRF	Número de Requerimientos Funcionales especificados.
NRFI	Número de Requerimientos Funcionales Implementados.
NRII	Número de Requerimientos Implementados no correctas.
NRNI	Número de Requerimientos funcionales No Implementados.
NSMC	Número de Scrum Meeting planeadas.
NSMR	Número de Scrum Meeting Realizadas.
NUD	Número de requerimientos de Usabilidad Definidas en las especificaciones.
NUNC	Número de requerimientos de Usabilidad que son Comprendidas por el usuario.
OMG	Object Management Group.
PAA	Porcentaje de actividades actualizadas según el avance realizado.
PC	Porcentaje de completud.
PCP	Porcentaje en la completud de la definición de Pruebas de software.
PCCU	Porcentaje de completud en la documentación de la especificación de casos de uso.
PCCUV	Porcentaje de completud de Casos de uso que definieron validaciones y excepciones.
PCII	Porcentaje de Correctud en Información de Interfaces.
PCIU	Porcentaje de Comprensión de Interfaces de Usuario.
PCP	Porcentaje en la Completud de la definición de Pruebas de software.
PCUI	Porcentaje de eficiencia en la identificación de los Casos de Uso.
PCR	Porcentaje de Correctud.
PCRB	Porcentaje de Completud en la Realización de Bitácoras, según el proceso establecido por integrante del equipo.
PCRT	Porcentaje de Completud de minutas de trabajo según las Reuniones de Trabajo realizadas.
PCSM	Porcentaje de Completud de Scrum Meeting realizadas.
PECU	Promedio de Elementos cumplidos en la especificación de los Casos de Uso.
PHE	Promedio de Horas Esfuerzo invertido en revisiones de validar los casos de uso.
PHEP	Promedio de Horas Esfuerzo invertido en ejecutar las Pruebas.
PyS	Planeación y Seguimiento.
RFA	Realización Fase de Análisis.
RFD	Realización de la Fase de Diseño.
RFP	Realización Fase de Pruebas.
RFR	Realización Fase de Requerimientos.
SQuaRe	Modelo de Procesos para la Industria de Software.
SPEM 2	Metamodelo de Ingeniería de Procesos de Software versión 2.

TA	Tarea de Apoyo.
TB	Tarea Base.
TE	Tiempo total Estimado de las tareas planeadas (min).
TECUP	Tiempo total Estimado de las tareas planeadas para la implementación de los Casos de Uso (min).
TES	Tiempo total Estimado de las tareas planeadas (hrs).
TSP	Modelo de Proceso de Software de Equipo.
TR	Tiempo Total Real de las tareas.
TRCUI	Tiempo Total Real de las tareas completadas para la implementación de los Casos de Uso (min).
UABC	Universidad Autónoma de Baja California.
UNAM	Universidad Nacional Autónoma de México.
UML	Unifield Modeling Language (Lenguaje de Modelado Unificado).
VyV	Verificación y Validación.

V RESUMEN

Método para establecer la relación entre la calidad del producto y la calidad del proceso de software utilizando métodos cuantitativos

En este documento se presentan un Metamodelo en el que se establece la relación entre el proceso y el producto de software, a través de métricas de calidad, con lo cual se puede observar cuantitativamente el impacto que tiene el proceso de software sobre la calidad del producto generado. Para establecer la relación se toman como base los siguientes conceptos:

Estructura de composición del proceso de software se define como un conjunto de actividades, y estás están compuestas por tareas (unidad atómica), en donde la tarea, tiene como propósito generar un artefacto, el cual forma parte de un producto de software.

Modelo de calidad de software es un conjunto de características de las cuales se subdividen en sub-características y estas en atributos específicos de calidad.

M*étrica de software* se define como un atributo que evalúa un componente y produce un dato simple, un símbolo o un número.

En base a estos conceptos, se definió un Metamodelo que describe la estructura jerárquica de las entidades de proceso, producto y el modelo de calidad, así como la relación entre ellos. En base a esto, se establece que la tarea se clasifica en dos tipos: Tareas Base (TB) y Apoyo (TA). Las TB son tareas que generan, elaboran, modifican un artefacto. Las TA son las tareas que verifican, validan, revisan un artefacto. Por otro lado, un Artefacto se clasifica en Tangibles (AT) y No Tangibles (ANT). Los AT están formados por otros artefactos más simples. Los ANT son artefactos no definidos formalmente. Los AT se subdividen en Contribuidor Directo (ATCD) y Contribuidor Indirecto (ATCI). Los ATCD son elementos principales para la generación de otro artefacto tangible y ATCI definen elementos de soporte para la generación de otro artefacto tangible. Por otro lado, el Metamodelo establece que tanto el artefacto como la tarea, tienen propiedades mediante las cuales se evalúan y describe su calidad. Estas propiedades pueden ser cuantificables por medio de las métricas de calidad, según corresponda. De esta manera, se establece un mapeo entre el proceso y el producto de software a través de las métricas de calidad de software.

Para evaluar la utilidad del Metamodelo se realizó un estudio donde se consideraron varios casos de estudio con proyectos reales, en el cual se concluyó lo siguiente: (1) Al definir el proceso de software en su nivel más fino de composición (tarea), contribuyó para que el mapeo sea directo con la causa raíz del defecto del producto de software. (2) Al tener identificadas las tareas del proceso que causaron los defectos en la calidad del producto de software, se concluyó que las TB son las que más contribuyen en la calidad del producto de software. (3) Las TB que generan ATCD tienen un mayor impacto en la calidad del producto en comparación con los ATCI, debido a que aportan elementos esenciales a otra tarea del proceso. (4) Al realizar el mapeo de las tareas del proceso con los atributos de calidad del producto, se observó que las tareas que impactaron a los atributos de calidad de Funcionalidad y fueron las tareas con propósito de planeación y seguimiento, el análisis de los requerimientos y la elaboración y ejecución de pruebas.

Palabras Clave: *Calidad de software, proceso de software, producto de software, métricas de calidad.*

VI ABRSTRACT

Method to establish the relationship between the software product quality and the quality of the software development process using quantitative measures

This document presents a Metamodel which seeks to establish the relationship between the quality of a software product and the quality of its development process using quality metrics. This lets us measure the impact that the software development process has over the quality of the end product. To establish this relationship we take into account the following concepts:

Composition structure of the software development process, it is defined as a set of *activities*, and these activities are composed of *tasks* (atomic unit). The purpose of a task is to generate an *artifact.* The artifacts are part of a software product.

Software Quality model is a set of features divided into sub-features and these sub-features are then divided into quality attributes.

A *software metric* is defined as an attribute that evaluates a component and produces a simple value, a symbol or a number.

Based on these concepts we define a Metamodel to describe a hierarchical structure for the process entities, products, the quality model and the relation between them.

Our Metamodel establishes two types of tasks: Base tasks (TB) and Support tasks (TA). Base tasks are those that generate or modify an artifact. The support tasks are defined as the tasks that verify, validate or go through an artifact. On the other hand, the artifacts are classified as tangible (AT) and non-tangible (ANT). An AT is composed of simpler artifacts. The ANTs are not formally defined artifacts. The ATs are then divided into two kinds, direct contributor (ATCD) and indirect contributor (ATCI). The ATCD are the main elements to generate another tangible artifact and the ATCI define elements to help on the generation of another tangible artifact. Aside from this, the meta-model establishes that an artifact as well as a task has properties which help evaluate and describe its quality. These properties can be measure using quality metrics as needed. From this, we establish a mapping from a development process to a software product using software quality metrics.

To evaluate the usefulness of our Metamodel we did some field studies considering different use cases from real projects, which led us to the following conclusion: (1) Defining a software development process to its smallest components, that is the tasks, helps define a direct mapping between a software product defect and its root cause. (2) By identifying the root cause tasks of a defect on the software product we found that the TBs are the ones that contribute the most to the software product quality. (3) The TBs that generate ATCDs have a bigger impact on the software quality compared to the ATCI, due to the elements required by other tasks which are generated by the ATCD. (4) Making a mapping from process tasks to quality attributes on the product we found that the tasks with a bigger impact on the quality attributes are those related to planning and following, the detailed requirements analysis and the design and execution of the software testing. And for the usability attribute the most important tasks are the ones with the purpose to specify the usability requirements and the execution of test focused on the usability of the software product.

Keywords: *Software Quality, software process, software product, quality metrics.*

CAPÍTULO I

INTRODUCCIÓN

I Introducción

I.1 Problemática sobre la separación de la calidad del producto de software y el proceso de software

Actualmente en la industria del software, el tema de la calidad se ha vuelto uno de los objetivos estratégicos principales en las organizaciones a la hora de pensar en mejorar la calidad de sus productos (Ivanisevich 1997). Con este fin, en los últimos años se han desarrollado modelos de calidad enfocados tanto al proceso de software así como al producto de software pero estudiados por separado. Tradicionalmente se ha manejado por separado la calidad del producto y la calidad del proceso, incluso así ha quedado plasmado en los modelos creados.

Por otro lado, los sistemas de información cada vez son más sofisticados y complejos para afrontar un mercado en continuo cambio. Las organizaciones necesitan tomar mejores decisiones y a tiempo para tener éxito en los proyectos. Con este panorama, la información objetiva es un requisito para la toma de decisiones críticas y basadas en hechos. (Comunicación 2008)

En esta sección se describe el planteamiento del problema que se pretende atender en este trabajo de tesis: **La problemática de la separación entre la calidad del proceso de software y la calidad del producto de software.**

A continuación se describe las propuestas de modelos existentes orientadas tanto en la calidad del producto de software como en la calidad del proceso de software, de esta manera se podrá observar como cada uno de estos modelos se ha dedicado a estudiar la calidad del producto de software y del proceso de software pero por separado.

Antes de iniciar con la descripción de la problemática, es necesario describir brevemente conceptos con respecto al producto y proceso de software, así como métricas de software, para de esta manera poder entrar en contexto. Los cuales se trataran en el capítulo II con más detalle.

Según (Satpathy 2002) y (Ortega 2003), un proceso de software es un conjunto de actividades, en la cual cada una de ellas se relaciona desde el análisis de requisitos hasta mantenimiento del software. Cada una de estas actividades toma productos como entrada y genera productos de salida. Un modelo de procesos está representado por sus elementos básicos: quien realiza (rol) qué (tarea) para, a partir de unas entradas (productos de trabajo) obtener unas salidas (productos de trabajo), (Ruiz 2008). Es decir las tareas representan el esfuerzo a hacer, los roles representan quien lo hace y los productos de trabajo representan las entradas que se utilizan en las tareas y las salidas que se producen.

Como se mencionó anteriormente, un proceso relaciona sus elementos entre sí, en base a esto, el presente trabajo se enfocará en la relación entre la tarea y el producto.

Un producto, según IEEE-12207 (ISO/IEC 2002), es el conjunto de programas de computadora, documentación y datos asociados. También se define como un subconjunto de artefactos de un proyecto de software incluyendo código, ejecutable, desarrollado y liberado para un cliente (Juárez-Ramírez 2008).

La relación que existe entre ellos es la que se genera con la ejecución de la tarea, ya que puede producir o modificar un artefacto y también un artefacto alimenta a una tarea como entrada.

El producto tiene atributos de calidad que pueden ser observables y medibles. Por su parte, el proceso también tiene atributos observables y medibles. Algunos aspectos que sustentan a esta propuesta, son los modelos de calidad, los cuales están orientados a la satisfacción de los clientes y a la mejora de los procesos (Mario G. Piattini 2003). Un modelo de calidad define los requerimientos de calidad y clasifica las características y sub-características de calidad, hasta llegar con los atributos específicos de calidad, tanto del producto o como del proceso de software (Marc-Alexis Côté 2008). Un atributo es una propiedad de calidad a la que se le puede asignar una métrica, entendiendo por métrica como un atributo que evalúa un componente y produce un dato simple, un símbolo o un número (International Organization for Standardization 2005)

I.1.1 Propuestas orientadas a la calidad del producto de software

Según (McCall 1977) propone el primer modelo de calidad, que define la relación entre las características de calidad y las métricas. Este modelo ha influido en muchas investigaciones posteriores en esta área. Este modelo se describirá con mayor detalle en el capítulo II.

En 1978 fue introducido el Modelo Boehm (Boehm 1978), que trata de definir la calidad del software cuantitativamente utilizando un conjunto de atributos y métricas. Este modelo se describirá con mayor detalle en el capítulo II.

Más tarde aparece ISO 9126 (International Organization for Standardization 2001) y actualmente SQuaRE (Modelo de Procesos para la Industria de Software) (International Organization for Standardization 2005), los cuales son modelos normalizados que permiten evaluar y comparar productos de software en base a las siguientes características de calidad: funcionalidad, confiabilidad, usabilidad, portabilidad, mantenibilidad y eficiencia.

De igual manera, Basili, Caldiera y Rombach introducen Goal Question Metric (GQM) (Basili 1994) proporciona una manera útil para definir mediciones tanto del proceso como de los resultados de un proyecto. Es un método de medición que puede ser más satisfactorio si se diseña orientado a las metas u objetivos que se quieren alcanzar.

GQM define una meta, refina esta meta en preguntas y define métricas que intentan dar información para responder a estas preguntas. Las preguntas ayudarán a medir si se está alcanzando la meta definida, por lo tanto se considerarán preguntas que sean potencialmente medibles. GQM se puede aplicar a todo el ciclo de vida del producto, procesos y recursos.

GQM sigue un proceso de seis pasos donde, los tres primeros tratan de identificar las métricas a partir de las metas del negocio y los tres últimos se basan en la recopilación de los datos de las medidas y su utilización eficaz en la toma de decisiones (Instituto Nacional de Tecnologías de la Comunicación 2008).

I.1.2 Propuestas orientadas a procesos de software

Los principales modelos orientados a la calidad de procesos son (Wang 1997) : ISO 9000:2000, ISO 90003, Modelo de Madurez de Capacidades (CMM, por sus siglas en ingles), su sucesor Modelo de Madurez de Capacidades Integrado (CMMI, por sus siglas en ingles) y SPICE (Software Process Improvement and Capability dEtermination)

ISO 9000:2000 (International Organization for Standardization 2000): Establecen la necesidad de implementar el proceso de medición con el objetivo de controlar la calidad del producto y la capacidad del proceso a través de métricas para controlar la calidad del proceso.

CMM (Humphrey 1989): es un modelo orientado a la calidad del software que clasifica a las empresas en niveles de madurez, basándose en la mejora continua de los procesos de desarrollo.

CMMI (Institute 2002): En 2002 se publicó la evolución de CMM con el objetivo de facilitar la integración de varios modelos de forma simultánea. Este modelo define una metodología de medición del software en las áreas de proceso Medición y Análisis y en las áreas de proceso de Gestión Cuantitativa de Proyectos y Ejecución de Procesos Organizacionales.

ISO 15504 (15504-2 2003): El modelo ISO/IEC 15504, también conocido como SPICE se desarrolló como un estándar internacional para la evaluación de los procesos software combinando la experiencia y el conocimiento aportado por modelos como CMM, Trillium y Bootstrap entre otros.

Al igual que los modelos anteriores, define seis niveles de madurez, cinco categorías de procesos y nueve atributos de procesos para medir el grado de adecuación de los procesos (Montilva 2007).

Para la evaluación del proceso de mejora se utilizarán tanto métricas de efectividad del proceso como la puntuación del cumplimiento de las distintas prácticas genéricas y básicas del modelo SPICE (Barafort 2005).

Este modelo define un marco para el proceso de medición, debido a que son necesarias para mostrar cuantitativamente el estado actual de los procesos y las prácticas

contra unas mejores prácticas de ingeniería de software, y para mostrar hasta qué punto los procesos de software son efectivos para alcanzar las necesidades y metas de negocio de una organización (Instituto Nacional de Tecnologías de la Comunicación 2008).

En la Tabla 1 muestra los enfoques de separación entre los modelos existentes (véase columnas dos y tres). Por ejemplo, los modelos iniciales o que han servido como base para los modelos actuales, como McCall 1977 (McCall 1977) y Boehm 1978 (Boehm 1978) están orientados a la calidad del producto; mientras que CMM 1995 (Wang 1997) e ISO 9000, 2000 (International Organization for Standardization 2000) están orientados al proceso.

En la actualidad, continúa dicha separación del estudio de la calidad, por ejemplo, los modelos más recientes como CMMI 2002 (Institute 2002), Modelo de Procesos para la Industria de Software, MoProSoft 2005 (Oktaba Hanna 2005) están orientados al proceso; mientras que ISO 9126 (International Organization for Standardization 1987) e ISO/IEC 25000 (International Organization for Standardization 2005) se enfocan a la calidad del producto.

Tabla 1: Comparativo de los modelos de calidad del proceso y del producto.

Modelo	Enfoque de calidad del Proceso	Enfoque de Calidad del Producto	Uso Explicito de Métodos Cuantitativos para evaluación de Calidad
CMM	√		√
CMMI	√		√
ISO 15504	√		√
ISO 9000:2000	√		
MOPROSOFT	√		√
PMCOMPETISOFT	√		√
ISO 90003	√		
PSP Personal Software Process	√		√
TSP Team Software Process	√		√
BOOTSTRAP	√		
ISO/IEC 19761		√	√
ISO/IEC 20926		√	√
ISO/IEC 20968		√	√
IEEE 14143		√	√
ISO 9126		√	√
Modelo De MCCALL		√	
Modelo De BOEHM		√	
FURPS		√	
Modelo DROMEY		√	
ISO/IEC 25000		√	√

De los modelos mencionados, varios de ellos consideran aspectos cualitativos para evaluar la calidad del software. Sin embargo, los aspectos cuantitativos son más significativos para el control de la calidad. En la Tabla 1 se indica cuáles modelos consideran un enfoque cuantitativo, bien sea con métricas u otro tipo de controles numéricos.

La calidad de cualquier producto no puede ser asegurada simplemente inspeccionando el producto (aspectos cualitativos) o desarrollando controles de calidad estadísticos (García 2003). Esta afirmación se basa en que existe una correlación directa entre la calidad del proceso y la calidad del producto obtenido. Por lo tanto, una organización no puede garantizar la entrega de productos de calidad centrando sus programas de calidad únicamente en el producto o en el proceso.

Para definir la calidad del software es esencial tener bien especificado la diferencia entre la calidad del producto de software y la calidad de proceso de desarrollo de software, sin embargo, las metas que se establecen para obtener calidad del producto van a determinar los objetivos del proceso de desarrollo, ya que la calidad del producto va a depender del proceso especificado (Pressman 2002) Sin un buen proceso de desarrollo es casi imposible obtener un buen producto. (Mendoza 2005). Por tal motivo, las organizaciones necesitan indicadores que les proporcionen información para saber si los objetivos que se establecen en el proceso de desarrollo de software, realmente cumplieron con las metas esperadas para obtener calidad en el producto de software. Es por eso, que para poder asegurar la calidad del proceso de software y por consiguiente la del producto de software, es necesario tener un buen proceso de medición, el cual proporcione información objetiva, valores, descriptores, indicadores o algún otro mecanismo mediante el cual se pueda obtener información que nos permita tomar decisiones sobre la calidad del proceso o del producto de software (Fenton 1997).

Un proceso de medición de software persigue tres objetivos fundamentales: (1) ayudar a entender qué ocurre durante el desarrollo y el mantenimiento, (2) permitir controlar qué es lo que ocurre en los proyectos y (3) poder mejorar los procesos y productos (Fenton 1997).

Las métricas son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo de software y los proyectos de mantenimiento (Briand 1996) y pueden ser utilizadas por profesionales e investigadores para tomar mejores decisiones (Fenton 1997).

I.2 Objetivos del trabajo de tesis

Los objetivos planteados para este trabajo de tesis están directamente relacionados con el planteamiento del problema descrito en la sección anterior (I.1), ya que se pretender establecer la relación que existe entre la calidad del proceso de software y la calidad del producto de software por medio de métodos cuantitativos. También, esta investigación se basa en uno de los temas principales marcados en el Consorcio International de Investigación de Procesos (IPRC, por sus siglas en ingles) (Oktaba 2006) convocado por el Instituto de Ingeniería de Software (SEI, por sus siglas en ingles) en el 2004, donde uno de los principales objetivos es construir un mapa-guía de la investigación en procesos de software para los próximos 10 años, enfocándose en la relación entre proceso y la calidad del producto, así como el modelado de la relación calidad del Producto-Proceso, en especial tratando de contestar las siguientes preguntas: (1) ¿Existe una relación directa entre alguna cualidad del producto y el proceso utilizado? y (2) ¿Cómo modelar y predecir la relación entre una cualidad del producto y el proceso?.

Por otro lado, el presente trabajo realizado apoya a la iniciativa del programa para el desarrollo de la industria del software (PROSOFT) en México (Prosoft 2010), en la cual uno de sus objetivos es “Impulsar a la industria de software y extender el mercado de tecnologías de información en nuestro país”, a través de la estrategia “Alcanzar niveles internacionales en capacidad de procesos.”

Estos objetivos se clasifican en dos niveles: Objetivos generales y objetivos específicos. También en la siguiente sección se describen las metas para cada objetivo específico así como las hipótesis que serán guía para el desarrollo de la investigación y que

proponen tentativamente las respuestas a las preguntas de investigación que se obtuvieron del planteamiento del problema.

I.2.1 Objetivos generales.

- Crear un modelo para establecer la relación que existe entre el proceso de software y los atributos de calidad del producto de software.
- Determinar el impacto que tiene el proceso de software sobre la calidad del producto de software generado.

I.2.2 Objetivos específicos

Objetivo específico 1 (OE1): Determinar cuáles atributos del proceso (actividad, artefacto, rol y herramientas) impactan la calidad del producto.
Objetivo específico 2 (OE2): Precisar la relación que existe entre las actividades del proceso y la mejora de la calidad del producto generado.

I.2.3 Metas

Las metas por alcanzar en esta investigación son las siguientes:

- **Meta 1** (M1) (OE1): Identificar los elementos de composición del proceso y los atributos de calidad del producto de software mediante los cuales se puede establecer la relación entre ellos.
- **Meta 2** (M2) (OE2): Definir un Metamodelo en el que se identifique la relación entre un proceso y el producto a través de los atributos de calidad.
- **Meta 3** (M3) (OE2): Probar el modelo propuesto en proyectos dentro del ámbito académico y empresarial.
- **Meta 4** (M4): Publicar los resultados obtenidos en foros nacionales e internacionales.

I.2.4 Hipótesis

En esta sección se describen las principales hipótesis a manejar en este trabajo de investigación. Las hipótesis expuestas están relacionadas con la calidad del producto y con cada uno de los elementos estructurales del proceso.

- **H $_{(0,1)}$** Si un proceso está definido a su nivel más fino de granularidad (tarea), la identificación de la causa del defecto del producto es más directa.
- **H $_{(0,2)}$** La calidad del producto final de software es impactada por las Actividades Base del proceso que generan artefactos tangibles.
- **H $_{(0,3)}$** La calidad del producto final de software es menos impactada por las actividades de apoyo del proceso que no generan artefactos tangibles.
- **H $_{(0,4)}$** La calidad del producto final de software está directamente relacionada con los artefactos tangibles de tipo contribuidor directo.
- **H $_{(0,5)}$** La calidad del producto final de software está indirectamente relacionada con los artefactos tangibles de tipo contribuidor indirecto.
- **H $_{(0,6)}$** La calidad del producto final de software está estrechamente relacionada con la ejecución correcta de las tareas del proceso de software.

I.3 Marco de trabajo

En esta sección se describe el marco de trabajo que permite visualizar el contexto dentro del cual serán desarrollados los mecanismos especificados en los objetivos, la metodología de trabajo, así como las condiciones para la experimentación. Para soportar esta descripción, primeramente se trataran unos conceptos básicos de la investigación empírica.

La investigación empírica puede ser definida como la investigación basada en la observación para descubrir un hecho desconocido o probar una hipótesis (Kallakuri 2000). La investigación empírica involucra la recolección de datos, mismos que se analizan para determinar el significado de dichos datos, teniendo como base las siguientes estrategias de investigación:

- Un experimento provee al investigador el control sobre algunas condiciones en las cuales el estudio toma lugar, manipulando factores independientes para obtener respuesta sobre los factores dependientes.
- Una observación de un espectro de datos, a través de anécdotas o casos de estudio, la cual investiga fenómenos de la vida real en el contexto de una teoría actual.
- Una demostración de tecnología sobre temas o aspectos selectos.

En base a esto, se puede decir que los investigadores asumen que los conceptos que describen y prescriben los medios óptimos de construir software pueden ser descubiertos y establecidos, (Wohlin 2007) esto es, los conocimientos se formulan mediante una investigación pura y/o a través de la observación de fenómenos en la práctica del desarrollo de software y se establecen mediante la puesta en práctica y la aprobación de los mismos por los practicantes. En (Pfleeger 1999) se argumenta que las bases de toda actividad de ingeniería de software empírica deben guiarse por la siguiente idea: "*Si vemos a profundidad y suficientemente, podemos encontrar las reglas que nos muestren las mejores formas de construir software de calidad*". Esto quiere decir que realizando los fenómenos observados que se analizan en la práctica y se da un tratamiento profundo a las observaciones realizadas y a los resultados, se podrá crear nuevos mecanismos, metodologías y herramientas que faciliten la creación de software con calidad (Juárez-Ramírez 2008).

Por otro lado, el progreso científico está fundado en el estudio y establecimiento de las discrepancias entre el conocimiento y la realidad. En (Knight J. C. y Brilliant 1998) se argumenta que para establecer la validez empírica de una teoría, es necesario probarla en varios aspectos y entidades de diferentes dominios.

En base a esto, es posible establecer los siguientes puntos en los que se concentra este trabajo de tesis: metodología propuesta para el desarrollo de la tesis y el ambiente de experimentación.

I.3.1 Metodología de trabajo

Para el desarrollo de la presente investigación se usó la siguiente metodología.

1. La primera fase del desarrollo de la investigación fue el estudio del estado del arte sobre los conceptos de calidad del proceso y del producto de software, así como el estudio del tema de métricas de software. También se analizaron los modelos y estándares relacionados con el tema de calidad del producto y del proceso de software.

2. Formulación de las hipótesis a desarrollar en la investigación, con el propósito de validar los objetivos y metas establecidas para esta investigación.

3. Desarrollo del Metamodelo que representa la relación entre el proceso y el producto de software a través del modelo de calidad, utilizando el Lenguaje de Modelado Unificado (UML, por sus siglas en inglés). En base a esto, se definió la metodología que identifica la relación entre los elementos del proceso y la calidad del producto de software.

4. Realización de casos de estudio, usando la metodología investigación-acción:

- Para la elección de los casos de estudio se optó por proyectos de desarrollo de software con diferentes metodologías de desarrollo y diferente nivel de especialización en el tema: Maestría y Licenciatura, así como proyectos de empresas dedicadas al desarrollo de software.
- Para cada caso de estudio se realizarán los siguientes actividades:
  - Evaluación de la calidad del producto final, utilizando las métricas de calidad que establece ISO 9126-3.
  - Obtención de los resultados de la evaluación de la calidad del producto de software.
  - Identificación de las posibles causas de fallo del producto evaluado, utilizando el método de Análisis de Modos y Efectos de Fallos (FMEA, por sus siglas en inglés).

- Mapeo de las posibles causas de fallo con las actividades y tareas del proceso de desarrollo.
- Evaluación de la calidad del proceso, aplicando métricas de calidad del proceso en las actividades y tareas identificadas como posibles causas de fallo, así como del proceso completo.
- Procesar los datos obtenidos con métodos estadísticos.

5. Descripción de los resultados de los casos de estudio y conclusiones obtenidas, así como trabajo a futuro.

I.3.2 Contexto de la experimentación

Wohlin (Wohlin 2007) argumenta que la ingeniería de software empírica esta directamente asociada con la medición. Las mediciones son trabajadas en casos de estudio, los cuales pueden variar desde experimentos controlados hasta estudios de campo, y pueden ser estudiados de carácter cualitativo o cuantitativo. Así mismo, (Wohlin 2007) indica que las mediciones y los datos deben ayudar a entender, evaluar y modelar un fenómeno en la ingeniería de software son: experimentos controlados, casos de estudio, inspecciones, análisis de datos históricos e investigación de acción. Por otro lado, señala que los estudios empíricos deben ser manejados en forma sistemática, de tal manera que generen resultados significativos.

Para la realización de este tipo de estudios, (Hearn 2004), (Kallakuri 2000), (Maicher 2005) y (Pfleeger 1997) sugieren el uso de una guía de preguntas para conducir la investigación. Algunas de estas preguntas son: ¿Qué aspectos de la ingeniería de software pueden ser investigados usando mi metodología/tecnología?, ¿Qué pasa cuando los usuarios emplean mi metodología/tecnología?, ¿Es efectiva mi metodología/tecnología?

En base a estas sugerencias descritas anteriormente, la experimentación para este trabajo de tesis se realizó en el ámbito académico y en el ámbito industrial.

En el ámbito académico, se utilizó dos materias a nivel Licenciatura de las carreas de Ingeniero en Computación (IC), una de ellas del plan de estudios IC de la Universidad

Autónoma de Baja California (UABC), y la segunda materia del plan de estudios de IC de la Universidad Nacional Autónoma de México (UNAM). A nivel Maestría, se utilizó una materia del Posgrado en Ciencias e Ingeniería de la Computación de la UNAM.

En el ámbito industrial, se utilizaron dos proyectos realizados por distintas empresas: (a) Empresa dedicada al desarrollo de Software a la medida. (b) Empresa dedicada a dar el servicio de Telemetría, la cual tiene dos partes en su actividad, una parte de servicios de hardware y una parte de desarrollo de software conectado al funcionamiento del hardware.

Los métodos de experimentación a utilizar son principalmente en base al estudio y análisis de datos históricos.

I.4 Estructura del trabajo de tesis

La estructura de este trabajo de tesis, después de la introducción, se describe a continuación:

Capítulo II: En este capítulo se describen los conceptos básicos sobre los elementos del proceso de software y del producto de software, modelos de calidad, ontología de métricas de software y se hace un análisis breve sobre el estado de arte.

Capítulo III: En este capítulo se describe el Metamodelo propuesto para establecer la relación entre el proceso de software y el producto de software a través de las métricas de calidad.

Capítulo IV: En este capítulo se expone la experimentación realizada respecto a las hipótesis planteadas, en términos: (1) las tareas de un proceso de software están directamente relacionadas con la calidad del producto de software final, (2) los artefactos generados por las tareas de un proceso de software impactan en la calidad del producto final, (3) la efectividad de las tareas del proceso de software impactan en la calidad del producto final.

Capítulo V: En este capítulo contiene la discusión de los resultados más significativos obtenidos en la experimentación.

Capítulo VI: Este capítulo contiene las conclusiones y el trabajo a futuro a realizar.

Capítulo VII: En este apartado contiene un listado de las referencias bibliográficas consultadas para la realización del trabajo de tesis.

Capítulo VIII: Este apartado contiene los siguientes apéndices: (1) Apéndice A: *Herramientas de investigación,* (2) Apéndice B: *Métricas de calidad del producto ISO 9126,* (3) Apéndice C: *Análisis de modos y efectos de fallas (FMEA, por sus siglas en ingles),* Apéndice D: *Srum,* Apéndice E: *Plantilla de retrospectiva Srum y* Apéndice F: *Descripción del Proceso de Administración de Proyectos Específicos (APE) y del procesos de Desarrollo y Mantenimiento de Software (DMS), de la Norma Mexicana NMX-I-059/03-nyce-2005.*

CAPÍTULO II

CONCEPTOS BÁSICOS Y ESTADO DEL ARTE

II Conceptos básicos y estado del arte

II.1 Calidad del software

El interés por la calidad del software crece de forma continua debido a que los clientes se vuelven más selectivos y comienzan a rechazar los productos poco fiables o que realmente no satisfacen sus necesidades. Ahora bien, ¿Qué es calidad? Según Deming (Deming 1988), calidad es "Conformidad con los requisitos y confianza en el funcionamiento". Por su parte Crosby (Crosby 1979) pone más énfasis en la prevención de defectos: "hacerlo bien a la primera".

El IEEE Standard 729-1983 (International Organization for Standardization 1983) establece que la calidad del software es el grado con el cual el usuario percibe el software satisfaciendo sus expectativas. Por otro lado, el ISO 8402 (International Organization for Standardization 1990) establece que la calidad es la suma de todos aquellos aspectos o características de un producto o servicio que influyen en su capacidad para satisfacer las necesidades, expresadas o implícitas.

El concepto de calidad de software tiene varias definiciones formales. Algunas de ellas ponen mayor énfasis en el producto y otras en el cliente o el servicio. Sin embargo, todas tienen un mensaje común, que es el de obtener una mejora, ya sea en un producto software, o los procesos que llevan a la creación del mismo (Instituto Nacional de Tecnologías de la Comunicación 2008).

Para aumentar la calidad se establecen una serie de metodologías formales y buenas prácticas que permiten tener control sobre lo que está ocurriendo a distintos niveles y así poder evaluarlo y mejorarlo. Gracias a ellas, se consigue que el producto final tenga las características que se desean (funcionalidad, usabilidad, mantenibilidad, fiabilidad, etc.) y que los procesos asociados se vean impactados, así como los tiempos de entrega disminuyan, los costos se reduzcan y la satisfacción del cliente final aumente. Utilizando métricas y herramientas ayudan a la mejora de la calidad (Instituto Nacional de Tecnologías de la Comunicación 2008).

Los expertos en calidad sugieren adoptar los pasos siguientes para mejorar la calidad de software (Gérald Lomprey 2008):

1. Establecer un sistema de evaluación y medición para determinar en qué medida se está actuando correctamente.
2. Documentar el proceso de desarrollo actual, por más caótico que sea.
3. Calcular el coste de corregir errores de software en términos de las mercancías o servicios vendidos por la compañía.
4. Hacer lo necesario para eliminar errores en requerimientos y diseño.
5. Entrevistar cuidadosamente a usuarios, directores y verificar que se ha comprendido lo que necesitan.
6. Realizar pruebas pronto y con frecuencia.

Por lo tanto, para conseguir una buena calidad del software es esencial establecer un programa de medidas a tomar con respecto a los proveedores. Es también importe utilizar los modelos y métodos apropiados para controlar el proceso de desarrollo del mismo.

II.2 Proceso de software

Según (Fuggetta 2000) un proceso de software es un conjunto coherente de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, instalar y mantener un producto software.

Por otro lado, según, (Satpathy 2002)y (Ortega 2003) un proceso es un conjunto de actividades, en la cual cada una de ellas se relaciona desde el análisis de requisitos hasta mantenimiento del software. Cada una de estas actividades toma productos como entrada y genera productos de salida.

En el caso de 15504-3 (International Organization for Standardization 2004) indica que el proceso de software es el proceso (o procesos), usado por una organización (o proyecto) para planificar, administrar, ejecutar, monitorear, controlar y mejorar las actividades relacionadas con el software.

Cada proceso está constituido por actividades, en las que cada una de ellas está compuesta por tareas (Véase Figura 1), en donde se puede observar que un proceso se compone por varias actividades y estás por un conjunto de tareas.

Una tarea es un conjunto de elementos o acciones atómicas. Una tarea consume una entrada (datos, información, controles) y produce una salida (datos, información, controles), (International Organization for Standardization 2002).

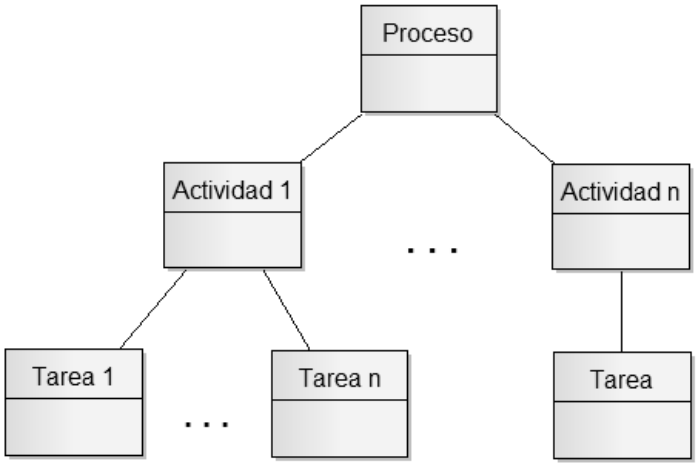

Figura 1: Composición estructural de actividad.

En (Dr. Mario Piattini Velthuis 2008-2009) se menciona que en una empresa o en un dominio de aplicación, los procesos de diferentes proyectos tienden a seguir patrones comunes, o bien se puede llamar "mejores prácticas", de las cuales son establecidas en los estándares. Por lo tanto se hace necesario intentar capturar estos aspectos comunes en una representación de proceso, la cual describe estas características comunes y fomenta la homogeneidad.

Según (Jean-Claude Derniame 1999) el proceso software es un concepto más amplio, basado en el de ciclo de vida, y cubre todos los elementos necesarios (tecnologías, personal, artefactos, procedimientos, etc.) relacionados con las actividades involucradas en la vida de un producto software, y por tal motivo, un proceso de software, por su naturaleza especial, se distinguen por las siguientes características:

1. Es complejo.

2. No es un proceso de producción típico.
3. No es un proceso de ingeniería “pura”.
4. No es (completamente) un proceso creativo.
5. Está basado en descubrimientos que dependen de la comunicación, coordinación y cooperación dentro de marcos de trabajo predefinidos.

II.3 Producto de software

El producto, según IEEE-12207 (International Organization for Standardization 2002), es el conjunto de programas de computadora, documentación y datos asociados. También se define como una entidad (Satpathy 2002), la cual un proceso genera como salida (producto salida) y también puede alimentar a un proceso como entrada (producto de entrada).

También se puede definir como producto de software como un subconjunto de artefactos de un proyecto de software incluyendo, desde la especificación de requerimientos, hasta el código ejecutable, desarrollado y liberado para un cliente (Juárez-Ramírez 2008).

Un artefacto de software es cualquier pieza de software desarrollada y usada durante el proceso de desarrollo y el mantenimiento. Algunos ejemplos de artefactos son: especificación de requerimientos, arquitectura, y modelos de diseño, planes de prueba, reportes de pruebas, modelos de procesos, planes de proyecto, y entidades de documentación.

Por lo tanto un producto de software *P* puede ser representado de la siguiente manera (Juárez-Ramírez 2008):

$$P= \{A_1 ..... A_n\}$$

Donde A_1 representa el i-ésimo artefacto que forma parte de un producto de software.

El producto tiene atributos que pueden ser observables y medibles.

Un atributo es una propiedad de un artefacto. Cada atributo tiene un nombre y puede tener un valor. Un artefacto tiene un conjunto de atributos que permite su identificación describiendo su contenido y su naturaleza (Juárez-Ramírez 2008).

II.4 Modelo de procesos de software

Un modelo de procesos de software es una abstracción o representación (textual, grafica o formal) en la que se capturan los aspectos más importantes de un proceso de software. Es una representación descriptiva de (Montilva 2007): las actividades, los recursos, los productos, los actores y las reglas que el proceso requiere para alcanzar sus objetivos.

El modelo un proceso de software es una actividad mediante la cual un proceso de software es representado o definido usando lenguajes apropiados que faciliten la comunicación de esa representación, documentación, la comprensión del proceso (Montilva 2007).

Uno de los marcos de referencia más representativos respecto a modelos de procesos es Metamodelo de Ingeniería de Procesos de Software versión 2 (SPEM 2, por sus siglas en inglés), formulado por la "Object Management Group" (OMG, por sus siglas en inglés) (Group 2008). SPEM es un Metamodelo modelo genérico para la definición de procesos software y está basado en el meta-Metamodelo modelo universal de Meta Object Facility (MOF, por sus siglas en inglés), que es una norma aprobado por el OMG para la definición, representación y gestión de metadatos de los procesos de software.

SPEM 2 sirve para definir procesos de desarrollo de software y sistemas y sus componentes. Su alcance se limita a los elementos mínimos necesarios para definir dichos procesos sin añadir características específicas de un dominio o disciplina particular.

El objetivo principal de SPEM 2 es representar procesos de software basándose en tres elementos básicos: rol, producto de trabajo y tarea (véase la Figura 2).

Las tareas representan el esfuerzo a hacer, los roles representan quien lo hace y los productos de trabajo representan las entradas que se utilizan en las tareas y las salidas que se producen. Por lo que la idea principal es que en un modelo de procesos esté representada

por sus elementos básicos: quién realiza (rol) qué (tarea) para, a partir de unas entradas (productos de trabajo) obtener unas salidas (productos de trabajo), (Ruiz 2008).

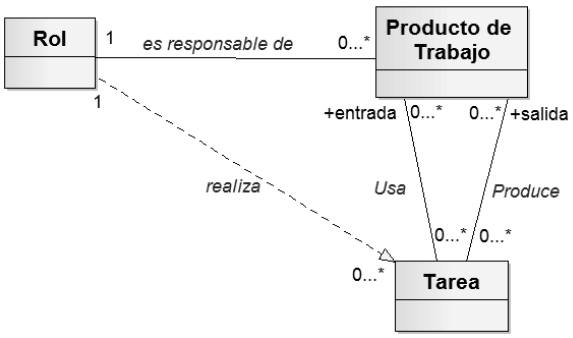

Figura 2: Elementos básicos de un proceso.

SPEM 2 establece una jerarquía de desglose de sus elementos para representar el esfuerzo a realizar a distintos niveles de detalle. A continuación se describen cada uno de ellos, del más general al más particular (vease Figura 3).

- Proceso de despliegue: Representa un proceso tan complejo como se necesite, que será el que sirva de base para realizar cierto tipo de proyectos.
- Patrón de capacidad: Representa un patrón de proceso, es decir, un fragmento de proceso que puede ser reutilizado más de una vez en un proceso de despliegue.
- Actividad: Es el elemento central para definir procesos ya que permite organizar los elementos básicos de proceso (roles, productos de trabajo y tareas).
- Tarea: Es la porción más pequeña de trabajo en un modelo de procesos en SPEM2.
- Un Producto de Trabajo: Es consumido, producido o modificado por tareas, (Group 2008).

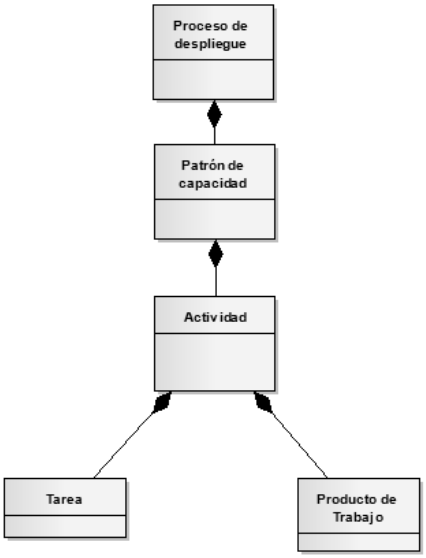

Figura 3: Estructura jerárquica del proceso de software.

Los elementos básicos que SPEM 2 utiliza como constructores y que de estos se basan para definir patrones, son: Tarea, Rol, Producto de Trabajo, Guía y Categoría.

II.4.1 Tarea

Una tarea describe una unidad de trabajo asignable y gestionable, es decir, es la unidad atómica de trabajo para definir procesos. Su granularidad es de unas pocas horas a unos pocos días, afectando a unos pocos productos de trabajo y vinculando a unos pocos roles.

Una tarea está asociada con varios roles, distinguiendo entre: realizador principal obligatorio (responsable) y realizados adicionales opcionales. Así como los Productos de Trabajo como: Entrada obligatoria, entradas opcionales y Salidas.

También las tareas se relacionan con herramientas que se recomienda usar y pasos que describen de forma secuencial el trabajo a realizar, así como habilidades que requieren habitualmente para llevar a cabo la tarea.

II.4.2 Rol

Un rol define un conjunto de habilidades, competencias y responsabilidades relacionadas, de un individuo o de un grupo.

Un rol está asociado con: Productos de Trabajo de los que es responsable, tareas de las que es responsable y habilidades, que el rol típicamente provee.

II.4.3 Producto de trabajo

Un Producto de Trabajo es producido o modificado por tareas. Es el elemento final que se desea obtener después de finalizar una actividad.

Un producto está asociado con: Roles, distinguiendo entre realizador principal obligatorio (responsable) y realizadores adicionales opcionales. Así como con las Tareas de entrada a otras tareas, Salidas de otras tareas o Productos de Trabajo.

Un Producto de Trabajo puede estar asociado con otros productos de Trabajo mediante asociaciones de los siguientes tipos:

- Composición, cuando las instancias de un producto de trabajo sirven para componer instancias de otro producto de trabajo. Ejemplo: “Actores” se emplean para componer “Casos de uso”.
- Agregación, si un producto de trabajo está formado por agregación de otros. Ejemplo: el “Manual de usuario” incluye el “Manual de instalación”.
- Es impactado por, cuando un producto de trabajo impacta en otro, es decir, si los cambios del primero obligan a cambiar el segundo. Ejemplo: si cambia el “Modelo de casos de uso”, es necesario adaptar a dicho cambio la “Realización de casos de uso”.

Existen tres tipos predefinidos (especializaciones) de Productos de Trabajo:

- Artefacto: Es de naturaleza tangible (modelo, documento, código, archivos, etc.). Un artefacto puede estar formados por otros artefactos más simples.
- Entregable: Provee una descripción y definición para empaquetar otros productos de trabajo con fines de entrega a un cliente interno o externo. Representa una salida de

un proceso que tiene valor para un usuario, cliente u otro participante (stakeholder). Está asociado con componentes de entregable, que son los productos de trabajo, habitualmente artefactos, que lo forman.

- Resultado: Un producto de trabajo de naturaleza intangible o que no está formalmente definido.

II.4.4 Propiedades

Todos los elementos de un proceso tienen las siguientes propiedades: nombre, nombre de presentación, descripción breve, descripción principal.

Otras propiedades específicas de cada elemento son:

- Tarea: Objetivo, factores clave, alternativas, lista de pasos que detalla el trabajo a realizar de forma ordenada.
- Rol: Factores clave, habilidades, propuestas de asignación, sinónimos.
- Producto de trabajo: ID exclusivo, objetivo factores clave, impacto de no tener motivos para no necesitar.
- Los Artefactos tienen esquematización breve y opciones de representación.
- Una Práctica tiene información adicional, objetivos, aplicación, problema, fondo, nivel de adopción, referencias, que es una lista de elementos de contenido referenciados.

II.4.5 Asociaciones

SPEM 2 define asociaciones básicas entre los elementos básicos (tarea, producto de trabajo y rol), de los cuales a continuación se describen:

Las asociaciones permitidas en SPEM 2 que pueden existir entre los diversos tipos de elementos son los siguientes:

- Tareas-Pasos: Lista ordenadas de pasos que se llevan a cabo en una tarea.

- Tareas-Roles: Relación entre el realizador principal con una tarea. También puede ser, realizadores adicionales.
- Tareas-Productos de Trabajo: Las tareas se basan de los productos de entrada obligatoria u opcional y generan productos de salida.
- Rol-Productos de Trabajo: Los productos de trabajo que son salida de tareas, lo realiza un rol, es decir el responsable de este producto.

Otras relaciones secundarias que pueden representar en SPEM2 de manera directa o indirecta son:

- Tarea-herramienta: Herramientas que se utilizan para hacer una tarea.
- Herramienta-Producto de Trabajo: Productos de trabajo que se producen utilizando una herramienta.
- Rol-Habilidades: Habilidades que posee un rol.
- Tarea-Habilidades: Habilidades requeridas para realizar una tarea.
- Producto de trabajo-Productos de trabajo: Un producto de trabajo (fuente) es necesario para poder producir otro producto de trabajo (generado).

II.4.6 Actividades

Una Actividad representa una unidad de trabajo general en un proceso. Una actividad puede tener estructura interna formada por agregación de elementos de desglose, que pueden ser de varios tipos, no solo de trabajo: actividades más simples (anidamiento), elementos de método en uso (roles en uso, productos de trabajo en uso), e hitos.

La complejidad de la estructura de desglose de una actividad puede variar entre 0 tareas (pueden estar formadas sólo por productos de trabajo en uso) o todo un proceso completo.

Una Actividad puede estar asociada con varias Actividades en Uso, que reutilizan la primera. Para ello, existen varias maneras de reutilización (herencia).

Una actividad representa una unidad de trabajo general asignable a realizadores específicos, representados por Roles en Uso. Dicha asignación se hace a través de Realizadores de Proceso.

Una actividad puede tener entradas y producir salidas, representadas por Productos de Trabajo en Uso. Para ésta última asociación se utilizan los llamados Parámetros de Proceso.

La Figura 4 muestra estos vínculos entre actividades y otros elementos de proceso.

