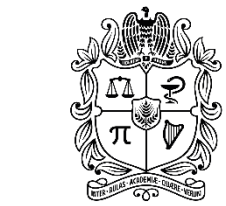


UNIVERSIDAD
NACIONAL
DE COLOMBIA

Modelo de gestión de conocimiento aplicado a las pruebas de software

Dario Enrique Soto Duran

Universidad Nacional de Colombia

Facultad de Minas

Medellín, Colombia

2017

Modelo de gestión de conocimiento aplicado a las pruebas de software

Dario Enrique Soto Duran

Tesis presentada como requisito parcial para optar al título de:

Doctor en Ingeniería de Sistema e Informática

Director:

Ph.D. Jovani Alberto Jiménez Builes

Universidad Nacional de Colombia

Facultad de Minas

Medellín, Colombia

2017

(Dedicatoria o lema)

A mis padres por cultivar la semilla del esfuerzo y el amor al estudio, a mi esposa por el apoyo incondicional para construir y compartir los éxitos de nuestra vida, a mis hijos por enseñarme el valor del amor y a mis hermanas y sobrino por su cariño y a todas las personas que de una u otra forma contribuyeron en el logro de este trabajo.

"Yo nunca pierdo. Yo gano o aprendo"

Nelson Mandela.

Agradecimientos

El autor expresa sus agradecimientos a:

Primero gracias a Dios y a María Auxiliadora, por regalarme la vida y en ella, a mi familia que son mi tesoro y fuente de inspiración.
A mi padre por enseñarme los principales valores de la vida, por su apoyo incondicional y el esfuerzo para impulsar mi vida.
A mi madre por su comprensión y amor para enseñarme que la felicidad de la vida se encuentra en el compromiso y la responsabilidad.
A mis hermanas y sobrino, por mantener su corazón junto al mío y encomendarme en sus oraciones.
A mi esposa por compartir sus sueños y devolverme la motivación en los momentos más difíciles.
A mis dos hermosos hijos por iluminarme con sus sonrisas y enseñarme a disfrutar la vida convirtiendo mi esfuerzo en felicidad.
Al maestro Jovani Alberto Jiménez Builes, por sus valiosos y oportunos aportes para alcanzar la meta propuesta, depositando su confianza, guía y tiempo en el fruto de mi esfuerzo y dedicación.
Al Tecnológico de Antioquia I.U., por contribuir a mi crecimiento personal y profesional en la construcción de este gran sueño académico y así continuar aportando en el camino de la calidad y la excelencia de esta bella institución.
Finalmente, un agradecimiento infinito a todos mis amigos y familiares por su presencia en los momentos más importantes de mi vida.
.

Resumen

Las pruebas de software son una actividad intensiva en conocimiento y relevante para asegurar la calidad del producto de software. Por otro lado, la gestión de conocimiento es una disciplina transversal en las organizaciones que contribuye al aprendizaje organizacional y a la mejora continua de los procesos. A nivel académico y empresarial, existen varias iniciativas para articular la gestión del conocimiento a las pruebas de software, siendo la adopción de estas prácticas costosas y con largos periodos de implementación, y en algunos casos, el esfuerzo invertido resulta infructuoso. Partiendo que el individuo es el principal dinamizador de cualquier estrategia para gestionar el conocimiento, las organizaciones intensivas en conocimiento deben ser analizadas como un sistema multicapa que integran tres niveles individual, grupal y organizacional.

En consecuencia, en esta tesis se presenta un modelo de gestión de conocimiento que articula las prácticas y actividades de las pruebas de software desde una estrategia individual basada en los principios de mejora continua y aprendizaje organizacional.

El modelo se soporta en la definición de un proceso de prueba personal basado en conocimiento (PTPk) que instancia la estructura e intencionalidad del proceso de software personal (PSP) e integra prácticas de calidad a través del aprendizaje basado en experiencias que inciden en la cultura y la conducta del profesional de pruebas de software.

Palabras clave: Pruebas de software, Gestión de Conocimiento, Proceso de pruebas Individual, Aprendizaje Individual, mejora de procesos de pruebas.

Abstract

Software testing is a knowledge-intensive and relevant activity to ensure the quality of the software product. On the other hand, knowledge management is a transversal discipline in organizations that contribute to organizational learning and the continuous improvement of processes. At academic and business level, there are several initiatives for the design of knowledge management to software testing, the adoption of these costly practices and with long implementation periods, and in some cases, the effort invested is unsuccessful. Starting from the fact that the individual is the main driver of any strategy to manage for knowledge, knowledge-intensive knowledge organizations should be analyzed as a multidisciplinary system that integrates three levels, individual, group and organizational.

Consequently, this thesis presents a knowledge management model that articulates the practices and activities of software testing from an individual strategy based on the principles of continuous improvement and organizational learning.

The model is supported in the definition of a personal testing process (PTPk) that instantiates the structure and intentionality of the personal process software (PSP) and integrates quality practices through learning based on experiences that affect culture and behavior a professional software tester.

Keywords: Software testing, Knowledge Management, Individual testing process, Individual Learning, improvement of testing processes.

Contenido

Pág.

1 Capítulo: Introducción 1
1.1 Motivación 1
1.2 Planteamiento del Problema 2
1.3 Pregunta de Investigación 6
1.4 Hipótesis 6
1.5 Objetivos 7
1.5.1 Objetivo General 7
1.5.2 Objetivos Específicos: 7
1.6 Contribución 7

2 Capítulo: Marco Teórico y Revisión de Literatura 9
2.1 Marco Teórico. 9
2.1.1 Conocimiento 9
2.1.2 Taxonomía del conocimiento 10
2.1.3 Gestión de Conocimiento 12
2.1.4 Marco de Procesos de la KM 14
2.1.5 Enfoques de KM 15
2.1.6 Aprendizaje Organizacional 17
2.1.7 Mejora de procesos de software 23
2.1.8 Pruebas de software 27
2.2 Revisión de Literatura 36
2.2.1 Factores que inciden en las pruebas de software 36
2.2.2 Experiencias de gestión de conocimiento aplicadas a las pruebas de software 41
2.3 Análisis y discusión. 43

3 Capítulo: Gestión de Conocimiento en el marco de la mejora de procesos 49
3.1 La gestión de conocimiento y la mejora de procesos 49
3.2 Integración de Conocimiento y aprendizaje en la mejora de proceso 51
3.2.1 Conocimiento experiencial 51
3.2.2 Conocimiento SECI 52
3.2.3 Aprendizaje organizacional desde el enfoque individual 52
3.2.4 Estrategia de mejora 53
3.2.5 Visión Integradora del aprendizaje Individual y la mejora de procesos 54
3.3 Caracterización de los modelos de mejora aplicadas a las pruebas de software 55

4 Capítulo: Modelo y Validación 59
4.1 Modelo PTPk 60

4.1.1 Marco de Conocimiento 60
4.1.2 Estructura del modelo 64
4.1.3 Fases del modelo 68
4.1.4 Método de estimación 70
4.2 Validación 75

5 Capitulo: Contribuciones de la Tesis 83
5.1 Artículos en Revistas Internacionales 84
5.2 Artículos en Revistas Nacionales 84
5.3 Trabajos Completos en Eventos Internacionales 84
5.4 Pasantías Nacionales e Internacionales 84
5.5 Trabajos Completos en Eventos Nacionales 85
5.6 Capítulos de Libro 85
5.7 Orientaciones 85
5.8 Proyectos y Otros. 86

6 Conclusiones y trabajos futuros 87
6.1 Conclusiones 87
6.2 Trabajos futuros 89

7 Anexos. 90

8 Bibliografía 91

Lista de figuras

Figura 1. Modelo SECI 17
Figura 2. Modelo experiencial de Kolb (Kolb,2014). 18
Figura 3. Aprendizaje Basado en bucles 19
Figura 4. Aprendizaje Analítico y Sintético..... 20
Figura 5. Modelo de Organización que Aprende..... 20
Figura 6. Principios de Mejora Continua. 24
Figura 7. Relación de PSP/TSP con CMMI. 25
Figura 8. Niveles de Pruebas de Software..... 29
Figura 9. Técnicas de Prueba..... 30
Figura 10. Elementos de un caso de prueba. 31
Figura 11. Principios de Mejora Continua..... 53
Figura 12. Niveles de abstracción del conocimiento 61
Figura 13. Estructura por niveles del modelo..... 65
Figura 14. Fases del Modelo 68

Lista de tablas

Tabla 1. Procesos de la Gestión de Conocimiento. 14
Tabla 2. Capacidades de KM 16
Tabla 3. Criterios para las técnicas de estimación de pruebas 32
Tabla 4. Valoración de Complejidad de casos de prueba 34
Tabla 5. Pesos en TCPs por Complejidad 34
Tabla 6. Pesos de Ajuste 35
Tabla 7. Pesos de Ajuste 35
Tabla 8. Pesos de Ajuste 36
Tabla 9. Estrategias mejora de procesos aplicadas al individuo. 55
Tabla 10. Modelos de mejora vs. Escuelas de KM 56
Tabla 11. Procesos de Calidad vs. Capacidades de KM. 57
Tabla 12. Área de Gestión del proceso 62
Tabla 13. Área de Gestión de proyectos 63
Tabla 14. Área de Gestión de pruebas 63
Tabla 15. Mejora de procesos de pruebas 64
Tabla 16. Métricas 70
Tabla 17. Puntos de Casos de Prueba 71
Tabla 18. Esfuerzo expresado TCP por proyecto ejecutado. 72
Tabla 19. Histórico de proyectos 72
Tabla 20. Esfuerzo total del proyecto expresado en TCPs 73
Tabla 21. Datos obtenidos del Nivel 0 77
Tabla 22. Datos obtenidos del Nivel 0.1 79
Tabla 23. Datos obtenidos del Nivel 1. 80
Tabla 24. Datos obtenidos del Nivel 2. 81

1 Capítulo: Introducción

El propósito de este capítulo es suministrar una visión general de la tesis presentando la motivación, el problema, la pregunta de investigación, la hipótesis, los objetivos, y finalmente exponiendo las contribuciones que brinda el desarrollo de este trabajo.

1.1 Motivación

El alto costo de las actividades de verificación y validación en general, y de pruebas en particular en el desarrollo y mantenimiento de un producto de software, se constituye en uno de los factores críticos de la industria del software. Sin incluir los costos organizacionales diferentes autores, establecen que el costo de pruebas de software supone entre un 30% y un 50% del costo total del producto, Sin embargo, en muchas ocasiones el esfuerzo previsto para la realización de pruebas se disminuye debido a los retrasos imprevistos en el desarrollo de los proyectos, induce la liberación de productos con un número de defectos sustancialmente superior al razonable. Además de causas coyunturales de retraso en costo, tiempo y alcance, no siempre se percibe la necesidad de invertir en una correcta planificación, gestión y ejecución de las pruebas, tanto por parte de los gestores como de los desarrolladores y usuarios finales de los productos de software.

Por ello, las organizaciones tratan de llevar a la práctica iniciativas de mejora de sus procesos en el área de pruebas de software. Sin embargo, esto supone un gran esfuerzo, así como inversiones considerables en personal, tiempo y dinero. Una actividad clave para poner en marcha estas iniciativas, es la definición del proceso y la administración del conocimiento. Dada la importancia del proceso de pruebas de software, en la actualidad

las características de los equipos de pruebas son variadas, incrementando de formar sustancial las actividades de externalización u outsourcing bajo la denominación de factorías de software, donde el principal elemento es la gestión y transferencia de conocimiento, indispensable en las organizaciones que desean mantener los niveles requeridos de competitividad y productividad.