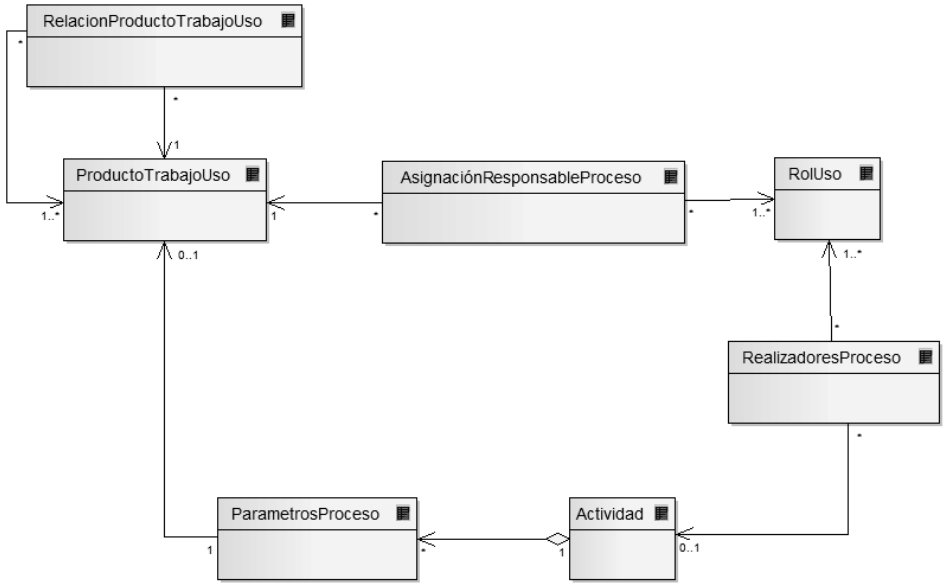

Figura 4: Relación entre elementos del proceso.

II.5 Modelo de calidad del producto

Los modelos de calidad están orientados a la satisfacción de los clientes y mejora de los procesos mediante la definición de principios y prácticas que conducen a mejores productos de software (Mario G. Piattini 2003). Aunque la calidad es un objeto importante para cualquier producto, no se puede olvidar que los productos de software, se construyen para ser utilizados. Por lo tanto, el principal objetivo de un producto es satisfacer una necesidad de un usuario, por consiguiente, ofrecer al usuario algún beneficio por su utilización.

La especificación de la calidad del software debe ser más detallada y exacta y el camino para ello es la formalización de la calidad mediante un modelo de calidad que

define las características de un producto que influyen a la hora de medir su calidad (Mario G. Piattini 2003).

Un Modelo de calidad debe tener los siguientes tres requerimientos:

- Un modelo de calidad debe de utilizarse desde el inicio hasta el fin del ciclo de vida, así como lo define IEEE 1061-1998 (International Organization for Standardization 1998), debe permitir la definición de los requerimientos de calidad y acentuar más en la clasificación apropiada de características, sub-características de calidad y métricas.
- Un modelo de calidad debe permitir las requeridas mediciones y subsiguiente agregación y evaluación de los resultados obtenidos (Marc-Alexis Côté 2008).

El proceso del modelado conceptual proporcionan el enlace entre las necesidades del usuario y la solución del software que las satisface (Chen P. 1999). Por lo tanto considerando el modelo conceptual en un sentido más amplio, podemos definirlo como la búsqueda y definición formal del conocimiento general sobre un dominio que un sistema de información (SI) necesita conocer para llevar a cabo las funciones requeridas. (Olivé 2000).

Algunos autores han definido la calidad de los modelos conceptuales como una lista de propiedades "deseables" que debe satisfacer el modelo de datos y han propuesto una serie de transformaciones con el objetivo de mejorar la calidad de los mismos. Pero aun así, el definir propiedades deseables no es suficiente para evaluar la calidad, ya que diferentes personas pueden dar diferentes interpretaciones sobre el mismo concepto, por lo que es necesario contar con medidas que permitan evaluar la calidad de los modelos conceptúeles de datos de forma cuantitativa y objetiva, evitando así los sesgos en el proceso de evaluación de su calidad. (D Moody 1998).

II.5.1 Estructura de un modelo de calidad del software

Los modelos de calidad SQuaRe (International Organization for Standardization 2005) e ISO 9126 (International Organization for Standardization 2001), categorizan la

calidad del software en características de las cuales se subdividen en sub características y estas en atributos específicos de calidad.

Siguiendo esta terminología, se entiende por característica de calidad de un producto de software, como un conjunto de propiedades mediante las cuales se evalúan y describe su calidad. Una sub-característica refina una propiedad considerada en la característica.

Un atributo es una propiedad de calidad a la que puede asignar una métrica, entendiendo por métrica un atributo que evalúa un componente y produce un dato simple, un símbolo o un número (International Organization for Standardization 2005).

Por lo que se puede decir que un modelo de calidad se define como un conjunto de características y sub-características, junto con las relaciones que existen entre ellas. La Figura 5 describe los niveles jerárquicos de un modelo de calidad.

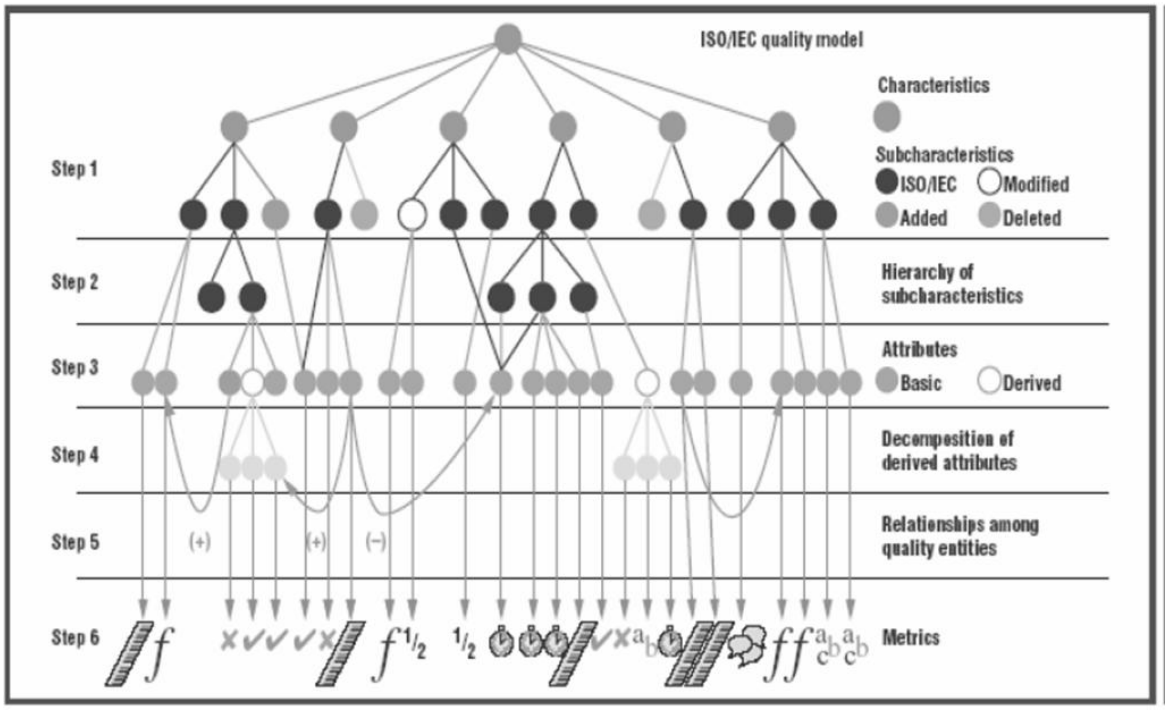

Figura 5: Jerarquía del modelo de calidad.

II.5.2 Modelos de calidad de software

Existen varios modelos de calidad de software que han surgido gradualmente: El modelo McCall, el modelo Boehm, Modelo Dromey, ISO 9126 y modelo de calidad sistemático, etc., cada uno de estos modelos define una estructura jerárquica, en donde un concepto se deriva de un conjunto de sub-conceptos, y cada uno de estos se va a evaluar a través de un conjunto de indicadores o métricas.

A continuación se describe cada uno de estos modelos de calidad, resaltando lo que cada una de ellos ha aportado.

II.5.2.1 Modelo McCall

El modelo de calidad McCall fue introducido en 1977, es uno de los primeros modelos de su tipo. Su principal enfoque es sobre los desarrolladores y el proceso de desarrollo. (Jetter 2006)

El modelo organiza los factores en tres ejes o puntos de vista desde los cuales el usuario puede contemplar la calidad de un producto, basándose en once factores de calidad organizados en torno a los tres ejes y a su vez cada factor se desglosa en otros criterios (Jetter 2006):

Revisión del Producto:

- Mantenibilidad: Modularidad, simplicidad, consistencia, concisión y auto descripción
- Flexibilidad: Auto descripción, capacidad de expansión, generalidad y modularidad.
- Facilidad de Pruebas: Modularidad, simplicidad, auto descripción y instrumentación.

Transición del Producto:

- Reusabilidad: Auto descripción, generalidad, modularidad, independencia entre sistema y software e independencia del hardware.
- Interoperabilidad: Modularidad de comunicaciones, compatibilidad de datos y estandarización en los datos.
- Portabilidad: Auto descripción, modularidad, independencia entre sistema y software e independencia del hardware.

Operación del Producto:

- Usabilidad: Facilidad de operación, facilidad de comunicación, facilidad de aprendizaje y formación.
- Integridad: Control de accesos, facilidad de auditoría y seguridad.
- Correctud: Completud, consistencia, trazabilidad o rastreabilidad.
- Confiabilidad: Precisión, consistencia, tolerancia a fallos, modularidad, simplicidad y exactitud.
- Eficiencia: Eficiencia en ejecución y eficiencia en almacenamiento.

II.5.2.2 Modelo Boehm

Modelo Boehm fue introducido en 1978, es similar al modelo de McCall ya que se representa con una estructura jerárquica de característica, en la que cada una de ellas contribuye a la calidad total (Maryoly Ortega 2001).

Básicamente este modelo trata de definir calidad del software cuantitativamente utilizando un conjunto de atributos y métricas.

Este modelo introduce características de alto nivel, intermedio y primitivas, cada uno de las cuales contribuye al nivel general de calidad. (Jetter 2006)

Las características de alto nivel representan requerimientos generales de uso: Utilidad, mantenibilidad y portabilidad.

Las características de nivel intermedio representan los factores de calidad, las cuales son (Jetter 2006): Portabilidad (Utilidad general), confiabilidad, eficiencia, usabilidad, mantenibilidad, fácil de probar, facilidad de entendimiento y flexibilidad

El nivel más bajo corresponde a características directamente asociadas a una o dos métricas de calidad, las cuales son: Portabilidad, confiabilidad, eficiencia, usabilidad y entendible.

A diferencia del modelo de McCall, este modelo incluye las necesidades del usuario y también se agregó característica de hardware.

II.5.2.3 Modelo Dromey

El modelo Dromey busca mejorar el entendimiento de la relación entre los atributos (características) del producto y las sub-características de calidad. Además trata de localizar las propiedades del producto de software que afecta a los atributos de calidad.(Jetter 2006)

Los modelos antes mencionados no establecen una conexión entre los atributos de calidad y las características del producto. Este modelo se enfoca principalmente en el producto de software, el código, por lo que propone un único nivel de propiedades de calidad entre un alto nivel de atributos de componentes del producto (véase Figura 6), (Jetter 2006).

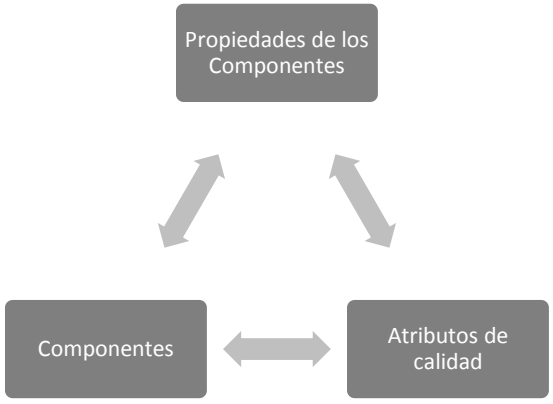

Figura 6: Modelo Dromey.

Este marco de trabajo establece mejor un modelo abajo-arriba, es decir por cada componente de calidad-propiedades son identificables, será más importante garantizar los atributos de alto nivel.

Dromey propone un conjunto de propiedades. En la Tabla 2 se presenta un resumen de las propiedades estructurales.

Tabla 2: Propiedades estructurales.

Propiedades de Exactitud	Propiedades de Estructural	Propiedades de Modularidad	Propiedades Descriptivas
Computable	Estructurado	Parametrizada	Especificado
Completo	Resuelto	Débilmente acoplados	Documentado
Asignado	Homogéneos	Encapsulado	Auto-descriptible
Preciso	Efectivo	Cohesivo	
Inicializado	No redundante	Genérico	
Progresivo	Directo	Abstracto	
Variante	Ajustable		
Consistente	Rango-Independiente		
	Utilizado		

Como un ejemplo se puede analizar la propiedad de calidad “Asignado”. Una variable es asignada si recibe un valor antes de ser usada. Por lo que la propiedad “Asignada” puede ser aplicada a variables. Esto significa, si en el código las variables están asignadas, se podrá desarrollar un producto de calidad (Jetter 2006).

Sobre esta base la conexión de los atributos de los niveles altos pueden ser construidos similarmente. Las propiedades de los productos caracterizan los requerimientos que pueden ser satisfechos para construir un nivel alto en los atributos de calidad del software. La parte más difícil de esto es la evaluación cuando las propiedades del producto no tienen una influencia significante en el atributo de calidad (Jetter 2006).

Es por eso, que Dromey selecciona una de la lista de atributos de calidad similares a los de ISO 9126 (Funcionabilidad, confiabilidad, usabilidad, eficiencia, Mantenibilidad y Portabilidad). Como una extensión al ISO 9126, Dromey añade el atributo reusabilidad, ya que lo considera importante (Jetter 2006).

II.5.2.4 Estándar ISO/IEC 9126

La Organización Internacional de Estándares (ISO) en conjunto con la Comisión Electrotécnica Internacional (IEC) propuso un estándar para la evaluación de la calidad del software, denominado ISO 9126.

El estándar ISO/IEC 9126 se compone de cuatro partes: modelo de calidad (International Organization for Standardization 2001), métricas externas (International Organization for Standardization 2003), métricas internas (International Organization for Standardization 2003) y métricas para la calidad en uso (International Organization for Standardization 2001). En la primera parte, se describen detalladamente seis características y sub-características de calidad para los productos de software (véase Tabla 3), (Marín 2007).

Tabla 3: Características y sub-cacacterísticas de calidad.

Característica	Sub-característica
Funcionalidad	Adecuación.
	Exactitud.
	Interoperabilidad.
	Seguridad.
	Adherencia a normas.
Confiabilidad/Fiabilidad	Madurez.
	Tolerancia a fallos.
	Recuperabilidad.
	Adherencia a normas.
Usabilidad	Fácil comprensión.
	Fácil aprendizaje.
	Operatividad.
	Software atractivo.
	Adherencia a normas.
Eficiencia	Comportamiento frente al tiempo.
	Uso de recursos.
	Adherencia a normas.
Mantenibilidad	Facilidad de análisis.
	Capacidad para cambios.
	Estabilidad.
	Facilidades para pruebas.
	Adherencia a normas.
Portabilidad	Adaptabilidad.
	Facilidad de instalación.
	Coexistencia.
	Facilidad de reemplazo.
	Adherencia a normas.

A continuación se describe cada una de las características de calidad establecidas:

Característica de funcionalidad: corresponde a la capacidad de los productos de software de proveer las funciones que satisfacen las necesidades de un dominio en particular. Esta característica en los modelos conceptuales implica que el modelo tiene todas las funciones necesarias para que puedan ocurrir los hechos del dominio, (Marín 2007).

Esta característica del software puede ser desglosada en varias sub-características:

- Adecuación. Capacidad del software de proporcionar un conjunto apropiado de funciones para tareas específicas y objetivos del usuario.
- Exactitud. Capacidad del software para proporcionar resultados correctos o que necesitan un determinado grado de precisión.
- Interoperabilidad. Capacidad del software de interaccionar con uno o más sistemas especificados.
- Seguridad. Capacidad del software de proteger la información y los datos.
- Adherencia a normas. Capacidad del software relacionada con el grado de conformidad con estándares, convenciones o regulaciones existentes.

Característica de confiabilidad/fiabilidad: consiste en mantener el nivel de rendimiento en condiciones específicas. Dado que el rendimiento se puede considerar como un hecho del dominio del modelo, la característica de confiabilidad concierne a un aspecto de la calidad semántica, es decir, a una correspondencia entre el modelo y el dominio (Marín 2007).

Se descompone en las siguientes sub-características:

- Madurez. Capacidad del software para evitar fallos como resultados de defectos del software.
- Tolerancia a fallos. Capacidad del software para mantener un nivel especificado de rendimiento en casos de fallos del software.
- Recuperabilidad. Capacidad para restablecer el nivel de rendimiento y de recuperación de datos afectados directamente en el caso de un fallo.

- Adherencia a normas. Capacidad del software relacionada con el grado de conformidad con estándares, convenciones o regulaciones existentes en leyes o prescripciones similares.

Característica de usabilidad: es definida como la capacidad de los productos de software para ser comprendidos, aprendidos, utilizados y agradables para un usuario en un dominio específico. Dado que la característica de usabilidad necesita de usuarios que interactúen con el producto, esta característica se corresponderá con uno o varios tipos de calidad asociados al conjunto que tiene la interpretación de la audiencia, es decir, a las interpretaciones que pueden realizar los usuarios de los modelos conceptuales (Marín 2007).

Se descompone en las siguientes sub-características:

- Fácil comprensión. La capacidad del software que permite al usuario si el producto es aceptable y cómo puede ser usado para tareas particulares y determinadas condiciones de uso.
- Fácil aprendizaje. Capacidad del producto software que permite al usuario aprender la aplicación software.
- Operatividad. Capacidad del producto software que permite al usuario controlar y usar la aplicación software.
- Software atractivo. Capacidad del producto software de ser atractivo al usuario.
- Adherencia a normas. Capacidad del software relacionada con el grado de conformidad con estándares, convenciones o regulaciones existentes en leyes o prescripciones similares.

Característica de eficiencia: Definida como la capacidad de entregar un rendimiento apropiado a los recursos utilizados en condiciones específicas, corresponde a un aspecto de la calidad semántica, ya que el rendimiento según la utilización de los recursos es un hecho del dominio de modelado de los productos de software (Marín 2007).

Se subdivide en las siguientes sub-características:

- Comportamiento frente al tiempo. Capacidad del producto software para proporcionar una respuesta y un tiempo de procesamiento apropiados al desarrollar sus funciones bajo condiciones establecidas.

- Uso de recursos. Capacidad del producto software para utilizar un apropiado número de recursos y tiempo de ejecución cuando el software desarrolla sus funciones bajo condiciones establecidas.
- Adherencia a normas. Capacidad del software relacionada con el grado de conformidad con estándares, convenciones o regulaciones existentes en leyes o prescripciones similares.

Característica de mantenibilidad: Consiste en poder realizar modificaciones a los productos de software. Estas modificaciones son necesarias para que los productos reflejen de mejor manera un dominio específico. Esta característica se puede entender como una propiedad del dominio de los modelos conceptuales, por lo que los modelos que cumplan con esta característica estarán contribuyendo a la calidad semántica (Marín 2007).

Se descompone en las siguientes sub-características:

- Facilidad de análisis. Capacidad del producto software para diagnosticar deficiencias o causas de fallos en el software.
- Capacidad para cambios. Capacidad del producto software que permite la ejecución de una modificación específica en ella misma.
- Estabilidad. Capacidad del producto de software para evitar defectos no esperados debidos a modificaciones en el mismo.
- Facilidades para pruebas. Capacidad del producto software que permite al software que ha sido modificado ser evaluado.
- Adherencia a normas. Capacidad del software relacionada con el grado de conformidad con estándares, convenciones o regulaciones existentes en leyes o prescripciones similares.

Característica de portabilidad: Consiste en la capacidad de los productos de software de ser transferidos de un entorno a otro, ya sea organizacional de software o de hardware. Esta característica consiste en la persistencia de los modelos de los productos de software, ya que es necesaria para poder realizar las transferencias a otros entornos (Marín 2007).

Esta característica del software puede ser desglosada en varias sub-características:

- Adaptabilidad. Capacidad del producto software para ser adaptado a diferentes entornos especificados sin aplicar acciones alejadas de aquellas que el propio software proporcione.
- Facilidad de instalación. Capacidad del producto software para ser instalado en un entorno específico.
- Coexistencia. Capacidad del producto software de coexistir con otros programas independientes en un entorno común y compartiendo recursos también comunes.
- Facilidad de reemplazo. Capacidad del producto software de ser utilizado en lugar de otro producto software específico para el mismo propósito que éste y en un entorno similar.
- Adherencia a normas. Capacidad del software relacionada con el grado de conformidad con estándares, convenciones.

De esta manera, se puede determinar el grado de calidad los productos según la evaluación de estas características y sub-características, en la cual una sub-característica puede ser medida con métricas internas o externas.

Una vez explicado la parte de calidad interna y externa del modelo de calidad, se adentrará en la calidad de uso. La calidad de uso es definida como "la capacidad del software que posibilita la obtención de objetivos específicos con efectividad, productividad, satisfacción y seguridad". (Véase Figura 7).

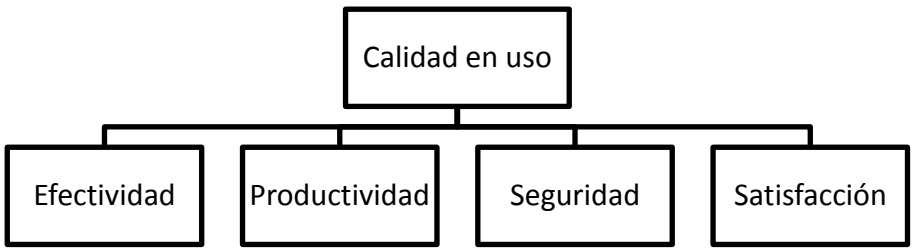

Figura 7: Estructura de la calidad en uso.

A continuación se describen las características (no contiene sub-características) de la calidad de uso:

- Efectividad: Es la capacidad de un producto de software, en la cual demuestra que se logra las metas específicas con exactitud y completamente en un contexto específico de uso.
- Productividad: Es la capacidad de un producto de software, para permitir a los usuarios emplear recursos apropiados con relación a la eficiencia alcanzada en un contexto específico de uso.
- Seguridad: Es la capacidad del producto de software para lograr niveles aceptables de riesgo hacia el personal, negocio, software, medio ambiente, en un contexto específico de uso.
- Satisfacción: Es la capacidad de un producto de software para satisfacer al usuario en un contexto específico de uso.

En la segunda, tercera y cuarta parte de la ISO 9126 se describen métricas y atributos para evaluar la calidad del software. Estas métricas y atributos están enfocadas a medir artefactos obtenidos en fases tardías del ciclo de desarrollo de software, complicando la detección y Correctud de los problemas de las etapas iniciales, que se propagan a las etapas posteriores (Marín 2007).

II.5.3 Marco de trabajo del modelo de calidad

El estándar ISO/IEC 9126 (International Organization for Standardization 2001) define un marco de conceptual de calidad que considera los siguientes factores: Calidad del Proceso, Calidad del Producto Software (Calidad Interna y Calidad Externa) y Calidad en Uso.

En la Figura 8, describe que la calidad del proceso contribuye a mejorar la calidad del producto y a su vez la calidad del producto contribuye a mejorar la calidad en uso (International Organization for Standardization 2003).

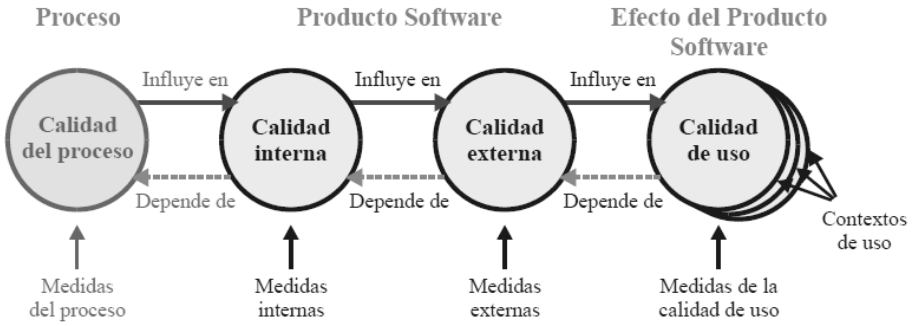

Figura 8: Marco de trabajo del modelo de calidad.

La evaluación del producto de software tiene como propósito satisfacer las necesidades de calidad, por lo que forma parte del ciclo de vida del desarrollo.

La calidad del producto del software puede ser evaluada por medio de la medición de la calidad interna o por medio de la calidad externa del software, o por la medición de la calidad en uso del software (International Organization for Standardization 2003).

Por lo tanto, evaluar y mejorar un proceso es significado de mejorar la calidad del producto, y la evaluación y la mejora de la calidad del producto significa la mejora de la calidad en uso. Así mismo, la evaluación de la calidad en uso puede retroalimentar la mejora del producto y la evaluación del producto puede retroalimentar al proceso (International Organization for Standardization 2003).

II.5.4 Modelo de ciclo de vida de la calidad del producto software

El estándar ISO/IEC 25000:2005 (International Organization for Standardization 2005) establece el modelo de calidad en el ciclo de vida del software en donde define tres principales faces durante el ciclo de vida del producto de software: producto en desarrollo, producto en operación y producto en uso.

La fase de producto en desarrollo está relacionada con la calidad interna de software. La fase del producto en operación está relacionada con la calidad externa, y la fase del producto en uso está relacionada con la calidad de software en uso. Ver figura 9.

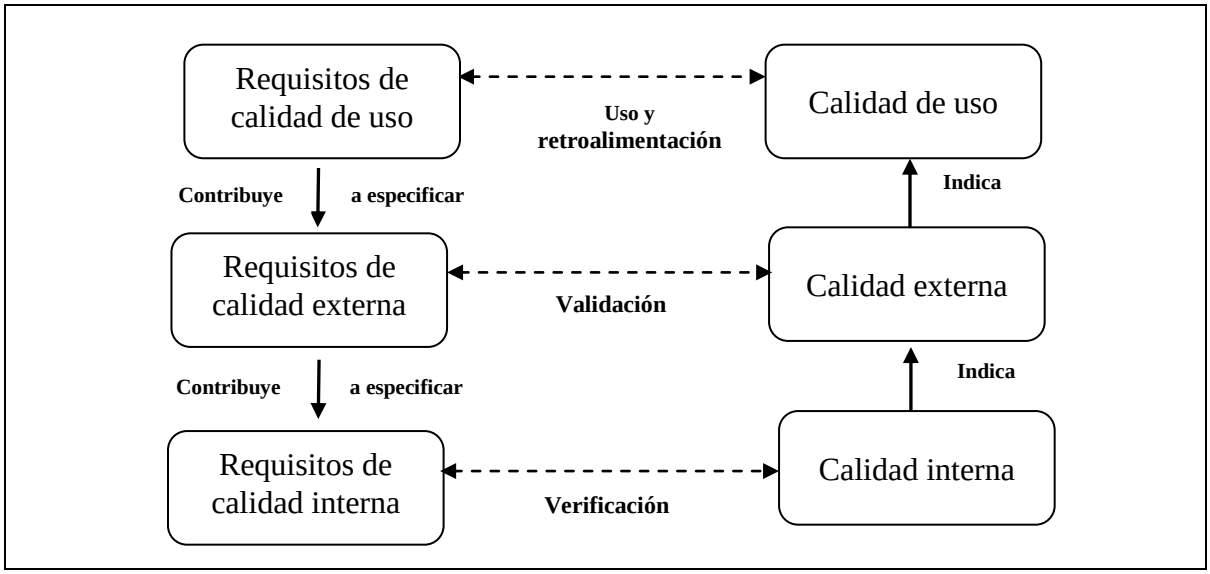

Figura 9: Modelo del ciclo de vida de la calidad del producto de software.

El modelo del ciclo de vida de la calidad del producto de software indica que la implementación de la calidad del software requiere un proceso similar al proceso de desarrollo de software por cada tipo de calidad: requerimientos, implementación y validación de los resultados.

Los requisitos de calidad de uso especifican el requerido nivel de calidad desde el punto de vista del usuario final, de esta manera, estos requisitos de calidad en uso son utilizados como objetivo de validación del producto de software por usuario.

Los requisitos de calidad externa de software especifican el nivel requerido de calidad desde el punto de vista externa. Estos incluyen requisitos derivados de los requisitos de calidad del usuario, incluyendo requerimientos de calidad en uso. Los requerimientos de calidad externa del software son utilizados como técnica principal para la verificación y validación del producto de software.

Los requisitos de calidad interna del software especifica el nivel requerido de calidad desde el punto de vista interno del producto. Estos incluyen requisitos derivados de las exigencias de la calidad externa del software. Los requisitos de calidad internas del software son utilizados para especificar las propiedades de productos de software

intermedios, como son los productos de software no ejecutables, así como documentación y manuales. También pueden ser utilizados para la definición de estrategias de desarrollo y criterios de evaluación y verificación durante el desarrollo.

Las necesidades de calidad interna deberían ser especificadas cuantitativamente en términos de métricas internas (International Organization for Standardization 2005).

II.6 Métricas de calidad

Es indiscutible que la medición es crucial para el progreso de todas las ciencias. El progreso científico se alcanza a través de observaciones y generalizaciones basadas en datos y mediciones, la derivación de estos en teorías y la confirmación o refutación de las teorías mediante hipótesis basadas en datos empíricos (Instituto Nacional de Tecnologías de la Comunicación 2008).

Las métricas son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo software y los proyectos de mantenimiento (Briand 1996).

En general, la medición persigue tres objetivos fundamentales: Entender qué ocurre durante el desarrollo y el mantenimiento, controlar qué es lo que ocurre en los proyectos y mejorar los procesos y los productos (Fenton 1997).

Una métrica del software se clasifica en tres categorías: Métricas del producto, métricas de proceso y métricas del proyecto.

- Las métricas de producto describen las características del producto como tamaño, complejidad, características de diseño, rendimiento y nivel de calidad.
- Las métricas del proceso se pueden utilizar para mejorar el desarrollo del software y el mantenimiento. Ejemplos de este tipo son: efectividad de la eliminación de defectos durante el desarrollo, el tiempo de respuesta del proceso de correctud, etc.

- Las métricas del proyecto describen las características del proyecto y su ejecución. Por ejemplo, puede ser el número de desarrolladores de software, el costo, calendario y productividad.
- Las métricas de calidad del software son un subconjunto de las métricas del software que se enfoca en los aspectos de calidad del producto, proceso y proyecto, sin embargo, las métricas de calidad de software están más asociadas a las métricas de proceso y producto que a las métricas de proyecto.

Las métricas de calidad del software son un subconjunto de las métricas del software que se enfoca en los aspectos de calidad del producto, proceso y proyecto, sin embargo, las métricas de calidad de software están más asociadas a las métricas de proceso y producto que a las métricas de proyecto (Instituto Nacional de Tecnologías de la Comunicación 2008).

Las métricas de calidad del producto se pueden dividir en métricas de calidad de producto final y métricas de calidad de producto en desarrollo.

- Las métricas de calidad del proceso miden el ciclo de vida del software, por lo que se pueden aplicar a medir el nivel de calidad del proceso de desarrollo o de mantenimiento.

Como se había mencionado en el tema de modelos de calidad del producto (International Organization for Standardization 2005), un atributo es una propiedad mesurable, física o abstracta, a la que se le puede asignar una métrica. Un atributo está relacionado con uno o más conceptos medibles.

Entendiendo por métrica como un atributo que evalúa un componente y produce un dato simple, un símbolo o un número. (International Organization for Standardization 2005). Una métrica está definida para uno o más atributos (Dr. Mario Piattini Velthuis 2008-2009).

La Ontología de Medición del Software identifica todos los conceptos involucrados con la medición del software, también proporciona definiciones exactas para todos los términos y clarificar las relaciones entre ellos (Mateus Ferreira 2006).

Para facilitar la comprensión de la ontología de la medición del software se prefirió por dividirla en cuatro sub-ontologías (Mateus Ferreira 2006):

Caracterización y objetivos de la medición software

Esta sub-ontología contiene los conceptos necesarios para establecer el ámbito y los objetivos del proceso de medición software. El principal objetivo del proceso de medición software es satisfacer ciertas necesidades de información identificando las entidades (que pertenecen a categoría de entidad) y los atributos de estas entidades (que son el enfoque del proceso de medición). Atributos y necesidades de información están relacionados por medio de conceptos medibles (que pertenecen a modelo de calidad).

En la Figura 10 se muestra la representación gráfica de los conceptos y relaciones de esta sub-ontología, utilizando el diagrama de clases UML:

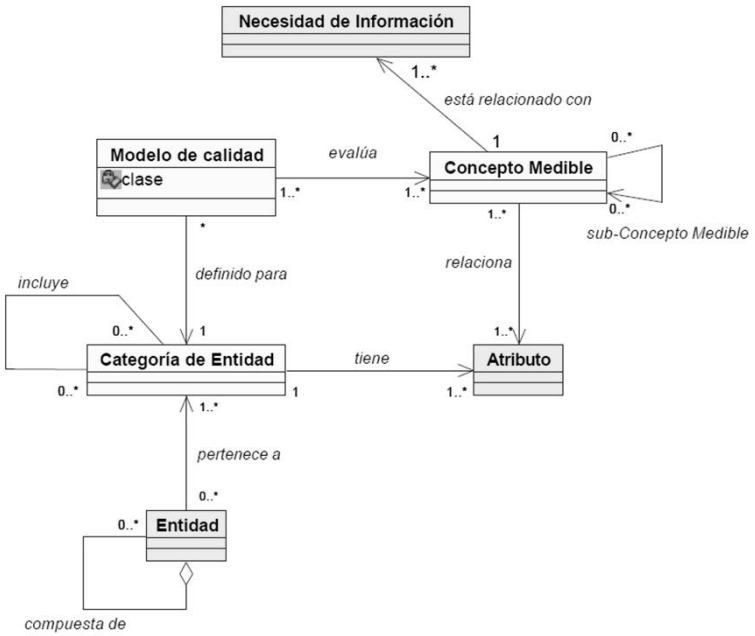

Figura 10: Sub-ontología caracterización y objetivos de la medición software.

Los conceptos centrales de esta sub-ontología son Necesidad de Información, Atributo y Entidad (Mateus Ferreira 2006).

Necesidad de información: Información necesaria para gestionar un proyecto (sus objetivos, hitos, riesgos y problemas).

Concepto medible: Relación abstracta entre atributos y necesidades de información.

Entidad: Un objeto que va a ser caracterizado mediante una medición de sus atributos.

Categoría de entidad: Una colección de entidades caracterizadas por satisfacer un cierto predicado común.

Atributo: Una propiedad mensurable, física o abstracta, que comparten todas las entidades de una categoría de entidad.

Modelo de calidad: Un conjunto de conceptos medibles y relaciones entre ellos que proporciona la base para especificar requisitos de calidad y evaluar la calidad de las entidades de una determinada categoría de entidad.

Medidas Software

Esta sub-ontología trata de establecer y aclarar los elementos clave en la definición de una medida software. Una medida relaciona un método de medición definido y una escala de medición (que pertenece a tipo de escala). La mayoría de las medidas pueden ser expresadas en una unidad de medición, y pueden ser definidas para más de un atributo (Medidas nominales son ejemplos de medidas que no pueden ser expresadas en unidades de medición) (Mateus Ferreira 2006).

Las medidas pueden ser de 3 tipos: Medidas base, medidas derivadas e indicadores.

La Figura 11 presenta el diagrama UML de los conceptos de esta sub-ontología e sus relaciones, donde el concepto central es Medida:

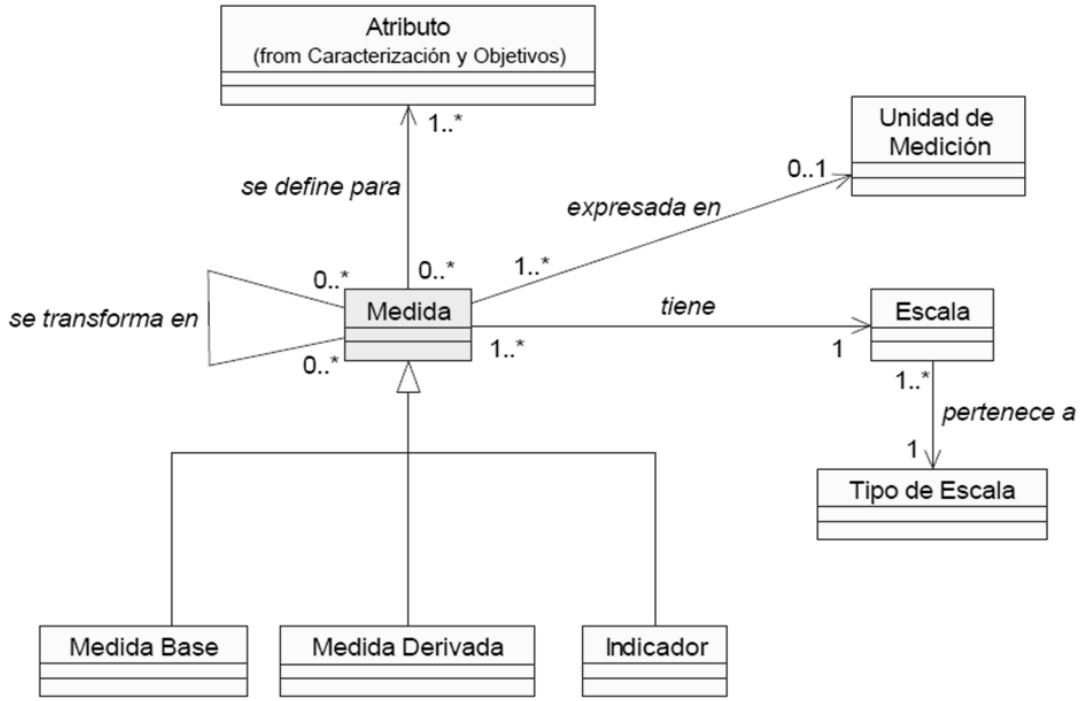

Figura 11: Sub-ontología de las medidas software.

Medida: La forma de medir (método de medición, función de cálculo o modelo de análisis) y la escala de medición.

Escala: Un conjunto de valores con propiedades definidas.

Tipo de escala: Indica la naturaleza de la relación entre los valores de la escala. Los tipos de escala son: nominal, ordinal, intervalo o razón/ratio. Independientemente de la escala de medición utilizada, cuando se obtienen datos, es necesario analizarlos para extraer información significativa. Existen varias medidas y estadísticas para consolidar los datos y poder realizar comparaciones, por ejemplo: Ratio, proporción, porcentaje, índice, etc. (Instituto Nacional de Tecnologías de la Comunicación 2008).

A continuación se definen algunas escalas de medición (Instituto Nacional de Tecnologías de la Comunicación 2008).

- Nominal: Consiste en clasificar elementos en categorías en función de un determinado atributo. Es el nivel más bajo de medición.

- Ordinal: Es un nivel más alto de medición que el nominal ya que, no solo se agrupan los elementos en categorías sino que se ordenan. Sin embargo, esta escala no ofrece información acerca de la magnitud de la diferencia entre elementos.
- Intervalo: Indica la diferencia exacta entre los puntos de medición. Esta escala requiere una unidad de medición bien definida que se puede tomar como estándar.
- Razón: Este es un tipo de escala de intervalos en el que se puede colocar una unidad cero de forma no arbitraria. Esta escala se puede aplicar a todas las operaciones matemáticas.
- Ratio: Es el resultado de dividir una cantidad por otra. El numerador y el denominador son dos categorías excluyentes.
- Proporción: La proporción se diferencia del ratio en que el numerador es parte del denominador, no son categorías excluyentes. Además, se pueden comparar múltiples categorías, en lugar de solo dos como en el ratio.
- Porcentaje: Una proporción se convierte en porcentaje cuando se expresa en términos de unidades por cien. Este tipo de escalas, representan un dato relativo, es importante dar suficiente información del contexto, como el número total de casos, para que la información se pueda interpretar correctamente.

Unidad de medición: Una cantidad particular, definida y adoptada por convención, con la que se puede comparar otras cantidades de la misma clase para expresar sus magnitudes respecto a esa cantidad particular.

Medida Base: Una medida de un atributo que no depende de ninguna otra medida, y cuya forma de medir es un método de medición.

Medida Derivada: Una medida que es derivada de otra medida base o derivada, utilizando una función de cálculo como forma de medir.

Indicador: Una medida que es derivada de otras medidas utilizando un modelo de análisis como forma de medir.

Formas de Medir

Esta sub-ontología introduce el concepto de forma de medir para generalizar los diferentes "métodos" usados por los tres tipos de medidas para obtener sus respectivos resultados de medición. Una medida base aplica un método de medición. Una medida derivada usa una función de cálculo (que se apoya en otra medida base y/o derivada). Por otro lado, un indicador utiliza un modelo de análisis (basado en un criterio de decisión) para obtener un resultado de medición que satisfaga a una necesidad de información (Mateus Ferreira 2006).

En la Figura 12 se representan los conceptos de esta sub-ontología y sus relaciones:

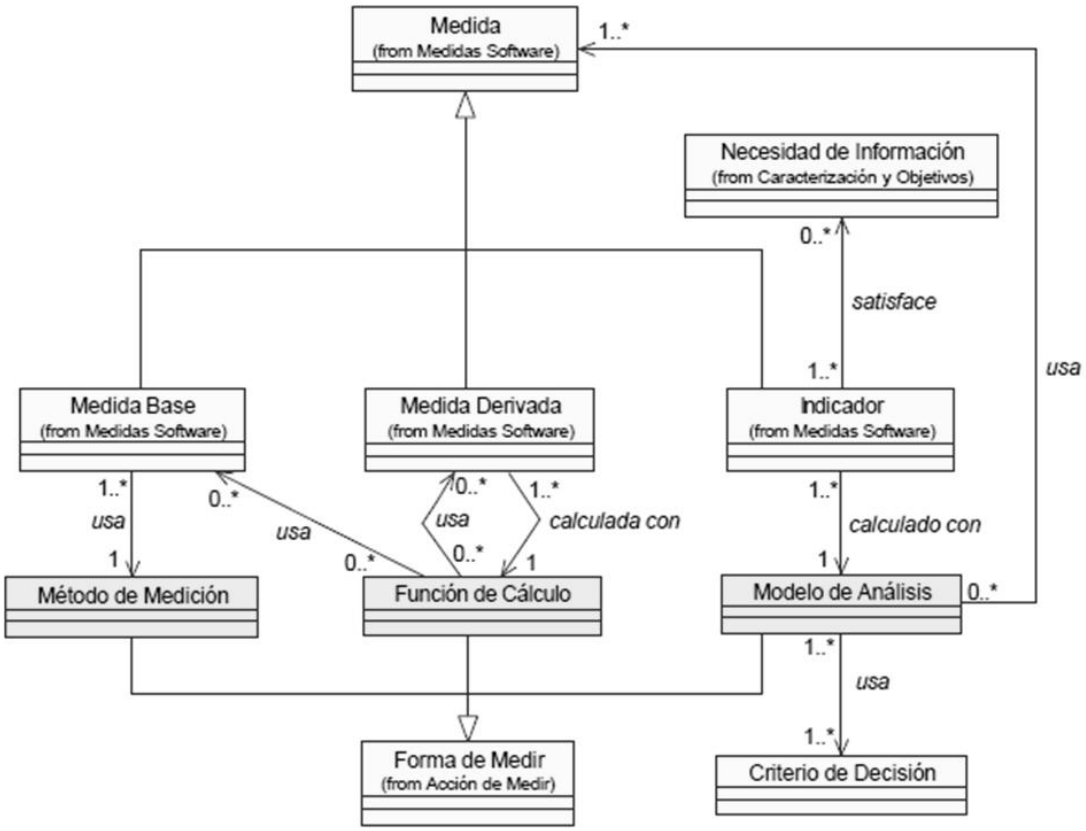

Figura 12: Sub-ontología de formas de medir.

Método de medición: La forma de medir una medida base. Secuencia lógica de operaciones, descritas de forma genérica, usadas para realizar mediciones de un atributo respecto de una escala específica.

Función de cálculo: La forma de medir una medida derivada. Algoritmo o cálculo realizado para combinar dos o más medidas base y/o derivadas.

Modelo de análisis La forma de medir un indicador. Algoritmo o cálculo realizado para combinar una o más medidas (base, derivadas o indicadores) con criterios de decisión asociados.

Criterio de decisión: Valores umbral, objetivos o patrones, usados para determinar la necesidad de una acción o investigación posterior o para describir el nivel de confianza de un resultado dado.

Medición

Esta sub-ontología establece la terminología relacionada con el acto de medir el software. Una medición (que es una acción) es un conjunto de operaciones cuyo objetivo es determinar el valor de un resultado de medición para un dado atributo de una entidad, utilizando una forma de medir. Los resultados de la medición se obtienen a través de la acción de llevar a cabo una medición (Mateus Ferreira 2006).

En la Figura 13 se representan los conceptos de esta sub-ontología y sus relaciones, donde el concepto central es la Medición.

Medición: Conjunto de operaciones que permite obtener el valor del resultado de la medición para un atributo de una entidad, usando una forma de medir.

Forma de medir: Secuencia de operaciones cuyo objeto es determinar el valor del resultado de la medición. Una forma de medir puede ser un método de medición, función de cálculo o modelo de análisis.

Resultado de la medición: Categoría o número asignado a un atributo de una entidad como resultado de una medición.

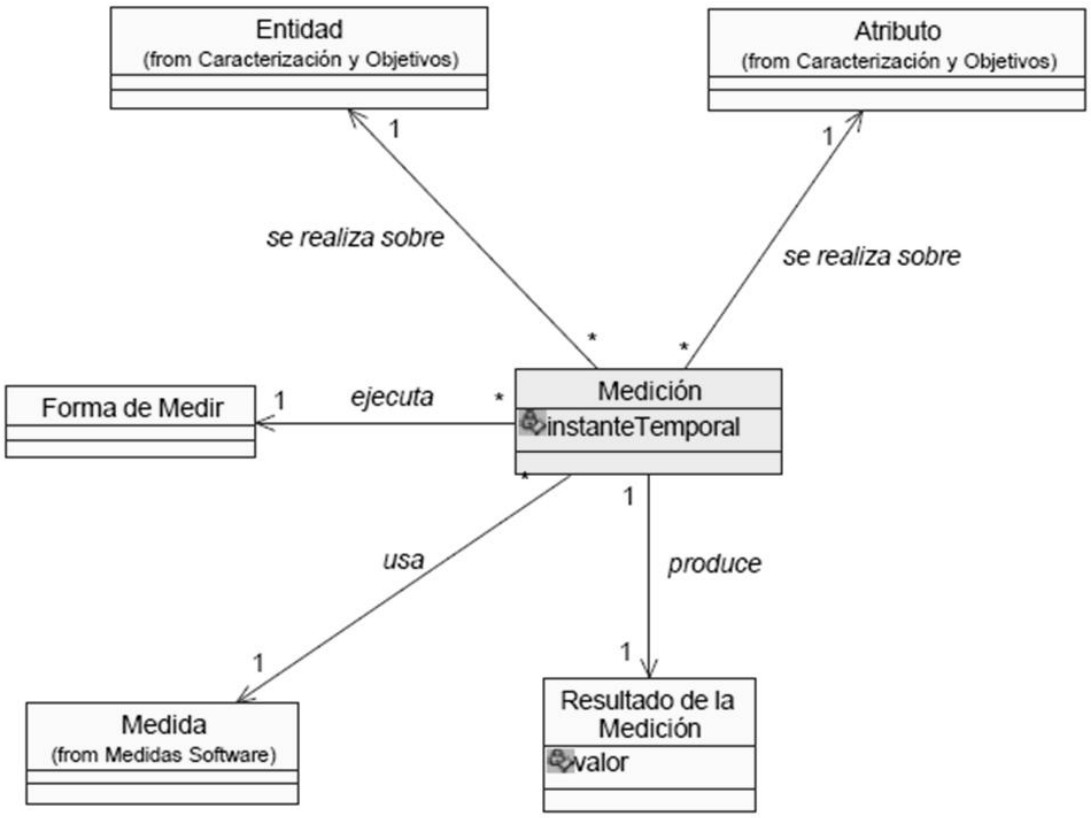

Figura 13: Sub-ontología de medición.

A través de esta ontología se establecen y clarifican los conceptos y relaciones relevantes relacionadas con la medición del software, a partir de los cuales se puede definir la información necesaria que hay que seleccionar y establecer para llevar a cabo un proceso de medición adecuado y efectivo. Además, se proporciona una forma de comunicación en una organización en el contexto de sus procesos de medición, basada en el uso de una terminología común, lo que facilita no sólo el entendimiento entre miembros del equipo responsables de realizar dichas actividades de medición, sino también la posibilidad de registrar los resultados de dicho proceso de una forma consistente e integrada (García 2004).

II.6.1 Métricas de Proceso de software

La medición del proceso implica las mediciones de las actividades relacionadas con el software siendo algunos de sus atributos típicos el esfuerzo, el coste y los defectos encontrados (Dr. Mario Piattini Velthuis 2008-2009).

Las mediciones son necesarias para mostrar cuantitativamente el estado actual de los procesos y las prácticas contra unas mejores prácticas de ingeniería de software y para mostrar hasta qué punto los procesos de software son efectivos para alcanzar las necesidades y metas de negocio de una organización (Instituto Nacional de Tecnologías de la Comunicación 2008).

Las mejores prácticas proporcionan un modelo general de la industria de software pero no reflejan las características específicas de una organización. Cada organización tiene sus propias necesidades y metas de negocio específicas que influenciarán la selección de las mejores prácticas que utilizarán para la mejora (Instituto Nacional de Tecnologías de la Comunicación 2008).

En la práctica, las necesidades y metas de negocio de una organización pueden imponer metas del proceso y prioridades diferentes, y pueden conducir a la identificación de metas de mejora que no se pueden expresar en términos de un nivel de capacidad para el proceso. Por ello, surge la necesidad de contar con un tipo distinto de medida, la medida de la efectividad de un proceso, orientado a las metas del proceso específicas de la organización (Instituto Nacional de Tecnologías de la Comunicación 2008).

Las medidas de efectividad indican hasta qué punto el proceso cumple las metas que se derivan directamente del análisis de las circunstancias específicas, necesidades y metas de negocio de la organización. Las medidas de la efectividad respaldan la elección de los procesos que se van a mejorar y el seguimiento de los resultados de la mejora.

Las medidas de efectividad se definen normalmente en base a las características de la salida del proceso. Por ejemplo, la efectividad del proceso de planificación se puede asociar a su eficiencia, precisión, fiabilidad o capacidad de ser repetido o una combinación de los anteriores (Instituto Nacional de Tecnologías de la Comunicación 2008).

Como las organizaciones tienen circunstancias y necesidades únicas, la elección de una medida de efectividad diferirá entre las distintas organizaciones. Sin embargo, su valor siempre debe ser medido y comparado con un valor objetivo o un rango como forma de medir la mejora (Instituto Nacional de Tecnologías de la Comunicación 2008).

En la Tabla 4 se exponen algunos ejemplos de métricas que pueden utilizarse para medir algunos procesos del ciclo de vida del desarrollo del software (Instituto Nacional de Tecnologías de la Comunicación 2008).

Tabla 4: Métricas de procesos de software.

Procesos	Métricas asociadas
Planificación y seguimiento y control de proyecto.	Desviación de esfuerzo de cada tarea completada.
	Desviación en calendario de cada tarea completada.
	% de tareas/hitos completados a tiempo.
Proceso de revisión de código.	Densidad de defectos de las revisiones.
	Eficiencia de las revisiones.
	Índice de revisiones.
Proceso de pruebas unitarias.	Densidad de defectos de las pruebas.
	Eficiencia de las pruebas.
Codificación y pruebas de sistema.	Tiempo en Correctud de defectos durante revisiones de código.
	Tiempo en Correctud de defectos durante pruebas de sistema.
Ingeniería.	% coste de baja calidad.
Proceso de mantenimiento.	Esfuerzo de resolución por parche/versión liberado.
	Tiempo de resolución por parche/versión liberado.
Proceso de gestión de incidencias.	Esfuerzo de resolución de incidencias por incidencia.
	Edad de las incidencias cerradas.
	Edad de las incidencias abiertas.
Proceso de gestión de la disponibilidad.	Nº caídas del sistema.
	Porcentaje de comprobaciones de mantenimiento preventivo realizadas frente a las planificadas.

II.6.2 Métricas internas, externas y de uso del producto de software

El estándar ISO/IEC 9126 contiene dos partes del modelo de calidad: Una parte del modelo de calidad es aplicable para modelar la calidad del producto de software interna y externa y la otra parte está dedicada a modelar la calidad en uso del producto de software.

- La calidad interna revisa los documentos de especificaciones, modelos o el análisis del código fuente.
- La calidad externa se refiere a las propiedades del software que interactúan con el ambiente en donde se desarrolla (Zeiss).

En base a esto, ISO/IEC 9126 define que las métricas internas son aplicadas a productos de software que no son ejecutables durante las fases del desarrollo, por ejemplo: Propósito del producto, definición de requerimientos, especificación de diseño o código fuente. Estas métricas proveen al usuario la habilidad de medir la calidad de productos intermedios y predecir la calidad del producto final. Por lo que permite al usuario identificar deficiencias en la calidad y tomar acciones correctivas lo más temprano posible en el ciclo de vida del desarrollo (International Organization for Standardization 2003).

Las Métricas externas son usadas para medir la calidad del producto de software en base al comportamiento del sistema. Estas métricas, se utilizan durante la fase de pruebas del ciclo de vida del proceso y durante las fases de operación (International Organization for Standardization 2003).

Por otro lado, las métricas de calidad en uso, miden cualquier necesidad de especificación del usuario que se encuentre en el producto, para obtener las metas específicas como efectividad, productividad, seguridad y satisfacción en un contexto específico de uso. Esto se puede obtener únicamente en un ambiente real del sistema.

Las especificaciones de los requerimientos calidad en uso se establecen por las métricas de calidad en uso, por métricas externas y en algunas veces por métricas internas. Estas especificaciones de requerimientos por métricas deberían ser usadas como criterios cuando el producto es evaluado (International Organization for Standardization 2001).

Así mismo ISO/IEC 9126-2 (International Organization for Standardization 2003) proporciona al usuario una guía de métricas para la evaluación de planificación, selección de métricas, diseño de métricas, aplicación de métricas e interpretación de medidas de datos.

Es recomendable que las métricas internas tengan fuerte relación con las métricas externas para que puedan ser usadas para predecir los valores de las métricas externas.
La interpretación de las medidas se puede realizar de tres formas:

- Medida directa. Una medida directa es una medida de un atributo que no depende de las medidas de otros atributos.
- Medida indirecta. Una medida indirecta es derivada de medidas de uno o más atributos.
- Indicadores. Son aquellas medidas que pueden ser estimadas o predichas desde otras medidas.

Las métricas tienen unas propiedades deseables que se detallan a continuación:

- Fiabilidad.
- Indicabilidad.
- Disponibilidad.
- Correctud.
- Imparcialidad.

Así mismo, cada métrica descrita en el estándar contiene la siguiente información (International Organization for Standardization 2003):

- Nombre.
- Propósito.
- Método de aplicación.
- Medida, fórmula y cómputo de datos.
- Interpretación del valor medido.
- Tipo de escala.
- Tipo de medida.
- Fuente de medida.
- Referencia a ISO/IEC 12207 SLCP.
- Audiencia.

II.7 Propuestas que establecen relación entre la calidad del proceso de software y la calidad del producto de software

De acuerdo con (Oliveira 2002) actualmente, la calidad del software puede dividirse en dos áreas. Por un lado se distinguen las técnicas y propuestas que pretenden asegurar la calidad de los productos software (propuestas orientadas al producto). Por otro lado, las propuestas que se centran en la definición, evaluación y mejora de los procesos de desarrollo software (propuestas orientadas al proceso), asumiendo que la calidad del software se ve directamente influenciada por la calidad de los procesos que lo sustentan (Monasor 2008).

A continuación se describen propuestas que establecen relación entre la calidad del producto y el proceso.

II.7.1 Calidad sistémica

Según (Callaos 1996), propone un enfoque del concepto de calidad del software, en el que se involucran tanto características orientadas a la calidad del proceso como a la calidad del producto. Se define como calidad sistémica en base a una matriz de calidad global que se basa en cuatro tipos de calidades, distinguiendo entre aspectos internos y contextuales de las perspectivas software y de procesos, considerando tanto los puntos de vista del cliente como del usuario.

Toda esta definición se justifica, porque según (Callaos 1996), el producto creado es diferente al sistema de actividades humanas, es decir, el proceso mediante el cual el sistema-producto es diseñado. Por lo cual permite balancear las diferentes perspectivas de la calidad del software (Proceso y Producto).

(Rojas 2007) y Pérez (María A. Pérez 1999), definen cada uno de los elementos de la matriz de Calidad Global:

- Aspectos Internos del Producto: Son determinados por actividades de diseño interno y programación, ya que un producto eficiente es conseguido cuando se aplican las prácticas correctas de diseño físico y programación.

- Aspectos Contextuales del Producto: Son determinados por las actividades de identificación de requerimientos, diseño de interfaces y diseño general de la red, debido a que la misma está relacionada con la adecuación y confort del usuario.
- Aspectos Internos del Proceso: Están asociados con las actividades de gerencia de proyectos, las cuales incluyen el cumplimiento de fechas de entrega, aumento de la productividad y ahorro de recursos.
- Aspectos Contextuales del Proceso: Se relacionan con las actividades generales de gerencia, tales como liderazgo, administración de cambio, relaciones humanas y grupales, ya que las mismas conducen a establecer buenas relaciones entre los integrantes del equipo responsable del desarrollo de Sistemas de software.

Es por eso, que con estos elementos definen la calidad global que establece el compromiso entre todas las partes, permitiendo englobar dos tendencias actuales de modelos de calidad, la del producto y la calidad del proceso, con un enfoque sistémico.

II.7.2 IEEE Std 1061

Así mismo, IEEE 1061 (International Organization for Standardization 1988) proporciona una metodología para establecer requisitos de calidad y para identificar, implementar, analizar y validar medidas de calidad tanto de productos software como del proceso de software. Esta metodología se aplica a todo el software en todas las fases de cualquier ciclo de vida del software, pero sin prescribir métricas específicas.

Este estándar cubre conceptos de los tres grupos (medidas, procesos y objetivos y-metas), pero no de una manera completa y exhaustiva – que no es la meta principal del estándar, de todas formas.

Se encontraron discrepancias entre este estándar y el del IEEE 610.12, que se producen y mantienen por la misma organización. Por ejemplo, la definición de “métrica” en IEEE 610.12 se diferencia de la definición dada en IEEE 1061 (Mateus Ferreira 2006).

II.7.3 Modelo Sistémico de Calidad (MOSCA) del Software

El MOdelo Sistémico de CAlidad de software (MOSCA) integra el modelo de calidad del producto (Maryoly Ortega 2001) y el modelo de calidad del proceso de (María A. Pérez 1999) para poder establecer un prototipo de Modelo Sistémico de Calidad para la medición de la calidad de los sistemas de software.

MOSCA se basa en esta perspectiva de calidad. Por tal motivo, el modelo de calidad del producto se software, creado por Ortega (Maryoly Ortega 2001), se basa en las 6 características de calidad del estándar internacional ISO/IEC 9126 (International Organization for Standardization 1987), en la cual plantea un conjunto de sub-características y métricas asociadas que miden la calidad de un producto de software con un enfoque sistémico.

A pesar de que con este modelo de calidad enfocado al producto puede medir de forma cuantitativa la calidad del producto a cierto grado, no establece claramente la relación con la medición de la calidad del proceso.

Por otro lado, se encuentra en modelo de calidad del proceso de software con un enfoque sistémico (María A. Pérez 1999). Este modelo está basado en la adaptación del modelo del proceso ISO/IEC 15504 (International Organization for Standardization 2003), en la cual define 5 niveles, donde cada nivel superior está conformado por elementos del nivel inferior.

Este modelo de calidad del proceso solo estudia la perspectiva de la medición de la calidad del proceso de software, dejando a un lado la calidad del producto.

En base a esto, el modelo de calidad Sistémica MOSCA propone integrar ambos modelos para garantizar la medición de la calidad sistémica global en las organizaciones, (ver figura 14) (Mendoza 2005).

MOSCA está constituida por cuatro niveles:

- Nivel 0: Dimensiones. Aspectos Internos y contextuales del proceso y del producto.
- Nivel 1: Categoría: se contempla once categorías: seis pertenecientes al producto y otras cinco al proceso de desarrollo.

- Nivel 2: Características. Cada categoría tiene asociado un conjunto de características, las cuales definen las aéreas claves a satisfacer la calidad tanto del producto y del proceso según corresponda
- Nivel 3: Métricas. Para cada característica se propone una serie de métricas utilizadas para medir la calidad sistémica. Dada la cantidad de métricas asociadas a cada una de las características que conforman MOSCA 679 en total.

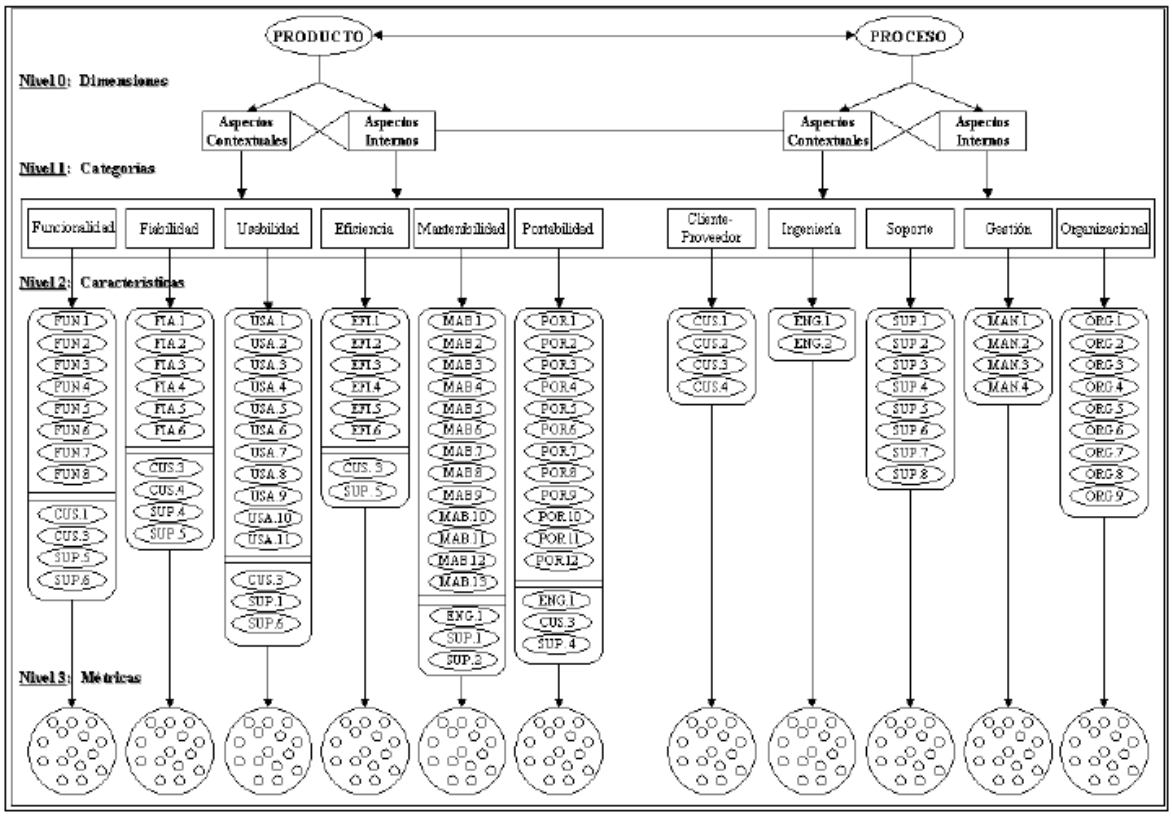

Figura 14: Modelo MOSCA.

Según (Humphrey 1989), establece que el producto final tiene una gran influencia por la calidad del proceso de desarrollo del mismo. Es por eso que MOSCA (Mendoza 2005) establece que cada categoría del producto tiene una dependencia específica de ciertas características del modelo de proceso, por lo que se determina que si las características del proceso se cumplen con calidad durante el proceso de desarrollo entonces la categoría del producto debería poseer los requerimientos mínimos de calidad.

Por ejemplo, la Tabla 5 describe las características del proceso que influyen en la Funcionalidad del producto de software.

En base a esta relación, MOSCA puede establecer un criterio para poder evaluar la calidad del producto de software.

Tabla 5: Relación entre el proceso de software y la característica de funcionalidad.

Característica del Proceso	Descripción de la relación
Proceso de adquisición de sistema o producto de software.	Este proceso tiene como objetivo obtener un producto que satisfaga las necesidades del cliente, garantizando la aceptación del mismo. Por lo tanto, siempre cuida la funcionalidad que el cliente está pidiendo.
Proceso de Determinación de Requerimientos.	Este proceso se encarga de verificar que las necesidades y los requerimientos del cliente sean satisfechos a través del proceso de desarrollo.
Proceso de Validación.	En este proceso confirma que los requerimientos para un uso específico del producto del sistema sean satisfechos, es decir, este proceso garantiza que la funcionalidad del producto de software se cumpla.
Proceso de Revisión Conjunta.	Este proceso se refiere al mantenimiento de un común entendimiento con el cliente sobre el progreso del proceso o del proyecto, en contraste con los objetivos del contrato. De esta manera, si se realiza correctamente, el producto estará desarrollado en base a los requerimientos establecidos por el cliente.

MOSCA establece un algoritmo con el cual puede medir la calidad sistémica dentro de una organización a través de la aplicación del modelo MOSCA. Este algoritmo inicia con la medición de la calidad del producto de software y después con la calidad del proceso de desarrollo. De esta manera, se obtiene una calificación que se pondera en: Calidad básica, calidad intermedia y calidad avanzada.

El modelo MOSCA especifica los procesos de una empresa que deben mejorar, así como las características que no son satisfechas por el producto de software desarrollado. Sin embargo, la relación que establece entre el producto y el proceso no es directa considerando el nivel jerárquico de los atributos del proceso como del producto, ni sigue un esquema preciso mapeando los atributos en base a ISO 9126 y SPEM 2, por lo que

ocasiona que pueda existir alguna perdida de información para poder ver el impacto directo que tiene un proceso de software en un producto final.

CAPÍTULO III

PROPUESTA DE METAMODELO PARA LA RELACIÓN PROCESO-PRODUCTO

III Propuesta de Metamodelo para la relación proceso-producto

En esta sección se presenta la propuesta de un Metamodelo que establece el mapeo entre los atributos del producto y los atributos del proceso.

Para desarrollar el Metamodelo se utilizaron los siguientes conceptos que sirvieron como base para definir cada uno de sus componentes (estos conceptos se describen a detalle en el capítulo II):

- Estructura del proceso de software: Metamodelo de Ingeniería de Procesos de Software versión 2 (SPEM 2, por sus siglas en inglés) (Group 2008).
- Modelo de Calidad: Estándar ISO/IEC 9126 (International Organization for Standardization 2001).
- Métricas de Software: Ontología de Medición del Software (Mateus Ferreira 2006).

El Metamodelo (véase Figura 15) describe la estructura jerárquica de las entidades de proceso, producto y el modelo de calidad, así como la relación entre ellos.

Estructura del Proceso

La estructura del proceso se define como un conjunto de Practicas Base, la cual se desglosa de los siguientes elementos (Véase en la figura 15):

- **Prácticas base:** Es un conjunto de Actividades (Ruiz 2008).
- **Actividad:** Es un conjunto de Tareas (Ruiz 2008).
- **Tarea:** Es la entidad más pequeña de trabajo de un proceso, tiene como propósito generar un artefacto, el cual forma parte de un producto de software (Ruiz 2008).
- **Relaciones:** Una tarea tiene relación con el Rol, debido a que el Rol realiza y aprueba la tarea.

Estructura del modelo de calidad

Siguiendo la definición de un modelo de calidad que establece SQueRe (International Organization for Standardization 2005), como un conjunto de características de las cuales se subdividen en sub-características y estas en atributos específicos de calidad.

Relación entre los elementos del proceso con el modelo de calidad

El artefacto (producto de software) como la tarea (proceso de software), tienen propiedades mediante las cuales se evalúan y describe su calidad. Estas propiedades (atributos de calidad) pueden ser cuantificables por medio de las métricas de calidad, según corresponda. De esta manera, se establece un mapeo entre el proceso y el producto de software a través de las métricas de calidad de software.

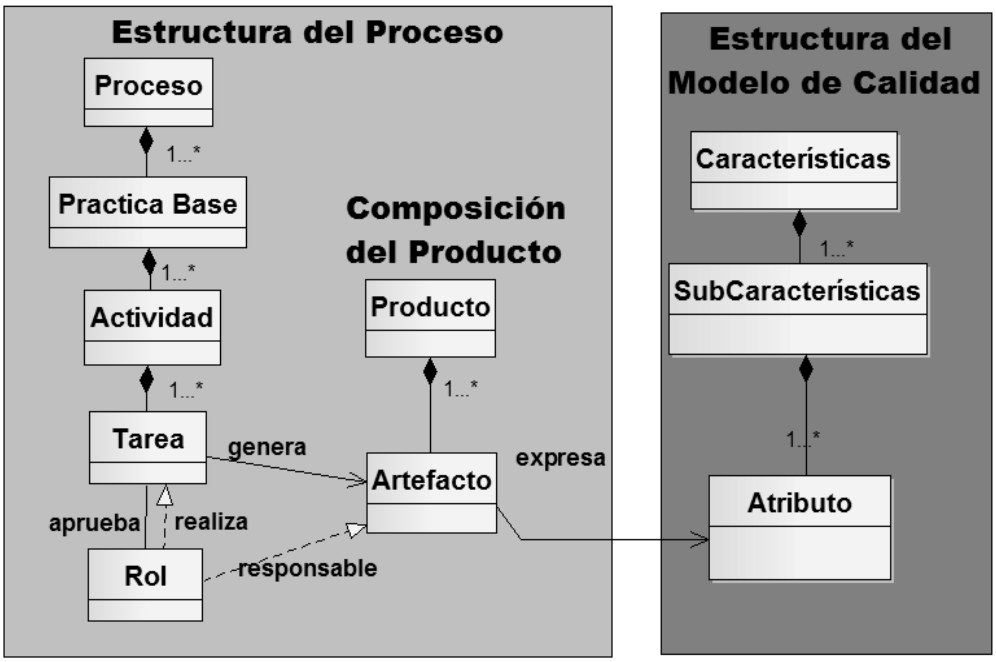

Figura 15: Relación entre proceso, producto y modelo de calidad.

Haciendo una descripción más detallada del diagrama anterior (Figura 15) y enfocando al nivel más fino jerárquicamente de las entidades del proceso, producto y modelo de calidad, se puede establecer un mapeo entre ellos a través de las métricas de

calidad del software. De esta forma, se propone una representación semi-formal de la relación que existe entre la calidad del proceso de software y la calidad del producto.

Al evaluar la calidad del producto o artefacto de trabajo, usando las métricas de calidad, se obtiene un valor cuantitativo, con el cual se puede saber el grado de defectos que tiene dicho producto. De esta manera, sabiendo el grado de calidad que tiene el producto se puede realizar una correspondencia con las tareas del proceso que generaron dicho producto, utilizando el método FMEA. Al tener identificadas las actividades del proceso responsables de dicho defecto, se puede aplicar las métricas de calidad a cada tarea y de esta manera se puede obtener un valor cuantitativo de la calidad del proceso. En el Metamodelo (Véase Figura 16) se puede ver que están relacionados los tres conceptos: proceso, producto y modelo de calidad a través de la aplicación de las métricas.

Proceso de software: Se establece que cada actividad del proceso tiene asignado un propósito el cual describe el objetivo principal de la actividad.

Propósito: En base al Modelo de Procesos para la Industria de Software, MoProSoft (Oktaba Hanna 2005) se tomó como base los propósitos descritos de las actividades del proceso Desarrollo y Mantenimiento de Software que pertenece a la categoría de Operación, de esta manera, se puede identificar los propósitos principales de cada una de las actividades del proceso de software.

Centrándose en el nivel más fino del proceso, es decir, la **tarea**, permite ver los siguientes elementos esenciales:

Objetivo: Su principal funcionalidad es asegurar el cumplimiento del propósito del proceso, el cual se clasifica en objetivo de calidad y en objetivo de producción.

Objetivo de calidad: Define el atributo de calidad que tiene la tarea, en el momento de que entra en ejecución y el cual es medible.

Objetivo de producción: Establece el cumplimiento de la realización o ejecución del proceso.

Por otro lado, la tarea genera un artefacto, el cual, el conjunto de artefactos generados constituyen al producto de software. Este elemento tiene atributos de calidad que pueden ser evaluados a través de las métricas de calidad.

Métricas de Software

En base a la ontología de medición de software (Mateus Ferreira 2006) y al estándar de ISO 9126 (International Organization for Standardization 2003) se identifican los siguientes elementos que tiene una métrica:

Propósito: Este elemento expresa el motivo por el cual se aplica la métrica, por lo que se mapea con el atributo de calidad, debido a que es el motivo que se quiere evaluar y obtener un valor cuantitativo (International Organization for Standardization 2003).

Audiencia: Este elemento identifica a interesados (roles) quienes les importa saber el resultado de la métrica (International Organization for Standardization 2003).

Medio de Aplicación: Es el artefacto de software en donde se aplica la medición (International Organization for Standardization 2003).

Fuente de Información: Este elemento representa la información fuente que se utiliza para poder aplicar las métricas (International Organization for Standardization 2003).

Las siguientes definiciones: **Indicador**, **Métrica Base**, **Métrica Derivada, Método de Medición, Función de cálculo, Escala y Tipo de Escala**, se puede observar en la descripción de la Ontología de medición de software en el capítulo II (ver en sección Medidas Software, y Formas de Medir).

En base a los conceptos de los elementos descritos, se establece el mapeo que existe entre el proceso y el producto de software aplicando métricas, de tal manera que se obtenga información cuantitativa de la calidad de los dos elementos (proceso y producto). Como se observa en la Figura 16.

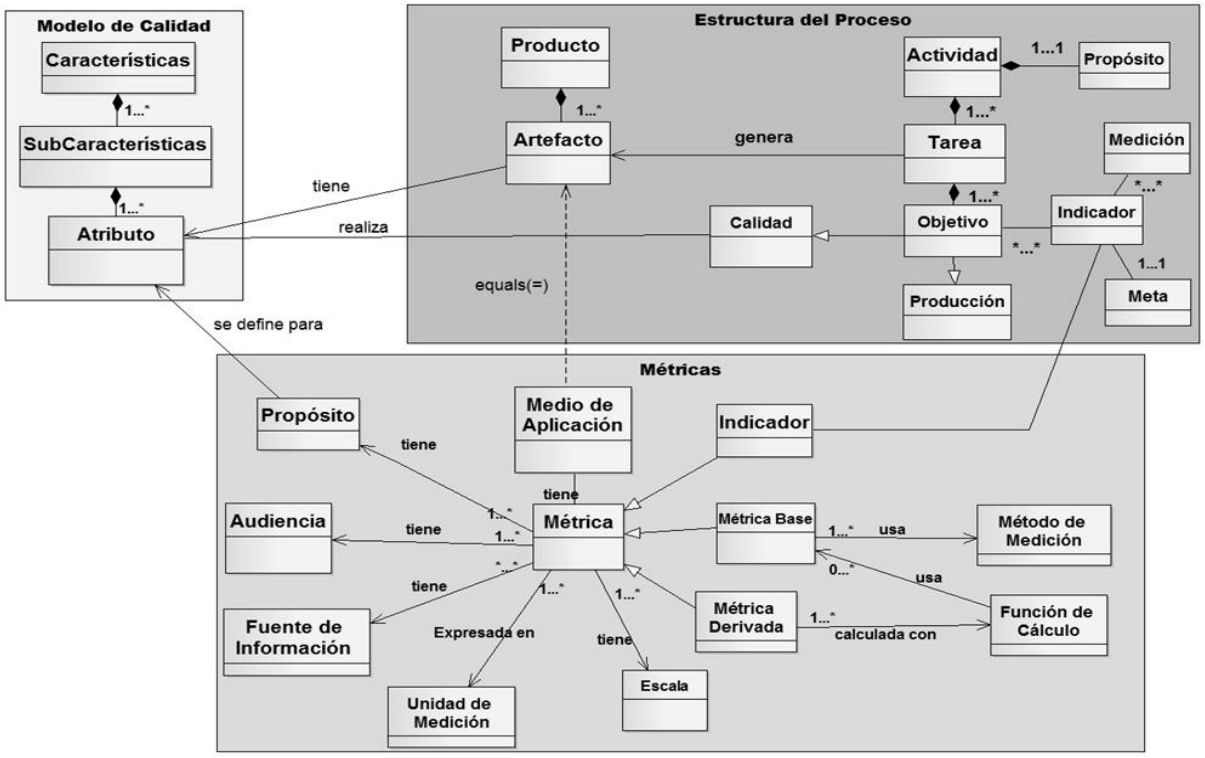

Figura 16: Mapeo entre el proceso-producto de software a través de las métricas de calidad.

Analizando la composición estructural tanto del proceso como del producto de software y haciendo uso de las definiciones de SPEM 2, se definen los siguientes elementos estructurales de estos elementos y su relación que tiene con la calidad del producto final de software (véase Figura 17).

Una tarea del proceso de software se clasifica en tarea base y en tarea de apoyo.

Tareas Base (TB): Corresponden a todas aquellas que tienen como objetivo generar, modificar, elaborar, realizar un producto o un artefacto de software.

Tareas de Apoyo (TA): Corresponden a todas aquellas que tienen como objetivo verificar, validar, revisar, aprobar, comprobar, identificar un producto o un artefacto de software.

Un artefacto de software se clasifica en artefactos tangibles y en artefactos no tangibles, según como establece SPEM 2 (Group 2008).

Artefactos Tangibles (AT): Según SPEM 2 (Group 2008), los artefactos tangibles están formados por otros artefactos más simples. En base a esto se clasifica este tipo de artefactos en dos tipos: contribuidor directo y contribuidor indirecto:

- **Artefactos Tangibles Contribuidor Directo (ATCD):** Se caracterizan por definir elementos principales (materia prima) para la generación de otro artefacto tangible. Es decir, los elementos que aporta este artefacto a otro, son elementos esenciales para la realización de tareas base que generan artefactos tangibles de contribuidor directo, por lo que contribuyen directamente al producto de software final.
- **Artefactos Tangibles Contribuidor Indirecto (ATCI):** Se caracterizan por definir elementos de soporte para la generación de otro artefacto tangible. Este tipo de artefacto se utiliza para monitorear, controlar o asegurar el desarrollo del producto de software final, es decir, contribuye al producto de software indirectamente.

Artefactos No Tangibles (ANT): son los que no están formalmente definidos y en base a quien lo produce o genera no existen físicamente.

Por lo tanto, las **TB** por su naturaleza producen **AT** (Ruiz 2008), de los cuales pueden ser **ATCD** o **ATCI**. Por consiguiente, las **TB** que generan **ATCD**, aporta elementos principales (materia prima) a otra actividad del proceso de software (por medio de un producto de entrada) el cual, se puede establecer que contribuye directamente en la calidad del producto de software final en comparación con las **TB** que generan **ATCI.**

Las **TA** por su naturaleza no generan **AT**, sino **ANT** (Ruiz 2008), los cuales pueden servir como apoyo para la elaboración de un **AT**, debido a que aportan menos elementos principales (materia prima) a otra actividad del proceso de software.

Por consiguiente, se dice que los **AT** contribuyen en la calidad del producto de software en un mayor peso que los **ANT**.

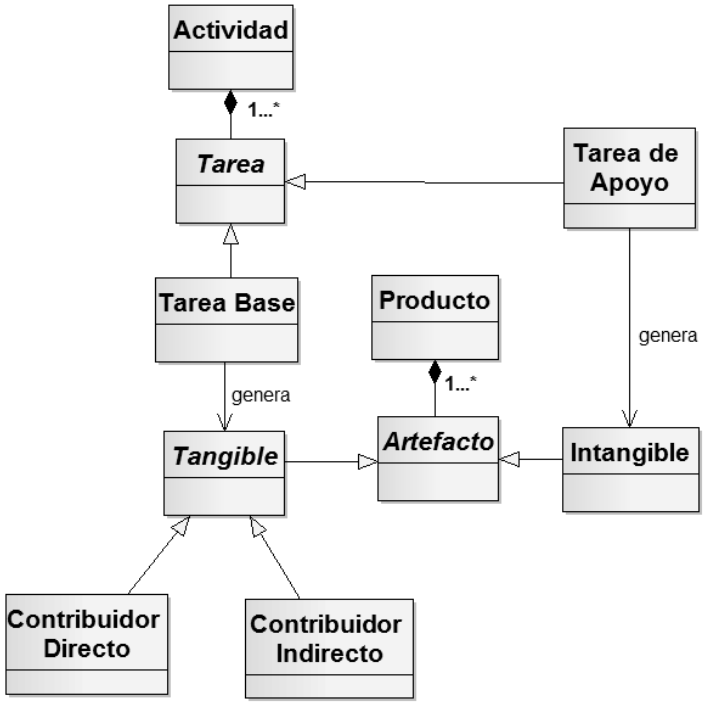

Figura 17: Composición estructural del proceso y del producto de software.

CAPÍTULO IV

EXPERIMENTOS REALIZADOS

IV Experimentos realizados

En este capítulo se exponen los experimentos realizados para probar lo planteado en las hipótesis de este trabajo de tesis. La mayoría de los experimentos fueron realizados en el ámbito académico, ya que aún es muy complicado que las empresas permitan observar sus procesos y analizar la información que generan, por lo que solo se pudo obtener información de dos empresas de la localidad. Sin embargo, cabe mencionar que los proyectos usados en la experimentación, en el ámbito académico, todos son proyectos reales.

IV.1 Aspectos generales para la experimentación

En esta sección se describen los elementos generales que soportan la experimentación realizada, describiendo los escenarios empleados y la recolección de datos.

Para la realización de los experimentos (ambiente académico e industria) se utilizó el método de Análisis de Modos y Efectos de Fallos (FMEA, por sus siglas en inglés), la cual ayudó a identificar las posibles causas de los defectos del producto de software. Este método se puede ver a detalle en el Apéndice C. También se utilizó el estándar ISO/IEC 9126 para aplicar métricas de calidad enfocadas al producto de software. Este estándar se puede ver a detalle en el Apéndice B. Para la identificación de los atributos estructurales de un proceso de software, se utilizó el Metamodelo de Ingeniería de Procesos de Software versión 2 (SPEM 2), el cual se puede ver a detalle en el Capítulo II.

Los escenarios empleados para la realización de los experimentos se describen a continuación:

En el ámbito Académico:

La materia de Temas Selectos de Ingeniería de Software que forman parte del plan de estudios de la Licenciatura de Ingeniería en Computación de la Facultad de Ingeniería en

Universidad Nacional Autónoma de México (UNAM), sirvió como caso de estudio, debido a que se realizó un proyecto de desarrollo de software para un cliente real.

Otro escenario fue en la materia de Temas Especiales de Ingeniería de Software de la Maestría en Ciencias e Ingeniería de la Computación de la UNAM, en donde se realizó un sistema de software para un cliente real.

En el ámbito de la Industria:

Se pudieron realizar 2 casos de estudio en el ámbito académico gracias a la vinculación que actualmente está teniendo la carrera de Ingeniería en Computación (IC) de la Facultad de Ciencias Químicas e Ingeniería (FCQI) con la industria.

Un caso de estudio, se tomo de un proyecto realizado por una empresa de la localidad dedicada al desarrollo de software a la medida. Con esta empresa no se realizó un convenio de vinculación con la UABC, pero permitió que se tomará un proyecto de desarrollo de software, el cual estaba en la fase de cierre.

Otro caso de estudio fue en un proyecto realizado por la empresa Argus Tecnologías S.A de C.V., esta empresa se dedica a implementar soluciones productivas de telemetría y geo-posicionamiento orientadas al mercado del transporte terrestre de personas o de carga, www.argustelematics.com. Esta empresa tiene un convenio de vinculación con la UABC en la FCQI. Por tal motivo se tomo como caso de estudio el proyecto que se realizó dentro del convenio, el cual era desarrollar un producto de software.

IV.2 Descripción de los casos de estudio

En esta sección se descubrirá las condiciones de experimentación de cada caso de estudio, debido a que cada uno fue un proyecto desarrollado en contextos diferentes.

IV.2.1 Caso de estudio 1

En este caso de estudio se desarrollo el proyecto llamado “Hemosist”, el cual es una aplicación Web que fue desarrollado por un grupo de 5 alumnos de la Maestría en Ciencias e Ingeniería de la Computación de la UNAM. Este proyecto se desarrollo dentro de la materia “Temas Especiales de Ingeniería de Software”. Además de formar parte de un proyecto de una materia, el interesado principal del proyecto fue un cliente real. Por lo que tenía una fecha de entrega definido y requerimientos específicos del producto de software.

Descripción del proyecto

El cliente requería de un sistema que le permitiera mantener datos históricos y obtener datos estadísticos de los pacientes del área de hemodinámica del Instituto Nacional de Cardiología. Las principales funcionalidades son: captura, modificación y consulta de los datos de los pacientes.

Los integrantes del equipo de trabajo tenían asignados roles de trabajo que establece el Modelo de Proceso de Software de Equipo (TSP, por sus siglas en inglés), debido a que este está dedicado a equipos.

Los roles asignados fueron los siguientes (en base a TSP): Líder del equipo (LD), Administrador de planeación (AP), Administrador de desarrollo (AD), Administrador de calidad y proceso (ACP), Administrador de apoyo (AA) e Ingeniero de software (IS).

Para poder asignar las roles se revisaron los perfiles de cada Rol y en base a este, el equipo realizaba propuestas a cerca de quién podría cumplir con dichas características.

La metodología de desarrollo que se utilizó fue el Proceso Unificado Ágil, por lo que la administración del proyecto se basó en los casos de uso identificados.

También se tomaron como base MoProSoft (véase a mas detalle en el Apéndice F) para el desarrollo de actividades y entregables del proceso de Administración de proyectos.

IV.2.2 Caso de estudio 2

Este caso de estudio se aplicó en el proyecto llamado “Seguimiento de Egresados de la Facultad de Ciencias”, el cual es una aplicación web que fue desarrollado por un grupo de 7 alumnos de la materia Temas Avanzados de Ingeniería de Software que forma parte del plan de estudios de la Licenciatura de Ingeniería en Computación (fecha de aprobación Agosto 2005) de la Facultad de Ingeniería en la UNAM. Además de formar parte de un proyecto de una materia, el interesado principal del proyecto fue un cliente real, que tenía definido fecha de entrega y requerimientos específicos para el producto de software.

Descripción del proyecto

El producto de software a desarrollar fue una aplicación Web, el cual el principal propósito es darle seguimiento a los egresados de la facultad de Ciencias de la UNAM. Las principales funcionalidades de esta aplicación fueron: Realizar encuestas de seguimiento de los egresados para poder llevar estadísticas, consultar eventos y propaganda de la facultad, así como dar a conocer los proyectos de vinculación y otras actividades.

El equipo de trabajo decidió utilizar una metodología de desarrollo Ágil, por lo que se seleccionaron SCRUM (ver apéndice D). En base a los principios de esta metodología, el equipo de trabajo se auto organizó para la definición de los roles, según las habilidades de cada integrante y el gusto por realizar las actividades del rol seleccionado. También se dirigieron en cada una de las fases y las mediciones del proyecto se realizaban a diario.

Los roles definidos fueron: Líder del equipo, programador y rol de Pruebas.

El nivel de conocimiento de los integrantes del equipo de desarrollo sobre los modelos agiles de desarrollo se puede considerar que es medio, debido a que anteriormente ya habían trabajando en proyectos de este tipo pero siempre con supervisión de un asesor. El proyecto se planeó en Seis Sprints (iteraciones). En cada Sprint se definieron los requerimientos que se iba a desarrollar, así como las actividades a realizar en cada iteración.

IV.2.3 Caso de estudio 3

Este caso de estudio que se aplicó en el proyecto llamado “Sistema de medición a distancia de parámetros tales como la temperatura, la presión y los niveles de volumen de un producto transportado dentro de un contenedor”, el cual fue desarrollado como proyecto de vinculación UABC y la empresa Argus Tecnologías S.A de C.V.. El equipo de trabajo estuvo integrado por un grupo de alumnos de las carreras de Ingeniero en Computación y de Electrónica (UABC), así como de los ingenieros de la empresa encargados de dirigir el proyecto.

El nivel de conocimiento sobre el desarrollo de software de los alumnos es de un nivel medio, debido a que ya habían participado en un proyecto de desarrollo de software, pero no con un cliente real ni con una fecha de entrega definida.

El proyecto fue planeado por el equipo de Ingenieros de la empresa en conjunto con líderes del equipo de UABC, en donde determinaron una fecha de entrega del proyecto y requisitos de calidad del producto de software.

Descripción del proyecto.

El proyecto se divide en 3 módulos, que son:

(1) Monitoreo de parámetros en el contenedor.

(2) Recepción y procesamiento de parámetros del contenedor.

(3) Aplicación de cliente final.

Para la evaluación de la calidad del producto de software se tomo la parte de la aplicación de cliente final, debido a que es donde se visualiza el funcionamiento completo del proyecto.

Para determinar las actividades del proceso de software se tomó como base las tareas establecidas en MoProSoft del proceso de Desarrollo y Mantenimiento de Software (DMS) (ver apéndice F).

IV.2.4 Caso de estudio 4

Este caso de estudio se aplicó en un proyecto en donde el producto de software fue una aplicación web. Este proyecto fue desarrollado por una empresa de la localidad dedicada al desarrollo de software a la medida y de servicios. El producto está enfocado en el mercado de la industria hotelera y de tiempos compartidos, por lo que los requerimientos de este producto fueron obtenidos en base a las necesidades de varios clientes y de las observaciones en campo del encargado del proyecto. Su principal propósito fue la administración de reservaciones y control administrativo y operativo de tiempos compartidos.

El proyecto fue elaborado por el equipo de desarrollo de la empresa en la cual desarrollaron los roles de: Líder de proyecto, Analista-Diseñador, Programador, Pruebas y Mantenimiento. El proyecto tenía un tiempo estimado para una entrega final del producto de software y empezar a comercializarlo.

En este caso de estudio se evaluó el producto final antes de iniciar la fase de comercializarlo.

Para determinar las actividades del proceso de software se tomó como base las tareas establecidas en MoProSoft de los procesos de Administración de Proyectos Específicos (APE) y de Desarrollo y Mantenimiento de Software (DMS) (ver apéndice F).

IV.3 Experimento 1: Contribución de las tareas del proceso de software en la calidad del producto de software.

La calidad del producto de software viene determinada por la calidad del proceso con el que se desarrolla. Teniendo un proceso definido y aplicando mejoras sobre este proceso, podrán incrementar la calidad de nuestros productos continuamente (Gérald Lomprey 2008).

En la propuesta del Metamodelo (ver capítulo III) se observa que la composición estructural de las actividades del proceso de software está compuesta por las Tareas Base y por las Tareas de Apoyo. En base a esto se estableció la hipótesis (H0, 2; H0, 3) de que las

tareas Base, por su naturaleza y la acción que tienen como propósito, aportan o contribuyen en la calidad del producto de software en comparación que las Tareas de Apoyo.

IV.3.1 Objetivos del experimento

Este experimento tuvo como objetivo principal comprobar las hipótesis: $H_{(0,2)}$ y $H_{(0,3)}$ (ver a detalle en el Capítulo I) pretendiendo lograr los siguientes objetivos:

1. Mostrar que las Tareas base inciden más en la calidad del producto de software en comparación de las Tareas de apoyo.
2. En base al objetivo anterior (1), mostrar que las Tareas base que generan artefactos tangibles, inciden más en la calidad del producto de software en comparación de las Tareas de apoyo que generan artefactos no tangibles o de resultado.

IV.3.2 Caso de estudio 1

IV.3.2.1 Metodología utilizada

El procedimiento recomendado para llevar a cabo este experimento y cumplir con los objetivos planteados fue el siguiente:

Paso 1: Obtener la información del proyecto: Se recolectó los artefactos generados a lo largo del proceso de desarrollo, como son el documento de requerimientos, documento de casos de uso, documento de diseño, documentos de administración del proyecto. Así mismo, se realizó una entrevista a cada uno de los integrantes del equipo, con el objetivo de obtener sus experiencias a lo largo del desarrollo del proyecto de software.

Paso 2: Evaluar el producto de software final, aplicando métricas de calidad al producto final. Las características de calidad seleccionada para evaluar la calidad son: Funcionalidad

y Usabilidad. Estas características fueron determinadas por el equipo de desarrollo y por el cliente final debido al tipo de información que se maneja y al usuario final.

Paso 3: Obtener el resultado de la calidad del producto, en base a las métricas aplicables. (La descripción de las métricas de calidad utilizadas se puede ver en el Apéndice B)

Paso 4: Identificar las posibles causas de los defectos identificados del producto, utilizando FMEA. (La descripción del método FMEA se puede ver en el Apéndice C)

Paso 5: Realizar el mapeo de las causas de los defectos identificados con las tareas del proceso desarrollados en el proyecto.

Paso 6: Clasificar las tareas mapeadas en base al tipo de tarea, identificando el tipo de artefacto que generan:

- Tarea Base
- Tarea Apoyo

Paso 7: Clasificar las tareas mapeadas en base al tipo de propósito de Actividad:

- Planeación y Seguimiento.
- Verificación y Validación.
- Realización Fase de Requerimientos.
- Realización Fase de Análisis.
- Realización Fase de Diseño.
- Realización Fase de Desarrollo.
- Realización Fase de Pruebas.
- Realización Fase de Cierre

IV.3.2.2 Herramientas utilizadas

Las herramientas utilizadas para este experimento fueron:

Métricas de calidad de IEC/ISO 9126: Estas métricas se utilizaron para evaluar las características de calidad del producto, tales como Funcionalidad y Usabilidad. La descripción de estas métricas se encuentra en el Apéndice B.

Metamodelo SPEM 2: Se utilizó la estructura de SPEM para la identificación de las tareas del proceso de desarrollo de software.

Plantilla de FMEA: La plantilla contiene la siguiente información (el detalle de esta plantilla puede verse en el apéndice C):

- Nombre del producto
- Modelo de fallo (defecto)
- Efectos de fallo (consecuencia)
- Grado de Severidad: Peores consecuencias
- Causa del fallo
- Grado de ocurrencia

Microsoft Excel: Utilizado para llevar a cabo las evaluaciones del producto y del proceso, así como para llevar un control de cada métrica aplicable.

IV.3.2.3 Información recolectada y tratamiento de los datos

La información recolectada y el tratamiento de los datos fueron en base a la metodología propuesta para el cumplimiento de los objetivos específicos planteados en este experimento.

Datos para la evaluación de la calidad del producto. (Paso 2 y paso 3)

Los valores de la medición obtenidos para la evaluación de las características de calidad del producto se muestran en la Tabla 6.

Para la característica de funcionalidad se evaluó una sub-característica, "adecuación", y tres atributos los cuales fueron: completud, correctud y correctud en información de interfaces. Los datos para aplicar las métricas fueron obtenidos en base a los artefactos de casos de uso y de diseño.

Para la característica de Usabilidad se evaluó una sub-característica "Fácil comprensión" y el atributo "comprensión". Los datos para poder aplicar la métrica fueron obtenidos en base al producto de software final y a los resultados obtenidos en las pruebas de usuario.

Tabla 6: Datos de la evaluación de calidad del producto de software, caso estudio 1.

Característica	Funcionalidad			
Sub-característica	Adecuación			
Atributo	Medida Base	Medida derivada	Función de calculo	Datos de entrada
Completud	NRNI= Número de requerimientos funcionales no implementados. NRF= Número de requerimientos funcionales especificados.	PC=Porcentaje de completud.	PC=(1-NRNI / NRF)*100%	NRNI =2, NRF =188
Correctud	NRII= Número de requerimientos implementados no correctos. NRFI=Número de requerimientos funcionales implementados.	PCR=Porcentaje de Correctud.	PCR= (1-NRII /NRFI)*100%	NRII =23, NRFI =187
Correctud en información de interfaces	NINC= Número de interfaces que sus datos no son correctos según las especificaciones. NII=Número de interfaces implementadas según las especificaciones.	PCII=Porcentaje de Correctud en información de interfaces.	PCII=(1-NINC/NII)*100%	NINC =11, NII =55
Característica	**Usabilidad**			
Sub-característica	**Fácil comprensión**			
Atributo	Medida Base	Medida derivada	Función de cálculo	Datos de entrada
Comprensión	NUNC= Número de requerimientos de usabilidad que son comprendidas por el usuario. NUD= Número de requerimientos de usabilidad definidas en las especificaciones.	PCIU=Porcentaje de comprensión de interfaces de usuario.	PCIU =(NUNC/NUD)*100%	NUNC =15, NUD =22

Causas de defecto del producto. (Paso 4)

En base a la evaluación de la calidad (funcionalidad y usabilidad) del producto de software se pudo obtener los defectos del producto de software, utilizando FMEA.