Hoy en día, las pruebas de software se ejecutan como proyectos paralelos al proceso de desarrollo, los cuales deben ser gestionados con las mejores prácticas para lograr el objetivo propuesto. Debido a la ausencia de un proceso de administración del conocimiento que responda a las necesidades del ámbito de las pruebas de software, se origina esta propuesta de investigación.

1.2 Planteamiento del Problema

Para las fábricas de software, cada proyecto se convierte en un reto y en cada uno de ellos se busca mejorar las prácticas y metodologías que utilizaron en proyectos anteriores.

Dentro del ciclo de vida de los proyectos de software, se desarrollan las pruebas como una actividad que busca garantizar que el producto desarrollado cumpla con las necesidades del cliente. Siendo, la calidad del producto valorada desde las especificaciones del producto e integra el trabajo en equipo para este propósito, como una comunidad de práctica. (Abdullah et al. 2011).

Las pruebas de software es un proceso basado en conocimiento, conjugando el conocimiento, tanto de negocio como metodológico con el objeto de generar una solución real a las necesidades del cliente. Sin embargo, el éxito del desarrollo de las pruebas de software depende en gran medida de las habilidades, conocimientos, intuición y experiencia de los miembros que integran el equipo de pruebas. (Kaner et al. 2002).

Para lograr un proceso de pruebas de software eficiente, se debe integrar eficazmente la gestión del conocimiento en el proceso de pruebas de software para que los activos de conocimiento se puedan transmitir y reutilizar por las organizaciones en el proceso de pruebas (Abdullah et al. 2011). Siendo indispensable el conocimiento y experiencia en

estas áreas para llevar cabo actividades de verificación y validación de productos software (Beer & Ramler 2008). El conocimiento juega un papel esencial en la aplicación de los métodos y técnicas de prueba, así como en el éxito de las mismas. (Vegas et al. 2006)

Los defectos son la principal causa de la baja calidad de los productos de software, así como del incremento del costo debido generalmente al tiempo y esfuerzo requerido para su corrección (Boehm 1987). Además, el costo asociado a resolver los defectos encontrados después de una release es exponencialmente mayor que solucionarlos mientras se construye el producto.

Las pruebas de software se basan en dos procedimientos, la verificación y la validación. La verificación consiste en hacer chequeos que certifiquen que el sistema funciona de acuerdo con las especificaciones y la validación consiste en asegurar que el sistema hace lo que se supone deba hacer, es decir que la especificación este correcta.

Las pruebas de software es un proceso determinado por un conjunto de actividades que permiten detectar defectos, con el propósito de solucionarlos por parte del equipo de desarrollo, convirtiéndose en un proceso iterativo hasta lograr una solución óptima y adecuada. Es por esto, que existe un reproceso tan evidente dentro de esta etapa del ciclo de vida de desarrollo ocasionado por múltiples causas.

Esta problemática se presenta en todos los proyectos de software, sin importar el tamaño, complejidad y organización; pero mientras más grande el proyecto puede llegar a ser más crítico y costoso, ya que cualquier solución implementada para corregir los defecto puede afectar otros artefactos del desarrollo, generando nuevos errores. Es en este punto es importante involucrar la gestión de conocimiento en el proceso de pruebas de software, ya que desde la perspectiva del analista de calidad se puede identificar situaciones previamente encontradas y las soluciones que se han aplicado a ellas. (Perona & Velásquez, 2012).

Es claro que el conocimiento de la aplicación debe ser compartido entre los miembros que participan en el desarrollo del proyecto, sin importar si hacen parte o no del grupo de implementación. Su interpretación debe ser la misma, si no se corre el riesgo de empezar

a hacer supuestos erróneos. Por esta razón se debe integrar diferentes fuentes de conocimiento para lograr desarrollar el proyecto correctamente.

Con la gestión del conocimiento se pueden establecer formas de manejar la información que otras personas conocen y que no se ha dejado almacenada en ningún medio, ayudando a formar grupos de trabajo que se encarguen de entrevistar los diferentes participantes o invitarlos a formar bases de datos de conocimiento acerca de todas las experiencias, de todos los incidentes que se les hayan presentado en algún momento del ciclo de desarrollo. De esta forma se podrían minimizar los costos en la solución de incidentes y permitir que las pruebas de software generen mayor impacto en el proceso de construcción del producto. Una vez se identifiquen los incidentes durante el proceso de pruebas, el desarrollador tiene la capacidad y la información a la mano para darle solución de manera más rápida y eficaz a dichos incidentes.

De acuerdo con el análisis realizado del proceso de pruebas de software, en algunas compañías, usando los principios primarios de la gestión de conocimiento, fueron identificados cinco problemas importantes en el proceso. (Xue-Mei et al., 2009).

1. Baja tasa de reusó de conocimiento en las pruebas de software: El conocimiento en las pruebas de software no se ha almacenado concienzudamente. Algunas bases de conocimiento se han implementado, pero generan gran discusión en los grupos de trabajo.

2. Barreras en la transferencia de conocimiento en las pruebas de software: La administración del conocimiento de pruebas es muy difícil de transmitir.

3. Ambiente pobre para compartir el conocimiento de las pruebas de software: Los miembros del equipo tienen poco tiempo en común para compartir experiencias.

4. Pérdida significativa en el conocimiento de las pruebas de software: Los conocimiento y experiencias obtenidas por el equipo de trabajo en pocas ocasiones se convierte en conocimiento público. Adicionalmente, la rotación de personal lleva a la pérdida de conocimiento en pruebas.

5. Imposible obtener rápidamente la más óptima distribución del recurso humano: La gestión de conocimiento es una integración entre personas, procesos y tecnología, donde las personas son la parte más importante. Sin tener un conocimiento adecuado de la información de cada persona, los recursos no se asignan adecuadamente dentro de los proyectos de software.

Según el estudio realizado a 30 organizaciones del sector del software, realizado por (Taipale, 2007), los factores que impactan el desarrollo y la mejora del proceso de pruebas son: La automatización de pruebas, el conocimiento sobre el proceso de pruebas, la adecuada definición de una estrategia de pruebas, las características del proyecto de pruebas asociadas a la gestión empresarial y el uso de componentes de software.

Se debería entonces entender la gestión de conocimiento, como una disciplina que permite consolidar el capital intelectual en las organizaciones. La gestión de conocimiento apoya no sólo los conocimientos técnicos de una empresa, sino también el dónde, quién, qué, cuándo y por qué suceden las cosas (Rus & Lindvall, 2002). Cuando una organización decide utilizar tecnologías relacionadas con la gestión del conocimiento con el propósito de hacer que esas herramientas lleven a una mejora de los procesos de la organización, o para instrumentar cambios organizativos o de cultura, se deben establecer los lineamientos técnicos y metodológicos necesarios, a partir de un análisis de los recursos disponibles.

En el contexto de la construcción de productos de software, el conocimiento dentro de un grupo de desarrollo de software está implícito dentro de los procesos y los productos. El conocimiento necesario comprende principalmente dos dominios: (1) el técnico, para poder manejar las herramientas y métodos necesarios de desarrollo, y (2) el dominio particular al que pertenece el proyecto en sí.

Se puede ver al conocimiento como un concepto fundamental dentro de las organizaciones, ya que potencializa los servicios de tecnología de información y los sistemas de soporte dentro de un proyecto. La idea para desarrollar es simple: dar a los desarrolladores el conocimiento preciso y actualizado que ellos necesitan, para solucionar problemas encontrados en un sistema, y mantener el entorno de las tecnologías de

información de este campo ejecutándose eficientemente (Bellinger, 2007). Para lograr esta meta, se debe coleccionar los defectos y soluciones obtenidos a lo largo de los proyectos pasados para construir una base de conocimiento.

Un modelo de gestión de conocimiento que administre la transferencia y los activos de conocimiento de las pruebas impacta en diferentes momentos del proceso como son: planeación del proyecto de pruebas, definición de las estrategias de prueba, reducción de costo del proyecto y soporte para la corrección de defectos entre otros.

1.3 Pregunta de Investigación

¿Cómo incorporar procedimientos y artefactos específicos para orientar y gestionar el conocimiento en el proceso de pruebas de software como una estrategia de mejora continúa orientada a los individuos?

1.4 Hipótesis

Teniendo en cuenta lo expuesto anteriormente, la **hipótesis** es:

El proceso de pruebas de software con la aplicación de un modelo de gestión de conocimiento incrementa el nivel de calidad del producto y contribuirá a la mejora continua del proceso.

1.5 Objetivos

1.5.1 Objetivo General

Diseñar un modelo de gestión de conocimiento para el proceso de pruebas de software que aporte a la mejora continua, basado en la experiencia y el aprendizaje individual.

1.5.2 Objetivos Específicos:

- Identificar los factores que inciden en el proceso de las pruebas de software, mediante una revisión de literatura.
- Caracterizar las iniciativas de gestión de conocimiento aplicadas a las pruebas de software.
- Determinar los activos de conocimiento basado en las teorías gestión del conocimiento para la mejora del proceso de pruebas.
- Desarrollar el modelo de gestión de conocimiento aplicado al proceso de pruebas.
- Validar el modelo mediante la técnica experimental, para determinar el aporte de la gestión de conocimiento en el proceso de pruebas.

1.6 Contribución

La finalidad de esta tesis es realizar una contribución al ámbito de la Ingeniería de Software en un área de importancia permanente como lo es el de la mejora de las prácticas y el proceso de pruebas de software, desde la perspectiva de la gestión del conocimiento.

La meta de este trabajo es definir un modelo de gestión del conocimiento para la mejora del proceso en los equipos de pruebas de los proyectos de desarrollo de software.

Esta tesis de doctorado busca constituirse en la base de una línea de investigación enfocada en el diseño de un modelo de gestión de conocimiento aplicado al proceso de pruebas de software que permitirá el desarrollo de una serie de trabajos de pregrado, maestría y doctorado. Basado en un proceso de mejora impactando en la eficiencia del proceso de pruebas independiente de las características de la estructura organizacional del equipo de pruebas.

.

2 Capítulo: Marco Teórico y Revisión de Literatura

En este capítulo se presenta el marco teórico sobre el área de investigación. Inicialmente se presenta la descripción de los conceptos asociados a la gestión de conocimiento y las pruebas de software. Posteriormente, se realiza la revisión de literatura sobre las experiencias y estudios más relevantes. En primera instancia, se identifican los factores que inciden en las pruebas de software y las iniciativas desarrolladas para mitigar la problemática en esta área. En segundo lugar, se caracterizan las experiencias y estudios desarrollados aplicando estrategias de gestión de conocimiento a las pruebas de software con el propósito de identificar las oportunidades de mejora en el contexto de la investigación. Con este capítulo se da solución a los objetivos específicos número uno y dos.

2.1 Marco Teórico.

2.1.1 Conocimiento

En la gestión de conocimiento existe una diversidad de definiciones y de caracterizaciones para explicar el termino de conocimiento.

Para comprender el concepto de conocimiento es importante analizar las tendencias occidental y oriental (Nonaka & Takeuchi, 1995).

Para los occidentales el conocimiento son las creencias justificadas por la verdad y para los orientales el conocimiento refleja la percepción del objeto en observación a través del

medio que permite conocerlo. Para la teoría del conocimiento, estos enfoques solo se asumen en torno a las condiciones del contexto como son: el objeto y el medio en que es observado (Hessen et al, 1970).

En consecuencia, el concepto de conocimiento se debe comprender desde la naturaleza, el contexto y el propósito de aplicación.