Para el análisis del defecto-causa se utilizó la plantilla de FMEA, la información del análisis fue:

- Identificación de los defectos del producto en base a la evaluación de la calidad del producto.
- Identificación de las causas de los defectos más significativas.

En la Tabla 7 se encuentra el mapeo de los defectos-causas identificados.

Tabla 7: Mapeo de defectos-causas adjudicadas al proceso, caso de estudio 1.

Categoría de Defectos	Causa adjudicada al proceso
No completar la implementación de las funcionalidades requeridas.	A1. No especificaron a detalle de los requerimientos funcionales.
	A2. No Identificación de todos los casos de uso (Mala identificación de todas las funcionalidades del sistema).
	A3. No validación de los casos de uso con el cliente.
	A4. Malas prácticas de estimación de los casos de uso.
	A5. El calendario no fue actualizado, en base a los reportes de seguimiento de las actividades.
Requerimientos funcionales implementados incorrectamente.	B1. No especificaron a detalle de los requerimientos funcionales.
	B2. No realizaron a detalle el plan de pruebas por caso de uso.
	B3. No se realizó a detalle el diseño de cada caso de uso especificado.
	B4. No se encontraron errores en los productos generados.
	B5. No se realizó un plan de pruebas de integración.
Redacción de la información incorrecta desplegada en pantalla.	C1. Mala redacción de los requerimientos: información a manejar en el sistema. Req.de información.
	C2. No se especificó a detalle las pruebas de interfaz de usuario.
	C3. No se encontraron defectos en las revisiones de interfaz de usuario, ni en los requerimientos.
	C4. No se definieron a detalle los requerimientos de interfaz de usuario.
	C5. No se especificó a detalle las pruebas de interfaz de usuario.
	C6. No se encontraron defectos en las revisiones de interfaz de usuario, ni en los requerimientos.
No comprensión de la Interfaz de Usuario.	D1. No se definieron a detalle los requerimientos de interfaz de usuario (requerimientos de usabilidad).
	D2. No se realizó un prototipo.
	D3. No se especificó a detalle las pruebas de interfaz de usuario.
	D4.No se encontraron defectos en las revisiones de interfaz de usuario, ni en los requerimientos especificados.

Mapeo de las causas de fallo con las tareas del proceso (Paso 5)

Al tener identificado las causas de los defectos encontrados en el producto de software, se realizó el mapeo de las causas de los defectos del producto de software con las tareas del proceso de software desarrolladas en este proyecto. En la Tabla 8 se puede ver el mapeo realizado.

Tabla 8: Mapeo de las causas adjudicadas al proceso con las tareas del proceso de software, caso de estudio 1.

Causa adjudicada al proceso	Tarea
No especificaron a detalle los requerimientos funcionales.	T1. Especificar a detalle los requerimientos funcionales.
No especificación de los requerimientos de interfaz de usuario.	T2. Especificar a detalle los requerimientos de interfaz de usuario y navegabilidad. (Requerimientos de Usabilidad).
No conocimiento ni experiencia de las tecnologías utilizadas en el proyecto.	T3. Definir el Plan de capacitación.
No identificación de todos los casos de uso (Mala identificación de todas las funcionalidades del sistema).	T4. Analizar los casos de uso generales y sus detalles.
No especificación de las validaciones de los casos de uso.	T5. Especificar las validaciones y excepciones de los casos de uso.
No validación de los casos de uso con el cliente.	T6. Validar los casos de uso definidos por el cliente/usuario.
Malas prácticas de estimación de los casos de uso	T7. Estimar el tamaño y esfuerzo por caso de uso.
El calendario no fue actualizado, en base a los reportes de seguimiento de las actividades.	T8. Dar seguimiento al calendario: Actualizar el calendario, en base a la bitácora o reporte de seguimiento semanal.
No documentación del prototipo de interfaz de usuario del sistema.	T9. Especificar y definir los componentes de la interfaz de usuario y navegabilidad.
No realizaron a detalle el plan de pruebas por caso de uso.	T10. Elaborar el plan de pruebas por cada caso de uso.
No se realizó a detalle el diseño de cada caso de uso especificado.	T11. Definir el diseño por caso de uso establecido.
No se encontraron errores en los productos generados.	T12. Aplicar las pruebas: usuario, funcionalidad, datos, interfaz de usuario.
No se realizó un plan de pruebas de integración.	T13. Aplicar el plan de pruebas de integración.

Clasificación de las tareas mapeadas. (Paso 6 y 7)

Al tener identificadas las tareas del proceso de software que generaron los defectos del producto de software, se realizó la clasificación de las tareas del proceso en base a al tipo de actividad y al propósito. Esta clasificación se basó en la propuesta del Metamodelo, en donde se describe la clasificación del tipo de tarea y el artefacto generado (ver Metamodelo propuesto en el capítulo III). En la Tabla 9 se muestra la clasificación de las tareas del proceso de software con respecto a su tipo y propósito.

Tabla 9: Clasificación de las tareas del proceso por tipo y propósito, caso de estudio 1.

ID	Tarea	Tipo	Artefacto	Propósito
T1	Especificar a detalle los requerimientos funcionales.	Base	Tangible	Realización Fase de Requerimientos.
T2	Especificar a detalle los requerimientos de interfaz de usuario y navegabilidad.	Base	Tangible	Realización Fase de Requerimientos.
T3	Definir el Plan de capacitación.	Base	Tangible	Planeación y Seguimiento.
T4	Analizar los casos de uso generales y sus detalles.	Base	Tangible	Realización Fase de Análisis.
T5	Especificar las validaciones y excepciones del caso de uso.	Base	Tangible	Realización Fase de Análisis.
T6	Validar los casos de uso definidos por el cliente/usuario.	Apoyo	No tangible	Verificación y validación.
T7	Estimar el tamaño y esfuerzo por caso de uso.	Base	Tangible	Planeación y Seguimiento.
T8	Dar seguimiento al calendario: Actualizar el calendario, en base a la bitácora o reporte de seguimiento semanal.	Base	Tangible	Planeación y Seguimiento.
T9	Especificar y definir los componentes de la interfaz de usuario y navegabilidad.	Base	Tangible	Realización de la Fase de Diseño.
T10	Elaborar el plan de pruebas por cada caso de uso.	Base	Tangible	Realización de la Fase de Pruebas.
T11	Definir el diseño por caso de uso definido.	Base	Tangible	Realización de la Fase de Diseño.
T12	Aplicar las pruebas: usuario, funcionalidad, datos, interfaz de usuario.	Base	Tangible	Realización de la Fase de Pruebas.
T13	Aplicar el plan de pruebas de integración.	Base	Tangible	Realización de la Fase de Pruebas.

IV.3.2.4 Resultados obtenidos

Evaluación del producto de software

Para el proceso de evaluación de la calidad del producto de software, en este caso de estudio, se evaluaron las características de calidad: Funcionalidad y Usabilidad. Estas características fueron determinadas por el equipo de desarrollo y por el cliente final debido al tipo de información que se maneja y al usuario que va a utilizar el producto de software. Cabe mencionar que el producto de software evaluado fue la versión final que se le entregó al cliente, según la fecha de entrega establecida en el plan de trabajo.

Los valores obtenidos de la medición para la evaluación de las características de calidad del producto se muestran en la Tabla 10.

La característica de funcionalidad fue evaluada con tres métricas de calidad diferentes, debido a que se quería saber los siguientes atributos de calidad:

Completud: El grado de qué tan completo esta el producto con respecto a su funcionalidad especificada en los requerimientos establecidos. La métrica utilizada fue Porcentaje de Completud, por lo que se encontró que el producto de software estuvo 98% completo en base a los requerimientos de funcionalidad establecidos.

Correctud: El grado con el cual el producto de software esta correcto con respecto a la funcionalidad implementada de los requerimientos establecidos. La métrica utilizada fue Porcentaje de Correctud, por lo que se encontró que el producto de software estuvo 87% correcto según los requerimientos de funcionalidad implementados.

Correctud en información de interfaces: El grado con el cual la información de las interfaces del producto fueron correctamente implementados en base a las especificaciones establecidas en los requerimientos de funcionalidad. La métrica utilizada fue Porcentaje de Correctud en información de interfaces por lo que se encontró que el producto de software estuvo 80% correcto en la información de interfaces establecidas en las especificaciones de los requerimientos funcionales.

Con respecto a la característica de Usabilidad, se evaluó en base a una sola métrica, debido a que se quería saber el siguiente atributo de calidad:

Comprensión: el grado de facilidad de comprensión de las interfaces del producto de software que tiene el usuario. La métrica utilizada fue Porcentaje de comprensión de interfaces de usuario, por lo que se encontró que el producto de software estuvo 68.19% de comprensión de interfaces de usuario en base en las funciones de interfaces de usuario definidas.

Tabla 10: Resultado de la evaluación de la calidad del producto para el caso de estudio 1.

Característica	Sub-Característica	Atributo	Métrica	Resultado Métrica
Funcionalidad	Adecuación	Completud	Porcentaje de Completud	98%
Funcionalidad	Adecuación	Correctud	Porcentaje de correctud	87%
Funcionalidad	Adecuación	Correctud en información de interfaces	Porcentaje de correctud en información de interfaces	80%
Usabilidad	Fácil de comprender	Comprensión	Porcentaje de comprensión de interfaces de usuario	68.19%

Clasificación de los tipos de tareas del proceso de software

Al tener identificadas las tareas causantes de los defectos del producto de software, se realizó la clasificación de las tareas en base a su tipo y a su propósito, esto con el fin de validar los objetivos específicos (ver en el apartado Objetivos del experimento) planteados en este experimento.

Los resultados obtenidos fueron los siguientes:

- Clasificación de las tareas por tipo de tarea y artefacto generado (véase Tabla 11).

Tabla 11: Tabla de resultados de la clasificación de tipo de tareas.

Tipo de Tarea	Artefacto	Frecuencia
Base	Tangible	12
Apoyo	No Tangible	1
Total		**13**

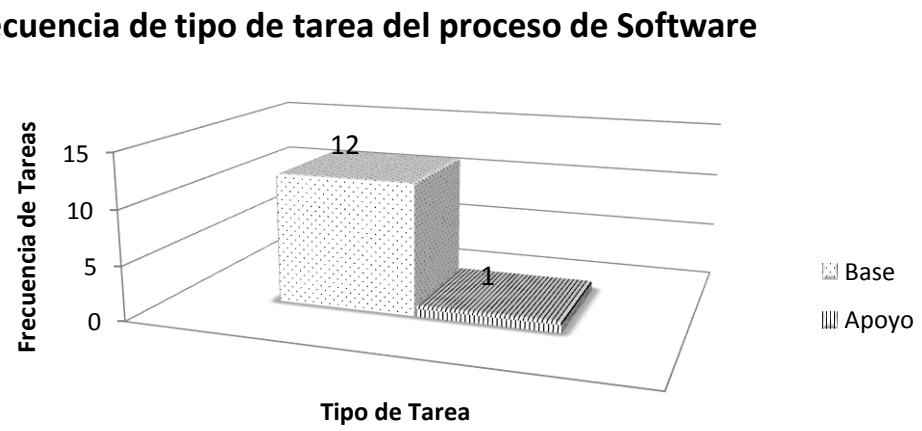

Figura 18: Gráfica del resultado de la clasificación de las tareas del proceso, caso de estudio 1.

En la gráfica de la Figura 18, se observa como las tareas de tipo base que generan artefactos tangibles son las que más contribuyen a la generación de defectos en comparación con las tareas de apoyo que generan artefactos no tangibles.

- Clasificación de las tareas por propósito (véase Tabla 12).

Tabla 12: Resultado de la clasificación de las tareas por propósito.

Propósito de Actividad	Frecuencia
Planeación y Seguimiento.	3
Realización Fase de Requerimientos.	2
Realización Fase de Análisis.	2
Realización Fase de Pruebas.	3
Verificación y validación.	1
Realización de la Fase de Diseño.	2
Total	13

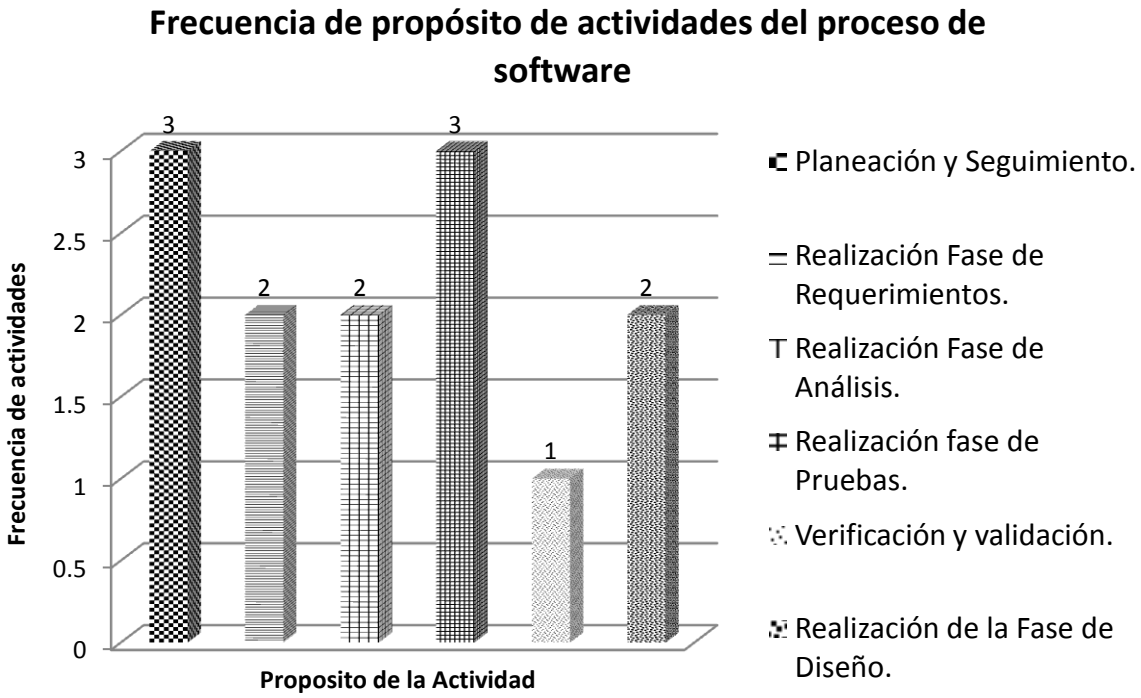

Figura 19: Gráfica del resultado de la clasificación de propósito de actividades del proceso de software, caso de estudio 1.

En Figura de la gráfica 19, se puede ver que en este caso particular las tareas que más impactaron fueron las que tienen como propósito la planeación y seguimiento asi como la realización de la fase de pruebas, por lo que se considera que las otras tareas con propósito de Fase de Requerimientos, Fase de Análisis, Fase de Verificación y Validación y de la Fase de Diseño impactaron en la calidad del producto pero a menor escala.

IV.3.3 Caso de estudio 2

IV.3.3.1 Metodología utilizada

El procedimiento recomendado para llevar a cabo este experimento y cumplir con los objetivos específicos planteados fue el siguiente.

En base a los principios de las metodologías agiles y en este caso de SCRUM (ver Apéndice D), se realizó la evaluación de la calidad del producto de software y la identificación de la causa del defecto del producto, por cada Iteración o Sprint definidos en el proyecto.

La metodología utilizada que se aplicó en cada iteración o Sprint del proyecto fue la siguiente:

Paso 1: Obtener la información de cada Sprint. Se recolectó los artefactos generados en el Sprint, como son los Sprint Backlog, Sprint Meeting (Reporte de junta), Revisión del Sprint, Retrospectiva del Sprint.

Paso 2: Evaluar el producto de software de cada Iteración o Sprint, aplicando las métricas de calidad al producto de software desarrollado. Las características de calidad seleccionada para evaluar la calidad fueron: Funcionalidad y Usabilidad. Estas características fueron determinadas por el equipo de desarrollo, debido al tipo de información que se maneja y al usuario que lo va a utilizar.

Paso 3: Es el mismo que en el caso de estudio 1.

Paso 4: Identificar las posibles causas de los defectos del producto de software desarrollado en el Sprint o Iteración, en base al análisis de retrospectiva que realizaba el equipo de desarrollo. El análisis de retrospectiva de cada Sprint se realiza en la etapa final de cada una de ellas, por lo que facilitó la identificación de las causas de los defectos. (Véase en el apéndice E, la descripción del Formato de Retrospectiva)

Paso 5, Paso 6 y Paso 7: Son los mismos que en el caso de estudio 1.

IV.3.3.2 Herramientas utilizadas

Las herramientas utilizadas para este experimento fueron:

Métricas de calidad de IEC/ISO 9126: Estas métricas se utilizaron para evaluar las características de calidad del producto, tales como Funcionalidad y Usabilidad. La descripción de estas métricas se encuentra en el Apéndice B.

Metamodelo SPEM 2: Se utilizó la estructura de SPEM para la identificación de las tareas del proceso de desarrollo de software.

Documento de Retrospectiva del Sprint: el documento contiene la siguiente información:

- Información de la reunión de retrospectiva: Fecha, número del Sprint, objetivo del Sprint, hora de inicio y fin de la reunión de retrospectiva.(Ver detalle en el Apéndice E)
    - Introducción del documento
    - Información general de Sprint
    - Métricas de calidad
    - Revisión del proceso:
        - Procesos más efectivos en el Sprint
        - Proceso no efectivos en el Sprint
        - Acciones de mejora
    - Varianza

Microsoft Excel: utilizado para llevar a cabo las evaluaciones del producto y del proceso, así como para llevar un control de cada métrica aplicable.

IV.3.3.3 Información recolectada y tratamiento de los datos

La información recolectada y el tratamiento de los datos fueron en base a la metodología propuesta para el cumplimiento de los objetivos planteados en este experimento.

Datos para la evaluación de la calidad del producto. (Paso 2 y paso 3)

Los datos obtenidos de la medición para la evaluación de las características de calidad del producto se muestran en la siguiente Tabla 13. La descripción de las métricas aplicadas a cada atributo son: (véase a detalle en el apéndice B).

PC=Porcentaje de Completud

PCR= Porcentaje de Correctud

PCII=Porcentaje de Correctud en Información de Interfaces

PCIU=Porcentaje de Comprensión de Interfaces de Usuario

Tabla 13: Datos de la evaluación de calidad del producto de software, caso estudio 2.

Característica		**Funcionalidad**				
Sub-Característica		**Adecuación**				
Métrica	Datos de Entrada Sprint 1	Datos de Entrada Sprint 2	Datos de Entrada Sprint 3	Datos de Entrada Sprint 4	Datos de Entrada Sprint 5	Datos de Entrada Sprint 6
PC	NRNI =0, NRF =3	NRNI = 0, NRF = 6	NRNI = 7, NRF =16	NRNI =6, NRF =20	NRNI =6, NRF =11	NRNI =6, NRF =32
PCR	NRII =3, NRFI =3	NRII =2, NRFI =6	NRII =4, NRFI =9	NRII =7, NRFI =14	NRII =2, NRFI =5	NRII =8, NRFI =26
PCII	NINC =5, NII =7	NINC =5, NII =7	NINC =6, NII =9	NINC =5, NII =14	NINC =5, NII =14	NINC =6, NII =14
Característica		**Usabilidad**				
Sub-Característica		**Fácil comprensión**				
Métrica	Datos de Entrada Sprint 1	Datos de Entrada Sprint 2	Datos de Entrada Sprint 3	Datos de Entrada Sprint 4	Datos de Entrada Sprint 5	Datos de Entrada Sprint 6
PCIU	NUNC =2, NUD =3	NUNC = 5, NUD = 6	NUNC = 3, NUD =6	NUNC = 5, NUD =14	NUNC = 0, NUD = 11	NUNC = 30, NUD = 32

Causas de defecto del producto. (Paso 4)

En base al análisis de retrospectiva de cada Sprint que se realiza en cada reunión de retrospectiva a finalizar el Sprint, se pudo identificar las causas de defecto del producto de software por cada Sprint.

Para el análisis de retrospectiva se utilizó una plantilla definida por el equipo de desarrollo, pero cuidando los puntos señalados por los principios de SCRUM (ver apéndice D).

En la plantilla de retrospectiva de cada Sprint se captura la siguiente información:

- Actividades del proceso que fueron más efectivos en el Sprint.
- Actividades del proceso que fueron menos efectivos en el Sprint.
- Acciones de mejora.

En las Tablas de la 14 a la 19, se encuentra la relación de los defectos del producto con las causas raíz de estos defectos identificados en cada sprint. Las causas de los defectos del producto fueron tomadas en base las actividades menos efectivas identificadas en la revisión de retrospectiva de cada Sprint.

Tabla 14: Mapeo del defecto-causa adjudicado al proceso del Sprint 1, caso de estudio 2.

Defecto	Causa adjudicada al proceso
D1. Validaciones de la información presentada: datos obligatorios y tipos de datos. D2. Mal uso de componentes en la Interfaz de Usuario. D3.Errores de ortografía de la información presentada.	C1. La comunicación con el cliente fue difícil.
	C2. Inactividad de un compañero.
	C3. Realizar las pruebas antes de que se entregue el sprint en curso.
	C4. No asistencia de todos los integrantes del equipo en la reunión de planeación.
	C5. No Planear y estimar el tiempo de cada actividad y registrar el tiempo actual de las actividades de desarrollo.
	C6. No se realizaron las revisiones de los planes y productos de trabajo entre colegas.

Tabla 15: Mapeo del defecto-causa adjudicado al proceso del Sprint 2, caso de estudio 2.

Defecto	Causa adjudicada al proceso
D2. Mal uso de componentes en la Interfaz de Usuario. D3. Errores de ortografía de la información	C1. La comunicación con el cliente fue difícil.
	C4. No asistencia de todos los integrantes del equipo en la reunión de planeación.

presentada. D4. Validaciones de las excepciones y mensajes de aviso.	C6. No se realizaron las revisiones de los planes y productos de trabajo entre colegas.
	C7. No se llevo el plan de admón. de configuración: perdida de versiones de productos de trabajo.
	C8. Tardanza en la definición del manejador para la base de datos a utilizar.
	C9. No Realizar las pruebas antes de que se entregue el sprint en curso.
	C10. No se realizó Plan de integración de los módulos.
	C11. No especificación detallada de los requerimientos. Solo se utilizo Usar la especificación de la encuesta.

Tabla 16: Mapeo del defecto-causa adjudicado al proceso del Sprint 3, caso de estudio 2.

Defecto	Causa adjudicada al proceso
D1. Validaciones de la información presentada: datos obligatorios y tipos de datos. D2. Mal uso de componentes en la Interfaz de Usuario. D4.Validaciones de las excepciones y mensajes de aviso.	C7. No se llevo el plan de admón. de configuración: perdida de versiones de productos de trabajo.
	C10. No se realizó Plan de integración de los módulos.
	C12. El cambio de tecnología causo: retraso de calendario, tiempo en la curva de aprendizaje, reconfiguración del ambiente de desarrollo, etc.
	C13. No se dio seguimiento a todas las actividades desarrolladas.
	C14. No se realizaron las pruebas unitarias.
	C15. No se realizaron las pruebas de integración.

Tabla 17: Mapeo del defecto-causa adjudicado al proceso del Sprint 4, caso de estudio 2.

Defecto	Causa adjudicada al proceso
D4. Validaciones de las excepciones y mensajes de aviso. D5. Validaciones de reglas de negocio.	C7. No se realizó un plan para la administración de la configuración.
	C15. No se realizaron las pruebas de integración.
	C16. No se realizó un plan de capacitación.
	C17. No se realizó un plan de infraestructura para las necesidades de desarrollo e implementación.
	C18. No se realizó documentación de diseño: no diagramas de clases, No diagramas de ER, no flujos.
	C19. No se realizaron minutas o reportes de las reuniones.
	C20. No se realizaron los prototipos.

Tabla 18: Mapeo del defecto-causa adjudicado al proceso del Sprint 5, caso de estudio 2.

Defecto	Causa adjudicada al proceso
D5. Validaciones de reglas de negocio. D6. No completar las funcionalidades requeridas.	C7. No se realizó el plan de administración de la configuración.
	C15. No se realizó un plan de pruebas de integración.
	C21. No se realizó plan de trabajo del Sprint.
	C22. Reusarse a aprender RichFaces para una interfaz y mejor manejo.
	C23. No asistir a las reuniones del equipo y así no conocer los avances que se van teniendo.
	C24. No visitar periódicamente la página del equipo y no actualizar las horas que cada uno ha dedicado al trabajo, retrasando con ello los documentos y mediciones del producto.

Tabla 19: Mapeo del defecto-causa adjudicado al proceso del Sprint 6, caso de estudio 2.

Defecto	Causa adjudicada al proceso
D6. No completar las funcionalidades requeridas.	C13. No se dio seguimiento a todas las actividades desarrolladas.
	C15. No plan de pruebas de integración.
	C25. No se realizó reporte de retrospectiva.

Mapeo de las causas de defecto con las tareas del proceso (Paso 5)

Al tener identificadas las causas de los defectos encontrados en cada Sprint o iteración, se realizó el mapeo de las causas de los defectos del producto de software con las tareas del proceso de software desarrolladas en este proyecto.

A continuación se describe el mapeo de las causas de defecto con las tareas del proceso por Sprint (véase en las Tablas 20-25):

Tabla 20: Mapeo de las causas adjudicadas al proceso con las tareas del proceso de software del Sprint 1, caso de estudio 2.

Causa de defecto	Tarea
C1. La comunicación con el cliente fue difícil.	T1. Verificar plan de comunicación con el cliente.
	T2. Definir herramientas para la recolección de los requerimientos con el cliente.
C2. Inactividad de un compañero.	T3. Elaborar el plan de trabajo.
	T4. Dar seguimiento al plan de trabajo.

	T5. Realizar el plan de capacitación.
	T6. Actualizar bitácoras de actividades.
C3. Realizar las pruebas antes de que se entregue el sprint en curso.	T7. Realizar plan de Pruebas unitarias.
	T8. Realizar plan de pruebas de integración.
	T9. Ejecutar pruebas unitarias.
	T10. Ejecutar pruebas de integración.
C4. No asistencia de todos los integrantes del equipo en la reunión de planeación.	T11. Realizar minutas en la juntas de trabajo.
C5. No Planear y estimar el tiempo de cada actividad y registrar el tiempo actual de las actividades de desarrollo.	T3. Elaborar el plan de trabajo.
	T12. Realizar calendario de trabajo.
	T4. Dar seguimiento al plan de trabajo.
C6. No se realizaron las revisiones de los planes y productos de trabajo entre colegas.	T13. Verificar y validar los planes de trabajo.
	T14. Verificar y validar los productos entregables.

Tabla 21: Mapeo de las causas adjudicadas al proceso con las tareas del proceso de software del Sprint 2, caso de estudio 2.

Causa de defecto	Tarea
C1. La comunicación con el cliente fue difícil	T1. Verificar el plan de comunicación con el cliente.
	T2. Definir herramientas para la recolección de los requerimientos con el cliente.
C4. No asistencia de todos los integrantes del equipo en la reunión de planeación.	T3. Realizar minutas en la juntas de trabajo.
C6. No se realizaron las revisiones de los planes y productos de trabajo entre colegas.	T4. Verificar y validar los planes de trabajo.
	T5. Verificar y validar los productos entregables.
C7. No se llevo el plan de admón. de configuración: perdida de versiones de productos de trabajo.	T6.Crear plan de administración de la configuración.
	T7. Definir herramienta para la admón. De la configuración.
	T8. Verificar y validar la admón. En la configuración.
C8. Tardanza en la definición del manejador para la base de datos a utilizar.	T9. Definir Arquitectura de software.
	T10. Definir herramienta a utilizar para el desarrollo.
	T11. Realizar el plan de adquisición y capacitación.
C9. No Realizar las pruebas antes de que se entregue el sprint en curso.	T12. Realizar plan de Pruebas unitarias.
	T13. Realizar plan de pruebas de integración.
	T14. Realizar reportes de pruebas unitarias.
	T15. Realizar reportes de pruebas de integración.
C10. No se realizó Plan de integración de los módulos.	T16. Realizar plan de Pruebas integración.
C11. No especificación detallada de los requerimientos. Solo se utilizo Usar la especificación de la encuesta.	T17. Definir los requerimientos.
	T18. Realizar Análisis de requerimientos.

Tabla 22: Mapeo de las causas adjudicadas al proceso con las tareas del proceso de software del Sprint 3, caso de estudio 2.

Causa de defecto	Tarea
C7. No se llevo el plan de admón. de configuración: perdida de versiones de productos de trabajo.	T1. Realizar el plan de admón. De configuración.
	T2. Dar seguimiento del plan de admón. De configuración.
C10. No se realizó Plan de integración de los módulos.	T3. Realizar plan de pruebas de integración.
C12. El cambio de tecnología causo: retraso de calendario, tiempo en la curva de aprendizaje, reconfiguración del ambiente de desarrollo, etc.	T4. Realizar plan de riesgos.
	T5. Realizar plan de capacitación.
C13. No se dio seguimiento a todas las actividades desarrolladas.	T6. Dar seguimiento al plan de trabajo.
	T7. Realizar bitácora de las actividades.
C14. No se realizaron las pruebas unitarias.	T8. Realizar el plan de pruebas unitarias.
	T9. Realizar el reporte de pruebas unitarias.
C15. No se realizaron las pruebas de integración.	T10. Realizar el plan de pruebas de integración.
	T11. Realizar el reporte de pruebas de integración.

Tabla 23: Mapeo de las causas adjudicadas al proceso con las tareas del proceso de software del Sprint 4, caso de estudio 2.

Causa de defecto	Tarea
C7. No se realizó un plan para la administración de la configuración.	T1. Realizar el plan de admón. De configuración.
	T2. Dar seguimiento del plan de admón. De configuración.
C15. No se realizaron las pruebas de integración.	T3. Realizar el plan de pruebas de integración.
	T4. Ejecutar el plan de pruebas de integración.
	T5. Realizar el reporte de pruebas de integración.
C16. No se realizó un plan de capacitación.	T6. Elaborar el plan de capacitación.
C17. No se realizó un plan de infraestructura para las necesidades de desarrollo e implementación.	T7. Realizar el plan de infraestructura y capacitación.
C18. No se realizó documentación de diseño: no diagramas de clases, No diagramas de ER, no flujos.	T8. Realizar el diseño de clases.
	T9. Realizar el diseño de diagramas Entidad-Relación.
	T10. Realizar los diagramas de flujos de datos.
C19. No se realizaron minutas o reportes de las reuniones.	T11. Generación de minutas en la juntas de trabajo.
C20. No se realizan prototipos.	T12. Realizar el diseño de prototipo.

Tabla 24: Mapeo de las causas adjudicadas al proceso con las tareas del proceso de software del Sprint 5, caso de estudio 2.

Causa de defecto	Tarea
C7. No se realizó el plan de administración de la configuración.	T1. Realizar el plan de admón. De configuración.
	T2. Dar seguimiento del plan de admón. De configuración.
C15. No se realizó un plan de pruebas de integración.	T3. Realizar el plan de pruebas de integración.
	T4. Ejecutar el plan de pruebas de integración.
	T5. Realizar el reporte de pruebas de integración.
C21. No se realizó plan de trabajo del Sprint.	T6. Realizar el plan de trabajo.
C22. Reusarse a aprender RichFaces para una interfaz y mejor manejo.	T7. Realizar el plan de capacitación.
C23. No asistir a las reuniones del equipo y así no conocer los avances que se van teniendo.	T8. Realizar minutas en la juntas de trabajo.
C24. No visitar periódicamente la página del equipo y no actualizar las horas que cada uno ha dedicado al trabajo, retrasando con ello los documentos y mediciones del producto.	T9. Dar seguimiento del plan de trabajo.
	T10. Realizar bitácora de las actividades.

Tabla 25: Mapeo de las causas adjudicadas al proceso con las tareas del proceso de software del Sprint 6, caso de estudio 2.

Causa de defecto	Tarea
C13. No se dio seguimiento a todas las actividades desarrolladas.	T1. Dar seguimiento al plan de trabajo.
	T2. Realizar bitácora de las actividades.
C15. No plan de pruebas de integración.	T3. Realizar el plan de pruebas de integración.
	T4. Ejecutar el plan de pruebas de integración.
	T5. Realizar el reporte de pruebas de integración.
C25. No se realizó reporte de retrospectiva.	T6. Realizar retroalimentación.

Clasificación de las actividades mapeadas. (Paso 6 y paso 7)

Al tener identificadas las tareas del proceso de software de cada Sprint, las cuales generaron los defectos del producto de software, se realizó la clasificación de las tareas del proceso en base a al tipo de actividad y al propósito. Esta clasificación se baso en la propuesta del Metamodelo, en donde se describe la clasificación del tipo de tarea y el artefacto generado (ver Metamodelo propuesto en el capítulo III).

En las siguientes Tablas 26-31, se muestran la clasificación de las tareas del proceso de software con respecto a su tipo y propósito por cada Sprint: (Paso 6 y paso 7):

Tabla 26: Clasificación de las tareas del proceso por tipo y propósito Sprint 1, caso de estudio 2.

Tarea	Tipo	Artefacto	Propósito
T1. Verificar un plan de comunicación con el cliente.	Apoyo	No Tangible	Planeación y Seguimiento.
T2. Definir herramientas para la recolección de los requerimientos con el cliente.	Base	Tangible	Realización Fase de Requerimientos.
T3. Elaborar el plan de trabajo.	Base	Tangible	Planeación y Seguimiento.
T4. Dar seguimiento al plan de trabajo.	Apoyo	No Tangible	Planeación y Seguimiento.
T5. Realizar el plan de capacitación.	Base	Tangible	Planeación y Seguimiento.
T6. Actualizar bitácoras de actividades.	Base	Tangible	Planeación y Seguimiento.
T7. Realizar plan de Pruebas unitarias.	Base	Tangible	Realización Fase de Pruebas.
T8. Realizar plan de pruebas de integración.	Base	Tangible	Realización Fase de Pruebas.
T9. Ejecutar pruebas unitarias.	Base	Tangible	Realización Fase de Pruebas.
T10. Ejecutar pruebas de integración.	Base	Tangible	Realización Fase de Pruebas.
T11. Realizar minutas en la juntas de trabajo.	Base	Tangible	Planeación y Seguimiento.
T12. Realizar calendario de trabajo.	Base	Tangible	Planeación y Seguimiento.
T13. Verificar y validar los planes de trabajo.	Apoyo	No Tangible	Verificación y Validación.
T14. Verificar y validar los productos entregables	Apoyo	No Tangible	Verificación y Validación.

Tabla 27: Clasificación de las tareas del proceso por tipo y propósito Sprint 2, caso de estudio 2.

Tarea	Tipo	Artefacto	Propósito
T1. Verificar el plan de comunicación con el cliente	Apoyo	No Tangible	Verificación y Validación
T2. Definir herramientas para la recolección de los requerimientos con el cliente	Base	Tangible	Realización Fase de Requerimientos
T3. Realizar minutas en la juntas de trabajo	Base	Tangible	Planeación y Seguimiento
T4. Verificar y validar los planes de trabajo	Apoyo	No Tangible	Verificación y Validación
T5. Verificar y validar los productos entregables.	Apoyo	No Tangible	Verificación y Validación
T6.Crear plan de administración de la configuración	Base	Tangible	Planeación y Seguimiento
T7. Definir herramienta para la admón. De la configuración	Base	Tangible	Planeación y Seguimiento
T8. Verificar y validar la admón. En la configuración	Apoyo	No Tangible	Verificación y Validación
T9. Definir Arquitectura de software	Base	Tangible	Realización Fase de Diseño
T10. Definir herramienta a utilizar para el desarrollo	Base	Tangible	Realización Fase de Diseño
T11. Realizar el plan de adquisición y capacitación	Base	Tangible	Planeación y Seguimiento
T12. Realizar plan de Pruebas unitarias	Base	Tangible	Realización Fase de

			Pruebas
T13. Realizar plan de pruebas de integración	Base	Tangible	Realización Fase de Pruebas
T14. Realizar reportes de pruebas unitarias	Base	Tangible	Realización Fase de Pruebas
T15. Realizar reportes de pruebas de integración	Base	Tangible	Realización Fase de Pruebas
T16. Realizar plan de Pruebas integración	Base	Tangible	Realización Fase de Pruebas
T17. Definir los requerimientos	Base	Tangible	Realización Fase de Requerimientos
T18. Realizar Análisis de requerimientos	Base	Tangible	Realización Fase de Análisis

Tabla 28: Clasificación de las tareas del proceso por tipo y propósito Sprint 3, caso de estudio 2.

Tarea	Tipo	Artefacto	Propósito
T1. Realizar el plan de admón. De Configuración.	Base	Tangible	Planeación y Seguimiento.
T2. Dar seguimiento del plan de admón. De configuración.	Apoyo	No Tangible	Planeación y Seguimiento.
T3. Realizar plan de pruebas de integración.	Base	Tangible	Realización Fase de Pruebas.
T4. Realizar plan de riesgos.	Base	Tangible	Planeación y Seguimiento.
T5. Realizar plan de capacitación.	Base	Tangible	Planeación y Seguimiento.
T6. Dar seguimiento al plan de trabajo.	Apoyo	No Tangible	Planeación y Seguimiento.
T7. Realizar bitácora de las actividades.	Base	Tangible	Planeación y Seguimiento.
T8. Realizar el plan de pruebas unitarias.	Base	Tangible	Realización Fase de Pruebas.
T9. Realizar el reporte de pruebas unitarias.	Base	Tangible	Realización Fase de Pruebas.
T10. Realizar el plan de pruebas de integración.	Base	Tangible	Realización Fase de Pruebas.
T11. Realizar el reporte de pruebas de integración.	Base	Tangible	Realización Fase de Pruebas.

Tabla 29: Clasificación de las tareas del proceso por tipo y propósito Sprint 4, caso de estudio 2.

Actividad	Tipo	Artefacto	Propósito
T1. Realizar el plan de admón. De configuración.	Base	Tangible	Planeación y Seguimiento.
T2. Dar seguimiento del plan de admón. De configuración.	Apoyo	No Tangible	Planeación y Seguimiento.
T3. Realizar el plan de pruebas de integración.	Base	Tangible	Realización Fase de Pruebas.
T4. Ejecutar el plan de pruebas de integración.	Base	Tangible	Realización Fase de Pruebas.
T5. Realizar el reporte de pruebas de	Base	Tangible	Realización Fase de Pruebas.

integración.			
T6. Elaborar el plan de capacitación.	Base	Tangible	Planeación y Seguimiento.
T7. Realizar el plan de infraestructura y capacitación.	Base	Tangible	Planeación y Seguimiento.
T8. Realizar el diseño de clases.	Base	Tangible	Realización Fase de Diseño.
T9. Realizar el diseño de diagramas Entidad-Relación.	Base	Tangible	Realización Fase de Diseño.
T10. Realizar los diagramas de flujos de datos.	Base	Tangible	Realización Fase de Diseño.
A11. Generación de minutas en la juntas de trabajo.	Base	Tangible	Planeación y Seguimiento.
T12. Realizar el diseño de prototipo.	Base	Tangible	Realización Fase de Diseño.

Tabla 30: Clasificación de las tareas del proceso por tipo y propósito Sprint 5, caso de estudio 2.

Tarea	Tipo	Artefacto	Propósito
T1. Realizar el plan de admón. De configuración	Base	Tangible	Planeación y Seguimiento
T2. Dar seguimiento del plan de admón. De configuración	Apoyo	No Tangible	Planeación y Seguimiento
T3. Realizar el plan de pruebas de integración	Base	Tangible	Realización Fase de Pruebas
T4. Ejecutar el plan de pruebas de integración	Base	Tangible	Realización Fase de Pruebas
T5. Realizar el reporte de pruebas de integración	Base	Tangible	Realización Fase de Pruebas
T6. Realizar el plan de trabajo	Base	Tangible	Planeación y Seguimiento
T7. Realizar el plan de capacitación	Base	Tangible	Planeación y Seguimiento
T8. Realizar minutas en la juntas de trabajo	Base	Tangible	Planeación y Seguimiento
T9. Dar seguimiento del plan de trabajo	Apoyo	No Tangible	Planeación y Seguimiento
T10. Realizar bitácora de las actividades	Base	Tangible	Planeación y Seguimiento

Tabla 31: Clasificación de las tareas del proceso por tipo y propósito Sprint 6, caso de estudio 2.

Tarea	Tipo	Artefacto	Propósito
T1. Dar seguimiento al plan de trabajo	Apoyo	No Tangible	Planeación y Seguimiento
T2. Realizar bitácora de las actividades	Base	No Tangible	Planeación y Seguimiento
T3. Realizar el plan de pruebas de integración	Base	Tangible	Realización Fase de Pruebas
T4. Ejecutar el plan de pruebas de integración	Base	Tangible	Realización Fase de Pruebas
T5. Realizar el reporte de pruebas de	Base	Tangible	Realización Fase de

integración			Pruebas
T6. Realizar retroalimentación	Base	Tangible	Planeación y Seguimiento

IV.3.3.4 Resultados obtenidos

Evaluación del producto de software

Para el proceso de evaluación de la calidad del producto de software, en este caso de estudio se evaluaron las características de calidad: Funcionalidad y Usabilidad. Estas características fueron seleccionadas por el equipo de desarrollo y por el asesor del proyecto, debido a que sus objetivos era conocer si el producto entregado al cliente cumple con la funcionalidad correspondiente, el grado con el cual el producto de software está libre de defectos, así como la facilidad de uso por el usuario principal. Ya que su principal objetivo es la satisfacción del cliente.

Estas características de calidad se definieron al inicio del proyecto y se acordó que se aplicaría en cada Sprint realizado, debido a que en cada Sprint se planeaba generar módulos completos del producto de software final.

Los valores obtenidos para la evaluación de las características de calidad del producto de software por Sprint se muestran en las Tablas 32 y 33.

La característica de funcionalidad fue evaluada con tres métricas de calidad diferentes, debido a que se quería saber los siguientes atributos de calidad:

Completud: El grado de que tan completo esta el módulo desarrollado en cada sprint con respecto a los requerimientos de funcionalidad establecidos en cada iteración, por lo que se encontró que el módulo generado en el Sprint 1 estuvo 100% completo, en el Sprint 2 estuvo 100% completo, en el Sprint 3 estuvo 56% completo, en el Sprint 4 estuvo 70% completo, en el Sprint 4 estuvo 45% completo, en el Sprint 6 estuvo 81% completo.

Correctud: El grado con el cual el módulo desarrollado en cada Sprint esta correcto con respecto a la funcionalidad implementada de los requerimientos establecidos. La métrica utilizada fue ***Porcentaje de Correctud***, por lo que se encontró que el módulo generado en el Sprint 1 estuvo en un 85% correcto, en el sprint 2 estuvo en un 94% correcto, en el Sprint 3 estuvo en un 60% correcto, en el Sprint 4 estuvo en un 30%

correcto, en el Sprint 5 estuvo en un 50% correcto, y en el Sprint 6 estuvo en un 82% correcto, según los requerimientos de funcionalidad establecidos en cada Sprint.

Correctud en información de interfaces: El grado con el cual la información de las interfaces del producto fueron correctamente implementados en base a las especificaciones establecidas en los requerimientos de funcionalidad de cada módulo desarrollado en cada Sprint. La métrica utilizada fue ***Porcentaje de Correctud*** en información de interfaces por lo que se encontró que el Sprint 1 estuvo un 29% correcto en la información de interfaces, en el Sprint 2 estuvo 29% correcto en la información de interfaces, en el Sprint 3 estuvo 33% correcto en la información de interfaces, en el Sprint 4 estuvo 64% correcto en la información de interfaces, en el Sprint 5 estuvo 64% correcto en la información de interfaces, el Sprint 6 estuvo 81% correcto en la información de interfaces.

Con respecto a la característica de Usabilidad, se evaluó en base a una métrica, debido a que se quería saber el siguiente atributo de calidad.

Comprensión: El grado de facilidad de entendimiento del de las interfaces que tiene el usuario en cada módulo desarrollado (producto de software) en cada Sprint. La métrica utilizada fue el ***Porcentaje de comprensión*** de interfaces de usuario, por lo que se encontró que en el Sprint 1 estuvo 67% fácil de comprender por el usuario, en el Sprint 2 estuvo 83% fácil de comprender por el usuario, en el Sprint 3 estuvo 50% de comprender por el usuario, en el Sprint 4 estuvo 36% de comprender por el usuario, en el Sprint 5 estuvo 0% de comprender por el usuario, en el Sprint 6 estuvo 94% de comprender por el usuario.

Tabla 32: Resultado de la evaluación de la característica de funcionalidad del producto por Sprint, para el caso de estudio 2.

Característica	Funcionalidad						
Sub-característica	Adecuación						
		Resultado Métrica					
Atributo	Métrica	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6
Completud	Porcentaje de Completud	100%	100%	56%	70%	45%	81%
Correctud	Porcentaje de correctud	85%	94%	60%	30%	50%	82%

Correctud en información de interfaces	Porcentaje de correctud en información de interfaces	29%	29%	33%	64%	64%	81%

Tabla 33: Resultado de la evaluación de la característica de usabilidad del producto por Sprint, caso de estudio 2.

Característica	Usabilidad						
Sub-característica	Fácil comprensión						
		Resultado Métrica					
Atributo	Métrica	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6
Comprensión	Porcentaje de comprensión de interfaces de usuario	67%	83%	50%	36%	0%	94%

Clasificación de los tipos de tareas del proceso de software

Al tener identificadas las tareas causantes de los defectos del producto de software de cada Sprint, se realizó la clasificación de las tareas en base a su tipo y a su propósito, esto con el fin de validar los objetivos específicos (ver en el apartado Objetivos del experimento) planteado en este experimento.

Los resultados fueron los siguientes:

Resultados por tipo de Tarea fueron los siguientes (véase Tabla 34 y figura 20):

Tabla 34. Tabla de resultados de la clasificación de tipo de tareas por Sprint.

Sprint	Base	Apoyo
Sprint 1	10	4
Sprint 2	13	5
Sprint 3	8	3
Sprint 4	11	1
Sprint 5	8	2
Sprint 6	5	1

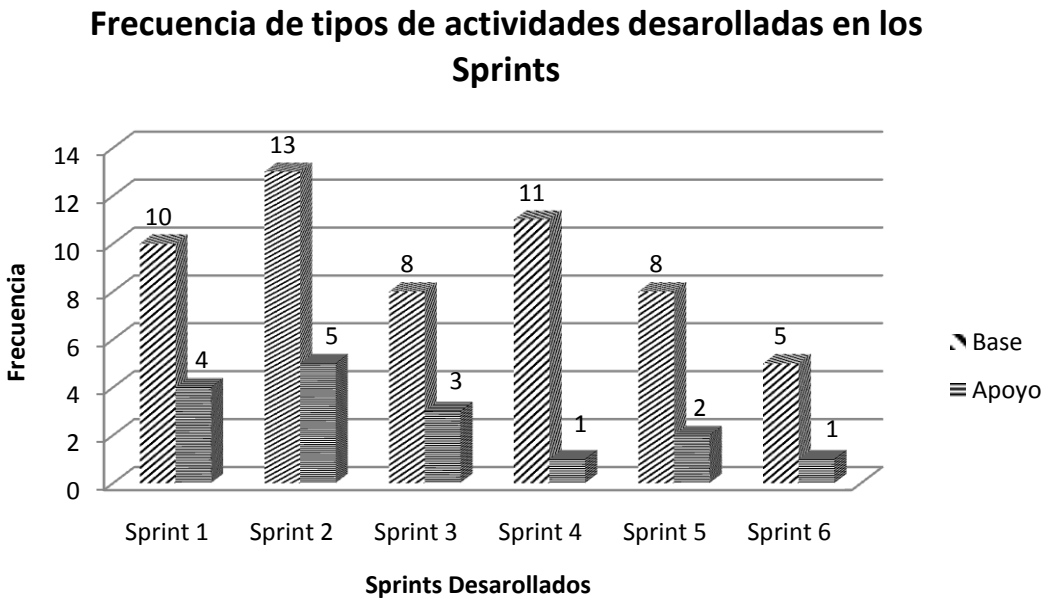

Figura 20: Gráfica del resultado de la clasificación de tipo de tarea del proceso por Sprint, caso de estudio 2.

En la gráfica de la Figura 20 se observa como las tareas de tipo Base que generan artefactos tangibles son las que más contribuyen a la generación de defectos del producto en comparación con las tareas de apoyo que generan artefactos no tangibles.

- Resultados por propósito de la actividad (véase Tabla 35 y Figura 21):

Tabla 35: Resultado de la clasificación de las tareas por propósito de cada Sprint.