Desde el punto de vista organizacional, se concibe el conocimiento como el insumo para la producción de riqueza, y su creación y aplicación como el principal soporte de la organización en lograr una ventaja competitiva sostenible (Nonaka & Toyama, 2003).

Los autores (Alavi & Leidner, 2001), definen “El conocimiento como la integración de experiencia estructurada, valores, información contextualizada y discernimiento experto que provee un marco de referencia para evaluar e incorporar nuevas experiencias e informaciones”.

Acorde con (Nonaka, 2008), el conocimiento es creado sólo por los individuos y la organización provee el contexto para que los individuos generen conocimientos.

2.1.2 Taxonomía del conocimiento

Existen diferentes clasificaciones de conocimiento de acuerdo con las características que inciden en su generación y uso, tales como:

De acuerdo con los autores (Nonaka & Takeuchi, 1995), el conocimiento se clasifica en: tácito y explícito. A continuación, se definen:

- Conocimiento tácito: Es aquel que físicamente no es palpable, difícil de expresar y de uso interno de cada individuo. Normalmente se asocia a habilidades que se fundamentan en acciones.
- Conocimiento Explicito: Es aquel que se puede compilar a través de un lenguaje formal y sistemático. Por ejemplo, manuales, libros y reglas.

Por su parte Schunk (Schunk, D., 1997), categoriza el conocimiento en tres tipos: : declarativo, procedimental y condicional. Dando la siguiente conceptualización:

- Declarativo: es el conocimiento basado en creencias y teorías.
- Procedimental: Es aquel conocimiento que se orienta a dar respuesta al cómo ejecutar las actividades y procedimientos para alcanzar un propósito o solución, y se asocia a las reglas y algoritmos.
- Condicional: Se refiere al conocimiento que permite discernir en qué condiciones se debe emplear los conocimientos declarativos y procedimentales.

Según Boisot (Boisot, M., 1998), plantea que el conocimiento puede ser codificable y en otros casos no y solo es posible si el conocimiento se puede difundir con facilidad. Para el autor, el conocimiento codificable es almacenable sin pérdida de información, mientras que el no codificable es aquel que no puede ser obtenido por escrito ni almacenado sin perder los aspectos esenciales de la experiencia a la que se refiere.

Por su parte (Bollinger & Smith, 2001), distinguen entre conocimiento individual y conocimiento colectivo, basado en las siguientes definiciones:

- Individual: conocimiento personal es el entendimiento, toma de conciencia o familiaridad acerca de un tema, adquirido por medio del estudio, la investigación, la observación o la experiencia práctica a lo largo del tiempo.
- Colectivo: conocimiento organizacional es, en general, lo que las personas conocen acerca de los clientes, productos, procesos, políticas, cultura y forma de actuar de la organización.

Rus y Lindvall (Rus & Lindvall, 2002) afirman que, dependiendo de las características asociadas al tipo de actividad de la ingeniería de software, los conocimientos se clasifican en:

- Organizacional: Se refiere al conocimiento para gestionar la organización, los objetivos del negocio y la gestión de los recursos humanos.
- De Gestión: Se refiere al conocimiento para planificar, liderar y realizar el seguimiento de un proyecto.

- Técnico: Abarca el conocimiento para desarrollar la cadena de valor del software como: requisitos, diseño, programación y pruebas de software.
- Del Dominio: Hace referencia al conocimiento del contexto de aplicación del negocio, ejemplo: académico, financiero y salud, entre otros.

Por otra parte, los autores (Itkonen et al., 2013), en el contexto especifico de pruebas de software distingue tres tipos de conocimiento:

- Conocimiento del dominio: hace referencia a las reglas de negocio que soportan la misión empresarial y los referentes normativos, legales y culturales que influyen en los procesos de negocio.
- Conocimiento del sistema: este se refiere al conocimiento embebido en la aplicación y no se puede identificar claramente en la documentación de diseño.
- Conocimientos generales de la ingeniería de software. Hace referencia al conocimiento y habilidades que deben tener los probadores frente a la cadena de valor que soporta la construcción de software.

Claramente existen muchas categorías adicionales a estas que permiten entender el valor del conocimiento. Sin embargo, un punto importante a considerar es el hecho que la organización por sí sola no puede crear conocimiento, sino que son las personas que la componen quienes establecen las nuevas percepciones, pensamientos y experiencias que establecen el conocer de la organización (Nonaka & Takeuchi, 1995). Bajo esta premisa, entender donde reside aquel conocimiento es de vital importancia para administrarlo y generar valor.

2.1.3 Gestión de Conocimiento

La gestión de conocimiento (KM por sus siglas en inglés) se soporta desde la comprensión del concepto de conocimiento y comprende tres elementos (Soto et al, 2017a):

1. Conceptualizar las diferencias y la relación existente en los conceptos: dato, información y conocimiento,

2. La instanciación que realiza un individuo a través del procesamiento cognitivo de los datos y la información, para ser usados en un contexto especifico.
3. La concepción de conocimiento, el cual es propia del individuo, su utilidad está sujeta a los procesos de compartir, interpretar e interiorizar de otras personas.

Los conceptos de dato, información y conocimiento son definidos por (Checkland & Holwell, 2006). como:

- Dato: es un valor aislado y representa un hecho observado o creado por las personas.
- Información: es el conjunto de datos que se le atribuye un significado.
- Conocimiento: colección de información que pueden ser apropiadas o interiorizadas por las personas, que puede ser útil para ellas, pero no provee generación de nuevo conocimiento.

En consecuencia, la gestión de conocimiento se fundamenta en la comprensión de la cadena datos, información y conocimiento, para optimizar y mejorar los objetivos estratégicos de la organización (Soto et al, 2017a).

La gestión de conocimiento al igual que el término conocimiento, presentan diversas definiciones y enfoques, a continuación, se referencian las siguientes:

Según Gupta y Sharma (Gupta & Sharma, 2004), la generación de conocimiento es el conjunto de procesos que gobiernan la creación, diseminación, y utilización de conocimiento.

Para los autores (Del Moral, et al., 2007), la gestión de conocimiento tiene un espectro más amplio y lo define como: el conjunto de principios, métodos, técnicas, herramientas, métricas y tecnologías que permiten obtener los conocimientos precisos, para quienes los necesitan, del modo adecuado, en el tiempo oportuno de forma eficiente y sencilla, con el fin de conseguir una actuación institucional o más inteligente posible.

IBM y LOTUS definen la gestión del conocimiento como "una disciplina sistemática que aprovecha el contenido y la experiencia para proporcionar innovación, capacidad de respuesta, competencia y eficiencia " (Pohs et al., 2001).

Mientras Microsoft afirma que la gestión del conocimiento “es nada más que la gestión de los flujos de información; permitiendo a las personas el acceso a la información para que puedan actuar rápidamente" (Gates, 1999).

Independiente la definición de Gestión, el conocimiento se concibe como la principal fuente de riqueza y por consiguiente los activos intangibles, como los procesos de Know-how, las mejores prácticas y el capital intelectual, adquieren una mayor importancia en la gestión efectiva del conocimiento. (Drucker, 2002, 2003, 2004; Krüger, 2006, López, 2005; Del Moral et al., 2007; Myers, 1996; North & Roque, 2008; Soto et al., 2006).

Según (Call, D. 2005), el éxito de la gestión de conocimiento se sustenta en facilitar el acceso a una mejor información para cumplir con una labor asignada, a pesar de que no proporciona la respuesta a los problemas, pero si facilita el aprendizaje para darle solución.

2.1.4 Marco de Procesos de la KM

De acuerdo con los autores (Soto et al, 2017a), los procesos de KM reciben diferentes denominaciones en la Tabla 1, se unifican los conceptos.

Tabla 1. Procesos de la Gestión de Conocimiento.

Procesos	Términos similares utilizados por diferentes autores
Definición de objetivos	Alineación Estratégica y Lineamientos Organizacionales.
Identificación	Mapeo y Fuentes de Información.
Incorporación	Adquisición, Desarrollo, Creación, Construcción; Externalización; Innovación.
Preservación	Indexación; Formalización; Codificación; Almacenamiento
Distribución	Transferencia, Socialización. Integración y Compartir
Utilización	Aplicación o Uso

Fuente: Elaboración propia

A continuación, se describe cada proceso acorde con los autores (Soto et al, 2017a):

- Definición de Objetivos: Hace referencia a los conocimientos, habilidades y capacidades requeridas por la organización, para alcanzar la estrategia de negocios establecida en los planes y objetivos estratégicos. Partiendo de estos argumentos, se establecen los objetivos de conocimiento estratégico, que soportan la estrategia de la organización.
- Identificación: En esta etapa se detectan los activos de conocimiento que tiene la organización (inventario de conocimiento) y el conocimiento requerido para soportar los objetivos de conocimiento, definidos en la fase anterior.
- Incorporación: El propósito es adquirir el conocimiento requerido para cumplir con el propósito estratégico y que no está disponible, según el análisis realizado en el proceso anterior.
- Preservación: Este proceso está compuesto por las actividades que permiten almacenar y mantener los conocimientos (tácito y explicito) disponibles.
- Distribución: Este proceso genera las condiciones para que el conocimiento se pueda compartir.
- Utilización: El objetivo del proceso es garantizar que el conocimiento se institucionalice o incorpore por los miembros de la organización. Siendo el fin de cualquier estrategia de KM.

2.1.5 Enfoques de KM

De acuerdo con los autores Galvis y Sánchez (Galvis & Sánchez, 2013), los enfoques se abordan desde las capacidades y las corrientes que soportan las iniciativas de KM.

Los programas de KM se componen de varias prácticas y estrategias que tienen su origen en diferentes escuelas, organizadas según (Earl, 2001) en tres categorías, como son:

- Tecnocrática: Se enfoca en el uso de la tecnología para gestionar el conocimiento de los miembros de la organización y la componen las siguientes escuelas:
    - Sistemas: Intercambio de conocimientos basado en la codificación soportado en recursos tecnológicos y repositorios.

  - Cartográficas: Se soporta en mapas de conocimiento y creación de directorios de conocimiento para la compartir el conocimiento.
  - Ingeniería: El aprendizaje organizacional se centra en los procesos y flujos de conocimiento.
- Comportamental: Se enfoca en la promoción y el estímulo de las personas para lograr el uso y transferencia del conocimiento
  - Organizacional: Genera un entorno de conectividad basado en redes de conocimiento.
  - Espacial: Propicia la socialización y transferencia del conocimiento basado en el diseño de espacios.
  - Estratégica: El conocimiento es visto como la esencia de la estrategia de la organización para la generación de valor.
- Económica: Se sustenta en la relación existente entre los activos de conocimiento con los ingresos de las organizaciones.

Otra perspectiva de análisis de las iniciativas de gestión de conocimiento planteada por los autores (Gold et al., 2001), son las capacidades para que las organizaciones puedan aprovechar de manera competitiva el conocimiento que posee y así generar nuevo conocimiento. Estas capacidades se categorizan en: capacidades de infraestructura de conocimiento y capacidades de procesos de conocimiento. En la tabla 2, se describen las capacidades abordadas por (Gold et al., 2001).

Tabla 2. Capacidades de KM

Categoría	**Capacidades**	**Alcance**
Infraestructura	Tecnología	Las tecnologías de información determinan la forma como el conocimiento se transfiere y se accede
	Cultura	La cultura organizacional y la conducta de las personas apoya y potencias las actividades de KM.
	Estructura	Las estructuras organizacionales facilitan la interacción entre las personas y los activos de conocimiento.
Procesos	Adquisición	Localización y obtención del conocimiento mediante la colaboración entre los individuos y socios de negocio.