	Planeación y Seguimiento (PyS)	Realización Fase de Requerimientos (RFR)	Realización Fase de Análisis (RFA)	Realización de la Fase de Diseño (RFD)	Realización Fase de Pruebas (RFP)	Verificación y validación (VyV)
Sprint 1	8	1	0	0	4	3
Sprint 2	4	2	1	2	5	4
Sprint 3	6	0	0	0	5	0
Sprint 4	5	0	0	4	3	0
Sprint 5	7	0	0	0	3	0
Sprint 6	3	0	0	0	3	0

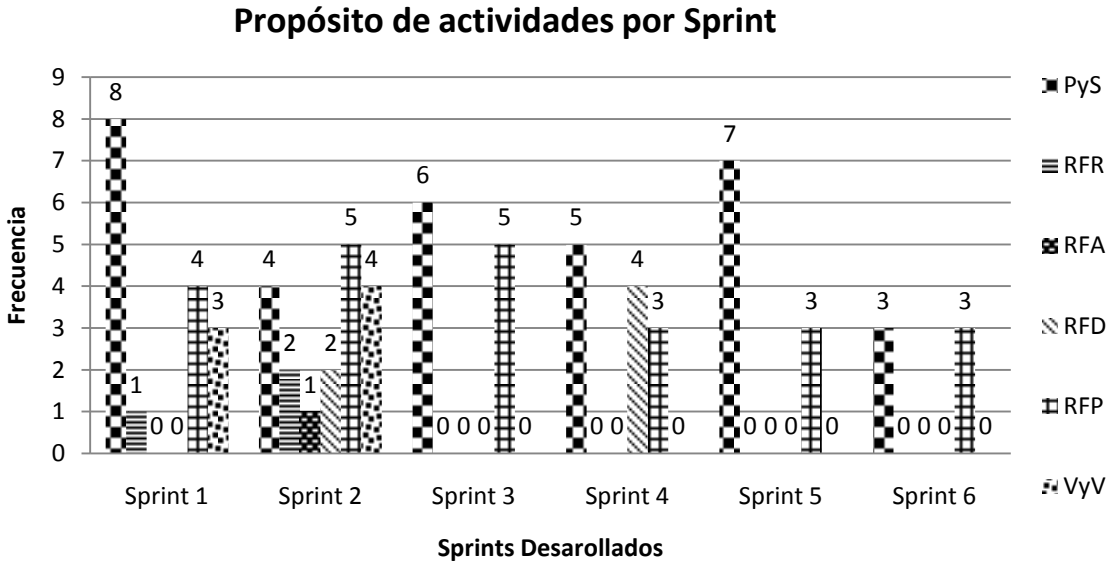

Figura 21: Gráfica del resultado de la clasificación de propósito de actividades del proceso de software por Sprint, caso de estudio 2.

En la gráfica de la Figura 21 se observa, que en este caso particular las tareas que más impactaron en cada Sprint fueron las que tienen como propósito la planeación, seguimiento y la realización de la fase de pruebas.

IV.3.4 Caso de estudio 3

IV.3.4.1 Metodología utilizada

El procedimiento recomendado para llevar a cabo este experimento y cumplir con los objetivos planteados es el siguiente:

Paso 1: Obtener la información del proyecto: se recolectaron los artefactos generados a lo largo del proceso de desarrollo, como son el documento de requerimientos, documento de casos de uso, plan de pruebas y reportes técnicos sobre el proceso de desarrollo, y de administración del proyecto.

Paso 2: Es el mismo paso que en el caso de estudio 1. Las características fueron determinadas por los líderes del proyecto de la empresa como de la UABC, teniendo en mente el cliente final.

Paso 3: Es el mismo paso que en el caso de estudio 1.

Paso 4: Es el mismo paso que en el caso de estudio 1.

Paso 5: Verificar y mapear las Actividades-tareas desarrolladas en el proyecto cumplen con las actividades-tareas establecidas en MoProSoft.

Paso 6: Realizar el mapeo de las causas de los defectos identificados con las tareas del proceso desarrollados en el proyecto.

Paso 7: Paso 6 del caso de estudio 1.

Paso 8: Paso 7 del caso de estudio 1.

IV.3.4.2 Herramientas utilizadas

Las herramientas utilizadas para este experimento fueron:

Métricas de calidad de IEC/ISO 9126: para poder evaluar las características de calidad del producto: Funcionalidad y Usabilidad, se utilizaron las métricas del estándar ISO 9126 que evalúan estas características. La descripción de estas métricas se encuentra en el Apéndice B.

MoProSoft: Se utilizó la definición de las Actividades-Tareas que establece el proceso de Desarrollo y Mantenimiento de Software de la categoría de Operación de MoProSoft, para identificar las tareas realizadas en el proyecto. Ver apéndice F

Plantilla de FMEA: La plantilla contiene la siguiente información (el detalle de esta plantilla, véase en el apéndice C):

- Nombre del producto
- Modelo de fallo (defecto)
- Efectos de fallo (consecuencia)
- Grado de Severidad: peores consecuencias
- Causa del fallo

- Grado de ocurrencia

Microsoft Excel: utilizado para llevar a cabo las evaluaciones del producto y del proceso, así como para llevar un control de cada métrica aplicable.

IV.3.4.3 Información recolectada y tratamiento de los datos

La información recolectada y el tratamiento de los datos fueron en base a la metodología propuesta para el cumplimiento de los objetivos específicos planteados en este experimento.

Datos para la evaluación de la calidad del producto. (Paso 2 y paso 3)

Los datos obtenidos para la evaluación de las características de calidad del producto se muestran en la Tabla 36.

Para la característica de funcionalidad se evaluó una sub-característica, "adecuación", y tres atributos los cuales son: Completud, correctud y correctud en información de interfaces. Los datos para aplicar las métricas fueron obtenidos en base a los artefactos de casos de uso y de diseño.

Para la característica de Usabilidad se evaluó una sub-característica "Fácil comprensión" y el atributo "comprensión". Los datos para poder aplicar la métrica fueron obtenidos en base al producto de software final y a los resultados obtenidos en las pruebas de usuario.

La definición de las métricas utilizadas para la evaluación de los atributos de calidad de cada una de las características seleccionadas se puede visualizar en la Tabla 36 (caso de estudio 1).

Tabla 36: Datos de la evaluación de calidad del producto de software, caso estudio 3.

Característica	Funcionalidad	
Sub-característica	Adecuación	
Atributo	Métrica	Datos de entrada
Completud	Porcentaje de completud.	NRNI =3, NRF =27
Correctud	Porcentaje de correctud.	NRII=1, NRFI =24
Correctud en información de interfaces	Porcentaje de correctud en información de interfaces.	NINC=2, NII =10

Característica	Usabilidad	
Sub-característica	Fácil comprensión	
Atributo	Métrica	Datos de entrada
Comprensión	Porcentaje de comprensión de interfaces de usuario.	NRNI=22, NRF =32

Causas de Fallo del producto. (Paso 4)

En base a la evaluación de la calidad (funcionalidad y usabilidad) del producto de software se pudo obtener los defectos del producto de software, utilizando FMEA.

Para el análisis del defecto-causa se utilizó la plantilla de FMEA, la información del análisis fue:

- Identificación de los defectos del producto en base a la evaluación de la calidad del producto.
- Identificación de las causas de los defectos más significativas.

En la Tabla 37 se encuentra el mapeo de los defectos-causas identificados.

Tabla 37: Mapeo del defecto-causa adjudicado al proceso, caso de estudio 3.

Categoría de defecto	Causas adjudicadas al proceso
A. No completar la implementación de las funcionalidades requeridas.	No se realizó correctamente la Revisión del Plan de Desarrollo Actual con los miembros del equipo de trabajo.
	No Verificación y validación de la Especificación de Requisitos.
B. Funcionalidad incorrecta de los requerimientos funcionales implementados.	No identificación y descripción de todos los componentes de software con base en el Análisis y Diseño.
	Falta de especificaciones en el requerimientos funcional.
	Falta de pruebas unitarias y con datos reales.
	No se realizó la Verificación y validación de Análisis y Diseño.
C. Datos no presentados en la interfaz de usuario según las especificaciones.	No Elaboración del prototipo en etapa temprana.
	Falta de definición de requerimientos de usabilidad en etapa temprana.
	Falta de pruebas unitarias y con datos reales.
	No Verificación y validación de la Especificación de Requisitos de Usabilidad.
D. No comprensión de la Interfaz de Usuario.	No Verificación y validación de la Especificación de Requisitos de Usabilidad.
	No se realizó completamente las Pruebas con el

	usuario final en etapas tempranas.
	Falta de definición de requerimientos de usabilidad en etapa temprana.
	No se realizó correctamente la Revisión del Plan de Desarrollo Actual con los miembros del equipo de trabajo.

Mapeo de las causas de defecto con las tareas del proceso (Paso 5 y 6)

Al tener identificado las causas de los defectos encontrados en el producto de software, se realizó el mapeo de las causas de los defectos del producto de software con las tareas del proceso de software desarrolladas en este proyecto. La definición de cada tarea mapeada se tomo de las definidas en el proceso de DMS de MoProSoft, como se muestra en la Tabla 38.

Tabla 38: Mapeo de las causas adjudicadas al proceso con las tareas del proceso de software, caso de estudio 3.

Causas adjudicadas al proceso	Tarea
No se realizó correctamente la Revisión del Plan de Desarrollo Actual con los miembros del equipo de trabajo.	T1. Revisar el plan de Desarrollo Actual con los miembros del equipo de trabajo.
No Verificación y validación de la Especificación de Requisitos.	T2.Verificar y validar la especificación de requisitos.
No identificación y descripción de todos los componentes de software con base en el Análisis y Diseño.	T3. Identificar y describir de todos los componentes de software con base en el Análisis y Diseño.
Falta de especificaciones en el requerimientos funcional.	T4.Elaborar el documento de los requerimientos funcionales.
Falta de pruebas unitarias y con datos reales.	T5. Realizar pruebas unitarias con datos reales.
No se realizó la Verificación y validación de Análisis y Diseño.	T6. Verificar y validar el Análisis y diseño.
No Elaboración del prototipo en etapa temprana.	T7. Elaborar el prototipo: Interfaz de usuario.
Falta de definición de requerimientos de usabilidad en etapa temprana.	T8. Definir los requerimientos de usabilidad en etapa temprana.
Falta de pruebas unitarias y con datos reales.	T9. Realizar pruebas unitarias y con datos reales.
No Verificación y validación de la Especificación de Requisitos de Usabilidad.	T10. Verificar y validar la especificación de requisitos de Usabilidad.
No Verificación y validación de la Especificación de Requisitos de Usabilidad.	T11. Verificar y validar la especificación de requisitos de Usabilidad.
No se realizó completamente las Pruebas con el usuario final en etapas tempranas.	T12. Realizar las pruebas con el usuario final en etapas tempranas.
Falta de definición de requerimientos de	T13. Definir requerimientos de usabilidad a

usabilidad en etapa temprana.	detalle.
No se realizó correctamente la Revisión del Plan de Desarrollo Actual con los miembros del equipo de trabajo.	T14. Revisar el plan de Desarrollo Actual con los miembros del equipo de trabajo.

Clasificación de las tareas mapeadas. (Paso 7)

Al tener identificadas las tareas del proceso de software que generaron los defectos del producto de software, se realizó la clasificación de las tareas del proceso en base a al tipo de actividad y al propósito. En la Tabla 39 se muestra la clasificación de las tareas del proceso de software con respecto a su tipo y propósito: (Paso 7):

Tabla 39: Clasificación de las tareas del proceso por tipo y propósito, caso de estudio 3.

Tarea	Tipo	Artefacto	Propósito
T1. Revisar el plan de Desarrollo Actual con los miembros del equipo de trabajo.	Apoyo	No Tangible	Planeación y Seguimiento.
T2.Verificar y validar la especificación de requisitos.	Apoyo	No tangible	Verificación y Validación.
T3. Identificar y describir de todos los componentes de software con base en el Análisis y Diseño.	Base	Tangible	Realización Fase de Análisis.
T4.Elaborar el documento de los requerimientos funcionales.	Base	Tangible	Realización Fase de Requerimientos.
T5. Realizar pruebas unitarias con datos reales.	Base	Tangible	Realización Fase de Pruebas.
T6. Verificar y validar el Análisis y diseño.	Apoyo	No tangible	Verificación y Validación.
T7. Elaborar el prototipo: Interfaz de usuario.	Base	Tangible	Realización Fase de Análisis.
T8. Definir los requerimientos de usabilidad en etapa temprana.	Base	Tangible	Realización Fase de Requerimientos.
T9. Realizar pruebas unitarias y con datos reales.	Base	Tangible	Realización Fase de Pruebas.
T10. Verificar y validar la especificación de requisitos de Usabilidad.	Apoyo	No tangible	Verificación y Validación.
T11. Verificar y validar la especificación de requisitos de Usabilidad.	Apoyo	No tangible	Verificación y Validación.
T12. Realizar las pruebas con el usuario final en etapas tempranas.	Base	Tangible	Realización Fase de Pruebas.
T13. Definir requerimientos de usabilidad a detalle.	Base	Tangible	Realización Fase de Requerimientos.
T14. Revisar el plan de Desarrollo Actual con los miembros del equipo de trabajo.	Apoyo	No tangible	Planeación y Seguimiento.

IV.3.4.4 Resultados obtenidos

Evaluación del producto de software

Para este caso de estudio se evaluaron los siguientes atributos de calidad del producto: (1) Completud, Correctud y Correctud en información de interfaces, que pertenecen a la característica de Funcionalidad. (2) Para la característica de Usabilidad, se evaluó el atributo de calidad Comprensión. (La descripción de cada atributo se puede ver en la sección de Resultados obtenidos del caso de estudio 1).

Los resultados obtenidos para la evaluación de las características de calidad del producto se muestran en la Tabla 40.

Tabla 40: Resultado de la evaluación de la calidad del producto para el caso de estudio 3.

Característica	Sub-Característica	Atributo	Métrica	Resultado Métrica
Funcionalidad	Adecuación	Completud	Porcentaje de completud.	88.89%
Funcionalidad	Adecuación	Correctud	Porcentaje de correctud.	95.83%
Funcionalidad	Adecuación	Correctud en información de interfaces	Porcentaje de correctud en información de interfaces.	80%
Usabilidad	Fácil de comprender	Comprensión	Porcentaje de comprensión de interfaces de usuario.	68.75%

En esta tabla se puede ver los resultados de la evaluación de la calidad del producto de software, con respecto a las siguiente atributos: "**Completud**" estuvo 88.89% completo en base a los requerimientos de funcionalidad establecidos; "**Correctud**" estuvo en un 95.83% correcto según los requerimientos de funcionalidad implementados; "**Correctud en información de interfaces**" estuvo en un 80% correcto en la información de interfaces establecidas en las especificaciones de los requerimientos funcionales; y en el atributo de "comprensión" estuvo en un 68.75% de comprensión de interfaces de usuario en base en las funciones de interfaces de usuario definidas.

Clasificación de los tipos de tareas del proceso de software

Al tener identificadas las tareas causantes de los defectos del producto de software, se realizó la clasificación de las tareas en base a su tipo y a su propósito, esto con el fin de validar los objetivos específicos (ver en el apartado Objetivos del experimento) planteados en este experimento.

Los resultados fueron los siguientes:

- Clasificación de las tareas por tipo de tarea y artefacto generado (véase en la Tabla 41 y Figura 22).

Tabla 41: Tabla de resultados de la clasificación de tipo de tareas.

Tipo de Tarea	Artefacto	Frecuencia
Base	Tangible	8
Apoyo	No Tangible	6
Total		**14**

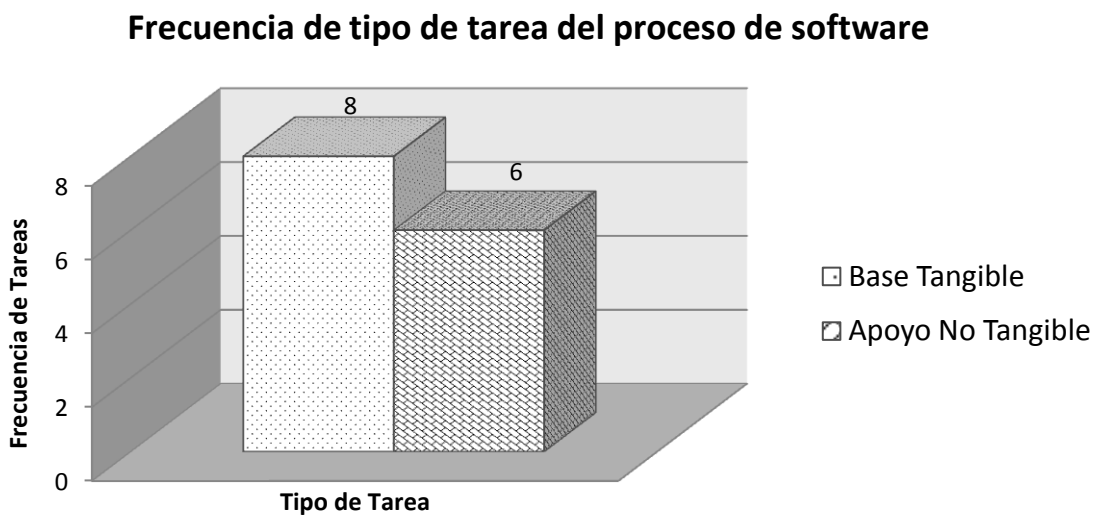

Figura 22: Gráfica del resultado de la clasificación de las tareas del proceso, caso de estudio 3.

En la gráfica de la Figura 22 se observa como las tareas de tipo Base que generan artefactos tangibles son las que más contribuyen a la generación de defectos en comparación con las tareas de apoyo que generan artefactos no tangibles.

- Clasificación de las tareas por propósito (véase en la Tabla 42 y Figura 23):

Tabla 42: Resultado de la clasificación de las tareas por propósito.

Propósito de Actividad	Frecuencia
Planeación y Seguimiento	2
Realización Fase de Requerimientos	3
Realización Fase de Análisis	1
Realización fase de Pruebas	3
Verificación y validación	4
Realización de la Fase de Diseño	1
Total	14

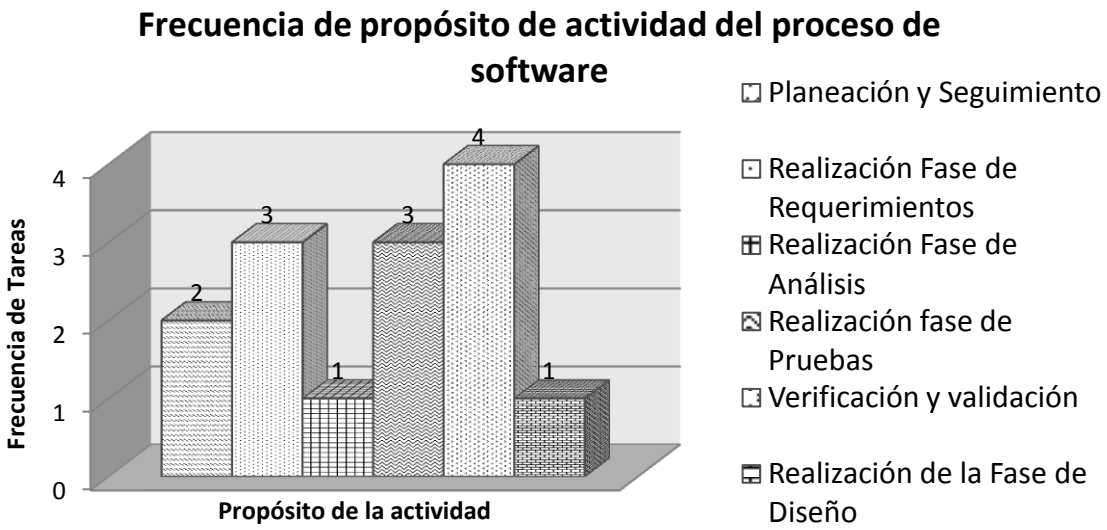

Figura 23: Gráfica del resultado de la clasificación de propósito de actividades del proceso de software, para el caso de estudio 3.

En la gráfica de la Figura 23 se observa, que en este caso particular las tareas que más impactaron fueron las que tienen como propósito la verificación y validación, realización de la fase de requerimientos y la de realización de la fase de pruebas.

IV.3.5 Caso de estudio 4

IV.3.5.1 Metodología utilizada

El procedimiento recomendado para llevar a cabo este experimento y cumplir con los objetivos planteados es el siguiente:

Paso 1: Obtener la información del proyecto: se recolectaron los artefactos generados a lo largo del proceso de desarrollo, como son calendario, plan de proyecto, reportes de seguimiento, documento de casos de uso, diagramas de base de datos, plan de pruebas y comentarios de los participantes del proyecto.

Paso 2: Evaluar el producto de software final, aplicando métricas de calidad al producto final. La característica de calidad seleccionada para evaluar la calidad es: Funcionalidad. Esta característica fue determinada por el dueño de la empresa, debido que en ese momento quería saber indicadores de funcionalidad.

Del paso 3 al paso 8: Son los mismos que en el caso de estudio 3.

IV.3.5.2 Herramientas utilizadas

Las herramientas utilizadas para este experimento fueron:

Métricas de calidad de IEC/ISO 9126: Para poder evaluar la características de calidad del producto: Funcionalidad, se utilizaron las métricas del estándar ISO 9126 que evalúan estas características. (La descripción de estas métricas se encuentra en el Apéndice B.)

MoProSoft: Se utilizó la definición de las Actividades-Tareas que establece los procesos de Administración de Proyectos Específicos (APE) y de Desarrollo y Mantenimiento de Software (DMS), que pertenecen a la categoría de Operación de MoProSoft, para identificar las tareas realizadas en el proyecto. (Ver apéndice F.)

Plantilla de FMEA: La plantilla contiene la siguiente información (el detalle de esta plantilla, véase en el Apéndice C):

- Nombre del producto.
- Modelo de fallo (defecto).
- Efectos de fallo (consecuencia).
- Grado de Severidad: Peores consecuencias.
- Causa del fallo.
- Grado de ocurrencia.

Microsoft Excel: Utilizado para llevar a cabo las evaluaciones del producto y del proceso, así como para llevar un control de cada métrica aplicable.

IV.3.5.3 Información recolectada y tratamiento de los datos

La información recolectada y el tratamiento de los datos fueron en base a la metodología propuesta para el cumplimiento de los objetivos específicos planteados en este experimento.

Datos para la evaluación de la calidad del producto. (Paso 2 y paso 3)

Los datos obtenidos para la evaluación de la característica de calidad del producto se muestran en la Tabla 43. Para la característica de funcionalidad se evaluó una sub-característica, "**Adecuación**", y dos atributos los cuales son: **Completud** y **Correctud**. Los datos para aplicar las métricas fueron obtenidos en base a los artefactos de casos de uso y de diseño.

La definición de las métricas utilizadas para la evaluación de los atributos de calidad de la característica Funcionalidad (Se puede visualizar en apéndice B).

Tabla 43: Datos de la evaluación de calidad del producto de software, caso estudio 4.

Característica	Funcionalidad	
Sub-característica	Adecuación	
Atributo	Métrica	Datos de entrada
Completud	Porcentaje de completud	NRNI =8, NRF =104
Correctud	Porcentaje de correctud	NRII=5, NRFI =96

Causas de Fallo del producto. (Paso 4)

En base a la evaluación de la calidad (funcionalidad) del producto de software se pudo obtener los defectos del producto de software, utilizando FMEA.

Para el análisis del defecto-causa se utilizó la plantilla de FMEA, la información del análisis fue: (1) Identificación de los defectos del producto en base a la evaluación de la calidad del producto. (2) Identificación de las causas de los defectos más significativas.
En la Tabla 44 se encuentra el mapeo de los defectos-causas identificados.

Mapeo de las causas de fallo con las tareas del proceso (Paso 5 y 6)

Al tener identificado las causas de los defectos encontrados en el producto de software, se realizó el mapeo de las causas de los defectos del producto de software con las tareas del proceso de software desarrolladas en este proyecto (véase Tabla 45). La definición de cada tarea mapeada se tomo de las definidas en el proceso de APE y DMS de MoProSoft.

Tabla 44: Mapeo del defecto-causa adjudicado al proceso, caso de estudio 4.

Categoría defecto	Causa
A. No completar la implementación de las funcionalidades requeridas.	No Elaborar el Plan de Adquisiciones y Capacitación, definiendo las características y el calendario en cuanto a recursos humanos, materiales, equipo y herramientas, incluyendo la capacitación requerida para que el equipo de trabajo pueda desempeñar el proyecto.
	No se elaboro el documento de los riesgos que pueden afectar el proyecto, que contemple riesgos relacionados con el equipo de trabajo incluyendo al Cliente y a los usuarios, riesgos con la tecnología o la metodología, riesgos con la organización del proyecto (costo, tiempo, alcance y recursos) o riesgos externos al proyecto.
	La distribución de las tareas a los miembros del equipo de trabajo según su rol no se realizó en base al Plan de Desarrollo actual.
B. Funcionalidad incorrecta de los requerimientos funcionales implementados.	No se revisó el Registro de Rastreo de los requerimientos del usuario a través del ciclo.
	No se realizó reuniones de revisión con el equipo de trabajo y con el Cliente.
	No verificar la Especificación de Requerimientos.
	No Corregir los defectos encontrados en la Especificación de Requerimientos con base en el Reporte de Verificación.

	No realizó documento de Diseño: diagramas de secuencia, clases.
	Se realizan las verificaciones de los documentos de: casos de uso y prototipo, pero no se genera un reporte de verificación.
	No hay retroalimentación con el cliente.
	No elaborar Plan de Pruebas de Integración.
	No verificar el Plan de Pruebas de Integración
	No corregir los defectos encontrados en el Plan de Pruebas de Integración con base en el Reporte de Verificación y obtener la aprobación de las correcciones.
	No se realizó el Reporte de Verificación del documento de Análisis y Diseno y Registro de Rastreo
	No aprobación de los entregables.

Tabla 45: Mapeo de las causas adjudicadas al proceso con las tareas del proceso de software, caso de estudio 4.

Causa adjudicadas al proceso	Categoría	Tarea
No Elaborar el Plan de Adquisiciones y Capacitación, definiendo las características y el calendario en cuanto a recursos humanos, materiales, equipo y herramientas, incluyendo la capacitación requerida para que el equipo de trabajo pueda desempeñar el proyecto.	APE	T1. Elaborar el Plan de Adquisiciones y Capacitación, definiendo las características y el calendario en cuanto a recursos humanos, materiales, equipo y herramientas, incluyendo la capacitación requerida para que el equipo de trabajo pueda desempeñar el proyecto.
No se elaboró el documento de los riesgos que pueden afectar el proyecto, que contemple riesgos relacionados con el equipo de trabajo incluyendo al Cliente y a los usuarios, riesgos con la tecnología o la metodología, riesgos con la organización del proyecto (costo, tiempo, alcance y recursos) o riesgos externos al proyecto.	APE	T2. Elaborar el documento de los riesgos que pueden afectar el proyecto, que contemple riesgos relacionados con el equipo de trabajo incluyendo al Cliente y a los usuarios, riesgos con la tecnología o la metodología, riesgos con la organización del proyecto (costo, tiempo, alcance y recursos) o riesgos externos al proyecto.
La distribución de las tareas a los miembros del equipo de trabajo según su rol no se realizó en base al Plan de Desarrollo actual.	DMS	T3. Distribución de las tareas a los miembros del equipo de trabajo según su rol, de acuerdo al Plan de Desarrollo actual.
No se revisó el Registro de Rastreo de los requerimientos del usuario a través del ciclo.	APE	T4. Revisar el Registro de Rastreo de los requerimientos del usuario a través del ciclo.
No se realizó reuniones de revisión con el equipo de trabajo y con el Cliente.	APE	T5. Realizar reuniones de revisión con el equipo de trabajo y con el Cliente.
No Verificar la Especificación de Requerimientos.	DMS	T6. Verificar la Especificación de Requerimientos.
No corregir los defectos encontrados en la Especificación de Requerimientos con base en el Reporte de Verificación.	DMS	T7. Corregir los defectos encontrados en la Especificación de Requerimientos con base en el Reporte de Verificación y obtener la aprobación de las correcciones.
No realizó documento de Diseño: diagramas de secuencia, clases.	DMS	T8. No realización de Diseño: diagramas de secuencia, clases.
Se realizan las verificaciones de los documentos de: casos de uso y prototipo, pero no se genera un reporte de verificación.	DMS	T9. Realizar el reporte de verificación.

No hay retroalimentación con el cliente.	DMS	T10. Realizar mecanismos de retroalimentación con el cliente.
No elaborar Plan de Pruebas de Integración.	DMS	T11. Elaborar el Plan de Pruebas de Integración.
No verificar el Plan de Pruebas de Integración.	DMS	T12. Verificar el Plan de Pruebas de Integración.
No corregir los defectos encontrados en el Plan de Pruebas de Integración con base en el Reporte de Verificación y obtener la aprobación de las correcciones.	DMS	T13. Corregir los defectos encontrados en el Plan de Pruebas de Integración.
No se realizó el Reporte de Verificación del documento de Análisis y Diseno y Registro de Rastreo.	DMS	T14. Realizar el Reporte de verificación del documento de Análisis y Diseno y Registro de Rastreo.
No aprobación de los entregables.	DMS	T15. Aprobación de los entregables por el cliente.

Clasificación de las tareas mapeadas. (Paso 7)

Al tener identificadas las tareas del proceso de software que generaron los defectos del producto de software, se realizó la clasificación de las tareas del proceso en base a al tipo de actividad y al propósito. En la Tabla 46 se muestra la clasificación de las tareas del proceso de software con respecto a su tipo y propósito: (Paso 7):

Tabla 46: Clasificación de las tareas del proceso por tipo y propósito, caso de estudio 4.

	Tarea	Tipo	Artefacto	Propósito
APE	T1. Elaborar el Plan de Adquisiciones y Capacitación, definiendo las características y el calendario en cuanto a recursos humanos, materiales, equipo y herramientas, incluyendo la capacitación requerida para que el equipo de trabajo pueda desempeñar el proyecto.	Base	Tangible	Planeación y Seguimiento.
APE	T2. Elaborar el documento de los riesgos que pueden afectar el proyecto, que contemple riesgos relacionados con el equipo de trabajo incluyendo al Cliente y a los usuarios, riesgos con la tecnología o la metodología, riesgos con la organización del proyecto (costo, tiempo, alcance y recursos) o riesgos externos al proyecto.	Base	Tangible	Planeación y Seguimiento.
DMS	T3. Distribución de las tareas a los miembros del equipo de trabajo según su rol, de acuerdo al Plan de Desarrollo actual.	Apoyo	No tangible	Planeación y Seguimiento.
APE	T4. Revisar el Registro de Rastreo de los requerimientos del usuario a través del ciclo.	Apoyo	No tangible	Realización Fase de Requerimientos.
APE	T5. Realizar reuniones de revisión con el equipo de trabajo y con el Cliente.	Apoyo	No tangible	Verificación y Validación.
DMS	T6. Verificar la Especificación de Requerimientos.	Apoyo	No tangible	Realización Fase de Requerimientos.
DMS	T7. Corregir los defectos encontrados en la	Base	Tangible	Verificación y Validación.

	Especificación de Requerimientos con base en el Reporte de Verificación y obtener la aprobación de las correcciones.			
DMS	T8. No realización de Diseño: diagramas de secuencia, clases.	Base	Tangible	Realización Fase de Diseño.
DMS	T9. Realizar el reporte de verificación	Base	Tangible	Verificación y Validación.
DMS	T10. Realizar mecanismos de retroalimentación con el cliente.	Apoyo	No tangible	Verificación y Validación.
DMS	T11. Elaborar el Plan de Pruebas de Integración.	Base	Tangible	Realización Fase de Pruebas.
DMS	T12. Verificar el Plan de Pruebas de Integración.	Apoyo	No tangible	Verificación y Validación.
DMS	T13. Corregir los defectos encontrados en el Plan de Pruebas de Integración	Base	Tangible	Realización Fase de Pruebas.
DMS	T14. Realizar el Reporte de verificación del documento de Análisis y Diseno y Registro de Rastreo	Base	Tangible	Verificación y Validación.
DMS	T15. Aprobación de los entregables por el cliente.	Base	Tangible	Verificación y Validación.

IV.3.5.4 Resultados obtenidos

Evaluación del producto de software

Para este caso de estudio, se evaluaron los siguientes atributos de calidad del producto: (1) **completud y correctud**. (La descripción de cada atributo se puede ver en la sección de Resultados obtenidos del caso de estudio 1).

Los resultados obtenidos de la evaluación de la característica de calidad del producto se muestran en la Tabla 47.

Tabla 47: Resultado de la evaluación de la calidad del producto para el caso de estudio 4.

Característica	Sub-Característica	Atributo	Métrica	Resultado Métrica
Funcionalidad	Adecuación	Completud	Porcentaje de completud	92.30%
Funcionalidad	Adecuación	Correctud	Porcentaje de correctud	94.79%

En esta Tabla se puede ver los resultados de la evaluación de la calidad del producto de software, con respecto a las siguiente atributos: "**completud**" estuvo 92.30% completo en base a los requerimientos de funcionalidad establecidos; "**correctud**" estuvo en un 94.79% correcto según los requerimientos de funcionalidad implementados.

Clasificación de los tipos de tareas del proceso de software

Al tener identificadas las tareas causantes de los defectos del producto de software, se realizó la clasificación de las tareas en base a su tipo y a su propósito, esto con el fin de validar los objetivos específicos (ver en el apartado Objetivos del experimento) planteados en este experimento.

Los resultados fueron los siguientes:

- Clasificación de las tareas por tipo de tarea y artefacto generado (véase en la Tabla 48 y Figura 24).

Tabla 48: Tabla de resultados de la clasificación de tipo de tareas.

Tipo de Tarea	Artefacto	Frecuencia
Base	Tangible	9
Apoyo	No Tangible	6
Total		**15**

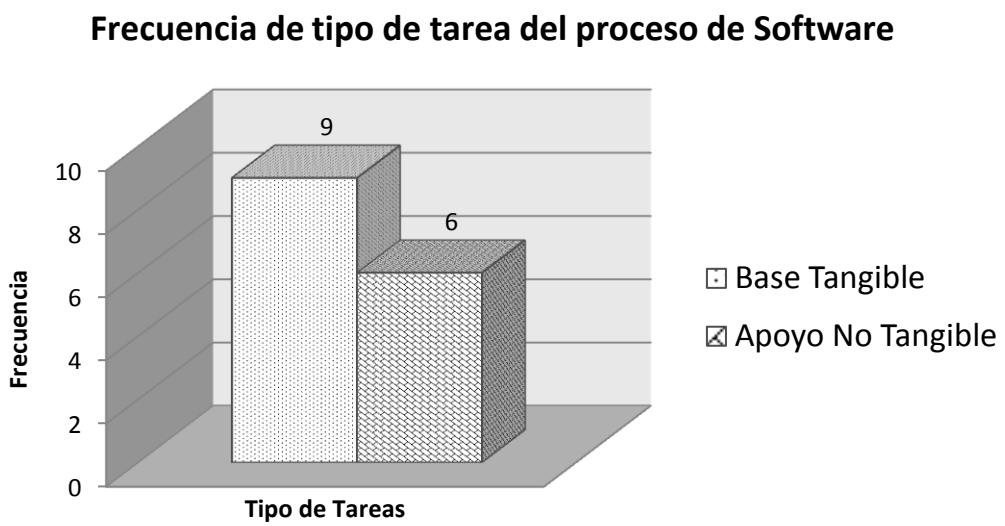

Figura 24: Gráfica del resultado de la clasificación de tipo de tarea del proceso, para el caso de estudio 4.

En la gráfica de la Figura 24 se muestra como las tareas de tipo Base que generan artefactos tangibles son las que más contribuyen a la generación de defectos en comparación con las tareas de apoyo que generan artefactos no tangibles.

- Clasificación de las tareas por propósito (véase en la Tabla 49 y Figura 25).

Tabla 49: Resultado de la clasificación de las tareas por propósito.

Propósito de Actividad	Frecuencia
Planeación y Seguimiento	3
Realización Fase de Requerimientos	2
Realización fase de Pruebas	2
Verificación y validación	7
Realización de la Fase de Diseño	1
Total	15

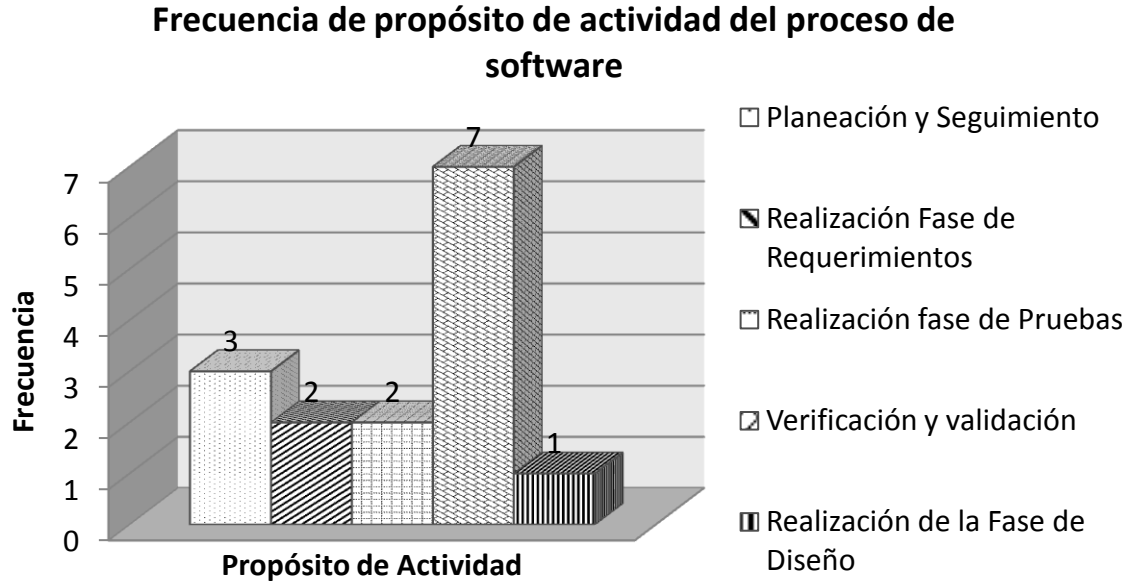

Figura 25: Gráfica del resultado de la clasificación de propósito de actividades del proceso de software, para el caso de estudio 4.

En la gráfica de la Figura 25 se observa que en este caso particular las tareas que más impactaron fueron las que tienen como propósito la verificación y validación y de Planeación y Seguimiento.

IV.4 Experimento 2: Contribución de los artefactos generados en el proceso de software en la calidad del producto de software.

SPEM (Ruiz 2008) establece que un producto de trabajo es producido o modificado medinte tareas, por lo que es el elemento que se desea obtener después de finalizar una tarea o actividad.

En base a esto, al ejecutarse una actividad de cualquier tipo en el proceso de software, se genera un producto o artefacto de software, el cual puede ser de dos tipos (SPEM 2):

- Tangible: Es de naturaleza tangible (modelo, documento, código, archivos, etc.) y que puede estar formados por otros artefactos más simples.
- Resultado: Un producto de trabajo de naturaleza intangible o que no está formalmente definido.

La propuesta del Metamodelo (ver capítulo III) describe la composición estructural del proceso y del producto de software, en donde se puede ver que cualquier tarea del proceso genera/produce un artefacto. Por consiguiente, al tener identificada el tipo de tarea del proceso de software así como el tipo de artefacto que se genera, se puede conocer cuáles tareas y artefactos impactan la calidad del producto.

IV.4.1 Objetivos del experimento

Este experimento tuvo como objetivo principal comprobar las hipótesis: $H_{(0,4)}$ y $H_{(0,5)}$ (ver a detalle en el Capítulo I) pretendiendo lograr el siguiente objetivo:

- Mostrar que los artefactos tangibles de tipo contribuidor directo tienen un mayor peso en la calidad del producto final, en comparación con los artefactos tangibles de tipo contribuidor indirecto.

IV.4.2 Caso estudio 1

IV.4.2.1 Metodología propuesta

El procedimiento recomendado para llevar a cabo este experimento y cumplir con el objetivo planteado fue:

Paso 1: Obtener la información del proyecto: Se recolectó los artefactos generados a lo largo del proceso de desarrollo, como son el documento de requerimientos, documento de casos de uso, documento de diseño, documentos de administración del proyecto. Así mismo, se realizó una entrevista a cada uno de los integrantes del equipo, con el objetivo de obtener sus experiencias a lo largo del desarrollo del proyecto de software.
Paso 2: Evaluar el producto de software final, aplicando las métricas de calidad al producto final. Las características de calidad seleccionada para evaluar la calidad son: Funcionalidad y Usabilidad. Estas características fueron determinadas por el equipo de desarrollo y por el cliente final debido al tipo de información que se maneja y al usuario que lo va a utilizar. La versión del producto es la que se le entregó al cliente según la fecha acordada.
Paso 3: Obtener el resultado de la calidad del producto, según las métricas aplicables. (Las métricas de calidad utilizadas se pueden ver en el Apéndice B)
Paso 4: Identificar las posibles causas de los defectos identificados del producto, utilizando FMEA. (La descripción del método se puede ver en el Apéndice C)
Paso 5: Realizar el mapeo de las causas de los defectos identificados con las tareas del proceso desarrollados en el proyecto.
Paso 6: Clasificar las tareas mapeadas en:

- Tareas base que generaron artefactos tangibles de tipo contribuidor directo.
- Tareas base que generaron artefactos tangibles de tipo contribuidor indirecto.

IV.4.2.2 Herramientas utilizadas

Estas herramientas son las mismas que se utilizaron para este caso de estudio en el experimento 1.

IV.4.2.3 Información recolectada y tratamiento de los datos

Los datos recolectados para los pasos del 1 al paso 5 de la metodología propuesta para este experimento son los mismos que los del experimento 1, por lo que solo se presentaran los datos recolectados para el paso 6 de la metodología propuesta.

Clasificación de las tareas mapeadas.

En la Tabla 50 se muestra la lista de clasificación por tipo de artefactos tangibles que producen las tareas base del proceso de software. Los tipos de artefactos tangibles son: tangible contribuidor directo y tangibles contribuidor indirecto (Paso 6 de la metodología).

Tabla 50: Clasificación de los artefactos por su tipo que corresponden a las tareas base del proceso, caso de estudio 1.

ID	Tarea	Clasificación	Propósito	Artefacto Tangible
A1	Especificar a detalle los requerimientos funcionales.	Base	Realización Fase de Requerimientos.	Contribuidor Directo.
A2	Especificar a detalle los requerimientos de interfaz de usuario y navegabilidad.	Base	Realización Fase de Requerimientos.	Contribuidor Directo.
A3	Definición del Plan de capacitación.	Base	Planeación y Seguimiento.	Contribuidor Indirecto.
A4	Analizar los casos de uso generales y sus detalles.	Base	Realización Fase de Análisis.	Contribuidor Directo.
A5	Especificación de las validaciones y excepciones del caso de uso.	Base	Realización Fase de Análisis.	Contribuidor Directo.
A7	Estimar el tamaño y esfuerzo por caso de uso.	Base	Planeación y Seguimiento.	Contribuidor Directo.
A8	Dar seguimiento al calendario: Actualizar el calendario, en base a la bitácora o reporte de seguimiento semanal.	Base	Planeación y Seguimiento.	Contribuidor Indirecto.
A9	Especificación y definición de	Base	Realización de la Fase de	Contribuidor

	los componentes de la interfaz de usuario y navegabilidad.		Diseño.	Directo.
A10	Elaborar el plan de pruebas por cada caso de uso.	Base	Realización de la Fase de Pruebas.	Contribuidor Directo.
A11	Definición del diseño por caso de uso definido.	Base	Realización de la Fase de Diseño.	Contribuidor Directo.
A12	Aplicación de las pruebas: usuario, funcionalidad, datos, interfaz de usuario.	Base	Realización de la Fase de Pruebas.	Contribuidor Directo.
A13	Aplicación del plan de pruebas de integración.	Base	Realización de la Fase de Pruebas.	Contribuidor Directo.

IV.4.2.4 Resultados obtenidos

Evaluación del producto de software

Los resultados de la evaluación de la calidad del producto de software final para este caso de estudio. (Se puede ver en la sección de resultados obtenidos del experimento1).

Identificación de artefactos tangibles generados por tareas de tipo base.

Al tener identificadas las tareas causantes de los defectos del producto de software, se realizó la clasificación de las tareas en base a su tipo y se identificó el tipo de artefacto que genera en este caso particular, esto con el fin de validar el objetivo específico (ver en el apartado Objetivos del experimento) planteado en este experimento.

Tabla 51: Tabla de resultados de la clasificación de tipo de artefacto.

Tipo de artefacto	Frecuencia
Tangible- Contribuidor directo	10
Tangible- Contribuidor indirecto	2
Total	12

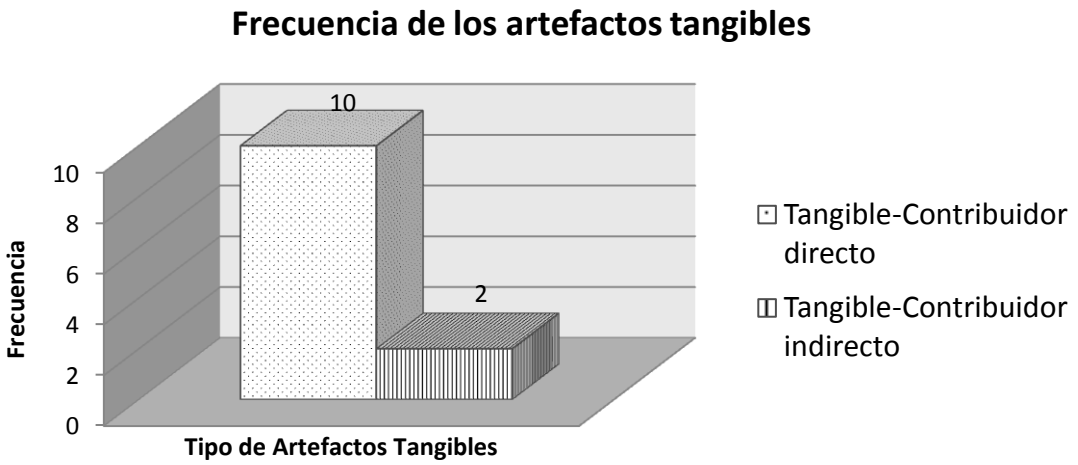

Figura 26: Gráfica del resultado de la clasificación de tipo de artefacto de las tareas base del proceso, para el caso de estudio 1.

En la gráfica de la Figura 26 se observa que en este caso particular los artefactos tangibles de tipo contribuidor directo tienen mayor impacto en la calidad del producto de software en comparación que los artefactos tangibles de contribuidor indirecto.

IV.4.3 Caso de estudio 2

IV.4.3.1 Metodología propuesta

El procedimiento recomendado para llevar a cabo este experimento y cumplir con el objetivo planteado fue:

Paso 1: Obtener la información de cada Sprint. Se recolectó los artefactos generados en el Sprint, como son los Sprint Backlog, Sprint Meeting (Reporte de junta), Revisión del Sprint, Retrospectiva del Sprint.

Paso 2: Evaluar el producto de software de cada Iteración o Sprint, aplicando las métricas de calidad al producto de software desarrollado. Las características de calidad seleccionada para evaluar la calidad fueron: Funcionalidad y Usabilidad. Estas características fueron

determinadas por el equipo de desarrollo, debido al tipo de información que se maneja y al usuario que lo va a utilizar.

Paso 3: Obtener el resultado de la calidad del producto, según las métricas aplicables. (Las métricas de calidad utilizadas se pueden ver en el apéndice B)

Paso 4: Identificar las posibles causas de los defectos del producto de software desarrollado en el Sprint o Iteración, en base al análisis de retrospectiva que realizaba el equipo de desarrollo. El análisis de retrospectiva de cada Sprint se realiza en la etapa final de cada una de ellas, por lo que facilitó la identificación de las causas de los defectos. (La descripción del Formato de Retrospectiva véase en el apéndice E)

Paso 5: Realizar el mapeo de las causas de los defectos identificados con las tareas del proceso desarrollados en el proyecto.

Paso 6: Clasificar las tareas mapeadas en:

- Tareas base que generaron artefactos tangibles de tipo contribuidor directo
- Tareas base que generaron artefactos tangibles de tipo contribuidor indirecto.

IV.4.3.2 Herramientas utilizadas

Estas herramientas son las mismas que se utilizaron para este caso de estudio en el experimento 1.

IV.4.3.3 Información recolectada y tratamiento de los datos

Los datos recolectados para los pasos del 1 al paso 5 de la metodología propuesta para el caso de estudio 2 son los mismos, por lo que solo se presentaran los datos recolectados para el paso 6 de la metodología propuesta.

En las siguientes Tablas 52-57 se muestran las listas de Clasificación de las tareas Base del proceso de software que generan artefactos de software tangible de contribuidor directo y artefactos de software tangibles de contribuidor indirecto. Cada tabla corresponde a un Sprint desarrollado (Paso 6 de la metodología).