	Conversión	Hace referencia al conocimiento estructura y compilado que facilite su distribución y uso
	Aplicación	Se refiere al uso alineado a las estrategias para resolver problemas y mejora de la eficiencia.
	Protección	Proceso encargado de asegurar que no se permita el uso inadecuado del conocimiento.

Fuente: Modificado de (Galvis & Sánchez, 2013)

2.1.6 Aprendizaje Organizacional

La generación de conocimiento organizacional sustenta la base de los procesos de aprendizaje organizacional acorde a los cambios del contexto y al desarrollo de capacidades de adaptación para asumir cambios. A nivel de las ciencias organizacionales y cognitivas, encontramos muchos modelos sobre cómo el conocimiento es transferido o aprendido a nivel individual y organizacional. A continuación, se describe las siguientes teorías:

- **Teoría de la creación de conocimiento:** De acuerdo con (Nonaka & Takeuchi, 1995), la creación de conocimiento organizacional es producto de la interacción entre los conocimientos tácito y explícito, es un proceso cíclico de transformación o conversión de conocimiento soportado en cuatro procesos: socialización, externalización, combinación e interiorización denominado (SECI), como se presenta en la figura 1.

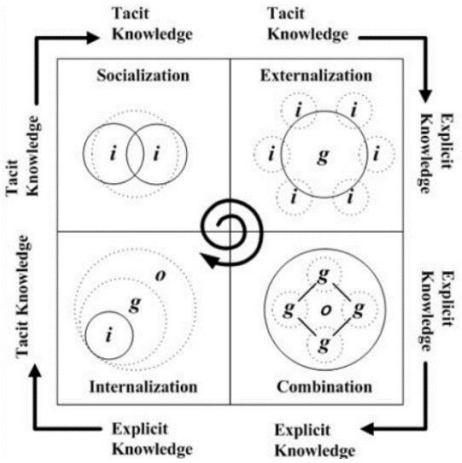


Figura 1. Modelo SECI
Fuente: Tomado de (Nonaka & Konno, 1998).

Socialización se refiere a la acción de transferir el conocimiento tácito a otra persona a través de la observación, la imitación y la práctica. La externalización consiste en pasar del conocimiento tácito al explícito. El conocimiento explícito puede tomar las formas de analogías, definiciones, procesos y modelos. El proceso de combinación convierte el conocimiento explícito a explícito, Ocurre cuando el conocimiento explícito preexistente se combina con el conocimiento externalizado. Internalización significa tomar el conocimiento explícito y convertirlo en conocimiento tácito individual en forma de modelos mentales o conocimientos técnicos.

- **Aprendizaje Basado en la experiencia**. Acorde con (Kolb,2014), aprender es un proceso por el cual el conocimiento se crea a través de la transformación de la experiencia. Plantea un modelo cíclico presentado en la figura 2, basado en cuatro momentos de la persona denominados: experiencia concreta, observación reflexiva, conceptualización abstracta y experimentación. La primera fase experiencia concreta el individuo asume nuevas experiencias. La siguiente etapa las personas trabajan en forma conjunta apoyados eventualmente por un facilitador, con el propósito de interpretar y reflexionar sobre esas nuevas experiencias. El tercer paso del modelo implica tomar como base los antecedentes y creencias para integrarla a sus capacidades. Por último, las nuevas capacidades y experiencias adquiridas son aplicadas en la toma de decisiones.

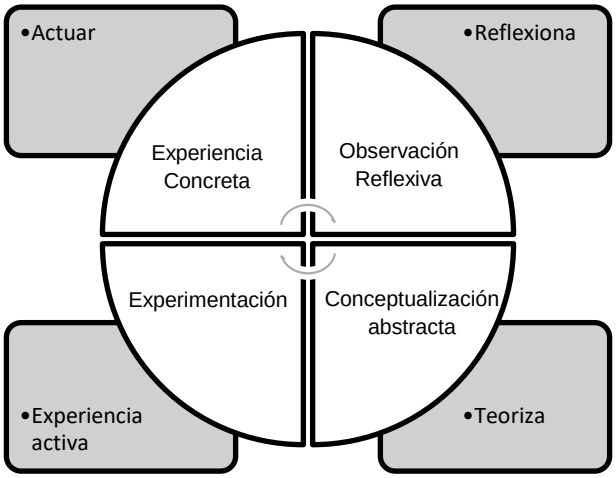


Figura 2. Modelo experiencial de Kolb (Kolb,2014).
Fuente: Elaboración propia

- **Aprendizaje Basado en Bucles.** Según los autores (Argyris & Schon, 1996), el aprendizaje de un sólo bucle es cuando el individuo aplicar el conocimiento de forma reiterativa sin cuestionar las causas o las razones que lo generan. Por su parte, el aprendizaje de doble bucle tiene lugar a la autocrítica y la reestructuración de las normas y las creencias para desarrollar nuevas premisas. En un nivel superior los autores plantean la capacidad que tiene la organización para "aprender a aprender" y generar nuevo aprendizaje basado en los enfoques de bucle simple y doble. Según figura 3.

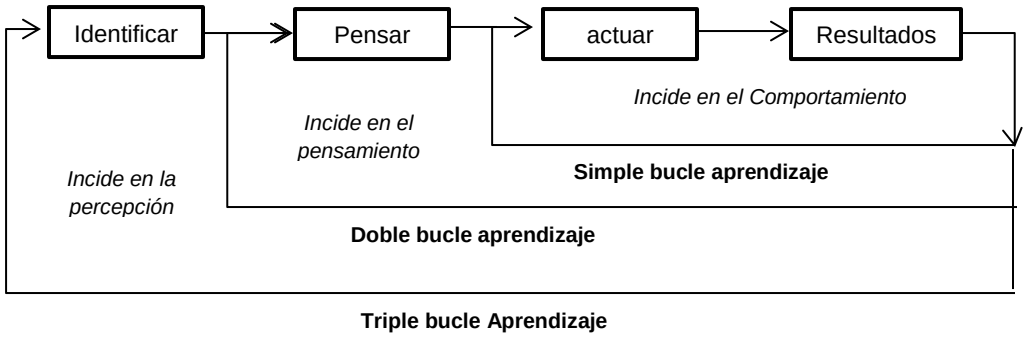


Figura 3. Aprendizaje Basado en bucles
Fuente: Elaboración propia

- **Aprendizaje Analítico y Sintético.** Según los autores (del Moral, et al., 2007) Existe dos enfoques de aprendizaje organizacional como se presentan en la figura 4, el primero denominado analítico que permite obtener los conocimientos en un área específica identificada como estratégica para la gestión de la organización. El segundo, el aprendizaje sintético, hace referencia al aprendizaje individual asociado al conocimiento de gestión o técnico, catalogado como una lección aprendida y puede ser distribuido al resto de la organización.

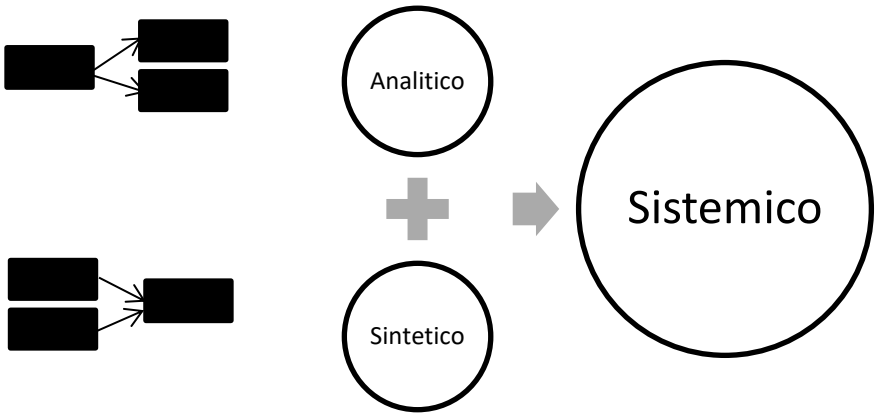


Figura 4. Aprendizaje Analítico y Sintético.
Fuente: Elaboración propia

Organización que aprende. Según Senge (Senge, 2005) es aquella que extiende sus capacidades para el futuro y Garvín (Garvín, 2000) como la organización que efectivamente es hábil en la creación, adquisición y transferencia de conocimientos, y la capacidad de apropiar el conocimiento en el comportamiento y la cultura organizacional. Para Harrison (Harrison, 2002), afirma que una organización que aprende está en constante incremento de la capacidad para producir resultados.

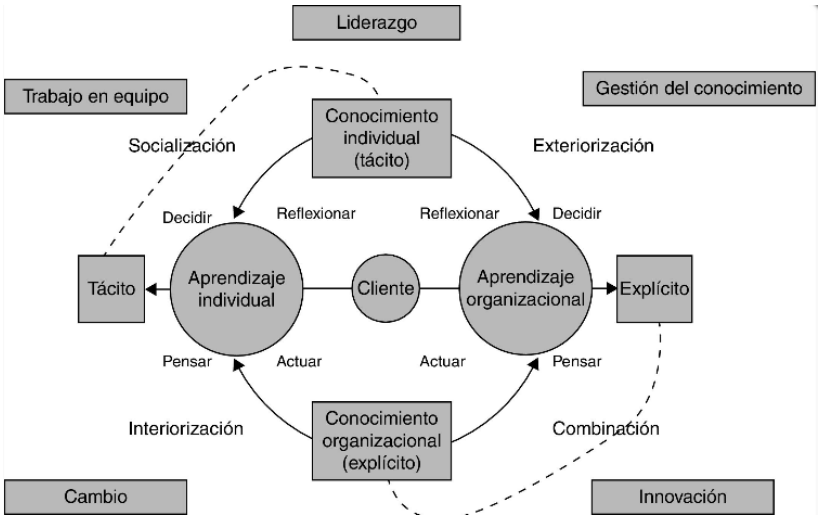


Figura 5. Modelo de Organización que Aprende.
Fuente: Tomado (Rodríguez & Trujillo, 2007)

De acuerdo con Rodríguez y Trujillo (Rodríguez & Trujillo, 2007), presentan el circuito según figura 5, denominado “conocimiento aprendizaje organizacional” (CAO), el cual integra el ciclo de aprendizaje experiencial (Kolb, 2014) y el proceso de conversión del conocimiento desarrollado por (Nonaka & Takeuchi, 1995). Esta fusión genera un modelo que muestra cómo el aprendizaje individual llega a convertirse en organizacional al transformar el conocimiento tácito en explícito, llegando a construir una organización que aprende.

Según Pawlowsky (Pawlowsky, 2001), describe el proceso de aprendizaje organizacional en niveles, procesos de KM, tipos de aprendizaje y orientación del aprendizaje en el marco de la organización que aprende. Define la estructura de niveles: individual, grupal y organizacional; y la relación existente entre estos niveles. También asocia los tipos de aprendizaje mencionados anteriormente: bucle simple y doble. Estableciendo la importancia de las estructuras cognitivas y conductuales de la organización desarrolladas a través de la cultura organizacional como elementos de la orientación del aprendizaje.

Los autores Castañeda y Fernández (Castañeda & Fernández Ríos, 2007), plantean que el aprendizaje organizacional se ejecutada en dos rutas: partiendo del nivel individual al organizacional y viceversa. Ratificando la estructura por niveles definida por Pawlowsky Pawlowsky, Kim, Crossan Lane y White. (Pawlowsky, 2001) (Crossan et al., 1999) (King, 2001) y la importancia del enfoque individual en el aprendizaje organizacional.

De acuerdo con Zapata (Zapata, 2005), toda iniciativa de aprendizaje organizacional y de gestión del conocimiento debe incorporar de manera explícita el aprendizaje individual.

Según Moreno y sus colegas (Moreno et al., 2001), afirman que la efectividad del aprendizaje individual se desarrollara de manera inconsciente por el individuo.

De acuerdo con el modelo de Crossan y sus colegas (Crossan et al., 1999) el aprendizaje tiene una relación entre los niveles individual, colectivo y organizacional, que se sustenta en cuatro procesos de aprendizaje: la intuición, la interpretación, integración y la institucionalización. Salazar (Salazar, 2015), desarrolla las definiciones de cada proceso así:

- La intuición “es el reconocimiento preconsciente del modelo y posibilidades inherentes en una corriente personal de experiencia. Este proceso puede afectar

las acciones individuales intuitivas, pero solo afecta a otras cuando ellas intentan interactuar con lo que es individual".

- La Interpretación "es la explicación, a través de palabras y/o acciones, o la comprensión de una idea de uno mismo o de otros. Este proceso va de lo proverbal a lo verbal, resultando en el desarrollo de un lenguaje".
- La Integración "es el proceso de desarrollo que comparte entendimiento entre individuos y coordina acciones a través de adaptación mutua".
- La institucionalización "es el proceso de integrar el aprendizaje que ha ocurrido en individuos y grupos dentro de la organización, e incluye sistemas, estructuras, procedimientos y estrategias".

Es de anotar que no es fácil determinar en qué nivel inicia y termina un proceso, pero si es claro que la institucionalización se da a nivel organizacional y la intuición a nivel individual. Sin embargo, la integración relaciona el plano grupal y organizacional, mientras la interpretación diferencia el nivel individual y grupal.

A partir de la revisión conceptual desarrollada por Zapata (Zapata, 2005), presenta una mejora al modelo de Crossan, Lane y White, integrando el concepto de capacidades humanas y el de subprocesos de aprendizaje, que permite una mayor comprensión del aprendizaje humano. Adicionando al aprendizaje por intuición, procesos cognoscitivos relacionados con el aprendizaje consciente, también sustentados por King (King, 2001), que se logra a través de actividades de aprendizaje conscientes como los procesos laborales y formativos.

Sumado al concepto de capacidades humanas, el enfoque de Castañeda y Pérez (Castañeda & Pérez, 2005), los cuales incorporan el componente de aprendizaje individual, expresado en cuatros momentos y descritos así:

- La atención: etapa definida como el proceso cognoscitivo que regula la exploración la percepción de las acciones que son modeladas por otras personas.
- La retención: consiste en la transformación de la información para representarla en términos de reglas y concepciones.
- La producción: proceso basado en la conversión de las representaciones simbólicas en acciones.

- La motivación: este proceso asocia la adquisición cognoscitiva y ejecución conductual, adoptando a la conducta los conocimientos y experiencias aprendidas que generen valor y resultan gratificantes.

2.1.7 Mejora de procesos de software

Para las organizaciones de software la mejora se refleja en el cumplimiento de los criterios establecidos en la ejecución de proyectos, como: el alcance, el costo, tiempo y satisfacción del cliente, entre otros, como propósitos y metas a cumplir (Capell, 2004) (Sommerville, 2004) (Niazi et al. 2006) (Allison, 2010) (Phillips, 2008). Para el cumplimiento de estos objetivos las organizaciones desarrollan una serie de esfuerzos denominadas en la Ingeniería de software como Mejora de Procesos de Software (SPI por sus siglas en inglés).

Por tal razón, para desarrollar el concepto SPI, se asumen las siguientes definiciones:

- Proceso es la secuencia de pasos que debe seguir un profesional experto para realizar una tarea específica (Pomeroy-Huff te al., 2009).
- Procesos Software se considera como la organización lógica de profesionales, herramientas y métodos, a actividades designadas a producir un resultado final específico (Pall, 1987).

Las definiciones referenciadas permiten comprender el proceso de software como la interacción de las prácticas, personas y el entorno para lograr la construcción de una solución de software.

Por consiguiente, la SPI es un enfoque estructurado para mejorar las capacidades de una organización de software para proporcionar servicios de calidad en forma competitiva (Mathiassen et al., 2003). Siendo una actividad repetitiva y con independencia del enfoque adoptado, requiere de cierto tiempo, recursos, medidas, y las iteraciones para su aplicación efectiva y exitosa (Nguyen et al., 2011) y enfocada a mantener procesos estables que permitan obtener realmente los resultados esperados (Capote et al. 2013).

Es importante anotar que “La mejora de procesos se basa en los principios de mejora continua” (Capote et al. 2013) como se describe en la figura 6.

Figura 6. Principios de Mejora Continua.
Fuente: Elaboración Propia

La mejora del proceso personal (PPI por sus siglas en inglés) asume los principios de la mejora continua, tomando como referencia un proceso definido y medible; se caracteriza por describir el conocimiento y las habilidades que necesitan los profesionales para mejorar su propio proceso (Pomeroy-Huff te al., 2009).

El PPI se concibe como un proceso de auto mejoramiento diseñado para ayudar a controlar, administrar y mejorar la forma en que se trabaja individualmente. Está estructurado por guías y procedimientos para apoyar al profesional a desarrollar el trabajo (Soto & Reyes, 2010).

A partir del concepto de proceso se concibe dos características denominadas habilitante y operacional. Se afirma que un proceso es habilitante porque resuelve el cómo hacer el trabajo e incluye todos los elementos requeridos para su uso, tales como: entradas requeridas, recursos asignados (por ejemplo, personas, hardware, tiempo, dinero) y criterio de salida. Un proceso es operacional porque describe puntualmente qué hacer y proporcionan los pasos detallados para que los equipos y las personas puedan realizar su trabajo, con el propósito de guiar y controlar el trabajo. El caso del proceso de software personal (PSP por sus siglas en inglés) y el proceso de software grupal (TSP por sus siglas

en inglés) son procesos con características operacionales y habilitantes (Pomeroy-Huff te al., 2009).

PSP y TSP son procesos que facilitan la implementación del modelo integrado de capacidad y madurez denominado CMMI (por sus siglas en inglés) (Humphrey, 2007). Los modelos de buenas prácticas denominados PSP y TSP, se soportan parcialmente en las áreas de proceso del modelo CMMI y siguen una implementación por niveles orientadas al individuo y al grupo respectivamente, como se muestra en la figura 7.

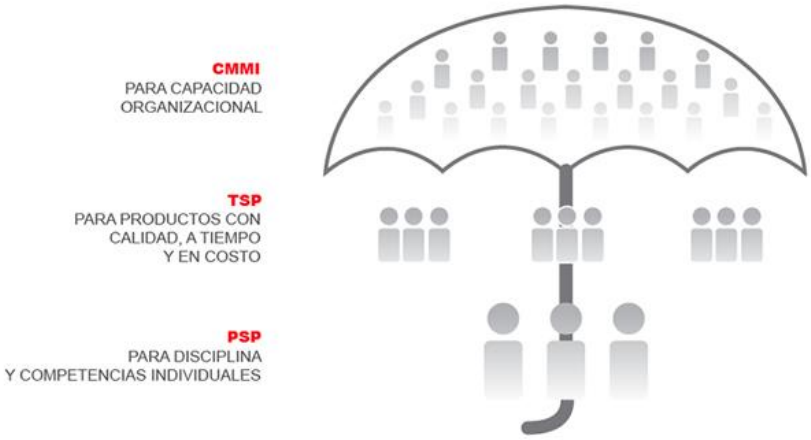


Figura 7. Relación de PSP/TSP con CMMI.
Fuente: Modificado de presentación de Jim Mchale del SEI.

El CMMI integra cuatro áreas de conocimiento, como son: ingeniería de software, ingeniería de sistemas, acuerdos con proveedores y desarrollo integrado de productos y servicios. El modelo cuenta con dos representaciones: por etapas y continuo. Estas representaciones permiten a la organización definir diferentes propósitos de mejora.

El PSP es una línea de trabajo de medición y análisis que permite al desarrollador caracterizar su proceso, con el propósito de mejorar su desempeño. Algunas de sus premisas son:

- El desempeño individual está dado por el conocimiento, la disciplina y el compromiso del individuo.
- La adopción de un proceso facilita el cumplimiento de los compromisos.

- La medición y el análisis de los datos derivados del desempeño permite identificar alternativas de mejora.
- Las lecciones aprendidas son fundamentales para la mejora de las prácticas personales.

El modelo TSP busca crear equipos autodirigidos y autónomos proporcionando directrices para ayudar a un equipo a establecer sus objetivos, a planificar sus procesos y a revisar su trabajo con el fin de que la organización pueda establecer prácticas de ingeniería avanzadas y así obtener productos eficientes, fiables y de calidad.

Los modelos PSP y TSP provee un esquema de trabajo, donde se definen roles, actividades y responsabilidades. Donde el PSP genera en el individuo el uso de buenas prácticas en el proceso, una vez apropiadas en la conducta permite integrarse al equipo de trabajo que desarrollo los lineamientos TSP.

Específicamente en las pruebas de software, de acuerdo con la revisión de literatura sistemática realizada por los autores Afzal, Alone, Glocksien y Torkar (Afzal et al., 2016) existen 18 enfoques de mejora de procesos, pero la gran mayoría no cuenta con información que soporte la implementación y no existe documentación para la evaluación. Esto hace difícil aplicar los enfoques a nivel empresarial.

Acorde con los autores (Soto et al, 2017b), los modelos más representativos en las pruebas de software son: el modelo integrado de capacidad y madurez de pruebas (TMMi por sus siglas en inglés), TestPAI, Mejora de proceso de prueba (TPI por sus siglas en inglés). A continuación, se describen:

- TestPAI, es un área de proceso integrada a CMMI, cuenta con dos representaciones continua y por etapas. Define 5 objetivos específicos con sus respectivas prácticas. Tiene una estructura compatible con CMMI.
- TMMi es un modelo de referencia para el proceso de pruebas. Instancia el concepto de niveles de madurez para la evaluación y mejora del proceso. Propone 5 niveles de madurez con sus correspondientes áreas claves de proceso, así como los objetivos específicos y genéricos y prácticas específicas y genéricas.
- TPI es modelo enfocado al proceso de pruebas. establece un conjunto de 20 áreas clave con diferentes niveles de madurez. Define un conjunto de áreas compuestas

por niveles. Donde cada nivel contiene un conjunto de Puntos de Comprobación y Sugerencias de Mejora.

Según (Heiskanen et al. 2012), los marcos TMMi y TPI tienen una mayor orientación a la gestión, dado el enfoque técnico que desarrollo el modelo TPI.

El análisis comparativo entre los marcos de referencia TMMi y TPI, desarrollado por (Afzal et al., 2016), cubren los mismos aspectos del proceso de prueba, pero difieren principalmente en la representación de los modelos. Sin embargo, TMMi presenta una descripción clara de las practicas genéricas y específicas que facilitan la implementación y evaluación, mientras que en el marco TPI los puntos de control permiten generar diferentes interpretaciones dada la superficialidad del proceso.

2.1.8 Pruebas de software

La literatura existente en el área provee una gran variedad de definiciones de pruebas de software (ST por sus siglas en inglés), de las cuales se referencian las siguientes:

La prueba se define como: “la actividad en que un sistema o un componente se ejecuta bajo condiciones controladas, los resultados son registrados y la evaluación es realizada sobre algún aspecto del sistema o componente” (ISO, 2010).