Tabla 52: Clasificación de los artefactos por su tipo que corresponden a las tareas base del proceso del Sprint 1, caso de estudio 2.

Actividad	Tipo Actividad	Tipo Artefacto
A2. Definir herramientas para la recolección de los requerimientos con el cliente	Base	Contribuidor directo
A3. Elaborar el plan de trabajo	Base	Contribuidor directo
A5. Realizar el plan de capacitación	Base	Contribuidor indirecto
A6. Actualizar bitácoras de actividades	Base	Contribuidor indirecto
A7. Realizar plan de Pruebas unitarias	Base	Contribuidor directo
A8. Realizar plan de pruebas de integración	Base	Contribuidor directo
A9. Ejecutar pruebas unitarias	Base	Contribuidor directo
A10. Ejecutar pruebas de integración	Base	Contribuidor directo
A11. Realizar minutas en la juntas de trabajo	Base	Contribuidor indirecto
A12. Realizar calendario de trabajo	Base	Contribuidor directo

Tabla 53: Clasificación de los artefactos por su tipo que corresponden a las tareas base del proceso del Sprint 2, caso de estudio 2.

Actividad	Tipo Actividad	Tipo Artefacto
A2. Definir herramientas para la recolección de los requerimientos con el cliente.	Base	Contribuidor indirecto
A3. Realizar minutas en la juntas de trabajo.	Base	Contribuidor indirecto
A6.Crear plan de administración de la configuración.	Base	Contribuidor indirecto
A7. Definir herramienta para la admón. De la configuración.	Base	Contribuidor indirecto
A9. Definir Arquitectura de software.	Base	Contribuidor directo
A10. Definir herramienta a utilizar para el desarrollo.	Base	Contribuidor directo
A11. Realizar el plan de adquisición y capacitación.	Base	Contribuidor indirecto
A12. Realizar plan de Pruebas unitarias.	Base	Contribuidor directo
A13. Realizar plan de pruebas de integración.	Base	Contribuidor directo
A14. Realizar reportes de pruebas unitarias.	Base	Contribuidor directo
A15. Realizar reportes de pruebas de integración.	Base	Contribuidor directo
A16. Realizar plan de Pruebas integración.	Base	Contribuidor directo
A17. Definir los requerimientos.	Base	Contribuidor directo
A18. Realizar Análisis de requerimientos.	Base	Contribuidor directo

Tabla 54: Clasificación de los artefactos por su tipo que corresponden a las tareas base del proceso del Sprint 3 caso de estudio 2.

Actividad	Tipo Actividad	Tipo Artefacto
A1. Realizar el plan de admón. De configuración.	Base	Contribuidor indirecto
A3. Realizar plan de pruebas de integración.	Base	Contribuidor directo
A4. Realizar plan de riesgos.	Base	Contribuidor indirecto
A5. Realizar plan de capacitación.	Base	Contribuidor indirecto
A7. Realizar bitácora de las actividades.	Base	Contribuidor indirecto
A8. Realizar el plan de pruebas unitarias.	Base	Contribuidor directo
A9. Realizar el reporte de pruebas unitarias.	Base	Contribuidor directo
A10. Realizar el plan de pruebas de integración.	Base	Contribuidor directo
A11. Realizar el reporte de pruebas de integración.	Base	Contribuidor directo

Tabla 55: Clasificación de los artefactos por su tipo que corresponden a las tareas base del proceso del Sprint 4, caso de estudio 2.

Actividad	Tipo Actividad	Tipo Artefacto
A1. Realizar el plan de admón. De configuración.	Base	Contribuidor indirecto
A3. Realizar el plan de pruebas de integración.	Base	Contribuidor directo
A4. Ejecutar el plan de pruebas de integración.	Base	Contribuidor directo
A5. Realizar el reporte de pruebas de integración.	Base	Contribuidor directo
A6. Elaborar el plan de capacitación.	Base	Contribuidor indirecto
A7. Realizar el plan de infraestructura y capacitación.	Base	Contribuidor indirecto
A8. Realizar el diseño de clases.	Base	Contribuidor directo
A9. Realizar el diseño de diagramas Entidad-Relación.	Base	Contribuidor directo
A10. Realizar los diagramas de flujos de datos.	Base	Contribuidor directo
A11. Generación de minutas en la juntas de trabajo.	Base	Contribuidor indirecto
A12. Realizar el diseño de prototipo.	Base	Contribuidor directo

Tabla 56: Clasificación de los artefactos por su tipo que corresponden a las tareas base del proceso del Sprint 5, caso de estudio 2.

Actividad	Tipo Actividad	Tipo Artefacto
A1. Realizar el plan de admón. De configuración.	Base	Contribuidor indirecto
A3. Realizar el plan de pruebas de integración.	Base	Contribuidor directo
A4. Ejecutar el plan de pruebas de integración.	Base	Contribuidor directo
A5. Realizar el reporte de pruebas de integración.	Base	Contribuidor directo
A6. Realizar el plan de trabajo.	Base	Contribuidor directo
A7. Realizar el plan de capacitación.	Base	Contribuidor indirecto
A8. Realizar minutas en la juntas de trabajo.	Base	Contribuidor indirecto
A10. Realizar bitácora de las actividades.	Base	Contribuidor indirecto

Tabla 57: Clasificación de los artefactos por su tipo que corresponden a las tareas base del proceso del Sprint 6, caso de estudio 2.

Actividad	Tipo Actividad	Tipo Artefacto
A2. Realizar bitácora de las actividades	Base	Contribuidor indirecto
A3. Realizar el plan de pruebas de integración	Base	Contribuidor directo
A4. Ejecutar el plan de pruebas de integración	Base	Contribuidor directo
A5. Realizar el reporte de pruebas de integración	Base	Contribuidor directo
A6. Realizar retroalimentación	Base	Contribuidor indirecto

IV.4.3.4 Resultados obtenidos

Evaluación del producto de software

Los resultados de la evaluación de la calidad del producto de software final para este caso de estudio 2 se pueden ver en la sección de resultados obtenidos del experimento 1.

Identificación de artefactos tangibles generados por tareas de tipo base.

Al tener identificadas las tareas causantes de los defectos del producto de software, se realizó la clasificación de las tareas en base a su tipo y se identificó el tipo de artefacto que genera en este caso particular, esto con el fin de validar el objetivo específico (ver en el apartado Objetivos del experimento) planteado en este experimento.

Tabla 58: Tabla de resultados de la clasificación de tipo de artefacto.

Sprints	Artefactos tangibles contribuidor directo	Artefactos tangible contribuidor indirecto
SPRINT 1	6	4
SPRINT 2	9	5
SPRINT 3	5	4
SPRINT 4	7	4
SPRINT 5	4	4
SPRINT 6	3	2

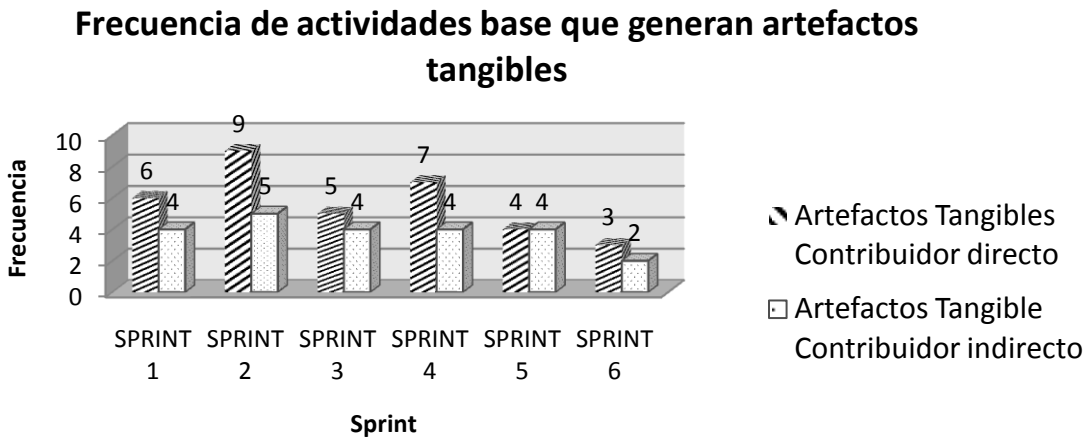

Figura 27: Gráfica del resultado de la clasificación de tipo de artefacto de las tareas base del proceso, para el caso de estudio 2.

En la gráfica de la Figura 27 se muestra, que en este caso de estudio, donde se evaluó por Sprint o iteración, los artefactos tangibles de contribuidor directo que son generados por las tareas base, representan mayor impacto en la calidad del producto de software en comparación de los artefactos de contribuidor indirecto.

IV.4.4 Caso estudio 3

IV.4.4.1 Metodología propuesta

El procedimiento recomendado para llevar a cabo este experimento y cumplir con el objetivo planteado fue:

Paso 1: Obtener la información del proyecto. Es la misma información del paso 1 del experimento 1.

Paso 2: Evaluar el producto de software final, aplicando las métricas de calidad al producto final. Las características de calidad seleccionada para evaluar la calidad son: Funcionalidad y Usabilidad.

Paso 3: Obtener el resultado de la calidad del producto, en base a las métricas aplicables. (La descripción de las métricas de calidad utilizadas se puede ver en el Apéndice B)

Paso 4: Identificar las posibles causas de los defectos identificados del producto, utilizando FMEA (ver apéndice C).

Paso 5: Verificar y mapear las Actividades-tareas desarrolladas en el proyecto cumplen con las actividades-tareas establecidas en MoProSoft.

Paso 6: Realizar el mapeo de las causas de los defectos identificados con las tareas del proceso desarrollados en el proyecto.

Paso 7: Clasificar las tareas mapeadas en:

- Tareas base que generaron artefactos tangibles de tipo contribuidor directo.
- Tareas base que generaron artefactos tangibles de tipo contribuidor indirecto.

IV.4.4.2 Herramientas utilizadas

Las herramientas utilizadas para este caso de estudio son las mismas que se emplearon en el experimento 1.

IV.4.4.3 Información recolectada y tratamiento de los datos

Evaluación del producto de software

Los resultados de la evaluación de la calidad del producto de software final para este caso de estudio 3 se pueden ver en la sección de resultados obtenidos del experimento 1.

Identificación de artefactos tangibles generados por tareas de tipo base

Al tener identificadas las tareas causantes de los defectos del producto de software, se realizó la clasificación de las tareas en base a su tipo y se identificó el tipo de artefacto que genera en este caso particular, esto con el fin de validar el objetivo específico (ver en el apartado Objetivos del experimento) planteado en este experimento.

En la Tabla 59 se muestra la lista de Clasificación de las tareas Base del proceso de software que generan artefactos de software tangible contribuidor directos y artefactos de software tangibles contribuidor indirecto (Paso 7 de la metodología)

Tabla 59: Clasificación de los artefactos por su tipo que corresponden a las tareas base del proceso, caso de estudio 3.

Tarea	Tipo	Tipo de Artefacto
T3. Identificación y descripción de todos los componentes de software con base en el Análisis y Diseño.	Base	Contribuidor Directo
T4.Elaboracion del documento de los requerimientos funcionales.	Base	Contribuidor Directo
T5. Realizar pruebas unitarias con datos reales.	Base	Contribuidor Directo
T7. Elaboración del prototipo: Interfaz de usuario.	Base	Contribuidor Directo
T8. Definición de los requerimientos de usabilidad en etapa temprana.	Base	Contribuidor Directo
T9. Realizar pruebas unitarias y con datos reales.	Base	Contribuidor Directo
T12. Realizar las pruebas con el usuario final en etapas tempranas.	Base	Contribuidor Directo
T13. Definir requerimientos de usabilidad a detalle.	Base	Contribuidor Directo

IV.4.4.4 Resultados obtenidos

Evaluación del producto de software

Los resultados de la evaluación de la calidad del producto de software final para este caso de estudio 3 se pueden ver en la sección de resultados obtenidos del experimento 1.

Identificación de artefactos tangibles generados por tareas de tipo base.

Al tener identificadas las tareas causantes de los defectos del producto de software, se realizó la clasificación de las tareas en base a su tipo y se identifico el tipo de artefacto que genera en este caso particular, esto con el fin de validar el objetivo específico (ver en el apartado Objetivos del experimento) planteado en este experimento.

Tabla 60. Tabla de resultados de la clasificación de tipo de artefacto.

Tipo de Artefacto	Frecuencia
Tangible-Contribuidor Directo	8
Tangible-Contribuidor Indirecto	0

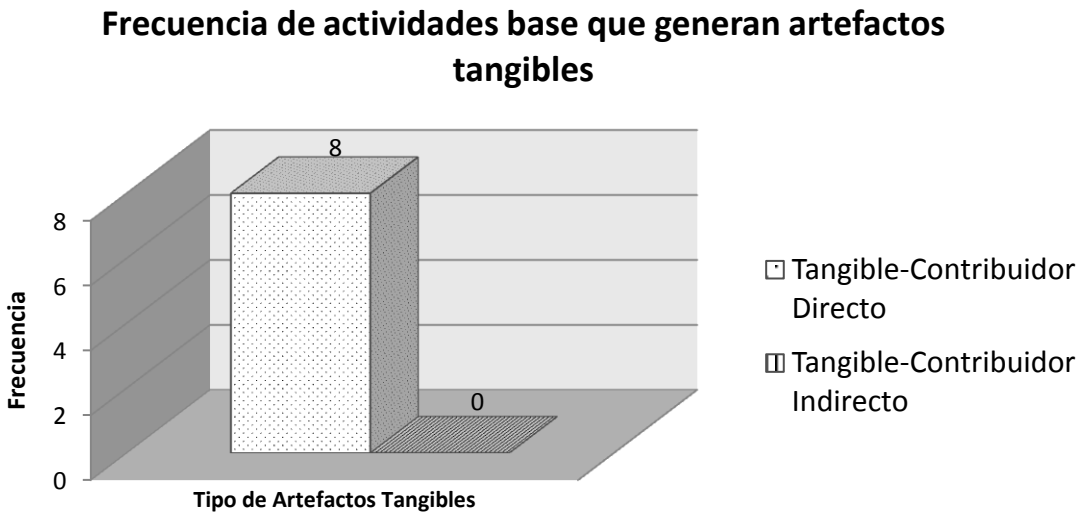

Figura 28: Gráfica del resultado de la clasificación de tipo de artefacto de las tareas base del proceso, para el caso de estudio 3.

En la gráfica de la Figura 28 se observa que en este caso particular los artefactos tangibles de tipo contribuidor directo tienen mayor impacto en la calidad del producto de software en comparación que los artefactos tangibles de tipo contribuidor indirecto.

IV.4.5 Caso estudio 4

IV.4.5.1 Metodología propuesta

El procedimiento recomendado para llevar a cabo este experimento y cumplir con el objetivo planteado es:

Paso 1: Obtener la información del proyecto. Es la misma información del paso 1 del experimento 1.

Paso 2: Evaluar el producto de software final, aplicando las métricas de calidad al producto final. Las características de calidad seleccionada para evaluar la calidad es: Funcionalidad.

Paso 3: Obtener el resultado de la calidad del producto, en base a las métricas aplicables. (La descripción de las métricas de calidad utilizadas se puede ver en el Apéndice B).

Paso 4: Identificar las posibles causas de los defectos identificados del producto, utilizando FMEA (ver apéndice C).

Paso 5: Verificar y mapear las Actividades-tareas desarrolladas en el proyecto cumplen con las actividades-tareas establecidas en MoProSoft.

Paso 6: Realizar el mapeo de las causas de los defectos identificados con las tareas del proceso desarrollados en el proyecto.

Paso 7: Clasificar las tareas mapeadas en:

- Tareas base que generaron artefactos tangibles de tipo contribuidor directo.
- Tareas base que generaron artefactos tangibles de tipo contribuidor indirecto.

IV.4.5.2 Herramientas utilizadas

Las herramientas utilizadas para este caso de estudio son las mismas que se emplearon en el experimento 1.

IV.4.5.3 Información recolectada y tratamiento de los datos

Evaluación del producto de software

Los resultados de la evaluación de la calidad del producto de software final para este caso de estudio 4 se pueden ver en la sección de resultados obtenidos del experimento 1.

Identificación de artefactos tangibles generados por tareas de tipo base.

Al tener identificadas las tareas causantes de los defectos del producto de software, se realizó la clasificación de las tareas en base a su tipo y se identificó el tipo de artefacto que genera en este caso particular, esto con el fin de validar el objetivo específico (ver en el apartado Objetivos del experimento) planteado en este experimento.

En la Tabla se muestra la lista de Clasificación de las tareas Base del proceso de software que generan artefactos de software tangible contribuidor directo y artefactos de software tangibles contribuidor indirecto (Paso 7 de la metodología).

Evaluación del producto de software

Los resultados de la evaluación de la calidad del producto de software final para este caso de estudio 3 se pueden ver en la sección de resultados obtenidos del experimento 1.

Identificación de artefactos tangibles generados por tareas de tipo base.

Al tener identificadas las tareas causantes de los defectos del producto de software, se realizó la clasificación de las tareas en base a su tipo y se identificó el tipo de artefacto que genera en este caso particular, esto con el fin de validar el objetivo específico (ver en el apartado Objetivos del experimento) planteado en este experimento.

En la Tabla 61 se muestra la lista de Clasificación de las tareas Base del proceso de software que generan artefactos de software tangible contribuidor directos y artefactos de software tangibles contribuidor indirecto (Paso 7 de la metodología).

Tabla 61: Clasificación de los artefactos por su tipo que corresponden a las tareas base del proceso, caso de estudio 4.

	Actividad	Tipo	Tipo Artefacto
APE	T1. Elaborar el Plan de Adquisiciones y Capacitación, definiendo las características y el calendario en cuanto a recursos humanos, materiales, equipo y herramientas, incluyendo la capacitación requerida para que el equipo de trabajo pueda desempeñar el proyecto.	Base	Contribuidor Indirecto
APE	T2. Elaborar el documento de los riesgos que pueden afectar el proyecto, que contemple riesgos relacionados con el equipo de trabajo incluyendo al Cliente y a los usuarios, riesgos con la tecnología o la metodología, riesgos con la organización del proyecto (costo, tiempo, alcance y recursos) o riesgos externos al proyecto.	Base	Contribuidor Directo
DMS	T7. Corregir los defectos encontrados en la Especificación de Requerimientos con base en el Reporte de Verificación y obtener la aprobación de las correcciones.	Base	Contribuidor Directo
DMS	T8. No realización de Diseño: diagramas de secuencia, clases.	Base	Contribuidor Directo

DMS	T9. Realizar el reporte de verificación	Base	Contribuidor Indirecto
DMS	T11. Elaborar el Plan de Pruebas de Integración.	Base	Contribuidor Directo
DMS	T13. Corregir los defectos encontrados en el Plan de Pruebas de Integración	Base	Contribuidor Directo
DMS	T14. Realizar el Reporte de verificación del documento de Análisis y Diseno y Registro de Rastreo	Base	Contribuidor Indirecto
DMS	T15. Aprobación de los entregables por el cliente.	Base	Contribuidor Indirecto

IV.4.5.4 Resultados obtenidos

Evaluación del producto de software

Los resultados de la evaluación de la calidad del producto de software final para este caso de estudio 4 se pueden ver en la sección de resultados obtenidos del experimento 1.

Identificación de artefactos tangibles generados por tareas de tipo base.

Al tener identificadas las tareas causantes de los defectos del producto de software, se realizó la clasificación de las tareas en base a su tipo y se identifico el tipo de artefacto que genera en este caso particular, esto con el fin de validar el objetivo específico (ver en el apartado Objetivos del experimento) planteado en este experimento.

Tabla 62: Tabla de resultados de la clasificación de tipo de artefacto.

Tipo de Artefacto	Frecuencia
Tangible-Contribuidor Directo	5
Tangible-Contribuidor Indirecto	4
Total	9

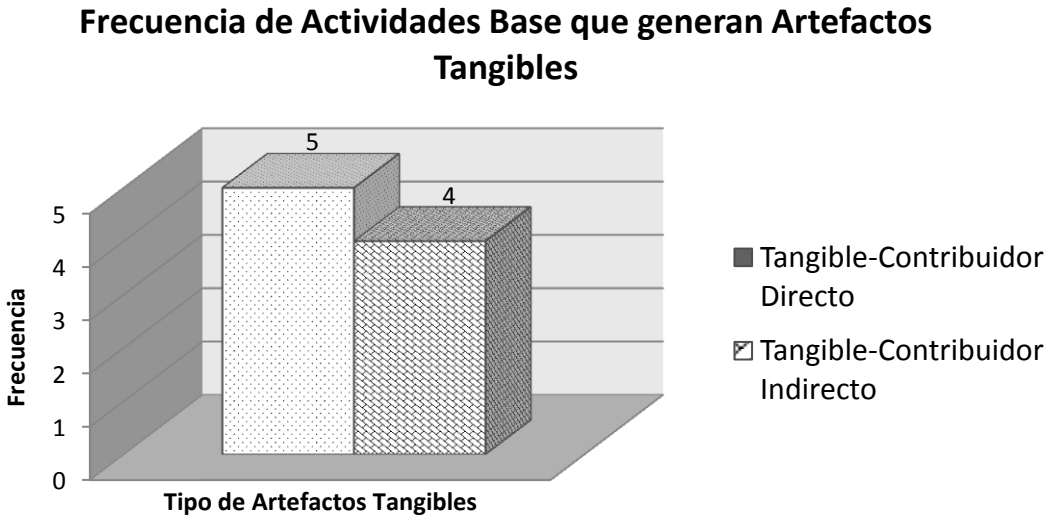

Figura 29: Gráfica del resultado de la clasificación de tipo de artefacto de las tareas base del proceso, para el caso de estudio 4.

En la gráfica de la Figura 29 se observa, que en este caso particular los artefactos tangibles de tipo contribuidor directo tienen mayor impacto en la calidad del producto de software en comparación que los artefactos tangibles de tipo contribuidor indirecto.

IV.5 Experimento 3: Impacto de la ejecución correcta de las tareas del proceso de software en la calidad del producto de software.

Según (Dr. Mario Piattini Velthuis 2008-2009), el proceso de medición implica las mediciones de las actividades relacionadas con el producto software siendo algunos de sus atributos típicos el esfuerzo, el coste y los defectos encontrados.

En base a la propuesta del Metamodelo (ver capítulo III), se observa que la tarea del proceso de software y el producto de software están estrechamente relacionados, debido a que la tarea genera un artefacto y un conjunto de artefactos componen al producto de software. En base a esto, para conocer de donde proviene la calidad del producto de software se debe de medir cuantitativamente la calidad de la tarea del proceso de software.

IV.5.1 Objetivos del experimento

Este experimento tuvo como objetivo principal comprobar la hipótesis: $H_{(0,6)}$ (ver a detalle en el Capítulo I) pretendiendo lograr el siguiente objetivo:

- Mostrar que al no ejecutar correctamente las tareas del proceso definido afecta directamente a la calidad del producto generado.

IV.5.2 Caso estudio 1

IV.5.2.1 Metodología propuesta

El procedimiento recomendado para llevar a cabo este experimento y cumplir con los objetivos planteados fue el siguiente:

Del paso 1 al paso 5: Son los mismos que en el experimento 1.

Paso 6: Al tener identificadas las tareas del proceso de software que causaron los defectos al producto de software, se clasificó en realizada o no realizada, según el proceso establecido.

Paso 7: Evaluar las tareas realizadas, aplicando métricas para evaluar la correctud de dichas tareas.

Paso 8: Obtener el resultado de las métricas aplicables.

IV.5.2.2 Herramientas utilizadas

Estas herramientas son las mismas que se utilizaron para este caso de estudio en el experimento 1.

IV.5.2.3 Información recolectada y tratamiento de los datos

Los datos recolectados para los pasos del 1 al paso 5 de la metodología propuesta para este experimento son los mismos que los del experimento 1, por lo que solo se presentaran los datos recolectados para el paso 6 y 7 de la metodología propuesta.

Clasificación de las tareas identificadas.

En la Tabla 63 se muestra la lista de clasificación de las tareas realizadas y no realizadas del proceso de software. (Paso 6 de la metodología).

Tabla 63: Clasificación de las tareas en realizadas o no realizadas, en base a las tareas identificadas como causante de los defectos del producto de software, caso de estudio 1.

Tarea	Realizada/No Realizada
T1. Especificar a detalle los requerimientos funcionales.	No realizada.
T2. Especificar a detalle los requerimientos de interfaz de usuario y navegabilidad. (Requerimientos de Usabilidad).	No realizada.
T3. Definir el Plan de capacitación.	Realizada.
T4. Analizar los casos de uso generales y sus detalles.	Realizada.
T5. Especificar las validaciones y excepciones de los casos de uso.	Realizada.
T6. Validar los casos de uso definidos por el cliente/usuario.	Realizada.
T7. Estimar el tamaño y esfuerzo por caso de uso.	Realizada.
T8. Dar seguimiento al calendario: Actualizar el calendario, en base a la bitácora o reporte de seguimiento semanal.	Realizada.
T9. Especificar y definir los componentes de la interfaz de usuario y navegabilidad.	No realizada.
T10. Elaborar el plan de pruebas por cada caso de uso.	Realizada.
T11. Definir el diseño por caso de uso definido.	No realizada.
T12. Aplicar las pruebas: usuario, funcionalidad, datos, interfaz de usuario.	Realizada.
T13. Aplicar el plan de pruebas de integración.	No realizada.

Datos para la evaluación de la correctud de las tareas realizadas del proceso de software (Paso 7).

En la Tabla 64 se muestra los datos de las tareas realizadas para evaluar la correctud de cada una de ellas.

En la tarea “T3. Definir el Plan de capacitación” se evaluó la desviación del esfuerzo total en tiempo de las tareas completadas según el plan de capacitación establecido. De esta manera, se puede saber si el tiempo estimado para la realización de cada tarea planeada resulto correcto con respecto a lo realmente realizado (esfuerzo realizado).

La tarea “T4. Analizar los casos de uso generales y sus detalles” se evaluó la eficiencia en la identificación de los casos de uso en la fase de análisis. De esta manera, se puede saber si la identificación de los casos de uso en la fase de análisis fue correcta con respecto a los implementados en el producto final. También se evaluó el grado de completud de los casos de uso en base a los elementos establecidos en el estándar de especificación de casos de uso. Por lo tanto, se puede saber si el detalle de la especificación de los casos de uso está completo.

En la tarea “T5. Especificar las validaciones y excepciones de los casos de uso” se evaluó el grado de completud de Casos de uso que definieron validaciones y excepciones en la especificación de los casos de uso, en base a la regla que establece que por lo menos debe de haber una validación y una excepción por caso de uso. De esta manera, se puede saber el grado de completud y detalle de la especificación de los casos de uso.

La tarea “T6. Validar los casos de uso definidos por el cliente/usuario” se evaluó la Eficiencia de las revisiones de validación de casos de uso, si el esfuerzo (tiempo) invertido es bastante (mucho tiempo invertido según el equipo de trabajo) y se encuentran pocos defectos, significa que no es eficiente las revisiones de validación. De esta manera, se puede saber si la tarea de validación de los casos de uso resulto eficiente con respecto al esfuerzo investido.

En la tarea “T7. Estimar el tamaño y esfuerzo por caso de uso” se evaluó la desviación del esfuerzo total (minutos) de las tarea completada para la implementación de cada caso de uso según el plan de trabajo establecido. Si el esfuerzo es positivo (tiempo

positivo) significa que hubo un retraso. Si el esfuerzo es negativo (tiempo negativo) significa que hubo ganancia de esfuerzo. De esta manera, se puede saber si el tiempo estimado para la realización de cada tarea planeada resulto correcto con respecto a lo realmente realizado (esfuerzo realizado).

En la tarea "T8. Dar seguimiento al calendario: Actualizar el calendario, en base a la bitácora o reporte de seguimiento semanal" se evaluó grado de completud de la realización de bitácoras, según el proceso establecido por integrante del equipo. De esta manera se puede saber si se puedo dar seguimiento al calendario correctamente en base a las bitácoras realizadas por todos los integrantes del equipo.

La tarea "T10. Elaborar el plan de pruebas por cada caso de uso" se evaluó el grado de completud de la definición de Pruebas de software. De esta manera, se puede saber si las pruebas planeadas estuvieron completas según las funcionalidades definidas por cada caso de uso especificado.

La tarea "T12. Aplicar las pruebas de funcionalidad" se evaluó la eficiencia en la detección de defectos en la aplicación las pruebas de funcionalidad. De esta manera, se puede saber si fue eficiente la detección de defectos en base al promedio de defectos establecidos en la aplicación de las pruebas de funcionalidad.

Tabla 64: Datos de la medición de las tareas identificadas como causante de los defectos del producto de software, caso estudio 1.

Tarea	T3. Definir del Plan de capacitación.			
Medida Base	Medida Derivada	Función de calculo	Criterios	Datos de entrada
TR= Tiempo Total real de las tarea completadas (min). TE=Tiempo total estimado de las tareas planeadas (min).	DEPC=Desviación del esfuerzo (tiempo) de cada tarea completada del plan de capacitación	DEPC=TR-TE	Si el esfuerzo es positivo (tiempo positivo) significa que hubo un retraso. Si el esfuerzo es negativo (tiempo negativo) quiere decir que hubo ganancia de esfuerzo.	TR= 2430 min, TE= 830min.
Tarea	T4. Analizar los casos de uso generales y sus detalles.			
Medida Base	Medida Derivada	Función de calculo		Datos de entrada
NCUID= Número de casos de uso identificados en la fase de análisis. NCUIM= Número de	PCUI=Porcentaje de eficiencia en la identificación de los casos de uso.	PCUI= (NCUID/ NCUIM)*100%		NCUID =185, NCUIM =188

casos de uso implementados.				
Medida Base	Medida Derivada	Función de calculo		Datos de entrada
PECU= promedio de elementos cumplidos en la especificación de los casos de uso. EECU= Número de elementos de la especificación de casos de uso según el estándar.	PCCU=Porcentaje de completud en la documentación de la especificación de casos de uso	PCCU = (PECU/EECU)*100%		PECU=8.6, EECU=12
Tarea	T5. Especificar las validaciones y excepciones de los casos de uso.			
Medida Base	Medida Derivada	Función de calculo	Criterios	Datos de entrada
NCUNV=Número de casos de uso que no definen validaciones y excepciones NCUID= Número de casos de uso identificados en la fase de análisis.	PCCUV=Porcentaje de completud de Casos de uso que definieron validaciones y excepciones	(PCCUV=1-NCUNV/ NCUID) *100%	Por lo menos debe de haber una validación y una excepción por caso de uso	NCUNV=71, NCUID =185
Tarea	T6. Validar los casos de uso definidos por el cliente/usuario.			
Medida Base	Medida Derivada	Función de calculo	Criterios	Datos de entrada
NDER= Número de defectos encontrados en revisiones de validaciones. PHE=Promedio de horas esfuerzo invertido en revisiones de validar los casos de uso.	ER=Eficiencia de las revisiones de validación de casos de uso.	ER= NDER / PHE	Si el esfuerzo (tiempo) invertido es bastante y se encuentran pocos defectos, significa que no es eficiente las revisiones de validación.	NDER = 0, PHE =8hrs.
Tarea	T7. Estimar el tamaño y esfuerzo por caso de uso.			
Medida Base	Medida Derivada	Función de calculo	Criterios	Datos de entrada
TRCUI= Tiempo Total real de las tarea completadas para la implementación de los casos de uso (min). TECUP=Tiempo total estimado de las tareas planeadas para la implementación de los casos de uso (min).	DEICU=Desviación de esfuerzo del total de las tareas completada para la implementación de cada caso de uso.	DEICU = TRCUI - TECUP	Si el esfuerzo es positivo (tiempo positivo) significa que hubo un retraso. Si el esfuerzo es negativo (tiempo negativo) quiere decir que hubo ganancia de esfuerzo.	TRCUI =3555 Min., TECUP =2630 Min.
Tarea	T8. Dar seguimiento al calendario: Actualizar el calendario, en base a la bitácora o reporte de seguimiento semanal.			
Medida Base	Medida Derivada	Función de calculo		Datos de entrada

NBR= Numero de bitácoras realizadas NBE= Numero de bitácoras establecidas en el proceso	PCRB=Porcentaje de completud en la realización de bitácoras, según el proceso establecido por integrante del equipo.		PCRB =(NBR/NBE)*100%	NBR=36, NBE=70
Tarea	T10. Elaborar el plan de pruebas por cada caso de uso.			
Medida Base	Medida Derivada		Función de calculo	Datos de entrada
NPP=Número de casos de pruebas diseñados y/o planeadas. NPR= Número de pruebas requeridos para obtener una cobertura adecuada.	PCP= Porcentaje en la completud de la definición de Pruebas de software		PCP=(NPP/NPR)*100%	NPP=112, NPR=187
Tarea	T12. Aplicar las pruebas de funcionalidad.			
Medida Base	Medida Derivada	Función de calculo	Criterios	Datos de entrada
NDEP= Número de defectos encontrados en pruebas. PHEP=Promedio de horas esfuerzo invertido en ejecutar las pruebas.	EP=Eficiencia en la ejecución de pruebas	EP= NDEP / PHEP	Si el esfuerzo (tiempo) invertido es bastante y se encuentran pocos defectos, significa que no es eficiente la ejecución de pruebas.	NDEP=0, PHEP=8 hrs.

IV.5.2.4 Resultados obtenidos

Evaluación del producto de software

Los resultados de la evaluación de la calidad del producto de software final para este caso de estudio 1 se pueden ver en la sección de resultados obtenidos del experimento 1.

Clasificación de las tareas mapeadas (realizadas y no realizadas)

En la Tabla 65 se puede ver el resultado de la frecuencia de la clasificación de las tareas identificadas que causan defectos en el producto de software. El resultado de la clasificación de las tareas fue: Tareas realizadas son 8 y las tareas no realizadas fueron 5.

Tabla 65: Resultado de la clasificación de las tareas identificadas en realizadas o no realizadas, caso de estudio 1.

Tareas	Frecuencia
Tareas Realizadas	8
Tareas No realizadas	5
Total	13

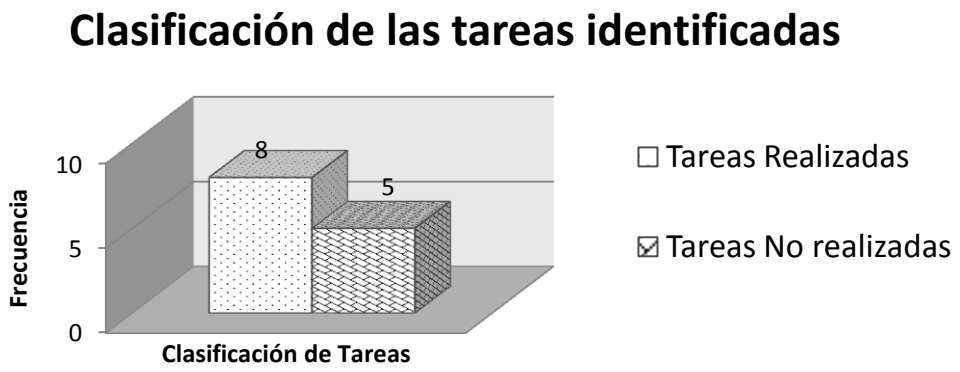

Figura 30: Gráfica del resultado de la clasificación de las tareas identificadas en realizadas o no realizadas, para el caso de estudio 1.

Las tareas T1, T2, T9, T11 y T13 no fueron realizadas por lo que no se pudo obtener información para poder ser evaluadas y saber el grado de correctud.

Evaluación de Correctud de las tareas realizadas del proceso de software

Los resultados obtenidos de la evaluación de correctud de las tareas realizadas del proceso de software, las cuales son las que están directamente relacionadas con la calidad del producto de software, se muestran en la Tabla 66.

Tabla 66: Resultado de la evaluación de las tareas realizadas causantes de los defectos del proceso, para el caso de estudio 1.

Tarea Evaluada	Métrica	Resultado	Observación.
T3. Definir el Plan de capacitación.	Desviación del esfuerzo (tiempo) total de las tarea completadas del plan de capacitación	600 min = 10 horas hombre de esfuerzo (Retraso)	Si el esfuerzo es positivo (tiempo positivo) significa que hubo un retraso. Si el esfuerzo es negativo (tiempo negativo) quiere decir que hubo ganancia de

			esfuerzo.
T4. Analizar los casos de uso generales y sus detalles.	Porcentaje de eficiencia en la identificación de los casos de uso.	98%	
	Porcentaje de completud en la documentación de la especificación de casos de uso	72%	
T5. Especificar las validaciones y excepciones de los casos de uso.	Porcentaje de completud de Casos de uso que definieron validaciones y excepciones.	62%	
T6. Validar los casos de uso definidos, por el cliente/usuario.	Eficiencia de las revisiones de validación de casos de uso por el cliente.	0 defectos/Hrs.	Si el esfuerzo (tiempo) invertido es bastante y se encuentran pocos defectos, significa que no es eficiente las revisiones de validación.
T7. Estimar el tamaño y esfuerzo por caso de uso.	Desviación de esfuerzo del total de las tareas completadas para la implementación de cada caso de uso.	925min= 15.41 horas hombre de esfuerzo (Retraso)	Si el esfuerzo es positivo (tiempo positivo) significa que hubo un retraso. Si el esfuerzo es negativo (tiempo negativo) quiere decir que hubo ganancia de esfuerzo.
T8. Dar seguimiento al calendario: Actualizar el calendario, en base a la bitácora o reporte de seguimiento semanal.	Porcentaje de completud en la realización de bitácoras, según el proceso establecido por integrante del equipo.	51%	
T10. Elaborar el plan de pruebas por cada caso de uso.	Porcentaje en la completud de la definición de Pruebas de software	59%	
T12. Aplicar las pruebas de funcionalidad	Eficiencia en la ejecución de pruebas	0 defectos/Hrs.	Si el esfuerzo (tiempo) invertido es bastante y se encuentran pocos defectos, significa que no es eficiente las revisiones de validación.

Para T3 se evaluó la desviación del esfuerzo total (tiempo) de las tareas completadas según el plan de capacitación, lo que resultó que hubo un retraso de tiempo de 10 horas hombre con respecto a lo planeado.

En T4 se evaluó la eficiencia en la identificación de los casos de uso, resultó con un 98% de eficiente en la identificación de los casos de uso en la fase de análisis con respecto a lo implementado en el producto final. Así mismo se evaluó el porcentaje de completud de

la documentación de la especificación de casos de uso, el cual resultó con un 72% completa con respecto al estándar de documentación que establece el proceso unificado Ágil.

Para T5 se evaluó el porcentaje de completud de casos de uso en la definición de validaciones y excepciones, el cual resultó con en 62% de completo con respecto a todos los casos de uso documentados, en base a la regla de que por lo menos debe de haber una validación y excepción definida por caso de uso.

En T6 se evaluó la eficiencia de las revisiones de validación de los casos de uso por el cliente, el cual resultó que en un promedio de 8 horas invertidas en la revisión de validación de los casos de uso no se encontraron defectos (0 defectos), por lo que se puede concluir que la tarea de validación de los casos de uso no fue eficiente.

Para T7 se evaluó la desviación del esfuerzo total de las tareas completadas para la implementación de los casos de uso con respecto a lo planeado, lo que resultó que hubo retraso de 15 horas con 41 minutos hombre trabajado extras.

En T8 se evaluó el porcentaje de completud en la realización de bitácoras, según el proceso establecido de todos los integrantes del equipo, lo que resultó con un 51% de completud de bitácoras realizadas según lo establecido en todo el proceso de software.

Para T10 se evaluó el porcentaje en la completud de la definición de Pruebas de software en base a los casos de uso implementados, lo que resultó con un 59% de completud de las pruebas planeadas con respecto a la cantidad de pruebas que debía de ser según la cantidad de casos de uso definidos.

En T12 se evaluó la eficiencia de la ejecución de las pruebas de funcionalidad por caso de uso definidos, el cual resultó que en un promedio de 8 horas invertidas en la ejecución de pruebas de funcionalidad no se encontraron defectos (0 defectos), por lo que se puede concluir que la tarea de ejecución de las pruebas de los casos de uso no fue eficiente debido a que se invirtió un tiempo considerado en le ejecución de las pruebas de funcionalidad y no se obtuvo ningún defecto cuando si existían.

IV.5.3 Caso estudio 2

IV.5.3.1 Metodología propuesta

El procedimiento recomendado para llevar a cabo este experimento y cumplir con los objetivos planteados es el siguiente:

Del paso 1 al paso 5, son los mismos que en el experimento 1.

Paso 6: Al tener identificadas las tareas del proceso de software que causaron los defectos al producto de software en cada Sprint, se clasificó en realizada o no realizada, según el proceso establecido.

Paso 7: Evaluar las tareas realizadas, aplicando métricas para evaluar la correctud de dichas tareas.

Paso 8: Obtener el resultado de las métricas aplicables.

IV.5.3.2 Herramientas utilizadas

Estas herramientas son las mismas que se utilizaron para este caso de estudio en el experimento 1.

IV.5.3.3 Información recolectada y tratamiento de los datos

Los datos recolectados para los pasos del 1 al paso 5 de la metodología propuesta para este experimento son los mismos que los del experimento 1, por lo que solo se presentaran los datos recolectados para el paso 6 y 7 de la metodología propuesta.

Clasificación de las tareas identificadas

En las Tablas 67-72 se muestra la lista de clasificación de las tareas realizadas y no realizadas por cada Sprint realizado. (Paso 6 de la metodología).

Tabla 67: Clasificación de las tareas en realizadas o no, en base a las tareas identificadas como causante de los defectos del producto de software del Sprint 1, caso de estudio 2.

Tarea	Realizado/No realizado
T1. Verificar un plan de comunicación con el cliente	No Realizado
T2. Definir herramientas para la recolección de los requerimientos con el cliente	No Realizado
T3. Elaborar el plan de trabajo	Realizado
T4. Dar seguimiento al plan de trabajo	No Realizado
T5. Realizar el plan de capacitación	No Realizado
T6. Actualizar bitácoras de actividades	No Realizado
T7. Realizar plan de Pruebas unitarias	Realizado
T8. Realizar plan de pruebas de integración	No Realizado
T9. Ejecutar pruebas unitarias	Realizado
T10. Ejecutar pruebas de integración	No Realizado
T11. Realizar minutas en la juntas de trabajo	No Realizado
T12. Realizar calendario de trabajo	Realizado
T13. Verificar y validar los planes de trabajo	No Realizado
T14. Verificar y validar los productos entregables.	No Realizado

Tabla 68: Clasificación de las tareas en realizadas o no, en base a las tareas identificadas como causante de los defectos del producto de software del Sprint 2, caso de estudio 2.

Tarea	Realizado/No Realizado
T1. Verificar el plan de comunicación con el cliente.	No Realizado
T2. Definir herramientas para la recolección de los requerimientos con el cliente.	No Realizado
T3. Realizar minutas en la juntas de trabajo.	Realizado
T4. Verificar y validar los planes de trabajo.	No Realizado
T5. Verificar y validar los productos entregables.	No Realizado
T6.Crear plan de administración de la configuración.	No Realizado
T7. Definir herramienta para la admón. De la configuración.	No Realizado
T8. Verificar y validar la admón. En la configuración.	No Realizado
T9. Definir Arquitectura de software.	No Realizado
T10. Definir herramienta a utilizar para el desarrollo.	No Realizado
T11. Realizar el plan de adquisición y capacitación.	No Realizado
T12. Realizar plan de Pruebas unitarias.	Realizado
T13. Realizar plan de pruebas de integración.	No Realizado
T14. Realizar reportes de pruebas unitarias.	No Realizado
T15. Realizar reportes de pruebas de integración.	No Realizado
T16. Realizar plan de Pruebas integración.	No Realizado
T17. Definir los requerimientos.	No Realizado
T18. Realizar Análisis de requerimientos.	No Realizado

Tabla 69: Clasificación de las tareas en realizadas o no realizadas, en base a las tareas identificadas como causante de los defectos del producto de software del Sprint 3, caso de estudio 2.

Tarea	Realizado/No Realizado
T1. Realizar el plan de admón. De configuración.	No Realizado
T2. Dar seguimiento del plan de admón. De configuración.	No Realizado
T3. Realizar plan de pruebas de integración.	No Realizado
T4. Realizar plan de riesgos.	No Realizado
T5. Realizar plan de capacitación.	No Realizado
T6. Dar seguimiento al plan de trabajo.	Realizado
T7. Realizar bitácora de las actividades.	No Realizado
T8. Realizar el plan de pruebas unitarias.	No Realizado
T9. Realizar el reporte de pruebas unitarias.	No Realizado
T10. Realizar el plan de pruebas de integración.	No Realizado
T11. Realizar el reporte de pruebas de integración.	No Realizado

Tabla 70: Clasificación de las tareas en realizadas o no realizadas, en base a las tareas identificadas como causante de los defectos del producto de software del Sprint 4, caso de estudio 2.

Tarea	Realizado/No realizado
T1. Realizar el plan de admón. De configuración.	No Realizado
T2. Dar seguimiento del plan de admón. De configuración.	No Realizado
T3. Realizar el plan de pruebas de integración.	No Realizado
T4. Ejecutar el plan de pruebas de integración.	No Realizado
T5. Realizar el reporte de pruebas de integración.	No Realizado
T6. Elaborar el plan de capacitación.	No Realizado
T7. Realizar el plan de infraestructura y capacitación.	No Realizado
T8. Realizar el diseño de clases.	No Realizado
T9. Realizar el diseño de diagramas Entidad-Relación.	No Realizado
T10. Realizar los diagramas de flujos de datos.	No Realizado
T11. Generación de minutas en la juntas de trabajo.	No Realizado
T12. Realizar el diseño de prototipo.	No Realizado

Tabla 71: Clasificación de las tareas en realizadas o no realizadas, en base a las tareas identificadas como causante de los defectos del producto de software del Sprint 5, caso de estudio 2.

Tarea	Realizado/No Realizado
T1. Realizar el plan de admón. De configuración.	No Realizado

T2. Dar seguimiento del plan de admón. De configuración.	No Realizado
T3. Realizar el plan de pruebas de integración.	No Realizado
T4. Ejecutar el plan de pruebas de integración.	No Realizado
T5. Realizar el reporte de pruebas de integración.	No Realizado
T6. Realizar el plan de trabajo.	Realizado
T7. Realizar el plan de capacitación.	No Realizado
T8. Realizar minutas en la juntas de trabajo.	Realizado
T9. Dar seguimiento del plan de trabajo.	Realizado
T10. Realizar bitácora de las actividades.	No Realizado

Tabla 72: Clasificación de las tareas en realizadas o no realizadas, en base a las tareas identificadas como causante de los defectos del producto de software del Sprint 6, caso de estudio 2.

Tarea	Realizado/ No realizado
T1. Dar seguimiento al plan de trabajo.	Realizado
T2. Realizar bitácora de las actividades.	No Realizado
T3. Realizar el plan de pruebas de integración.	No Realizado
T4. Ejecutar el plan de pruebas de integración.	No Realizado
T5. Realizar el reporte de pruebas de integración.	No Realizado
T6. Realizar retroalimentación.	No Realizado

Datos para la evaluación de la correctud de las tareas realizadas del proceso de software (Paso 7).

En la Tabla 73 se pueden ver los valores obtenidos de la medición de las tareas realizadas del proceso para el Sprint 1.

En la tarea "T1. Elaborar el plan de trabajo" se evaluó el porcentaje de completud de las reuniones de Scrum Meeting realizadas según las planeadas para el Sprint 1. Es estas reuniones el objetivo principal es planear las actividades del día y actualizar el estatus de las actividades realizadas.

La tarea "T7.Realizar plan de pruebas unitarias" se evaluó el porcentaje en completud de la definición del plan de pruebas. De esta manera, se puede saber si las pruebas planeadas estuvieron completas según las funcionalidades definidas en este Sprint.

La tarea "T9.Ejecutar pruebas unitarias" se evaluó la eficiencia en la ejecución de pruebas, en base a la cantidad de defectos detectados según el tiempo investido en la

ejecución de las pruebas. De esta manera, se puede saber si la tarea fue realizada eficientemente en este Sprint.

En la tarea "T12.Realizar calendario de trabajo" se evaluó la desviación del esfuerzo total (tiempo) de las tareas completadas según el plan de trabajo establecido. Si el esfuerzo es positivo (tiempo positivo) significa que hubo un retraso. Si el esfuerzo es negativo (tiempo negativo) significa que hubo ganancia de esfuerzo. De esta manera, se puede saber si el tiempo estimado para la realización de cada tarea planeada resultó correcto con respecto a lo realmente realizado (esfuerzo realizado).

De las tareas no realizadas no se obtuvieron información por lo que no se evaluó el grado de correctud de las tareas del proceso de software definido.

Tabla 73: Datos de la medición de las tareas identificadas como causante de los defectos del producto de software del Sprint 1, caso estudio 2.