Acorde con el estándar IEEE 1059/1993 (IEEE, 1993), las pruebas son: “un proceso de analizar un elemento de software para detectar las diferencias entre las condiciones existentes y las condiciones necesarias, y evaluar las características de un elemento de software.”

Según Graham, Van Veenendaal, & Evans (Graham et al. 2008), la norma ISTQB identifica los siguientes procesos en los proyectos de pruebas de software: la planificación y control del proyecto, análisis y diseño de casos de prueba, implementación y ejecución. Los proyectos de pruebas deben describir la evaluación de los criterios de salida y la presentación de informes, y por último el cierre de la prueba del software.

Acorde con los autores Myers y sus colegas (Myers, et al, 2015). El propósito de las pruebas de software es la detección y corrección de los defectos de los productos de

software. Las pruebas se enfocan exclusivamente en el ciclo de vida del software para asegurar que el producto cumpla con las especificaciones definidas.

La calidad según la ISO 9000 (García et al, 2014), es el grado en el que un conjunto de características cumple con los requisitos, lo que indica la satisfacción del interesado del software. Las pruebas de software contribuyen en gran medida al cumplimiento de los objetivos de calidad de un proyecto de software.

Las actividades de verificación y validación realizado durante el proceso de software acorde con ISO / IEC 29119, están asociadas al proceso de pruebas (Villamizar et al,2016).

De acuerdo con las diferentes definiciones las pruebas de software integran las actividades tendientes a verificar el proceso y validar el producto (ISO, 2013). Las pruebas evalúan dos aspectos el nivel y tipo de prueba (Soto et al, 2017b).

Nivel de pruebas

El Nivel de pruebas hace referencia al alcance de evaluación del producto, dependiendo de su propósito, uso, comportamiento o estructura (Bourk & Fairley, 2013) y existen los siguientes niveles (figura 8):

- Pruebas unitarias: El objetivo es probar las unidades más pequeñas del software, el componente o módulo de software (Bruegge, et al., 2002).
- Pruebas de componentes: El objetivo de la prueba de componentes es escoger un modulo, aislarlo del resto del código, y determinar si se comporta exactamente como se espera. Cada componente, se prueba por separado antes de integrarlo en un servicio (Gélvez, 2010)..
- Pruebas de integración: Se centran en verificar el ensamblaje correcto entre los distintos componentes, una vez que han sido probados unitariamente (Bourk & Fairley, 2013).
- Pruebas de sistema: Son pruebas de integración del sistema completo. Permiten probar el sistema en su conjunto y con otros sistemas con los que se relaciona, para verificar que las especificaciones funcionales y técnicas se cumplen. Estas pruebas se realizan en un entorno fuera del alcance del desarrollador (Bourk & Fairley, 2013).
- Pruebas de aceptación: Los usuarios prueban el sistema o aplicación para establecer si está listo para la salida a producción. (Bourk & Fairley, 2013).

- Pruebas de regresión: Ejecución de casos de pruebas anteriormente ejecutados cuando se implementan cambios en el software (Presman, 2002).

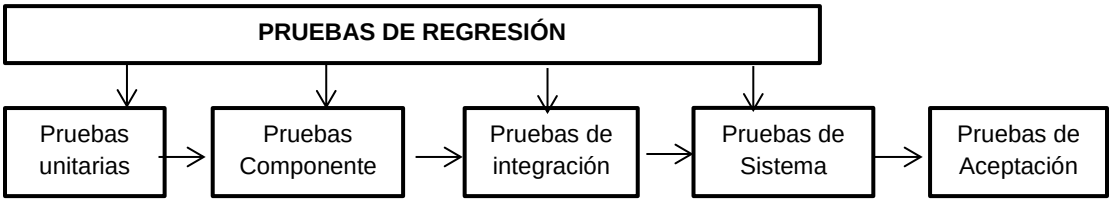


Figura 8. Niveles de Pruebas de Software.
Fuente: Modificado de propia.

Tipos de prueba

El tipo de prueba se asocia con el aspecto del producto a evaluar, a continuación, se detallan algunas:

- Las pruebas funcionales, que a su vez se dividen en pruebas de requisitos funcionales, pruebas de control de acceso y seguridad, y pruebas de volumen.
- Las pruebas de usabilidad evalúan la facilidad de uso de las interfaces con el usuario.
- La prueba de confiabilidad involucra la ejecución de otros tipos de pruebas como: pruebas de integridad, estructurales y de estrés.
- Las pruebas de rendimiento incluyen: pruebas de “benchmark” (procedimiento que evalúan el rendimiento), pruebas de carga y pruebas de perfiles.
- Las pruebas de infraestructura se dividen en pruebas de configuración y pruebas de instalación.
- Las pruebas de regresión.

Técnicas de pruebas

Para la aplicación de los procesos de verificación y validación acorde con el estándar ISO 29119, se tienen las siguientes técnicas (ISO, 2013) :

- Técnicas Estáticas: Es una técnica de evaluación de artefactos del proceso de desarrollo denominada generalmente revisiones, pretende detectar de forma

manual defectos en cualquier producto durante el proceso de construcción. Los tipos de revisiones que se realizan son: Revisiones Informales, Inspecciones y Auditorias.

- Técnicas Dinámicas: Estas técnicas permite evaluar un sistema o uno de sus componentes. Se ejecuta en circunstancias previamente definidas, comparando los resultados obtenidos y los resultados esperados para identificar las fallas en el software.
    - Caja Blanca o Estructural: Se enfocan en examinar los detalles procedimentales del código fuente en el contexto lógico del desarrollo.
    - Caja Negra o Funcionales: Estas pruebas se realizan al sistema de software en tiempo de ejecución orientada a evaluar los aspectos externos derivados de la funcionalidad del producto.

Acorde con la ISTQB (Graham et al. 2008), existen otro grupo de técnicas basadas en la experiencia del probador. En la figura 9, se describen las técnicas de pruebas.

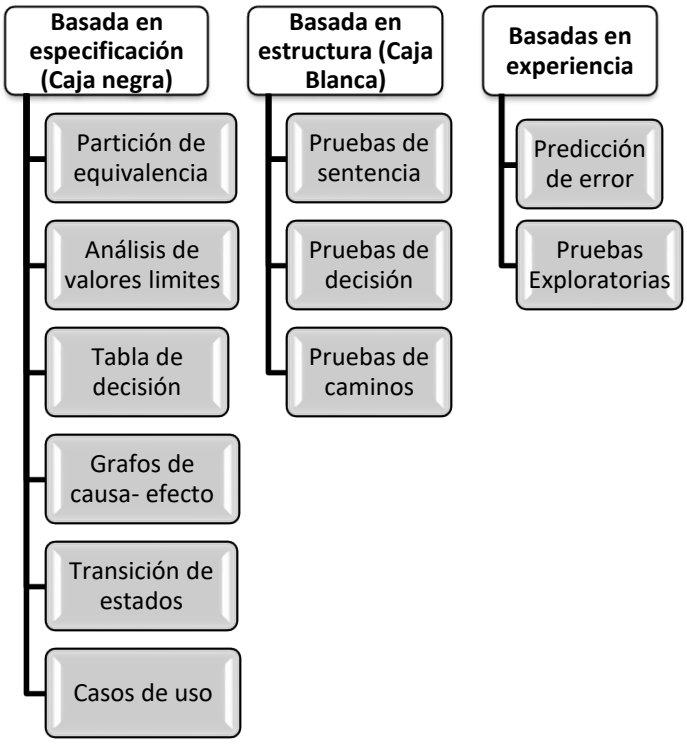


Figura 9. Técnicas de Prueba.
Fuente: Tomada de (Narváez, 2015).

Caso de prueba

El caso de prueba es importante porque suministran la información necesaria para la ejecución y control de la prueba.

Un caso de prueba está conformado por varios elementos como: datos de entradas, precondiciones, resultados esperados y postcondiciones, que permiten evaluar un aspecto especifico y a través de criterios valorar el cumplimento de los requerimientos (ISO, 2013) (Bath & Veenendaal, 2014).

Un caso de prueba debe ser preciso, efectivo, trazable, evolutivo y eficiente. Otra característica a nivel del entorno de pruebas es la capacidad para que se reinicie el ambiente en caso de que se deba repetir la prueba (Graham et al. 2008).

Los casos de pruebas deben ser repetibles y atómicos para identificar el error en el momento que se genere. Por otro parte, el caso de prueba puede ser un referente para establecer una unidad de medida del proceso (Narváez, 2015).

A continuación, la figura 10 describe los campos más relevantes de un caso de prueba.

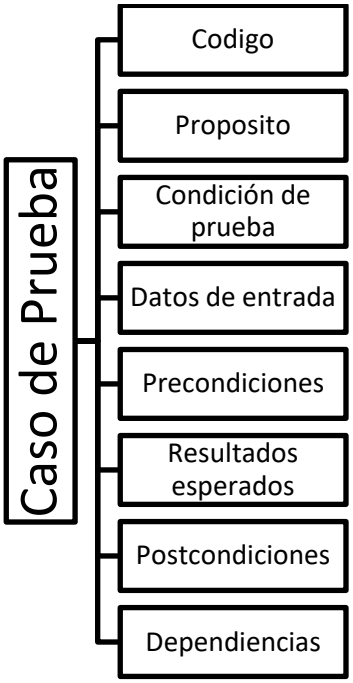


Figura 10. Elementos de un caso de prueba.
Fuente: Modificado de (Perry, 2007) (Aristegui, 2010)

Técnicas de estimación

Existen diferentes técnicas para la estimación de proyectos en pruebas de software. Según Torres (Torres, 2014), las técnicas se basan en los siguientes criterios:

Tabla 3. Criterios para las técnicas de estimación de pruebas

Criterio	Descripción
Esfuerzo del software	Las técnicas se fundamentan en el esfuerzo invertido para el desarrollo del producto (Nageswaran, 2001). Para ello, se calcula el esfuerzo de desarrollo a través de modelos como: • Modelo constructivo de costos (COCOMO por sus siglas en inglés) • Modelo de administración del ciclo de vida del software (SLIM por sus siglas en inglés) • Modelo de análisis de puntos de función (FPA por sus siglas en inglés). Una vez obtenido el esfuerzo del desarrollo del producto de software se multiplica por un factor de ajuste que expresa el esfuerzo de las pruebas.
Tamaño del software	La estimación parte de la premisa que el esfuerzo de pruebas es directamente proporcional al tamaño del software. El autor (Jones, 2008) parte de puntos de función para calcular el número de casos de prueba. Según Nageswaran (Nageswaran, 2001) para calcular los casos de prueba, toma como entrada los casos de uso y un factor de conversión basado en las características del entorno de pruebas. El método desarrollado por (Veenendaal & Dekkers, 1999) conocido como análisis de puntos de prueba. Calcula los puntos de prueba a partir de los puntos de función aplicando un factor ambiental y un factor de productividad para obtener el esfuerzo de pruebas.
Basados en casos de prueba	Este criterio tiene como base el número de casos de prueba y las características del entorno de las pruebas. Los autores Arahnay Borba (Aranha & Borba, 2009) realizan la estimación del esfuerzo de pruebas a partir de un factor de ajuste aplicado a la complejidad y tamaño de los casos de prueba. Este método estima el esfuerzo para casos de prueba de manera individual y los que se generan de manera automática. Xiaochum y sus colegas (Xiaochum et al., 2008) desarrollan un método basado en el algoritmo de aprendizaje y datos históricos.

Basados en actividades de prueba	Las aproximaciones en esta categoría dividen el proceso de pruebas en etapas que agrupan un conjunto de actividades de prueba. El esfuerzo total se obtiene operando el esfuerzo en cada etapa representado por medio de unas características que afectan el esfuerzo.

Fuente: Modificado a partir de (Torres, 2014)

De acuerdo con lo expuestos en la tabla 3, para desarrollo del modelo se toma el criterio de casos de prueba como base para la estimación.

De acuerdo con los autores (Borade & Khalkar, 2013), los puntos de casos de prueba cubren la estimación de las pruebas de caja negra y los puntos de función la estimación de las pruebas de caja blanca. Por lo tanto, exponen la necesidad de fusionar estas técnicas para mejorar el método de estimación, manteniendo la base del cálculo en los puntos de casos de prueba y aplicando un factor de ajuste derivado de las características previstas en la productividad de las pruebas.

La productividad se compone de dos elementos, el desempeño individual y el factor ambiental. El factor de desempeño individual es específico de cada probador y depende del conocimiento y la habilidad para desarrollar las actividades en su entorno organizacional. El factor ambiental describe la influencia externa del entorno en las actividades de prueba, incluida la disponibilidad de herramientas de prueba, el tipo de prueba y la disponibilidad de material de prueba entre otros (Borade & Khalkar, 2013).

Los autores (Chaudhary & Yadav, 2012), desarrollan la técnica denominada análisis de puntos de casos de prueba (TCPA por sus siglas en inglés) basada en los argumentos expuestos por (Borade, & Khalkar, 2013).

La técnica tiene en cuenta diferentes factores que influyen en la complejidad del diseño y ejecución de los casos de prueba. A continuación, se describe los pasos para su implementación:

- El probador una vez comprendidos los requerimientos y la estrategia de prueba a realizar, plantea una estimación conceptual de la complejidad de cada caso a desarrollar teniendo como base su experiencia y el análisis del requerimiento. Establece una valoración cualitativa (con tres juicios 1. Simple (S), 2. Media (M) y 3. Alto(A)) para los siguientes criterios descritos en la tabla 4:

Tabla 4. Valoración de Complejidad de casos de prueba

Codigo	Complejidad	No. Casos de prueba
1	simple	
2	medio	
3	alto	
Complejidad (sumatorio de Pesos)		

Fuente: Elaboración propia

La técnica establece que los puntos de casos de prueba(TCPs) se derivan de la complejidad descrita en la tabla 5:

Tabla 5. Pesos en TCPs por Complejidad

Tipo de Caso de Prueba	TCPs
Simple	6
Medio	8
Alto	12

Fuente: Elaboración propia

Con esta información se calcula los casos de prueba sin ajuste (TCPsa) y se calculan así:

TCPsa = (Nº casos S x TCPs S) + (Nº casos M x TCPs M) +(Nº casos A x TCPs A).

Para el ajuste de los puntos de los casos de prueba, se debe considerar un factor de ajuste denominado tareas administrativas o de gestión (TA) que no están consideradas en el diseño y ejecución de casos de pruebas como son: estrategia de pruebas, configuración del entorno de pruebas, reuniones, pruebas de regresión, configuración y parametrización de herramientas de pruebas entre otros. Inicialmente considerado este ajuste TA en 0.1, el cual puede variar de acuerdo con las condiciones de la organización. Para el ajuste se deben considerar dos escenarios:

- **Proceso automatizado:** En el proceso automático se deben tener en cuenta los factores del entorno. Los cuales se calculan teniendo con base a los pesos descritos en la tabla 6, así:

Tabla 6. Pesos de Ajuste

Código	Importancia del factor	Pesos de Ajuste
1	N/A	0
2	Impacto Bajo	0,05
3	Impacto Moderado	0,1
4	Impacto Medio	0,25
5	Impacto Significativo	0,5

Fuente: Elaboración propia

Partiendo de los valores definidos, asignar según el criterio del probador y las condiciones ambientales como se presenta en la tabla 7:

Tabla 7. Pesos de Ajuste

Factores	Pesos de Ajuste
Complejidad del dominio/tipo de aplicación	
Integración con otros dispositivos o Sistemas	
Soporte multi-idioma o multi-país	
Calidad de la documentación	
Total Peso de Ajuste	

Fuente: Elaboración propia

El factor de ajuste (FA) final es el siguiente:

Factor de Ajuste = 1 + Peso de Ajuste Total

Por tanto, el (TCPAca) con ajustado se calcula así:

TCPAca = TCPsa x (FA +TA)

- **Proceso manual:** De acuerdo en el tipo de prueba, se aplica un factor de corrección adicional a los TCPs calculados anteriormente, como se presenta en la tabla 8.

Tabla 8. Pesos de Ajuste

Tipo de Pruebas (TP)	Factor de ajuste (FA)
Pruebas funcionales	1
Usabilidad	1,15
Interfaces	1,25
Base de Datos	1,35
Rendimiento (manual)	1,35
Seguridad	1,40
Algoritmos de cálculo	1,40

Fuente: Elaboración propia

Por lo tanto, el (TCPMca) con ajuste se calcula así:

$$TCPMca = TCPsa \times (FA+TA)$$

Una vez calculado la complejidad de los puntos de casos de prueba, la técnica propone establecer un promedio de diseño y ejecución de casos de prueba por individuo basado en la experiencia.

2.2 Revisión de Literatura

En este aparte se caracterizan y analizan los estudios que pretenden identificar los factores que inciden en el desempeño del proceso de pruebas y las experiencias de gestión de conocimiento aplicadas al contexto de las pruebas de software.

2.2.1 Factores que inciden en las pruebas de software

Partiendo de algunas características inherentes del producto de software, como: la flexibilidad al cambio, la complejidad de las funcionalidades y las imprecisiones de los requisitos, las pruebas de software se tornan en una actividad compleja y difícil de desarrollar. En consecuencia, se puede afirmar que existen múltiples factores que impiden el éxito del proceso y algunas causas están asociadas a los variables que intervienen en

un proyecto de desarrollo, tales como: el tiempo, los recursos, la infraestructura, la cualificación de los profesionales, la metodología empleada, los requisitos del software entre otros (Kruchten, 2004).

Los autores (Kasurinen et al., 2009) realizan un estudio cualitativo en 26 organizaciones de software que implementan prácticas de pruebas y categorizan los factores que influyen en el desempeño de las pruebas de software, así:

- Herramientas de prueba, permite clasificar los problemas relacionados con la complejidad y el soporte en el uso de las herramientas.
- La automatización de pruebas, en esta categoría describe problemas de cualquier nivel o tipo de automatización.
- Gestión de conocimiento, en esta tipificación permite asociar los problemas y barreras existente para el intercambio y transferencia de conocimientos.
- Diseño del producto, en esta categoría agrupa las deficiencias y posibilidades relacionadas con la arquitectura del producto originadas en la fase de diseño que no generan falta de claridad en las pruebas del producto.
- La estrategia de prueba y planificación, en esta tipificación se describen los problemas causados por las prioridades de prueba y la falta de claridad para la selección de la estrategia de prueba.
- El personal de prueba, en esta categoría se asocian los problemas relacionados con el conocimiento y la falta de una conducta que propicie el uso de las prácticas de calidad en el proceso de pruebas.
- Los recursos de prueba, permite clasificar los problemas relacionados con la disponibilidad de herramientas, financiación o tiempo para completar las fases de prueba.

Los autores Fernández-Sanz y sus colegas (Fernández-Sanz et al., 2009), presentan los resultados obtenidos del estudio de percepción aplicado a 127 profesionales de la industria; tomando como referencia las 23 causas posibles que inciden negativamente sobre las prácticas de pruebas de software identificadas en el panel de investigadores (académica) y especialistas (industria) promovido por la red académica denominada REPRIS. Concluyen que los factores con mayor prevalencia son asociados a la falta de

conocimiento del personal y a las decisiones administrativas derivadas de los retrasos ocasionados en la construcción del producto.

Los autores Rodríguez, Pinheiro y Albuquerque (Rodríguez et al., 2010) identifican un conjunto de factores que dificultan la implementación del proceso de pruebas en siete organizaciones de software brasileñas, basado en un enfoque empírico articulado con una revisión previa de literatura. La investigación identifica los siguientes factores: una planificación inadecuada, poca importancia de las pruebas en la organización, baja calidad de los artefactos y no existe una cultura de calidad en el personal de pruebas, las herramientas no garantizan el desarrollo del proceso, desconocimiento del dominio del negocio, falta de recursos para el desarrollo de pruebas y no existe compromiso por parte de la alta dirección.

Los autores (Garousi & Zhi, 2013), realizan un estudio cualitativo aplicado a 246 probadores de diferentes organizaciones de software en Canadá. Cuyos hallazgos permiten identificar debilidades y oportunidades para mejorar el proceso de pruebas de software. Los elementos identificados son:

- La capacitación al personal de pruebas es indispensable.
- Las pruebas que mayor esfuerzo y dedicación reciben en los proyectos de desarrollo son las pruebas funcionales y unitarias.
- Los usos de las pruebas basadas en mutación tienen un particular interés en las factorías de pruebas.
- Las pruebas al final del proceso de desarrollo es un enfoque practicado en las pequeñas y medianas organizaciones de software.
- Las pruebas de aceptación aprobadas y el número de defectos encontrados por intervalo de tiempo se consideran las métricas de aseguramiento de calidad más importantes para liberar un producto.
- Los probadores en la mayoría de compañías canadienses son personas externas al equipo de desarrollo manteniendo una proporción promedio de 1 probador por cada 5 desarrolladores.

De acuerdo con (Kaner et al, 2008) (Natali et al. 2004) (Beer & Ramler, 2008), la experiencia y el conocimiento del equipo es un factor relevante en las pruebas de software. Dado que es una actividad cognitiva y no mecánica ni repetitiva que involucra varias

funciones mentales como el lenguaje, la imaginación y la percepción, entre otros (Soto et al, 2017a). La integración de la gestión de conocimiento al proceso de pruebas de software aumenta la eficiencia de las pruebas al permitir el reusó y la transferencia de conocimiento (Abdullah et al., 2011).

De acuerdo con los autores (Xue-Mei et al., 2009), los principales factores de gestión de conocimiento que afectan las pruebas de software son:

- Un bajo nivel de reusó del conocimiento.
- Existentes barreras que no permiten la transferencia de conocimiento.
- Los ambientes no propician las estrategias para compartir el conocimiento.
- Un alto nivel de rotación del personal de pruebas, generando reprocesos y problemas en el desarrollo de las actividades.

En la dimensión de la gestión de conocimiento, los autores (Mäntylä et al., 2012), plantean que el conocimiento del dominio incrementa la efectividad de los probadores.

Según (García et al., 2006) en los programas de mejora de procesos en el contexto del software se presentan dificultades que no permite una correcta implementación, tales como:

- Dificultad en el uso de los modelos de calidad por la limitada documentación y el alto costo de implementación
- El retorno de inversión en la implementación de los programas no es fácil de estimar.
- En algunas organizaciones (pequeñas y medianas) el costo de las actividades de mejora de proceso es elevado frente a los beneficios obtenidos a corto plazo.
- La falta de un proceso definido de pruebas.
- El personal presta mayor atención a las herramientas de automatización que al proceso de pruebas.
- Se asocian las actividades de mejora como una carga adicional al plan de trabajo.
- Falta de una cultura para asumir las actividades de mejora.
- Falta de compromiso de la alta dirección para respetar los tiempos estimados de pruebas.