Tarea	**T3. Elaborar el plan de trabajo**			
Medida Base	Medida Derivada	Función de cálculo		Datos de entrada
NSMR= Número de Scrum Meeting realizadas. NSMC=Número de Scrum Meeting planeadas	PCSM=Porcentaje de completud de Scrum Meeting realizadas.	PCSM1=(NSMR / NSMC)*100%		NSMR=0, NSMC=5,
Tarea	**T7. Realizar plan de Pruebas unitarias**			
Medida Base	Medida Derivada	Función de cálculo		Datos de entrada
NPP=Número de casos de pruebas diseñados y/o planeadas. NPR= Número de pruebas requeridos para obtener una cobertura adecuada.	PCP=Porcentaje en la completud de la definición de Pruebas de software	PCP=(NPP/NPR)*100%		NPP=3, NPR=20
Tarea	**T9. Ejecutar pruebas unitarias**			
Medida Base	Medida Derivada	Función de calculo	Criterios	Datos de entrada
NDEP= Número de defectos encontrados en pruebas. PHEP=Promedio de horas esfuerzo invertido en ejecutar las pruebas.	EP=Eficiencia en la ejecución de pruebas	EP= NDEP / PHEP	Si el esfuerzo (tiempo) invertido es bastante y se encuentran pocos defectos, significa que no es eficiente la ejecución de pruebas.	NDEP=3, PHEP=4 hrs.
Tarea	**T12. Realizar calendario de trabajo**			
Medida Base	Medida Derivada	Función de calculo	Criterios	Datos de entrada
TR= Tiempo Total real de las	DEPT=Desviación	DEPT1=TR-	Si el esfuerzo es	TR=27 hrs.,

tarea completadas (hrs). TES=Tiempo total estimado de las tareas planeadas (hrs).	del esfuerzo (tiempo) de cada tarea completada del plan de trabajo	TE	positivo (tiempo positivo) significa que hubo un retraso. Si el esfuerzo es negativo (tiempo negativo) quiere decir que hubo ganancia de esfuerzo	TE=24 hrs.

En la Tabla 74 se pueden ver los valores obtenidos de la medición de las tareas realizadas del proceso para el Sprint 2.

La tarea "T3.Realizar minutas en la juntas de trabajo" se evaluó el porcentaje de completud de minutas de trabajo según las reuniones de trabajo realizadas. De esta manera, se observó si las minutas de trabajo fueron realizadas según las reuniones realizadas.

En la tarea "T12.Realizar plan de Pruebas unitarias" se evaluó el porcentaje en la completud de la definición de Pruebas de software. De esta manera, se puede saber si las pruebas planeadas estuvieron completas según las funcionalidades definidas en este Sprint.

Tabla 74: Datos de la medición de las tareas identificadas como causante de los defectos del producto de software del Sprint 2, caso estudio 2.

Tarea	**T3. Realizar minutas en la juntas de trabajo.**		
Medida Base	Medida Derivada	Función de cálculo	Datos de entrada
NMR= Número de minutas realizadas. NRT=Número de reuniones de trabajo realizadas.	PCRT=Porcentaje de completud de minutas de trabajo según las reuniones de trabajo realizadas.	PCRT2=(NMR / NRT)*100%	NMR=5, NRTP=10
Tarea	**T12. Realizar plan de Pruebas unitarias**		
Medida Base	Medida Derivada	Función de cálculo	Datos de entrada
NPP=Número de casos de pruebas diseñados y/o planeadas. NPR= Número de pruebas requeridos para obtener una cobertura adecuada.	PCP=Porcentaje en la completud de la definición de Pruebas de software.	PCP=(NPP/NPR)*100%	NPP=17, NPR=31

En la Tabla 75 se pueden ver los valores obtenidos de la medición de las tareas realizadas del proceso para el Sprint 3.

En la tarea "T6.Dar seguimiento al plan de trabajo" se evaluó la desviación del esfuerzo total (tiempo) de las tareas completada según el plan de trabajo. Si el esfuerzo es positivo (tiempo positivo) significa que hubo un retraso. Si el esfuerzo es negativo (tiempo negativo) significa que hubo ganancia de esfuerzo. De esta manera, se puede saber si el tiempo estimado para la realización de cada tarea planeada resulto correcto con respecto a lo realmente realizado (esfuerzo realizado).

Tabla 75: Datos de la medición de las tareas identificadas como causante de los defectos del producto de software del Sprint 3, caso estudio 2.

Tarea	**T6. Dar seguimiento al plan de trabajo.**			
Medida Base	Medida Derivada	Función de cálculo	Criterios	Datos de entrada
TR= Tiempo Total real de las tarea completadas (hrs.). TES=Tiempo total estimado de las tareas planeadas (hrs.).	DEPT=Desviación del esfuerzo (tiempo) de cada tarea completada del plan de trabajo.	DEPT=TR-TE	Si el esfuerzo es positivo (tiempo positivo) significa que hubo un retraso. Si el esfuerzo es negativo (tiempo negativo) quiere decir que hubo ganancia de esfuerzo.	TR=82 hrs., TE=67 hrs.

En el Sprint 4 no se realizó ninguna tarea identificada como causante de los defectos del producto de software.

En la Tabla 76 se pueden ver los valores obtenidos de la medición de las tareas realizadas del proceso para el Sprint 5

En la tarea "T6.Realizar el plan de trabajo" se evaluó la desviación del esfuerzo total (tiempo) de las tareas completada según el plan de trabajo. De esta manera, se puede saber si el tiempo estimado para la realización de cada tarea planeada resultó correcto con respecto a lo realmente realizado (esfuerzo realizado).

Para la tarea "T8.Realizar minutas en la juntas de trabajo" se evaluó el porcentaje de completud de minutas de trabajo según las reuniones de trabajo realizadas. De esta manera, se observó si las minutas de trabajo fueron realizadas según las reuniones realizadas.

En la tarea "T9.Dar seguimiento del plan de trabajo" se evaluó el porcentaje de actividades actualizadas según el avance realizado en el plan de trabajo establecido para este sprint. De esta manera, se muestra si dio seguimiento al plan de trabajo.

Tabla 76: Datos de la medición de las tareas identificadas como causante de los defectos del producto de software del Sprint 5, caso estudio 2.

Tarea	**T6. Realizar el plan de trabajo.**			
Medida Base	Medida Derivada	Función de cálculo	Criterios	Datos de entrada
TR= Tiempo Total real de las tarea completadas (hrs.). TES=Tiempo total estimado de las tareas planeadas (hrs.).	DEPT=Desviación del esfuerzo (tiempo) de cada tarea completada del plan de trabajo	DEPT=TR-TE	Si el esfuerzo es positivo (tiempo positivo) significa que hubo un retraso. Si el esfuerzo es negativo (tiempo negativo) quiere decir que hubo ganancia de esfuerzo	TR=16 hrs., TE=13 hrs.
Tarea	**T8. Realizar minutas en la juntas de trabajo.**			
Medida Base	Medida Derivada		Función de cálculo	Datos de entrada
NMR= Número de minutas realizadas. NRTP=Número de reuniones de trabajo planeadas.	PCRT=Porcentaje de completud de minutas de trabajo según las reuniones de trabajo planeadas.		PCRT5=(NMR / NRTP)*100%	NMR=4, NRTP=6
Tarea	**T9. Dar seguimiento del plan de trabajo.**			
Medida Base	Medida Derivada		Función de cálculo	Datos de entrada
NANA=Número de actividades no actualizadas registradas en el plan de trabajo. NAP=Número de actividades planeadas.	PAA=Porcentaje de actividades actualizadas según el avance realizado		PAA=(1- NANA/ NAP)*100%	NANA=2, NAP=5

En la Tabla 77 se pueden ver los valores obtenidos de la medición de las tareas realizadas del proceso para el Sprint 6

En la tarea “T1.Dar seguimiento al plan de trabajo” se evaluó el porcentaje de actividades actualizadas según el avance realizado en el plan de trabajo establecido para este sprint. De esta manera, podemos saber si dio seguimiento al plan de trabajo.

Tabla 77: Datos de la medición de las tareas identificadas como causante de los defectos del producto de software del Sprint 6, caso estudio 2.

Tarea	**T1. Dar seguimiento al plan de trabajo.**		
Medida Base	Medida Derivada	Función de cálculo	Datos de entrada
NANA=Número de actividades no actualizadas registradas en el plan de trabajo. NAP=Número de actividades planeadas.	PAA=Porcentaje de actividades actualizadas según el avance realizado	PAA=(1- NANA/ NAP)*100%	NANA=6, NAP=12

IV.5.3.4 Resultados obtenidos

Evaluación del producto de software

Los resultados de la evaluación de la calidad del producto de software final para este caso de estudio 2. (Se pueden ver en la sección de resultados obtenidos del experimento 1).

Clasificación de las tareas mapeadas (realizadas y no realizadas)

En la siguiente Tabla 78 se puede ver el resultado de la frecuencia de la clasificación de las tareas identificadas que causan defectos en el producto de software de cada Sprint. El resultado de la clasificación de las tareas fue: En el Sprint 1: las tareas realizadas fueron 4 y las tareas no realizadas fueron 10. En el Sprint 2: las tareas realizadas fueron 2 y las tareas no realizadas fueron 16. En el Sprint 3: la tarea realizada es 1 y las no realizadas fueron 10. En el Sprint 4: las tareas realizadas son 0 y las no realizadas fueron 10. En el Sprint 5: las tareas realizadas fueron 3 y las no realizadas fueron 7. En el Sprint 6: la tarea realizada fue 1 y las no realizadas fueron 5.

Tabla 78: Resultado de la clasificación de las tareas identificadas en realizadas o no realizadas, caso de estudio 2.

	Tareas	
	Realizada	No realizada
Sprint 1	4	10
Sprint 2	2	16
Sprint 3	1	10
Sprint 4	0	12
Sprint 5	3	7
Sprint 6	1	5

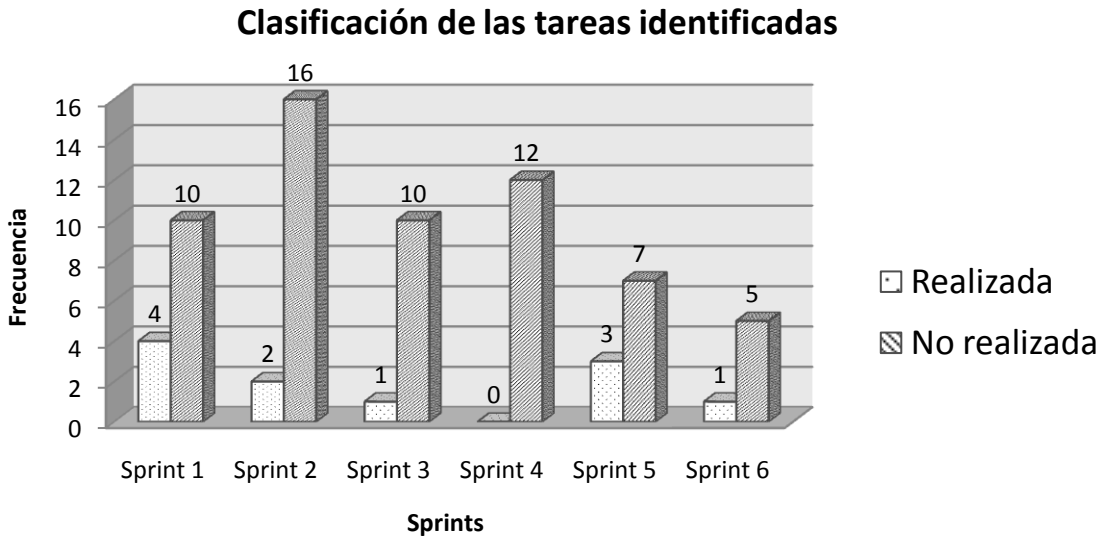

Figura 31: Gráfica del resultado de la clasificación de las tareas identificadas en realizadas o no realizadas, para el caso de estudio 2.

Evaluación de correctud de las tareas realizadas del proceso de software.

Los valores obtenidos de las mediciones de la evaluación de correctud de las tareas realizadas por cada Sprint, las cuales son las que están directamente relacionadas con la calidad del producto de software, se muestran en la Tabla 79.

Tabla 79: Resultado de la evaluación de las tareas realizadas causantes de los defectos del proceso, para el caso de estudio 2.

Sprint	**1**		
Tarea Evaluada	**Métrica**	**Resultado**	**Observación**
T3. Elaborar el plan de trabajo	Porcentaje de completud de Scrum Meeting realizadas.	0%	
T7. Realizar plan de Pruebas unitarias	Porcentaje de completud de la definición de Pruebas de software	15%	
T9. Ejecutar pruebas unitarias	Eficiencia en la ejecución de pruebas	0.75 defectos/ hrs.	Si el esfuerzo (tiempo) invertido es bastante y se encuentran pocos defectos, significa que no es eficiente las revisiones de validación.
T12. Realizar calendario de trabajo	Desviación del esfuerzo (tiempo) total de las tareas completadas según el plan de trabajo.	3 horas hombre de esfuerzo (Retraso)	Si el esfuerzo es positivo (tiempo positivo) significa que hubo un retraso. Si el esfuerzo es negativo (tiempo negativo) quiere decir que hubo ganancia de esfuerzo.

Sprint	**2**		
Tarea Evaluada	**Métrica**	**Resultado**	**Observación**
T3. Realizar minutas en la juntas de trabajo	Porcentaje de completud de minutas de trabajo según las reuniones de trabajo realizadas.	50%	
T12. Realizar plan de Pruebas unitarias	Porcentaje de completud de la definición de Pruebas de software	55%	

Sprint	**3**		
Tarea Evaluada	**Métrica**	**Resultado**	**Observación**
T6. Dar seguimiento al plan de trabajo	Desviación del esfuerzo (tiempo) total de las tareas completadas según plan de trabajo	15 hombre de esfuerzo (Retraso)	Si el esfuerzo es positivo (tiempo positivo) significa que hubo un retraso. Si el esfuerzo es negativo (tiempo negativo) quiere decir que hubo ganancia de esfuerzo.

Sprint	**5**		
Tarea Evaluada	**Métrica**	**Resultado**	**Observación**
T6. Realizar el plan de trabajo	Desviación del esfuerzo (tiempo) total de las tareas completadas según el plan de trabajo	3 hombre de esfuerzo (Retraso)	Si el esfuerzo es positivo (tiempo positivo) significa que hubo un retraso. Si el esfuerzo es negativo (tiempo negativo) quiere decir que hubo ganancia de esfuerzo.
T8. Realizar minutas en la juntas de trabajo	Porcentaje de completud de minutas de trabajo según las reuniones de trabajo realizadas.	67%	
T9. Dar seguimiento del plan de trabajo	Porcentaje de actividades actualizadas según el avance realizado	60%	

Sprint	**6**		
Tarea Evaluada	**Métrica**	**Resultado**	**Observación**
T1. Dar seguimiento al plan de trabajo	Porcentaje de actividades actualizadas según el avance realizado	50%	

En el Sprint 1 se evaluaron cuatro tareas:

La T3 se evaluó el porcentaje de completud de Scrum Meeting realizadas según el calendario, el cual resultó con un 0% de completo la realización de reuniones, en base a que debe de ser diaria la realización de Scrum Meeting para planear las actividades del día, según las prácticas que establece Scrum.

En la T7 se evaluó el porcentaje de completud de la definición de las pruebas de software en base a las funcionalidades descritas en el Sprint 1, el resultado fue 15% de completud de las pruebas planeadas con respecto a la cantidad de pruebas que debían de ser según la cantidad de funcionalidades implementadas.

En la T9 se evaluó la eficiencia de la ejecución de pruebas de funcionalidad, el resultado fue 0.75 defectos por hora, por lo que se puede considerar que fue ineficiente la ejecución de las pruebas en base al tiempo investido en la revisiones y el numero de defectos encontrados.

En la T12 se evaluó la deviación del esfuerzo total (horas) de las tareas completadas según el plan de trabajo establecido, el resultado fue de 3 horas de hombre de esfuerzo por lo que hubo un retraso según lo planeado en las actividades planeadas para este Sprint.

En el Sprint 2 se evaluaron dos tareas:

La T3 se evaluó el porcentaje de completud de minutas de trabajo según las reuniones realizadas, el resultado fue de 50% de completud de las minutas realizadas en base a las reuniones realizadas.

En la T12 se evaluó el porcentaje de completud de la definición de pruebas de software en base a las funcionalidades definidas en el Sprint 2, el resultado fue de 55% de completud de las pruebas planeadas con respecto a la cantidad de pruebas que debían de ser según la cantidad de funcionalidades implementadas.

En el Sprint 3 se evaluó una tarea:

La T6 se evaluó la desviación del esfuerzo total (horas) para la realización de las tareas completadas según el plan de trabajo establecido, el resultado fue de 15 horas de hombre de esfuerzo por lo que hubo un retraso según lo planeado en las actividades planeadas para este Sprint.

En el Sprint 4 no se evaluó ninguna tarea debido a que no se realizaron las tareas causantes de la mala calidad del producto de software.

En el Sprint 5 se evaluó tres tareas:

En la T6 se evaluó la desviación del esfuerzo total (horas) para la realización de las tareas completadas según el plan de trabajo establecido, el resultado fue de 3 horas de hombre de esfuerzo por lo que hubo un retraso según lo planeado en las actividades planeadas para este Sprint.

En la T8 se evaluó el porcentaje de completud de minutas de trabajo según las reuniones de trabajo realizadas, el resultado fue de 67% de completud de las minutas realizadas en base a las reuniones realizadas.

En la T9 se evaluó el porcentaje de actividades actualizadas según el avance realizado, el resultado fue de 60% de actividades actualizadas según el plan de trabajo para este Sprint.

En el Sprint 6 se evaluó 1 tarea:

En la T1 se evaluó el porcentaje de actividades actualizadas según el avance realizado, el resultado fue de 50% de actividades actualizadas según el plan de trabajo para este Sprint.

CAPÍTULO V

DISCUSIÓN DE RESULTADOS

V Discusión de resultados

En este capítulo se discuten los resultados más significativos sobre los principales conceptos propuestos en este trabajo de tesis. En esta discusión se da a conocer los aspectos que lograron mostrarse en la experimentación y los aspectos pendientes por mostrar.

V.1 Discusión de los resultados obtenidos de la experimentación

V.1.1 Resultados de la contribución de las tareas del proceso de software en la calidad del producto de software

En la propuesta del Metamodelo (véase capítulo III) se puede observar el mapeo que existe entre el proceso de software y la calidad del producto de software a través de las métricas. Este mapeo se realiza con los elementos más atómicos de la composición estructural de estos dos elementos: tarea y artefacto.

En base al resultado de la medición de 4 atributos de calidad del producto final (completud, correctud y correctud en información de la característica Funcionalidad y comprensión de la característica de Usabilidad) se identificaron los tipos de defectos del producto y de esta manera, se establecieron las causas potenciales de estos defectos, para así poderlas mapear con las tareas del proceso de software y obtener el tipo de tarea y propósito que contribuyen en la calidad del producto de software.

Por consiguiente, el experimento 1 (véase en IV.3) se planteo en base a las hipótesis $H_{(0,2)}$ y $H_{(0,3)}$ establecidas al inicio de la investigación, en donde se establece que las tareas base contribuyen más en la calidad del producto de software.

En base a esto, el experimento se realizó en cuatro casos de estudio diferentes, lo cual permitió observar los resultados esperados del experimento planteado.

En cuestión del resultado de la evaluación de la calidad del producto de software se obtuvo los siguientes resultados, para el caso de estudio 1, (véase en IV.3.1) resultó que la calidad del producto en los atributos de completud, correctud y correctud en la información

de interfaces (Funcionalidad) del producto de software se considera que es buena pero no alcanza una calidad al 100% , el atributo de comprensión (Usabilidad) se considera que fue media, en base a esto se realizó el análisis de los defectos y se identificaron las causas potenciales, las cuales fueron mapeadas con las tareas del proceso de software establecido. Por lo que, el caso de estudio 1, se considera que el proceso de software fue llevado sistemáticamente y ordenado, lo cual permitió identificar fácilmente las tareas del proceso, por lo que al clasificar el tipo de tareas causantes de los defectos de la calidad del producto de software, resultó que las tareas de tipo base, contribuyeron más en la calidad de producto de software en comparación con las tareas de apoyo; también resultó que las actividades con propósito de planeación y seguimiento y de realización de la fase de pruebas son las que más apoyan en la calidad del producto de software.

El caso de estudio 2 se aplicó la metodología propuesta del experimento en cada iteración o sprint del proceso de software por lo que se puede ver los resultados más detallados (véase en IV3.2). Los atributos de calidad medidos fueron completud, correctud y correctud en información de interfaces (Funcionalidad) y comprensión (Usabilidad).

Los resultados obtenidos de la calidad del producto de software fueron los siguientes, véase la siguiente gráfica de la Figura 32: El atributo de completud inició al 100% en el primer y segundo sprint pero en el tercer sprint decayó casi un 50%, en el sprint 4 volvió a mejorar pero aun sin alcanzar el 100 %, en el Sprint 5 volvió a decaer, pero en el Sprint 6 regreso a una calidad considerablemente buena, en la cual el sexto Sprint representa el producto final de software. El atributo de correctud inició con un una calidad buena, en el segundo sprint mejoro aun más, pero en el Sprint 3, 4 y 5 fue decayendo gradualmente pero, en el Sprint 6 subió su calidad considerablemente buena.

El atributo de correctud en información de interfaces comenzó con una calidad muy baja, pero mientras se iban agregando más especificaciones al producto de software y la integración con el producto final fue mejorando hasta alcanzar un nivel bueno en el sexto Sprint.

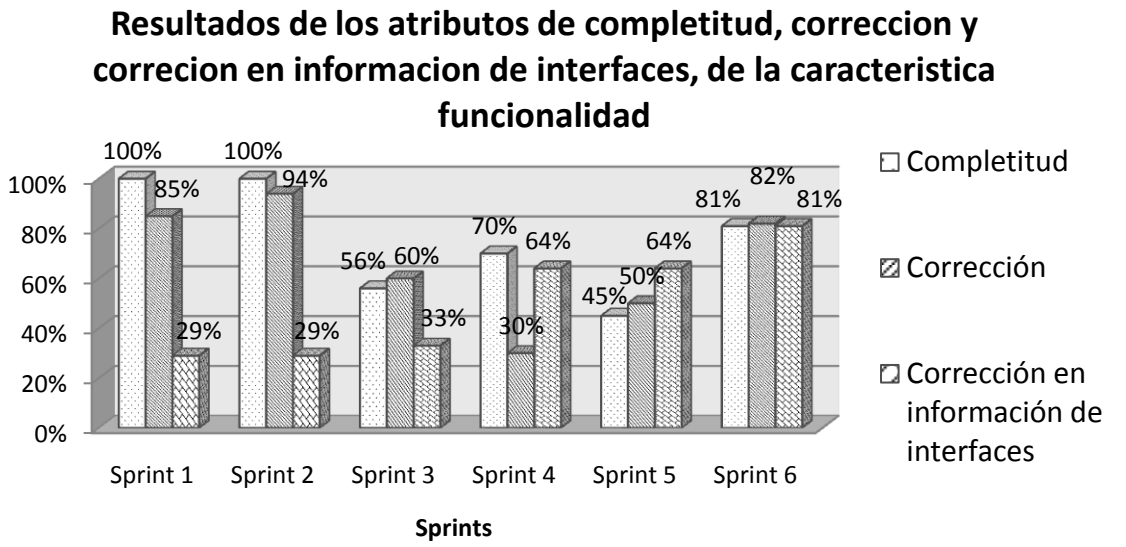

Figura 32: Gráfica de los resultados de los atributos de completud, correctud y correctud en información de interfaces de funcionalidad, para el caso de estudio 2.

El atributo de comprensión de la característica de Usabilidad resultó en el Sprint 1 con una calidad media, pero mientras se fue agregando mas especificaciones de usabilidad al producto, la calidad fue disminuyendo hasta llegar a 0% de comprensión en el Sprint 5 debido a que no se hizo ninguna mejora en ese sprint, hasta terminar en el sprint 6 con una buena calidad de comprensión de las interfaces implementadas. (Ver figura 33)

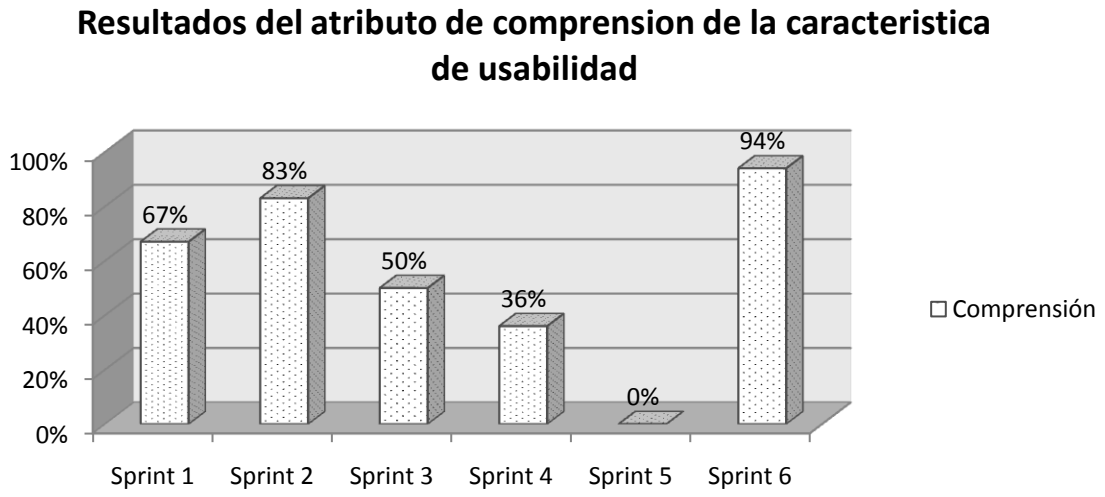

Figura 33: Gráfica de los resultados del atributo de comprensión de usabilidad, para el caso de estudio 2.

En base a estos resultados de la calidad del producto por Sprint se pudo identificar las causas que llevaron a cada Sprint a obtener la calidad del producto generado. La identificación de las causas se obtuvo en base al análisis de retrospectiva entre los integrantes del equipo al final de cada Sprint, en la cual se concluía con las tareas causantes que impactaron más en la calidad del producto. Por lo que al clasificar las tareas identificadas en todos los Sprints se obtuvo que las tareas base fueron las que más impactaron en la calidad del producto de software en comparación con las Actividades de Apoyo.

También se identificó el propósito de las actividades, en donde se obtuvo que las actividades con propósito de planeación y seguimiento y de la realización de la fase de pruebas apoyaron más en la calidad del producto de software en comparación con los otros propósitos establecidos.

El caso de estudio 3 (véase en IV.3.3), resultó que la calidad del producto en los atributos de completud, la correctud y correctud en la información de interfaces (Funcionalidad) del producto de software se considera que es buena, y el atributo de calidad comprensión resulto con calidad media, aun así no alcanzaron un 100% de calidad. En base a esto, se realizó el análisis de los defectos y se identificó las causas de dichos defectos mapeándolos con las tareas del proceso y resulto que las tareas base fueron las que más contribuyeron a que no se cumpliera con el 100 % de la calidad de los atributos medidos en comparación con las tareas de apoyo.

También resultó que las actividades con propósito de verificación y validación y, las de realización de la fase de requerimientos y pruebas son las que más apoyan en la calidad del producto de software.

El caso de estudio 4 (véase en IV3.4), resultó que la calidad del producto en los atributos de completud y de correctud (Funcionalidad) del producto de software se considera que es buena pero no se alcanzo el 100% de calidad, por lo que se hizo el análisis de los defectos y la identificación de las causas de dichos defectos mapeándolos con las tareas del proceso y resulto que las tareas base contribuyeron a no alcanzar la calidad esperada en esos dos atributos de calidad en comparación con las actividades de apoyo.

También resultó que las actividades con propósito de verificación y validación y de planeación y seguimiento son las que más apoyaron en la calidad del producto de software.

Por lo que se puede concluir, que en base a los resultados obtenidos en estos casos de estudio realizados, las tarea base son las que más impactan en la calidad del producto de software en comparación con las tareas de apoyo. Por lo que la calidad del producto en cuestión de los atributos evaluados, depende de que se lleven a cabo las tareas base del proceso de software sin importar la metodología de desarrollo utilizado, logrando así el cumplimiento de las hipótesis $H_{(0, 2)}$ y $H_{(0, 3)}$ planteadas.

Por otro lado, también se puede concluir que las actividades del proceso de software que tienen como propósito planeación y seguimiento, realización de fase de pruebas y verificación y validación son las que más apoyan en la calidad del producto de software en comparación con las otras actividades que tienen otros propósitos.

Realizando un análisis de las frecuencias obtenidas en los tipos de propósitos de las actividades del proceso con respecto a los atributos de calidad evaluados, se puede hacer un mapeo en donde la completud del producto de software está relacionada con las tareas que tienen como propósito la planeación y seguimiento, por otro lado el atributo de correctud y correctud en la información de interfaces se relaciona con las actividades que tienen como propósito la realización de pruebas y de verificación y validación.

V.1.2 Resultados de la contribución de los artefactos de software generados en la calidad del producto de software

Continuando con el estudio de la composición estructural del proceso de software, se observa que en la propuesta del Metamodelo se estableció que una tarea de tipo base generaba artefactos tangibles y que estos a su vez se clasificaban en dos tipos: tangible de contribuidor directo y tangible de contribuidor indirecto. En base a esto se puede analizar cómo estos tipos de tareas base con sus respectivos tipos de artefactos apoyan a la calidad del producto de software.

Siguiendo la metodología del experimento 1 y evaluando los mismos atributos de calidad del producto de software, este experimento 2 (véase en IV.4) se planteo en base a

las hipótesis $H_{(0,4)}$ y $H_{(0,5)}$ establecidas al inicio de la investigación, en donde se plantea que los artefactos de contribuidor directo contribuyen más en la calidad del producto de software.

En el caso de estudio 1(véase IV.4.2), el resultado fue que los artefactos de tipo contribuidor directo son los que más impactan en la calidad del producto de software en comparación con los artefactos tangibles de contribuidor indirecto.

En el caso de estudio 2 (véase IV.4.3), se obtuvo en todos los Sprints que las tareas base que generan artefactos tangibles de contribuidor directo, aportan más a la calidad del producto de software en comparación con los artefactos de contribuidor indirecto.

En el caso de estudio 3 y 4 (véase IV.4.4 y IV4.5), se confirmó nuevamente que los artefactos tangibles de contribuidor directo aportan más a la calidad del producto de software en comparación con los artefactos de contribuidor indirecto.

En base a estos resultados se puede concluir que los artefactos de contribuidor directo aportan más elementos al producto final, debido a que se generan de actividades base, que también se comprobó (experimento 1) que son las que contribuyen más en la calidad del producto de software en comparación con las de apoyo, por tal motivo al realizarse las actividades base, sin importar el tipo de metodología o modelo que se utilice para el desarrollo del producto de software, se deben de generar los artefactos tangibles tanto de contribuidor directo como indirectos para tener un producto final de buena calidad.

V.1.3 Resultados del impacto que tienen la ejecución correcta de las tareas del proceso de software sobre la calidad del producto de software

Siguiendo con el estudio del Metamodelo propuesto (véase capítulo III), se puede ver que las tareas del proceso de software y los artefactos del producto de software están estrechamente relacionados, debido a que la tarea genera un artefacto, y este es entrada para una tarea, por lo que se puede suponer que "realizando" y haciendo "correctamente" la tarea se obtiene un buen artefacto.

Para conocer la calidad del producto de software se midieron cuatro atributos de calidad del producto final (completud, correctud y correctud en información de la

característica Funcionalidad y comprensión de la característica de Usabilidad) se identificaron los tipos de defectos del producto y de esta manera, se establecieron las causas potenciales de estos defectos, para así poderlas mapear con las tareas del proceso de software y después, clasificar las tareas mapeadas en realizadas y no realizadas; y de las tareas realizadas se evaluó la calidad de dichas tareas.

El experimento 3 (véase en IV.5) se planteó en base a la hipótesis $H_{(0,6)}$ establecida al inicio de la investigación. En base a esto, el experimento se realizó en dos casos de estudio diferentes, lo cual permitió observar los resultados esperados del experimento planteado. Este experimento, solo se pudo hacer en dos casos de estudio (1 y 2) debido a que se necesitaba más datos para poder evaluar la calidad de las tareas realizadas.

El resultado de la evaluación de la calidad, de los casos de estudio se puede ver en los resultados de los experimentos anteriores (experimento 1 y experimento 2).

Para el caso de estudio 1, se puede concluir que las tareas no realizadas (5 tareas) tienen un mayor impacto en la calidad del producto debido a que son de tipo base (según el experimento 1) y por consiguiente al no realizarlas, no aportan ningún elemento al producto de software. De las tareas realizadas (8 tareas), se evaluaron para saber en forma cuantitativa si se ejecutaron correctamente; de esta manera, teniendo los tipos de defectos que afectaron la calidad del producto final de software y conociendo el resultado de la medición de las tareas causantes de dichos defectos, se puede establecer el impacto que tiene la ejecución de las tareas del proceso de software en la calidad del producto de software.

En base al resultado del caso de estudio 1, se concluye que las tareas que más afectaron al atributo de "completud", en base al resultado obtenido de las métricas establecidas para cada tarea realizada, fueron: la T3 porque hubo un retraso de tiempo de 10 horas hombre con respecto a lo planeado. Por lo que se puede decir que la planeación de las actividades para la capacitación del equipo de trabajo no estuvo bien planeada y eso ocasionó que el equipo no pudiera terminar la implementación todas las funcionalidades en el tiempo planeado. La T4 porque no se completo la documentación de la especificación de cada caso de estudio con respecto al estándar de documentación que establece el proceso unificado Ágil, por lo que se concluye que esta tarea contribuyó para que el producto de

software no fuera completado en sus funcionalidades definidas debido a que las especificaciones de los casos de uso no estaban detalladas. La T5 porque no estaban completas las especificaciones de los casos de uso definidos en cuestión de validaciones y excepciones, por lo que ocasionó que esta tarea apoyara a que no se completara con las funcionalidades descritas, debido a que falto más detalle en validaciones y excepciones de cada caso de uso implementado. La T6 porque las revisiones de los casos de uso por el cliente no fueron eficientes, debido a que no se encontraron ningún defecto en el tiempo invertido de las revisiones, por lo que se puede concluir que esta tarea al no realizarse eficientemente, la completud del producto de software no fue detectada en fases tempranas por el cliente. La T7 porque hubo un retraso en el tiempo planeado de la implementación de los casos de uso establecidos, lo que ocasionó que no se alcance a completar la implementación de todos los casos de uso en el tiempo estimado. También se concluye que la T1 no realizada impactó en este atributo de calidad del producto debido a que no se especificaron los requerimientos funcionales desde un inicio.

Para el atributo de calidad “correctud”, se concluye que las tareas que más afectaron al atributo, según al resultado obtenido de las métricas establecidas para cada tarea realizada, fueron: la tarea T10 porque no fue completado el plan de pruebas en base a las funcionalidades implementadas en el producto, por lo que ocasionó que no se probara completamente la funcionalidad del producto. La T12 porque la ejecución de las pruebas de funcionalidad no fueron eficientes, debido a que no se encontraron defectos en la ejecución del pruebas entre los integrantes del equipo en el tiempo invertido en la ejecución de la tarea, por lo que se puede concluir que esta tarea al no realizarse eficientemente impacto en la correctud del producto de software. Al igual, se concluye que las T11 y T13 no realizadas, afectaron al atributo de calidad del producto, debido al propósito especifico de estas tareas.

Para el atributo de calidad “correctud en información de interfaces”, se concluye que al igual que el atributo anterior, las tareas T10 y T12 fueron las que más afectaron en la calidad de este atributo, al igual que las tareas no realizadas T1, T2 y T13, debido a que aportaban elementos esenciales para la especificación de la información de las interfaces.

Para el atributo de calidad “comprensión”, se concluye que al igual que el atributo anterior, las tareas T10 y T12 fueron las que más afectaron en la calidad de este atributo, al igual que las tareas no realizadas T1, T2, T9, T12 y T13, debido a que estas tareas por su propósito debían de aportar elementos esenciales para la fácil comprensión de la interfaz de usuario.

Para el caso de estudio 2, se concluye que, al igual que el caso de estudio anterior, las tareas no realizadas en cada Sprint contribuyeron a que la calidad del producto de software final no fuera la esperada, debido a que son de tipo base (según el experimento 1) y por consiguiente al no realizarlas, no aportan ningún elemento al producto de software. En este caso de estudio se puede justificar el no realizar ciertas actividades, por el tipo de metodología que utilizaron (SCRUM) el cual les permite decidir cuales tareas realizar y cuáles no, según el acuerdo por el equipo de trabajo. Sin embargo, en base al análisis de retrospectiva que se hizo al final de cada Sprint, los integrantes del equipo señalaron las tareas que debían realizar para la mejora de la calidad del producto de software.

En base a esto se identificó que en todos los Sprints la mayoría de las tareas no se realizaron por lo que se concluye que estas tareas al no ser realizadas y que en su mayoría son de tipo base, impactaron negativamente a la calidad del producto de software.

De las tareas realizadas (11 tareas), se evaluaron para saber en forma cuantitativa si se ejecutaron correctamente y de esta manera, poder ver el impacto que tiene la ejecución de dichas tareas en la calidad del producto de software.

En el Sprint 1 las tareas que obtuvieron un resultado negativo son las tareas T3, T7, T9 y T12, por lo que al no ejecutarse correctamente contribuyen a la generación de defectos del producto de software, de los cuales se adjudica a los atributo de calidad: correctud en información de interfaces (funcionalidad) y de comprensión (Usabilidad), debido a que los otros atributos evaluados resultaron con buena calidad.

En el Sprint 2 las tareas que impactaron a los atributos de calidad correctud en información de interfaces (funcionalidad) y de comprensión (Usabilidad), son T3 y T12, debido a que no se obtuvo un buen resultado en la evaluación.

En el Sprint 3 la tarea realizada obtuvo un resultado negativo por lo que se considera que contribuyó en los atributos de calidad evaluados: completud, correctud en información de interfaces (funcionalidad) y de comprensión (Usabilidad).

En el Sprint 4, las tareas identificadas como causantes de los defectos del producto no se realizaron por lo que no hay tarea realizada que evaluar.

En el Sprint 5, las tareas realizadas que obtuvieron un resultado no satisfactorio son la T8 y la T9, por lo que se considera que contribuyeron en los atributos de calidad del producto, así como las tareas no realizadas.

En el Sprint 6, la tarea realizada T1 obtuvo un resultado no satisfactorio, por lo que contribuyó en los atributos de calidad del producto, así como las tareas no realizadas.

En base a estos resultados de cada Sprint tanto del producto como de las tareas del proceso se concluye que las tareas no realizadas en cada Sprint y las que no se ejecutaron correctamente impactaron en la calidad del producto de software final (producto obtenido en el Sprint 6).

Analizando los dos casos de estudio se puede decir que el no realizar y el no ejecutar correctamente las tareas base del proceso, impactan negativamente en la calidad de producto de software, por lo que se concluye que la hipótesis $H_{(0,6)}$ planteada se cumple.

CAPÍTULO VI

CONCLUSIONES Y TRABAJO FUTURO

VI Conclusiones y Trabajo Futuro

En esta sección se exponen las conclusiones y aportaciones específicas de este trabajo de tesis, así como el trabajo futuro.

En base a los objetivos planteados en esta investigación y a los experimentos realizados se llegó llegar a las siguientes conclusiones.

El desarrollo del Metamodelo permitió identificar los elementos de composición del Proceso de Software, Producto de Software y Modelo de Calidad, llegado a su nivel más fino de granularidad, es decir la tarea, artefacto y atributo de calidad correspondiente, de esta manera, teniendo establecido cada uno de estos elementos, se estableció el mapeo entre ellos a través de las métricas de software.

En base a esta descripción semi-formal (Metamodelo) de la relación que existe entre el proceso de software y la calidad del producto, se realizaron experimentos en cuatro casos de estudio distintos, con el propósito de llegar a los objetivos planteados en esta investigación así como comprobar si las definiciones establecidas en el Metamodelo son correctas. Para cada experimento realizado, se aplicó la siguiente metodología: evaluar la calidad del producto final, utilizando las métricas de calidad. En base al resultado cuantitativo de las métricas aplicadas en la calidad del producto, se puede saber el grado de defectos que tiene dicho producto. De esta manera, conociendo el grado de calidad que tiene el producto se puede hacer el análisis de defectos en donde se identifica la causa raíz de los defectos, utilizando el método FMEA. Al tener identificadas las causas de los defectos se hace la correspondencia con las tareas del proceso que generaron dicho producto de software. Con esta información se demuestran los siguientes aspectos:

- Al definir el proceso de software en su nivel más fino de composición, es decir la tarea, contribuyó para que el mapeo sea directo con la causa raíz del defecto del producto de software; como se puede ver en todos los casos de estudio planteados en los experimentos (véase Capítulo IV).
- Por otro lado, al tener identificadas las tareas del proceso que causaron los defectos en la calidad del producto de software, se clasificaron estas tareas según su tipo (Base o apoyo), que corresponde a la descripción de

composición del proceso de software del Metamodelo. En base a los resultados obtenidos en los experimentos se concluyó que las tareas Base son las que más contribuyen en la calidad del producto de software en comparación con las tareas de apoyo, por lo que si no se realizan este tipo de tareas el impacto es mayor en la calidad del producto.

- Al igual, al tener identificadas las tareas Base del proceso que causaron los defectos para la calidad del producto final, se clasificó el tipo de artefacto Tangible que generan estas tareas, es decir, artefacto Tangible de Contribuidor Directo y artefacto Tangible de Contribuidor Indirecto (esta descripción véase en el Metamodelo de la sección de composición estructural del proceso y del producto de software en el Metamodelo, capítulo III) en el cual, según en los casos de estudios aplicados, se concluyó que las tareas Base que generan artefactos tangibles de tipo contribuidor directo aportan elementos esenciales a otra tarea del proceso (como artefacto de entrada), de los cuales se demostró que tienen un mayor impacto en la calidad del producto en comparación con los artefactos tangibles de contribuidor indirecto.
- Analizando el tipo de tareas del proceso de software causantes de los defectos en la calidad de producto de software, se observó (caso de estudio 1 y 2) que las tareas que no fueron realizadas tuvieron un mayor impacto en la calidad del producto, debido a que son de tipo base y por consiguiente al no realizarlas no aportaron ningún elemento al producto de software final. Por otro lado, las tareas que fueron realizadas se evaluaron para saber en forma cuantitativa si se ejecutaron correctamente y de esta manera, conocer el grado con el cual las tareas realizadas impactaron en la calidad del producto de software. En base a esto, se concluye que las tareas ejecutadas que más impactaron negativamente en los atributos de calidad del producto de software son:
    a. Para el atributo de calidad “completud” del producto de software, fueron las tareas que tienen como propósito la planeación y

seguimiento del proceso de software, el análisis a detalle de los requerimientos del producto de software y el desarrollo y ejecución de los planes de pruebas unitarias y de integración.

b. Para el atributo de calidad "correctud" del producto de software, fueron las tareas que tienen como propósito la planeación y seguimiento de proceso de software, el desarrollo y ejecución de los planes de pruebas.
c. Para el atributo de calidad "correctud en información de interfaces", del producto de software, fueron las tareas que tienen como propósito el desarrollo y el plan pruebas de usabilidad, así como la definición detallada de los requerimientos de funcionalidad y usabilidad.
d. Para el atributo de calidad "comprensión", fueron las tareas que tienen como propósito la especificación de requerimientos de usabilidad, especificación y definición de los componentes de la interfaz de usuario y navegabilidad, así como las pruebas de unitarias y de integración enfocadas a la usabilidad del producto.

Según Loard Kelvin, *"cuando puedes medir aquello de lo que estás hablando y expresarlo en números, sabes algo sobre ello, pero cuando no lo puedes medir, y no lo puedes expresar en números, tu conocimiento acerca de ello es insatisfactorio";* de esta manera, en base a esta investigación, se puede resaltar la importancia de la aplicación de las métricas de software tanto para la calidad del producto de software como para la evaluación de la ejecución de las tareas del proceso de software, porque debido a esto se puede saber con cierta seguridad el impacto que tiene el proceso de software sobre la calidad del producto de software.

Este trabajo de investigación aporta a la industria de software, la justificación por la cual es importante la realización de las tareas base del proceso de software, independientemente de la metodologías de desarrollo utilizada (tradicional o ágil) o el modelo de procesos, debido a que generan artefactos tangibles que aportan elementos esenciales para la construcción de un producto de software de calidad. El no realizar

correctamente este tipo de tareas el impacto en la calidad del producto resultaría negativo por lo que no se tendría la satisfacción del cliente.

VI.1 Trabajo a futuro

En esta sección se describe el trabajo a realizar en un futuro cercano, el cual permite continuar la investigación tomando como base los logros obtenidos hasta el momento.

Características de calidad del producto de software

Realizar más casos de estudio en donde se aplique todas las características de calidad del producto (Funcionalidad, Usabilidad, Mantenibilidad, Confiabilidad, Eficiencia y Portabilidad), así como todos los atributos correspondientes, y de esta manera poder mapear sus defectos correspondientes de la calidad del producto con las tareas del proceso de software establecido y por consiguiente, conocer el impacto que tiene el proceso de software realizado en la calidad del producto generado.

Elementos estructurales del proceso de software

Identificar los elementos de composición del Rol y de las Herramientas del proceso de software para poder realizar el mapeo con la calidad del producto de software, y de esta manera identificar los elementos que impactan directamente e indirectamente en la calidad del producto.

Especificar pesos en los resultados obtenidos

Madurar el método para la identificación de las causas de defectos del producto de software en donde se pueda establecer un peso específico a los resultados obtenidos. Para poder realizar este trabajo, se necesita realizar un conjunto de casos de estudios, en cualquier circunstancia (metodología de desarrollo, tipo de proyecto, tipo de experiencia en el tema, etc.), el cual permita obtener datos significativos para establecer los pesos de cada resultado en la calidad del producto como del proceso de software.

Exposición de los resultados de los casos de estudio

Exponer y publicar un artículo sobre el resultado de los casos de estudios realizados y dar a conocer las aportaciones y conclusiones establecidas en este trabajo de tesis.

VI.2 Publicaciones

Durante el desarrollo de este trabajo de tesis, el cual abarco dos años y medio de maestría, se realizaron dos presentaciones en eventos en donde se dieron a conocer el tema de tesis.

Los eventos fueron:

- Participación en el *Workshop on Computational Modelling and Complex Systems 2009*, Tijuana B.C, en donde se expuso el Metamodelo de "Relación entre Proceso, Producto y Modelo de Calidad".
- Participación en el *Coloquio Nacional de Investigación en Ingeniería de Software y Vinculación Academia-Industria*, en León Guanajuato del 29 de septiembre al 1 de Octubre del 2010. En este evento se publicó el artículo con el título de "*Método para establecer la relación entre la calidad del producto y la calidad del proceso de software, utilizando métodos cuantitativos*". Los autores: Luz Adriana Cárdenas Martínez, Reyes Juárez-Ramírez, Guillermo Licea, Antonio Rodríguez Díaz. Estatus: Aceptado.

CAPÍTULO VII

REFERENCIAS BIBLIOGRÁFICAS

VII Referencias Bibliográficas

Avison, D. L., F., Myers, M. y Nielsen, A. (1999). *Action Research*. Communications of the ACM 42: 94-97.

Barafort, B. (2005). *Information Security Management and ISO/IEC 15504: the link opportunity between Security and Quality*.

Basili, V., G. Caldiera, and D. Rombach (1994). *Goal Question Metric Paradigm in Encyclopaedia of Software Engineering*. New York., Wiley-Interscience.

Boehm, B. W., Brown, J. R., Kaspar, J. R., Lipow, M. L. & MacCleod, G. (1978). *Characteristics of Software Quality*. A. Elsevier. New York.

Bohem B, R. T. (2003). *Balancing Agility and Discipline*. *A guide for the perplexed*. A. Wesley.

Briand, S. M., V. Basili (1996). *Property-based Software Engineering Measurement*. I. T. o. S. Engineering.

Callaos, N. a. B. C. (1996). *Designing with Systemic Total Quality*. Proceedings of the 24th International Conference on Software Engineering, Orlando, Florida.

Chen P., T. B. y. W. L. (1999). *Future directions of conceptual modeling*. Conceptual Modeling: Current issues and future directions. A. J. Chen P., Kangassalo H., Thalheim B.: 258-271.

Instituto Nacional de Tecnologías de la Comunicación (2008). *Guía de mejores prácticas de calidad de producto*, Instituto Nacional de Tecnologías de la Comunicación.

Crosby, P. B., Ed. (1979). *Quality is free: the art of making quality certain*. New York.

D Moody, S. G. (1998). *Improving the Quality of Entity Relationship Models-Experience in Research and Practice*. Proceedings of the Seventeenth International Conference on Conceptual Modelling (ER '98), Singapore.

Deming, W. E. (1988). *Out of the crisis: quality, productivity and competitive position*, Cambridge Univ. Press.

Dr. Mario Piattini Velthuis, D. I. G. R. d. G. (2008-2009). *Calidad de sistemas de información: Métricas.* España.

Fenton, N. E. y. P., S.L. (1997). *Software Metrics. A rigurous and practical approach*. P. Pub.

French, W. L. y. B., C. H. (1996). *Organizational Development: Behavioral Science Interventions for Organization Improvement*. London, United Kingdom: New Jersey, Prentice-Hall Inc.

Fuggetta, A. (2000). *Software Process: A Roadmap*. The Future of Software Engineering, ACM Press 2000.

García, F. (2003). *Gestión Integrada del Modelado y de la Medición del Proceso Software*. Ciudad Real, España, Universidad de Castilla-La Mancha.

García, F. (2004). *FMESP: Marco de Trabajo Integrado para el Modelado y la Medición del Proceso de Software.* Ciudad Real, España, Universidad de Castilla-La Mancha. Doctorado.

Gérald Lomprey, S. H. (2008). *La importancia de la Calidad en el desarrollo de productos de software*. México, Facultad de Ingeniería y Tecnología, Universidad de Montemorelos.

Glass, R. V., I. y Ramesh, V. (2002). *Research in Software Engineering: an analysis of the literature*. Information and Software Technology 44: 491-506.

Group, O. M. (2008). *Software & Systems Process Engineering Meta-Model Specification*.

Hearn, P. (2004). *Knowledge Anywhere Anytime- or The social Life of Knowledge-Workshop Report*.

Humphrey, W. S. (1989). *Managing the Software Process*. The SEI Series in Software Engineering.

Institute, S. E. (2002). CMMI *for Systems Engineering/Software Engineering*. Pittsburgh.

International Organization for Standardization, I. (1983). ISO/IEC 729-1983: IEEE Standard Glossary of Software Engineering Terminology.

International Organization for Standardization, I. (1987). ISO 9126, *Software Product Evaluation - Quality Characteristics and Guidelines for their User*. Geneva, Switzerland.

International Organization for Standardization, I. (1988). 1061-1998. IEEE Standard for a Software Quality Metrics Methodology.

International Organization for Standardization, I. (1990). ISO 8402, ISO/IEC Information Technology - Information Resources Dictionary System (IRDS) - Framework.

International Organization for Standardization, I. (2000). *ISO/ IEC 9000. 2000 Quality management systems*. Fundamentals and vocabulary. Geneva, Switzerland.

International Organization for Standardization, I. (2001). *ISO/IEC. 2001a. ISO/IEC 9126-1: Software Engineering-Software product quality-Part 1: Quality model*. Geneva, Switzerland.

International Organization for Standardization, I. (2002). ISO/IEC 12207:1995, Industry Implementation of International Standard. Standard for Information Technology – Software Life Cycle. Geneva, Switzerland.

International Organization for Standardization, I. (2003). ISO/IEC. 2003a. ISO/IEC TR 9126-2: Software Engineering-Software product quality-Part 2: External metrics. Geneva, Switzerland.

International Organization for Standardization, I. (2003). ISO/IEC. 2003b. ISO/IEC TR 9126-3: Software engineering-Software product quality-Part 3: Internal metrics. . Geneva, Switzerland.

International Organization for Standardization, I. (2004). ISO/IEC. 2001b. ISO/IEC DTR 9126-4: *Software engineering-Software product quality-Part 4: Quality in use metrics*. Geneva, Switzerland.

International Organization for Standardization, I. (2005). ISO/IEC-25000:2005, *Software engineering-Software product Quality Requirements and Evaluation (SQuaRE)*-Guide to SQuaRE. Geneva, Switzerland.

International Organization for Standardization, I. (2003). *ISO/IEC 15504-2 Software engineering — Process assessment.*

International Organization for Standardization, I. (2004). *ISO_IEC_15504-3 Guidance on performing an assessment.*

Ivanisevich, J., P. Lorenzi, S. Skinner, and P. Crosby (1997). *Management Quality and Competitiveness*. IrWin/MsGraw-Hill. New York.

Jean-Claude Derniame, e. a. (1999). *Software Process: Principles, Methodology, and Technology*. LNCS #1500, Springer-Verlag.

Jetter, A. (2006). *Assessing Software Quality Attributes*. Department of Informatics. Zurich, University of Zurich.

Juárez-Ramírez, J. R. (2008). *Formación de la Rastreabilidad y la correspondencia entre artefactos de software aplicando teoría de grafos, axiomas, diagramas de árbol y permutaciones*. Maestría y Doctorado en Ciencias e Ingeniería. Tijuana Baja California, Universidad Autónoma de Baja California. Doctor en Ciencias.

Kallakuri, P. y. E., S. (2000). *Experimental Studies in Empirical Software Engineering*.

Knight J. C. y Brilliant, S. S. (1998). *Report on Empirical Research in Software Engineering: A workshop*. National Science Foundation U. o. M. a. U. o. Virginia. Greenbelt, MD.

Kock, N. y. L., F. (2001). *Information Technology & People (special issue on Action Research in Information Systems).*

Langford, J. W. (1995). *Logística: Principios y usos*.

Larman (2004). *Agile & iterative Development. A manager´s guide*. A. Wesley.

Maicher, L., Sigel, A. y Garshol, L.M. (2005). *Reconstructing Requirements Coverage Views from Design and Test using Traceability Recovery via LSI*. Proceedings of the 3rd ACM International Workshop in Engineering Forms of Software Engineering Long Beach California, USA.

Marc-Alexis Côté, W. S., Elli Georgiadou (2008). *Software Quality Model Requirements for Software Quality Engineering.*

María A. Pérez, T. R., Luis E. Mendoza, Anna C. Grimán (1999). *Systemic quality model for system development process: case study1*. Systemic Quality Model for System Development Process.

Marín, B. (2007). *Calidad en modelos conceptuales: un análisis multidimensional de modelos cuantitativos basados en la ISO 9126*. Revista de Procesos y Métricas.

Mario G. Piattini, F. O. G. (2003). *Calidad en el desarrollo y mantenimiento del software*. Madrid, España, RA-MA.

Maryoly ORTEGA, M. A. P. (2001). *A Systemic Quality Model for Evaluating Software Products*, Venezuela.

Mateus Ferreira, F. G., Francisco Ruiz, Manuel F. Bertoa, Coral Calero, Antonio Vallecillo, Mario Piattini, Beatriz Mora (2006). *Medición del Software Ontología y Metamodelo*. Departamento de Tecnologías y Sistemas de la Información. Información. Castilla la Mancha, Universidad de Castilla la Mancha.

McCall, J. A., Richards, P. K., & Walters, G. F. (1977). *Factors in software quality. Air Development Center Air Force Systems Command.* Griffiths Air Force Base, N.Y. Rome.

Mendoza, L. E. (2005). *Prototipo de Modelo Sistémico de Calidad (MOSCA) del Software*. Computación y Sistemas 8: 196-217.

Monasor, M. J. (2008). *Calidad de Proceso*. Una Orientación hacia el Desarrollo Distribuido de Software. España, Universidad de Castilla-La Mancha.

Montilva, J. A. (2007). Modelo de Procesos de Software. Venezuela, Universidad de los Andes.

Moody, D. (2000). *Building links between IS research and professional practice: improving the relevance and impact of IS research*. Proceedings of the 21st International Conference on Information Systems, Brisbane, Australia.

NYCE, A. C. (2005). *NMX-I-059/03-NYCE-2005. Tecnología de la Información-Software-Modelos de procesos y evaluación para desarrollo y mantenimiento de Software.* México, D.F. 970-9841-09-2.

Oktaba, H. (2006). *Tendencias Internacionales en Procesos de Software*. Ciudad de México, AMCIS, UNAM.

Oktaba Hanna, A. E. C. y. B. M. B. (2005). *Modelo de Procesos de Software (MoProSoft) por niveles de capacidad de procesos.* México D.F.

Olivé, A. (2000). *An introduction to conceptual modeling of information systems*. In Chapter 2: Advanced Database Tecnology and Design, Eds. Marios Piattini y Oscar Diaz, Artech House: 25-57.

Oliveira, K. M. d., A.R. Rocha, and K.C. Weber, (2002). *Workshop on software quality, in Proceedings.* 24th International Conference on Software Engineering, Orlando, Florida.

Oliver Mäckel, S. A. *Software FMEA Opportunities and benefits of FMEA in the development process of software-intensive technical systems*. München, Simulation and Risk Management.

Ortega, M. P., María and Rojas, Teresita (2003). *Construction of a Systemic Quality Model for evaluating a Software Product*. Software Quality Journal.

Pete Deemer, G. B. (2007). *An Introduction to Agile Project Management with Scrum*.

Pfleeger, S. L. (1999). *Albert Einstein and Empirical Software Engineering IEEE Computer.* 32 (10): 32-37.

Pfleeger, S.L. y L. Hatton (1997). *Investigating the influence of formal methods*. IEEE Computer. 30, 2: 33-43.

Philips, P. A. (1998). *Disseminating and Applying the Best Evidence*. Medical Journal of Australia (MJA) 168: 260-261.

Pressman, R. S. (2002). *Ingeniería del Software - un enfoque práctico*. España.

Prosoft. (2010). *Programa para el Desarrollo de la Industria del Software [Prosoft]*. www.prosoft.economia.gob.mx.

Rojas, T. (2007). *Mejora de la calidad del proceso a través de infocas: un estudio de caso*. Revista de la Facultad de Ingeniería de la U.C.V.

Ruiz, F. (2008). *Guía de Uso de SPEM 2 con EPF Composer*. España, Universidad de Castilla-La Mancha.

Satpathy, M. (2002). *A Typed Generic Process Model for Product Focused Process Improvement.* Computer Software and Applications Conference.

Stamatis, D. H. (1995). *Failure Mode and Effect Analysis*. FMEA from Theory to Execution. Milwaukee, Wisconsin.

Steven Kmenta, P. F., Kosuke Ishii (1999). A*dvanced failure modes and effects analysis of complex processes*. Proceedings of the 1999 ASME Design Engineering Technical Conferences, Las Vegas, Nevada.

Wadsworth, Y. (1998). *What is participatory Action Research?* Action Research International.

Wang, Y., et al., (1997). *Quantitative Evaluation of the SPICE, CMM, IS0 9000 and BOOTSTRAP*. Proceedings of the 3rd International Software Engineering Standards Symposium (ISESS '97). .

Wohlin, C. (2007). *Empirical Software Engineering: Teaching Methods and Conducting Studies.* Lecture Notes in Computer Science 4336/2007, 135-142.

Zeiss, B. *Applying the ISO 9126 Quality Model to Test Specifications*.

CAPÍTULO VIII

APÉNDICE

VIII Apéndice

VIII.1 Apéndice A: Herramienta de investigación "Investigación - Acción"

Desde el punto de vista científico, las áreas de sistemas de información e ingeniería de software deberían tratarse con un carácter experimental (García 2004)La investigación es estas áreas se centra en conocer la naturaleza de los procesos, productos y sus interrelaciones en el contexto de un sistema de software o de un sistema organizacional (en el caso de sistemas de información).

La ingeniera de software como cualquier otra disciplina científica aplicada (como la química o la medicina) tiene dos objetivos fundamentales(Philips 1998):

- Aumentar el conocimiento (aspecto teórico) para entender por qué las cosas ocurren en un área particular de interés
- Mejorar las prácticas (aspecto práctico) de forma que los resultados de la investigación sean útiles.

Por lo tanto, en la ingeniería de software es fundamental establecer una buena base teórica, teniendo en mente el objetivo de obtener resultados útiles en la práctica. Sin embargo, con frecuencia se proponen, por ejemplo nuevas ideas y metodologías, pero sin aplicación en la práctica, existiendo una clara desconexión entre la investigación teórica y su aplicación. (Moody 2000)

En las áreas de la ingeniería se tiene el reto de eliminar la barrera entre la teoría y la práctica (Glass 2002). En el caso de la ingeniería de software es necesario que la investigación este orientada a objetivos prácticos, y que la industria del software aplique los resultados obtenidos en la investigación (García 2004)

Parte de la investigación llevada a cabo en las áreas de sistemas de información e ingeniería de software es de tipo cuantitativo, basándose principalmente en técnicas estadísticas (García 2004). Por otro lado, existen muchas áreas dentro de la ingeniería de software que pueden ser tratadas con métodos cualitativos. Sin embargo, tal como se puede ver en algunos estudios realizados como el de (Glass 2002), los métodos cualitativos no han

sido muy aplicados en ingeniería de software; en especial la investigación-acción no ha sido puesta en práctica con mucha frecuencia dentro de esta disciplina.

La aplicación de los métodos cualitativos, y en especial, la investigación-acción es bastante reciente. Hasta finales de los años 90 e incluso en los primeros años del nuevo siglo no ha recibido mucha atención y menos una completa aceptación por parte de los investigadores en el área de sistemas de información (García 2004). Esta situación ha sido expuesta en los trabajos de (Avison 1999).

La investigación-acción es un método que requiere ser entendido y valorado en cuanto a su utilidad. Dos de las definiciones más significativas son las siguientes:

(French 1996): *"La investigación-acción es el proceso de recopilar de forma sistemática datos de la investigación acerca de un sistema actual en relación con algún objetivo, meta o necesidad de ese sistema; de alimentar de nuevo con esos datos al sistema; de emprender acciones por medio de variables alternativas seleccionadas dentro del sistema, basándose tanto en los datos como en la hipótesis; y de evaluar los resultados de las acciones, recopilando datos adicionales*."

(Wadsworth 1998): "*La investigación –acción consiste en la participación de todas las partes involucradas (actores) en la investigación, examinado la situación existente (que sienten como problemática) con los objetivos de cambiarla y mejorarla."*

(Kock 2001) indican que la investigación-acción tiene una doble finalidad:

- Generar un beneficio al cliente de la investigación
- Generar conocimiento relevante (de investigación).

Por lo tanto, la investigación-acción es una forma de investigar de carácter colaborativo en el que se busca unir teoría y práctica entre investigadores y practicantes mediante un proceso de naturaleza cíclica. Este tipo de investigación está orientada a la producción de nuevo conocimiento útil en la práctica que se obtiene mediante el cambio y/o búsqueda de soluciones a situaciones reales que le ocurren a un grupo de practicantes (Avison 1999). Esto se consigue gracias a la intervención de un investigador en la realidad del mencionado grupo y los resultados de esta experiencia deben ser beneficiosos tanto para el investigador como para los practicantes (García 2004).

VIII.2 Apéndice B: Métricas de calidad ISO/IEC 9126

Las métricas utilizadas para las características de calidad son (ISO/IEC 9126) (International Organization for Standardization 2003):

Tabla 80: Métricas para la evaluar las características de funcionalidad y usabilidad del producto de software

Característica	**Funcionalidad**			
Sub-característica	**Adecuación**			
Atributo	Medida Base	Medida derivada	Función de calculo	Descripción
Completud	NRNI= Número de requerimientos funcionales no implementados. NRF= Número de requerimientos funcionales especificados.	PC=Porcentaje de completud	PC=(1-NRNI / NRF)*100%	Porcentaje de requerimientos funcionales no implementados en la evaluación comparada con el número de requerimientos funcionales descritos en la especificación de requerimientos.
Correctud	NRII= Número de requerimientos implementados no correctas. NRFI=Número de requerimientos funcionales implementados.	PCR=Porcentaje de correctud	PCR= (1-NRII / NRFI)*100%	Porcentaje de requerimientos implementados incorrectamente comparados con el número de requerimientos funcionales descritos en la especificación de requerimientos.
Correctud en información de interfaces	NINC= Número de interfaces que sus datos no son correctos según las especificaciones. NII=Número de interfaces implementadas según las especificaciones.	PCII=Porcentaje de Correctud en información de interfaces	PCII=(1-NINC/NII)*100%	Porcentaje de interfaces de usuario que sus datos presentados han sido implementados incorrectamente comparados con el número de interfaces implementadas según las especificaciones.

Característica	**Usabilidad**			
Sub-característica	**Fácil comprensión**			
Atributo	Medida Base	Medida derivada	Función de calculo	Descripción
Comprensión	NUNC= Número de requerimientos de usabilidad que son comprendidas por el usuario. NUD= Número de requerimientos de usabilidad definidas en las especificaciones.	PCIU=Porcentaje de comprensión de interfaces de usuario.	PCIU =(NUNC / NUD)*100%	Porcentaje de requerimientos de usabilidad que son comprendidas por el usuario comparados con el número de requerimientos de usabilidad definidas en las especificaciones.

VIII.3 Apéndice C: Análisis de modos y efectos de falla (por sus siglas en inglés FMEA)

FMEA es un procedimiento de administración de operaciones para el análisis de modos de fallas potenciales dentro de un sistema para determinar el efecto de las fallas y clasificar por severidad. Se utiliza en varias fases de la vida del producto.

FMEA fue desarrollado por la NASA en Estados Unidos para el proyecto de Apolo. En la industria del automóvil es un procedimiento estándar para la planeación y el desarrollo. En otras áreas de la industria FMEA se utiliza como una metodología para la administración de la calidad.(Oliver Mäckel)

El valor de la técnica se caracteriza por la simplicidad y la eficiencia de la técnica, además es recomendada como un estándar para el desarrollo de sistemas críticos.

La literatura ofrece varias definiciones de lo que es modo de fallas. Según la industria del Automóvil, un modo de falla es "la forma en que un producto o un proceso falla afectando su principal función". Algunas fuentes definen que el modo de falla es una descripción de una serie de causas-efectos no deseados. Otros definen que el modo de fallas es un enlace de una serie de causas-efectos. Para evitar confusiones, se introduce el término "escenario de fracaso" para describir una secuencia no deseado causas y efectos. Este término se aplica especialmente en los defectos que afectan al cliente y pueden ser potenciales o actuales. (Steven Kmenta 1999)

Otras definiciones de esta técnica son (Langford 1995):

- Análisis de efectos se refiere a estudiar la consecuencia de las fallas.
- Modo de falla es la forma por la cual un defecto es observado. Generalmente se describe como la forma que ocurra la falla.
- Efectos de falla son las consecuencias inmediatas de las fallas sobre operación, funcionalidad o estatus de algún elemento.
- Nivel identificador es un identificador para el nivel de complejidad.
- Causa de la falta. Defectos en diseño, el proceso, la calidad, o el uso de la parte, que son la causa subyacente de la falta o que inician un proceso que conduzca a la falta.

- Severidad considera la consecuencia potencial peor de una falta, determinada por el grado de lesión, de daños materiales, o de daño del sistema que podría ocurrir en última instancia.
- Defecto es una **imperfección en alguien o algo**, también se define como la **carencia de alguna cualidad propia de algo**. El concepto se utiliza como sinónimo de **error**, **fallo** o **desperfecto**. Los defectos pueden ser perceptibles por los sentidos, advertirse en el funcionamiento de alguna cosa o estar vinculados a algo más simbólico o subjetivo.

La siguiente Tabla describe las tres principales fases de FMEA(Steven Kmenta 1999):

Tabla 81: Fases de FMEA

Fase	Pregunta	Salida
Identificar	¿Qué puede salir mal?	Fallas: causas y efectos
Análisis	¿Qué tan probable es un fracaso y cuáles son las consecuencias?	Evaluación de la prioridad de riesgos
Actuar	¿Cómo se puede eliminar la causa o corregir la gravedad?	Soluciones de diseño, planes de prueba, cambios de manufactura y Correctud de error.

Tipos de FMEA

Diseño: Se usa para analizar componentes de diseños. Se enfoca hacia los Modos de Falla asociados con la funcionalidad de un componente, causados por el diseño.

Proceso: Se usa para analizar los procesos de manufactura y ensamble. Se enfoca a la incapacidad para producir el requerimiento que se pretende, un defecto. Los Modos de Falla pueden derivar de causas identificadas en el FMEA de Diseño.

A continuación se describe el procedimiento para la elaboración de FMEA para diseño y proceso (Steven Kmenta 1999):

1. Determinar el proceso o producto a analizar.
    - FMEA para diseño (FMAD): Enumerar que es lo que se espera del diseño del producto, que es lo que quiere y necesita el cliente, y cuáles son los requerimientos de producción. Así mismo listar el flujo que seguirá el

producto a diseñar, comenzando desde el abastecimiento de materia prima, el(los) procesos (s) de producción hasta la utilización del producto por el usuario final. Determinar las áreas que sean más sensibles a posibles fallas.

- FMEA para procesos (FMEAP): Listar el flujo del proceso que se esté desarrollando, comenzando desde el abastecimiento de la materia prima, el proceso de transformación hasta la entrega al cliente (proceso siguiente). Determinar las áreas que sean más sensibles a posibles fallas. En el caso de empresas de servicios no hay materias primas, para estos casos se toman en cuenta las entradas del proceso.

2. Establecer los modos potenciales de falla.

   Para cada una de las áreas sensibles a fallas determinadas en el punto anterior se deben establecer los modos de falla posibles. Modo de falla es la manera en que podría presentarse una falla o defecto. Para determinarlas nos cuestionamos ¿De qué forma podría fallar la parte o proceso?

   Un modo de falla puede estar originado por una o más causas. Estas pueden ser independientes entre sí. También pueden combinarse entre ellas, es decir, que el modo de dallo está condicionado a que se presenten ambas. Y por último, puede que las causas estén encadenadas, es decir si una falla no se presenta si no aparece antes otra.

   Lo más importante es establecer la cadena de sucesos en el orden correcto para una mejor comprensión del problema y una adecuada valoración de los índices de ocurrencia, de los cuales se hablara más adelante (Stamatis 1995).

3. Determinar el efecto de la falla.

   Efecto: Cuando el modo de falla no se previene ni corrige, el cliente o el consumidor final pueden ser afectados.

4. Determinar la causa de la falla.

   Causa: Es una deficiencia que se genera en el Modo de Falla.

5. Describir las condiciones actuales.

   Anotar los controles actuales que estén dirigidos a prevenir o detectar la causa de la falla.

- Evitar o eliminar causas de falla.
- Identificar o detectar falla anticipadamente.
- Reducir impactos / consecuencias de falla.

6. Determinar el grado de severidad: Para estimar el grado de severidad, se debe de tomar en cuenta el efecto de la falla en el cliente. Se utiliza una escala del 1 al 10: el '1' indica una consecuencia sin efecto. El 10 indica una consecuencia grave.
7. Determinar el grado de ocurrencia: Es necesario estimar el grado de ocurrencia de la causa de la falla potencial. Se utiliza una escala de evaluación del 1 al 10. El "1" indica remota probabilidad de ocurrencia, el "10" indica muy alta probabilidad de ocurrencia.
8. Determinar el grado de detección: Se estimará la probabilidad de que el modo de falla potencial sea detectado antes de que llegue al cliente. El '1' indicará alta probabilidad de que la falla se pueda detectar. El '10' indica que es improbable ser detectada.
9. Calcular el número de prioridad de riesgo (NPR): Es un valor que establece una jerarquización de los problemas a través de la multiplicación del grado de ocurrencia, severidad y detección, éste provee la prioridad con la que debe de atacarse cada modo de falla, identificando ítems críticos.

   NPR = Grado de Ocurrencia * Severidad * Detección.

   Prioridad de NPR:

   - 500 – 1000 Alto riesgo de falla
   - 125 – 499 Riesgo de falla medio
   - 1 – 124 Riesgo de falla bajo
   - No existe riesgo de falla

Se deben atacar los problemas con NPR alto, así como aquellos que tengan un alto grado de ocurrencia no importando si el NPR es alto o bajo.

10. Acciones recomendadas: Anotar la descripción de las acciones preventivas o correctivas recomendadas, incluyendo responsables de las mismas. Anotando la fecha compromiso de implantación. Se pueden recomendar acciones encaminadas hacia:

- Eliminar o disminuir la OCURRENCIA de la causa del modo de falla. (modificaciones al diseño o al proceso, Implementación de métodos estadísticos, ajuste a herramental, etc.
- Reducir la SEVERIDAD del modo de falla. (Modificaciones en el diseño del producto o proceso).
- Incrementar la probabilidad de DETECCIÓN. (Modificaciones en el diseño del producto o proceso para ayudar a la detección).

11. Una vez realizadas las acciones correctivas o preventivas, se recalcula el grado de ocurrencia, severidad, detección y el NPR.

Cada vez que haya alguna modificación en el proceso o en el producto se debe de actualizar FMEA.

Una vez que el equipo de desarrollo identifica y prioriza los escenarios de fracaso, pueden tomar decisiones para mejorar la confiabilidad, calidad y seguridad del producto o del proceso.

Para aplicar el procedimiento se debe de llenar una plantilla con la siguiente información. Solo se describirán los elementos de la plantilla que se utilizaron para los casos de estudio (Stamatis 1995).

- **Nombre del producto/proceso:** se escribe el nombre del producto sobre el que se va a aplicar.
- **Modo de fallo:** Un modo de fallo significa que un elemento o sistema no satisface o no funciona de acuerdo con la especificación o simplemente no se obtiene lo que se espera de él. El fallo es una desviación o defecto de una función o especificación. Con esa definición, un fallo puede no ser inmediatamente detectable por el cliente pero se considera como tal.
- **Efectos de fallo:** Se describe los efectos del fallo tal como lo haría el cliente. Los efectos corresponden a los síntomas. Si un modo de fallo tiene muchos efectos, a la hora de evaluar, se elige el más grave. Los efectos típicos de fallo pueden ser por diseño y proceso.

- **Grado de la severidad:** este índice de gravedad valora el nivel de las consecuencias sentidas por el cliente.
- **Causas del fallo:** se describe las causas potenciales de fallo atribuibles a cada modo de fallo. La causa potencial de fallo se define como indicio de una debilidad del diseño o proceso cuya consecuencia es el modo de fallo. Las causas relacionadas deben ser lo más concisas y completas posibles.
- **Grado de ocurrencia**: es la probabilidad de que una causa específica se produzca y dé lugar al modo de fallo. El índice de la ocurrencia representa más bien un valor intuitivo más que un dato estadístico matemático, a no ser que se dispongan de datos históricos de fiabilidad.

VIII.4 Apéndice D: SCRUM

SCRUM fue desarrollado por Ken Schwaber y Jeff Sutherlan. Es una buena forma de iniciar métodos ágiles en ambientes d desarrollo disciplinados (Bohem B 2003).

Este método ágil se basa en equipos auto-dirigidos, con mediciones diarias y sin método prescriptivo (C. 2004; Larman 2004). No es propiamente un método o metodología de desarrollo e implantarlo como tal resulta insuficiente. SCRUM define métodos de gestión y control para complementar la aplicación de otros métodos ágiles como XP que, centrados en prácticas de tipo técnico, carecen de ellas. (Larman 2004)

SCRUM se guía en los principios ágiles (Larman 2004):

- Equipos auto-dirigidos y organizados.
- No se agrega trabajo extra a una iteración.
- Reunión diaria corta con preguntas específicas.
- Iteraciones de máximo 30 días.
- Demostraciones a los clientes al final de cada iteración.
- Cada iteración se planea según las necesidades del cliente.
- Su énfasis es empírico, no es un proceso definido.
- Es un método flexible en cuanto a lo ceremonial, qué productos generar, de qué formalidad.
- Se enfatiza la administración del proyecto.
- Se trabaja en un cuarto o espacio para cada proyecto.
- No se puede añadir trabajo al equipo en las iteraciones. Si algo falta, se puede dejar a otra iteración o quitar algo.
- Lo que se reporta en una reunión, se remueve antes de la siguiente.

Al final de cada iteración se hace una reunión de revisión (de máximo 4 horas) dirigida por el Scrum master, se muestra el producto y se informa al dueño las funciones que incluye, diseño, bondades, debilidades y posibles problemas (Pete Deemer 2007).

Se hace la retroalimentación y lluvia de ideas para la próxima iteración pero sin compromisos.

En la sesión de planeación del sprint es donde se establecen los compromisos del equipo y otros involucrados para el sprint (Pete Deemer 2007).

Antes de iniciar cada iteración, el equipo revisa las tareas pendientes y selecciona la parte que entregará como un incremento de funcionalidad al finalizar la iteración (Sprint) (Pete Deemer 2007).

El equipo debe revisar los requisitos, considerar la tecnología a utilizar, evaluar su conocimiento y en forma colectiva determinar la forma en la que implementara la funcionalidad, ver figura 34.

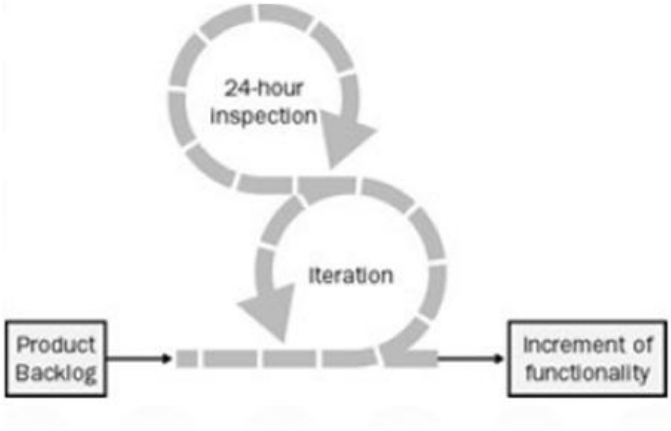

Figura 34: Iteración de SCRUM

SCRUM utiliza los siguientes roles (Larman 2004) :

- Product Owner: Representa a todos los interesados en el producto final. Marca las prioridades del producto y lleva el control de las estimaciones.
- Scrum Master: Responsable del proceso de Scrum. Se encarga de la incorporación de Scrum en la cultura de la organización. Asegura el cumplimiento de los roles y responsabilidades. Formación y entrenamiento en el proceso.
- Scrum Team: Debe transformar las tareas del Sprint Backlog en un incremento de funcionalidad en el software. Desarrollar el producto con

calidad, auto-gestionado, auto-organizado, multi-funcional. No mayor a ocho elementos.

SCRUM define una pequeña cantidad de artefactos para el seguimiento del proyecto y control de las actividades asociadas al sprint (Larman 2004).

- Sprint Backlog: Lista de tareas (realistas) extraídas del Product Backlog que serán convertidas en un incremento de funcionalidad.

  Es recomendable que las tareas tengan una duración entre 4 y 16 hrs., en caso de tareas mayores deben intentar descomponerse en sub-tareas de ese rango de tiempo.
- Product Backlog: Es el listado con los requisitos del sistema. Es un documento dinámico que incorpora constantemente las necesidades del sistema. Se mantiene durante todo el ciclo de vida.
- Gráfica de progreso: Gráfica del avance del sprint backlog que muestra el avance diario y lo que falta.

Reunión de retrospectiva

Los que acuden a la reunión de retrospectiva es el Srcum Master, el equipo de trabajo y en ocasiones el propietario del Producto.

Para esta reunión se plantean las siguientes preguntas, y todos los miembros del equipo responden a estas:

¿Qué cosas fueron bien en el último sprint?

¿Qué cosas se podrían mejorar?

El Scrum Master anota todas las respuestas y el equipo prioriza las mejoras posibles y no proporciona respuestas, sino que ayuda al equipo a encontrar la mejor forma de trabajar con SCRUM. Las acciones de mejora localizadas que se puedan implementar en el próximo Sprint deben introducirse en el Backlog como elementos no funcionales.

VIII.5 Apéndice E: Plantilla de retrospectiva SCRUM

Fecha:	**Realizado por:**
Numero de Sprint:	**Objetivo del Sprint:**
Hora inicio de la reunión Retrospectiva:	
Hora Fin de la reunión Retrospectiva:	

1. Introducción

2. Información General del Sprint(Iteración)

3. Métricas de Calidad

NOTA: Ir agregado la información de cada Sprint finalizado

Sprint	#Casos de Prueba	# Defectos encontrados	# Defectos Corregidos	#Esfuerzo (Hrs/h)
Sprint 1				
Sprint 2				
Sprint 3				
Sprint 4				
Sprint 5				
Sprint 6				
Total				

4. Revisión del Proceso

4.1 Procesos que fueron más efectivos en el Sprint

# Votos	Actividades que se hicieron Bien

4.2 Procesos que han sido de efecto negativo en el Sprint

#Votos	Necesidades de mejora

5. Acciones de mejora

Nota: Las acciones de mejora deberán ser introducidas inmediatamente en el siguiente Sprint.

6. Varianza

Sprint	Est. Hrs	Act. Hrs	Varianza	% Varianza
Sprint 1				
Sprint 2				
Sprint 3				
Sprint 4				
Sprint 5				
Sprint 6				
Promedio				

VIII.6 Apéndice F: Descripción del Proceso de Administración de Proyectos Específicos y del Proceso Desarrollo y Mantenimiento de Software de la Norma Mexicana NMX-I-059/03-NYCE-2005

La norma Mexica NMX-059/03-NYCE-2005, tiene como propósito principal fomentar la estandarización a través de la adopción de las mejores prácticas en gestión e ingeniería de software (NYCE 2005). Esta norma se basa en el Modelo de Procesos para la Industria de Software (MoProSoft), la cual fue creada con el objetivo de proporcionar a la industria de software en México, en su mayoría es pequeña y mediana, un modelo basado en las mejores prácticas internacionales en gestión e ingeniería de software. La adopción del modelo permitirá elevar la capacidad de las organizaciones para ofrecer servicios con calidad y alcanzar niveles internacionales de competitividad. (Oktaba Hanna 2005)

Este modelo de procesos define tres categorías las cuales cada una comprende un conjunto de procesos. La categoría de Operación tiene los procesos de Administración de Proyectos Específicos (APE) y Desarrollo y Mantenimiento de Software (DMS).

El Proceso de APE tiene como propósito establecer y llevar a cabo sistemáticamente las actividades que permitan cumplir con los objetivos de un proyecto en tiempo y costos esperados. Las actividades definidas para este proceso se describen en las siguientes tablas:

Tabla 82: Tareas de planificación de APE

Actividad	**A1. Planificación**
ID	Tarea
A1.1	Revisar con el Responsable de Gestión de Proyectos la Descripción del Proyecto
A1.2	Con base en la Descripción del Proyecto, definir el Proceso Específico del proyecto a partir del proceso de Desarrollo y Mantenimiento de Software de la organización o a partir del acuerdo establecido con el Cliente. Se considera el alcance, la magnitud y complejidad del proyecto.
A1.3	Definir conjuntamente con el Cliente el Protocolo de Entrega de cada uno de los entregables especificados en la Descripción del Proyecto.
A1.4	Identificar el número de ciclos y las actividades específicas que deben llevarse a cabo para producir los entregables y sus componentes identificados en la Descripción del Proyecto. Identificar las actividades específicas que deben llevarse a cabo para cumplir con los objetivos del proyecto, definir las actividades para llevar a cabo revisiones periódicas al producto o servicio que se está ofreciendo y para efectuar revisiones entre colegas. Identificar las actividades para llevar a cabo el Protocolo de Entrega. Documentar el resultado como Ciclos y Actividades.

A1.5	Identificar y documentar la relación y dependencia de cada una de las actividades.
A1.6	Establecer el Tiempo Estimado para desarrollar cada actividad considerando la información histórica y las Metas Cuantitativas para el Proyecto.
A1.7	Elaborar el Plan de Adquisiciones y Capacitación, definiendo las características y el calendario en cuanto a recursos humanos, materiales, equipo y herramientas, incluyendo la capacitación requerida para que el equipo de trabajo pueda desempeñar el proyecto.
A1.8	Conformar el Equipo de Trabajo, asignando roles y responsabilidades basándose en la Descripción del Proyecto.
A1.9	Asignar fechas de inicio y fin a cada una de las actividades para generar el Calendario de trabajo tomando en cuenta los recursos asignados, la secuencia y dependencia de las actividades.
A1.10	Evaluar y documentar el Costo Estimado del proyecto, tomando en cuenta las Metas Cuantitativas para el Proyecto.
A1.11	Identificar, describir y evaluar los riesgos que pueden afectar el proyecto, que contemple riesgos relacionados con el equipo de trabajo incluyendo al Cliente y a los usuarios, riesgos con la tecnología o la metodología, riesgos con la organización del proyecto (costo, tiempo, alcance y recursos) o riesgos externos al proyecto. Identificar la probabilidad e impacto de cada riesgo estimando sus implicaciones en los objetivos del proyecto (análisis cuantitativo). Priorizar los efectos de los riesgos sobre los objetivos del proyecto (análisis cualitativo). Desarrollar procedimientos para reducir el impacto de los riesgos. Documentar en el Plan de Manejo de Riesgos o actualizarlo.
A1.12	Generar el Plan del Proyecto o actualizarlo antes de iniciar un nuevo ciclo. Además el Plan del Proyecto se puede actualizar a causa de Solicitud de Cambios por parte del Cliente, Acciones Correctivas o Preventivas provenientes de Gestión de Proyectos o Acciones Correctivas de este proceso.
A1.13	Generar el Plan de Desarrollo en función del Plan del Proyecto o actualizarlo antes de iniciar un nuevo ciclo. Además el Plan de Desarrollo se puede actualizar a causa de Solicitud de Cambios por parte del Cliente, Acciones Correctivas o Preventivas provenientes de Gestión de Proyectos o Acciones Correctivas de este proceso.
A1.14	Verificar el Plan del Proyecto y el Plan de Desarrollo (Ver1).
A1.15	Corregir los defectos encontrados en el Plan del Proyecto y en el Plan de Desarrollo con base en el Reporte de Verificación y obtener la aprobación de las correcciones.
A1.16	Validar el Plan del Proyecto y el Plan de Desarrollo (Val1).
A1.17	Corregir los defectos encontrados en el Plan del Proyecto y Plan de Desarrollo con base en el Reporte de Validación y obtener la aprobación de las correcciones.
A1.18	Dar inicio formal a un nuevo ciclo una vez que se haya asegurado el cumplimiento de las condiciones iniciales del ciclo.

Tabla 83: Tareas de realización de APE

Actividad	**A2. Realización**
ID	Tarea
A2.1	Acordar con el Responsable de Desarrollo y Mantenimiento del proyecto la asignación de tareas al Equipo de Trabajo incluyendo a los subcontratistas.
A2.2	Acordar la distribución de la información necesaria al equipo de trabajo con base en el Plan de Comunicación e Implantación.
A2.3	Revisar con el Responsable de Desarrollo y Mantenimiento del proyecto la Descripción del Producto, el Equipo de Trabajo y Calendario.
A2.4	Dar seguimiento al Plan de Adquisiciones y Capacitación. Aceptar o rechazar la Asignación de Recursos humanos o subcontratistas. Distribuir los recursos a los

	miembros del equipo para que puedan llevar a cabo las actividades.
A2.5	Manejar la relación con subcontratistas que implica planificar, revisar y auditar las actividades, asegurando la calidad de los productos o servicios contratados y el cumplimiento con los estándares y especificaciones acordadas.
A2.6	Recopilar y analizar los Reportes de Actividades, Reportes de Mediciones y Sugerencias de Mejora y productos de trabajo.
A2.7	Registrar los costos y recursos reales del ciclo.
A2.8	Revisar el Registro de Rastreo de los requerimientos del usuario a través del ciclo.
A2.9	Revisar los productos generados durante el ciclo, que forman parte de la Configuración de Software.
A2.10	Recibir y analizar las Solicitudes de Cambios e incorporar los cambios aprobados en el Plan del Proyecto y en el Plan de Desarrollo. En caso de cambios a requerimientos se incorporan al inicio de un nuevo ciclo.
A2.11	Conduce reuniones de revisión con el equipo de trabajo y con el Cliente, generando Minutas con puntos tratados y acuerdos tomados.

Tabla 84: Tareas de realización de APE

Actividad	**A3. Evaluación y Control**
ID	Tarea
A3.1	Evaluar el cumplimiento del Plan del Proyecto y el Plan de Desarrollo, con respecto al alcance, costo, calendario, equipo de trabajo, proceso y se establecen Acciones Correctivas.
A3.2	Dar seguimiento y controlar el Plan de Manejo de Riesgos. Identificar nuevos riesgos y actualizar el plan.
A3.3	Generar el Reporte de Seguimiento del proyecto, considerando los Reportes de Actividades.

Tabla 85: Tareas de cierre de APE

Actividad	**A4.Cierre**
ID	Tarea
A4.1	Formalizar la terminación del ciclo o del proyecto de acuerdo al Protocolo de Entrega establecido en el Plan del Proyecto y obtener el Documento de Aceptación.
A4.2	Efectuar el cierre con subcontratistas de acuerdo al contrato establecido.
A4.3	Generar el Reporte de Mediciones y Sugerencias de Mejora de este proceso, de acuerdo al Plan de Mediciones de Procesos.
A4.4	Identificar las Lecciones Aprendidas e integrarlas a la Base de Conocimiento. Como ejemplo, se pueden considerar mejores prácticas, experiencias exitosas de manejo de riesgos problemas recurrentes, entre otras.

El Proceso de DMS tiene como propósito la realización sistemática de las actividades de obtención de requisitos, análisis, diseño, construcción, integración y pruebas de productos de software cumpliendo con los requisitos especificados.

Las actividades definidas para este proceso se describen en las siguientes tablas:

Tabla 86: Tareas de realización de la fase de inicio de DMS

Actividad	**A1. Realización de la fase de inicio**
ID	Tarea
A1.1	Revisar con los miembros del equipo de trabajo el Plan de Desarrollo actual para lograr un entendimiento común y obtener su compromiso con el proyecto.
A1.2	Elaborar el Reporte de Actividades registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas.

Tabla 87: Tareas de realización de la fase de requisitos de DMS

Actividad	**A2. Realización de la fase de requisitos**
ID	Tarea
A2.1	Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al Plan de Desarrollo actual.
A2.2	Documentar o modificar la Especificación de Requerimientos. • Identificar y consultar fuentes de información (clientes, usuarios, sistemas previos, documentos, etc.) para obtener nuevos requerimientos. • Analizar los requerimientos identificados para delimitar el alcance y su factibilidad, considerando las restricciones del ambiente del negocio del cliente o del proyecto. • Elaborar o modificar el prototipo de la interfaz con el usuario. • Generar o actualizar la Especificación de Requerimientos.
A2.3	Verificar la Especificación de Requerimientos (Ver1).
A2.4	Corregir los defectos encontrados en la Especificación de Requerimientos con base en el Reporte de Verificación y obtener la aprobación de las correcciones.
A2.5	Validar la Especificación de Requerimientos (Val1).
A2.6	Corregir los defectos encontrados en la Especificación de Requerimientos con base en el Reporte de Validación y obtener la aprobación de las correcciones.
A2.7	Elaborar o modificar Plan de Pruebas de Sistema.
A2.8	Verificar el Plan de Pruebas de Sistema (Ver2).
A2.9	Corregir los defectos encontrados en el Plan de Pruebas de Sistema con base en el Reporte de Verificación y obtener la aprobación de las correcciones.
A2.10	Documentar la versión preliminar del Manual de Usuario o modificar el manual existente.
A2.11	Verificar el Manual de Usuario (Ver3).
A2.12	Corregir los defectos encontrados en el Manual de Usuario con base en el Reporte de Verificación y obtener la aprobación de las correcciones.
A2.13	Incorporar Especificación de Requerimientos, Plan de Pruebas de Sistema y Manual de Usuario como líneas base a la Configuración de Software.
A2.14	Elaborar el Reporte de Actividades registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas.

Tabla 88: Tareas de realización de la fase de análisis y diseño de DMS

Actividad	**A3. Realización de la fase de análisis y diseño**
ID	Tarea
A3.1	Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al Plan de Desarrollo actual.
A3.2	Documentar o modificar el Análisis y Diseño: • Analizar la Especificación de Requerimientos para generar la descripción de la estructura interna del sistema y su descomposición en subsistemas, y éstos a su vez en componentes, definiendo las interfaces entre ellos. • Describir el detalle de la apariencia y el comportamiento de la interfaz con base en la Especificación de Requerimientos de forma que se puedan prever los recursos para su implementación. • Describir el detalle de los componentes que permita su construcción de manera evidente. • Generar o actualizar el Análisis y Diseño. • Generar o modificar el Registro de Rastreo.
A3.3	Verificar el Análisis y Diseño y el Registro de Rastreo (Ver4).
A3.4	Corregir los defectos encontrados en el Análisis y Diseño y en el Registro de Rastreo con base en el Reporte de Verificación y obtener la aprobación de las correcciones.
A3.5	Validar el Análisis y Diseño (Val2).
A3.6	Corregir los defectos encontrados en el Análisis y Diseño con base en el Reporte de Validación y obtener la aprobación de las correcciones.
A3.7	Elaborar o modificar Plan de Pruebas de Integración.
A3.8	Verificar el Plan de Pruebas de Integración (Ver5).
A3.9	Corregir los defectos encontrados en el Plan de Pruebas de Integración con base en el Reporte de Verificación y obtener la aprobación de las correcciones.
A3.10	Incorporar Análisis y Diseño, Registro de Rastreo y Plan de Pruebas de Integración como líneas base a la Configuración de Software.
A3.11	Elaborar el Reporte de Actividades registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas.

Tabla 89: Tareas de realización de la fase de construcción de DMS

Actividad	**A4. Realización de la fase de construcción**
ID	Tarea
A4.1	Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al Plan de Desarrollo actual.
A4.2	Construir o modificar el(los) Componente(s) de software: • Implementar o modificar Componente(s) con base a la parte detallada del Análisis y Diseño. • Definir y aplicar pruebas unitarias para verificar que el funcionamiento de cada componente esté acorde con la parte detallada del Análisis y Diseño. • Corregir los defectos encontrados hasta lograr pruebas unitarias exitosas (sin defectos). • Actualizar el Registro de Rastreo, incorporando los componentes construidos o modificados.
A4.3	Verificar el Registro de Rastreo (Ver6).
A4.4	Corregir los defectos encontrados en el Registro de Rastreo con base en el Reporte de Verificación y obtener la aprobación de las correcciones.
A4.5	Incorporar Componentes y Registro de Rastreo como líneas base a la Configuración de Software.
A4.6	Elaborar el Reporte de Actividades, registrando las actividades realizadas, fechas

	de inicio y fin, responsable por actividad y mediciones requeridas.

Tabla 90: Tareas de realización de la fase de integración y pruebas de DMS

Actividad	**A5. Realización de la fase de integración y pruebas**
ID	Tarea
A5.1	Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al plan de desarrollo actual.
A5.2	Realizar integración y pruebas. • Integrar los componentes en subsistemas o en el sistema del Software y aplicar las pruebas siguiendo el Plan de Pruebas de Integración, documentando los resultados en un Reporte de Pruebas de Integración. • Corregir los defectos encontrados, con base en Reporte de Pruebas de Integración, hasta lograr una prueba de integración exitosa (sin defectos). • Actualizar el Registro de Rastreo.
A5.3	Documentar el Manual de Operación o modificar el manual existente.
A5.4	Verificar el Manual de Operación (Ver7).
A5.5	Corregir los defectos encontrados en el Manual de Operación con base en el Reporte de Verificación y obtener la aprobación de las correcciones.
A5.6	Realizar las pruebas de sistema siguiendo el Plan de Pruebas de Sistema, documentando los resultados en un Reporte de Pruebas de Sistema
A5.7	Corregir los defectos encontrados en las pruebas de sistema con base en el Reporte de Pruebas de Sistema y obtener la aprobación de las correcciones.
A5.8	Documentar el Manual de Usuario o modificar el existente.
A5.9	Verificar el Manual de Usuario (Ver8).
A5.10	Corregir los defectos encontrados en el Manual de Usuario con base en el Reporte de Verificación y obtener la aprobación de las correcciones.
A5.11	Incorporar Software, Incorporar Software, Manual de Operación y Manual de Usuario como líneas base a la Configuración de Software.
A5.12	Elaborar el Reporte de Actividades registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas.

Tabla 91: Tareas de realización de la fase de cierre de DMS

Actividad	**A6. Realización de la fase de cierre**
ID	Tarea
A6.1	Documentar el manual de mantenimiento o modificar el existente.
A6.2	Verificar el manual de mantenimiento (Ver9).
A6.3	Corregir los defectos encontrados en el manual de mantenimiento con base en el reporte de verificación y obtener la aprobación de las correcciones.
A6.4	Incorporar manual de mantenimiento como línea base a la configuración de software.
A6.5	Identificar las Lecciones aprendidas e integrarlas a la base de conocimiento. Como ejemplo, se pueden considerar mejores prácticas, experiencias exitosas de manejo de riesgos, problemas recurrentes, entre otras.
A6.6	Generar el reporte de mediciones y sugerencias de mejora.
A6.7	Elaborar el reporte de actividades registrando las actividades realizadas, fechas de inicio y fin, responsable por actividad y mediciones requeridas.