Por su parte, la revisión de literatura sistemática realizada por los autores Afzal, Alone, Glocksien y Torkar (Afzal et al., 2016), plantean como factor determinante para la mejora de procesos de pruebas de software, la selección de un modelo de referencia que permita direccionar y priorizar a la organización en el propósito de mejora. Un principio para asegurar el proceso de mejora continua es el uso de un proceso estable y definido, para así, obtener los resultados esperados (Capote et al., 2013).

Aunque existen modelos de procesos para mejorar las prácticas de pruebas (CMMi, TMM, TMMI, TPI y TMap), el esfuerzo dedicado al desarrollo y mejora del proceso de pruebas software es escaso. Además, que la puesta en marcha de programas de mejora requiere de una fuerte inversión en recursos (personas, infraestructura, tiempo) que no todas las organizaciones pueden desarrollar (Esteban, 2012).

De acuerdo con Elberzhager, Münch y Assmann (Elberzhager et al, 2014), una alternativa prometedora para mejorar la eficacia y la eficiencia del aseguramiento de la calidad del software es el conocimiento obtenido del análisis realizado a los datos derivados de las actividades tempranas (inspecciones) de la detección de defectos, porque facilitan la definición de un enfoque de pruebas alineado a los riesgos identificados durante el proceso. Los autores (Janczarek & Sosnowski, 2015) también coinciden que los datos registrados en los activos del proceso de pruebas es la principal fuente para la predicción de defectos.

El enfoque basado en riesgos del producto es un factor determinante para mejorar el proceso y la calidad del producto (Felderer & Ramler, 2016). Siendo el enfoque de riesgos el principal insumo para la toma de decisiones del proceso (Erdogan et al., 2014) (Felderer & Schieferdecker, 2014).

Según los autores (Mäntylä & Itkonen, 2014), plantean las siguientes premisas que inciden en el cumplimiento del propósito de las pruebas de software:

- La detección de defectos es una actividad que requiere de múltiples roles y factores.
- las actividades para detectar defectos se desarrollan en dos escenarios: el explícito (aquellas actividades donde el único objetivo es la detección de

defectos) y el implícito (aquellas actividades cuyo propósito nos es propiamente la detección de defectos).

- Las actividades explicitas tienen mayor éxito en la detección de defectos estructurales del software y las actividades implícitas tienen un alto impacto en la detección defectos funcionales.
- Existe una baja documentación de los casos de pruebas en la detección de defectos de software.

Desde el nacimiento de la industria del software, existe un gran interés en cuantificar el esfuerzo en términos de tiempo y costo requerido para el desarrollo de cualquier tarea. Acorde con la afirmación de Tom DeMarco (DeMarco, 1986) ''No puedes controlar lo que no puedes medir", por lo tanto, la estimación del esfuerzo en las pruebas de software es un factor relevante para gestionar. Aún más relevante, son las métricas para evaluar el cumplimiento del proceso (Ferrer et al., 2013).

En las pruebas de software la automatización es una de las alternativas para garantizar la efectividad, frente al aumento de la complejidad del software. Según (Bertolino, 2007), uno de los desafíos es lograr la automatización del proceso de pruebas al 100%. Evidenciando que la mayor tendencia de esta clasificación es la generación automática de los casos de prueba, teniendo como premisa que la estrategia y la selección de los casos de prueba son el soporte de cualquier proceso de automatización.

2.2.2 Experiencias de gestión de conocimiento aplicadas a las pruebas de software

Las pruebas son un proceso extenuante y costoso que consume hasta el 50% de los costos totales del desarrollo de software (Bath & Veenendaal, 2014). Diferentes autores (Ossaba & Isaza, 2012) (Abdullah et al., 2011) afirman que la integración de la gestión de conocimiento al proceso de pruebas de software aporta de forma significativa al incremento de la efectiva del proceso, dado que las pruebas son una actividad altamente soportada en el conocimiento.

Existe una extensa literatura de la gestión de conocimiento en la ingeniería de software. Sin embargo, estas iniciativas son limitadas en el dominio especifico de las pruebas de software (De Souza et al., 2015).

Los autores (Xu-Xiang & Zhang, 2010) proponen un marco de mejora de las pruebas basado en PHVA (planear-hacer-verificar-actuar) independiente de metodología de desarrollo. Describe las actividades y medidas para evaluar el desempeño. El modelo propuesto integra la mejora de proceso y las pruebas de software sobre la estrategia de conocimiento explicito representado en los artefactos que gestiona el conocimiento del proceso.

La investigación desarrollada por los autores (Abdullah et al, 2011) definen el modelo arquitectónico de un sistema para gestionar conocimiento basado en la estrategia de las comunidades de práctica. La comunidad de practica integra los roles, como son: el diseñador, el programador y el probador del sistema.

Los autores (Li y Zhang, 2012) plantean un modelo de gestión del conocimiento para el reusó del conocimiento asociado a los casos de prueba basado en una representación ontológica. Realizan una caracterización del dominio de los casos de pruebas con el propósito de documentar e identificar los elementos relevantes de conocimiento que pueden aportar al reusó. Integra herramientas como mapas de conocimiento, repositorio de datos, ontologías de dominio para la captura del conocimiento tácito y uso del conocimiento explícito.

De acuerdo con los autores (Desai y Shah, 2011), sustentan el estudio a partir de la problemática existente en los procesos para adquirir, codificar y almacenar los conocimientos asociados en el dominio de las pruebas de software. Desarrollan una plataforma de software para darle solución a la problemática planteada. El sistema de gestión de conocimiento integra los procesos: crear, explorar, refinar, almacenar, gestionar y diseminar el conocimiento, facilitando el acceso a las experiencias de los individuos.

Los autores (Andrade et al, 2013), implementan un sistema de lecciones aprendidas, partiendo de la estructuración del conocimiento para su almacenamiento y recuperación.

Se centra en el propósito de crear una memoria organizacional basada en lecciones aprendidas y soportada en un repositorio organizacional.

El estudio de los autores (Nogeste y Walker, 2006), describe el desarrollo de un sistema de gestión de conocimiento basado en la efectividad de las pruebas de regresión con la captura y almacenamiento del conocimiento tácito para su posterior reusó en los procesos de planificación de los proyectos de pruebas.

Los autores (Kerkhof et al, 2003) presentan un modelo para gestionar el conocimiento con el propósito de consolidar una estrategia de aprendizaje organizacional para promover la transferencia de conocimiento. Los procesos de conocimiento son considerados competencias y expone las estrategias que tienen un enfoque corporativo porque se centra en los procesos y no el uso del conocimiento.

Los autores (De Souza et al., 2015), se enfocan en una estrategia de reusó basada en patrones ontológicos. También desarrollan una revisión sistemática de literatura en la aplicación de la gestión de conocimiento a las pruebas de software, incluyendo en las investigaciones de ingeniería de conocimiento en la definición de la gestión de conocimiento. En el análisis realizado concluye que las iniciativas orientadas al plano organizacional del conocimiento no han sido completamente validadas, y en la gran mayoría se incorporan el uso de técnicas de ingeniería de conocimiento para resolver problemáticas técnicas del proceso de pruebas.

2.3 Análisis y discusión.

En este trabajo se analiza el concepto de conocimiento que soporta la diferencia efectiva entre información y conocimiento. Donde el conocimiento no se encuentra en el contenido, estructura o utilidad de la supuesta información, sino que el conocimiento es la información procesada (relativa a procedimientos, conceptos e ideas) por las personas a partir de experiencias para ser usada en la generación de valor y en pro del aprendizaje organizacional. Aunque las organizaciones de software buscan convertirse en organizaciones que aprenden, muy pocas tienen éxito en lograrlo (Jennex & Olfman, 2004).

La gestión de conocimiento se asocia a las estrategias de mejora de procesos y al aprendizaje organizacional, pero dada la diversidad de definiciones es importante determinar su alcance.

En el contexto de los sistemas expertos, lenguajes formales e inteligencia artificial, el concepto gestión de conocimiento ha sido tomado como sinónimo de ingeniería de conocimiento. Sin embargo, el significado de las palabras “gestión” e “ingeniería” presentan diferencias funcionales y objetivas. Por lo tanto, el concepto de 'gestión' se instancia para "desarrollar los procesos y actividades tendientes al cumplimiento de un objetivo en el contexto organizacional” y el concepto “ingeniería” aborda la aplicación de "conocimientos y técnicas del saber científico".

En consecuencia, la “gestión del conocimiento” establece los lineamientos que el proceso debe seguir, mientras que la 'ingeniería del conocimiento' implementa las técnicas y herramientas necesarias para cumplir con los lineamientos propuestos (Newman, 1996).

Por otra parte, Earl (Earl, 2001) articula las iniciativas de gestión de conocimiento en el ámbito ingenieril cuando los procesos y flujos de conocimiento son el producto clave para garantizar la transferencia y almacenamiento. Siendo las prácticas para la construcción del producto, la gestión del recurso humano y el entrenamiento del personal procesos claves de la gestión de conocimiento en el contexto de la ingeniería de software (Galvis & Sánchez, 2013).

Entonces, se puede afirmar que los factores que afectan las pruebas de software son diversos, pero de acuerdo con las causas identificadas se pueden agrupar en dos categorías: la gestión del conocimiento y la gestión del proceso.

Teniendo como premisa que las pruebas de software son una actividad intensiva en conocimiento, se puede identificar que los factores que con mayor relevancia afectan el éxito de las iniciativas en este campo, son:

- Barreras para la transferencia de conocimientos.
- Pérdida de conocimiento.
- Baja tasa de reusó del conocimiento.
- El conocimiento no se comparte adecuadamente.

- El conocimiento no se considera adecuadamente para procesos de planificación.

Estos factores propician la repetición de los defectos, aunque existan individuos en la organización con el conocimiento y la experiencia requerida para prevenirlos (Andrade et al., 2013). Aumentando las barreras existentes para consolidar el aprendizaje organizacional, que en principio es difícil de lograr porque la gran mayoría de conocimientos en la organización son tácitos, dado que se originan en la experiencia y por sus características son complejos de articular.

Dado lo complejo y costoso del proceso de pruebas (Harrold 2000) (Bath & van Veenendaal 2014), el proceso se convierte en el soporte fundamental de la efectividad en la detección de defectos y entre los factores que la impacta se encuentran:

- Falta de compromiso de la alta dirección.
- Modelo y metodologías existentes rígidas y complejas de implementar en las organizaciones de software.
- Ausencia o baja calidad de los artefactos de desarrollo.
- La ausencia de un proceso definido.

El compromiso de la alta dirección es un elemento esencial en cualquier iniciativa empresarial de mejora de procesos, sin embargo, el tamaño de la organización y la estructura del equipo de pruebas inciden en la efectividad del proceso. La metodología o modelo para el desarrollo del proceso de pruebas es un factor que asegura en gran medida el éxito del proyecto, alineado a la política y estrategia establecida en los proyectos de pruebas. Actualmente con el estándar ISO 29119, se unifican los conceptos y criterios de la aplicación de las pruebas, resolviendo el interrogante ¿qué se debe hacer? pero sin resolver ¿el cómo se hace? siendo una oportunidad para soportar investigaciones en el campo de la mejora de procesos y la gestión de conocimiento.

Otro elemento, en esta dimensión es la inadecuada planeación del proceso, focalizada en la dificultad para identificar los riesgos que generan una selección inapropiada de la estrategia. Los activos del proceso de construcción son insumos primordiales para el proceso de pruebas de software, pero en algunos casos estos resultan incompletos y defectuosos afectando la ejecución de las pruebas. Esta situación, es reiterativa en la

tercerización de pruebas y diferentes autores proponen alternativas como: el trabajo cooperativo entre desarrolladores y probadores, estudios en análisis de riesgos, reusó de experiencias y la optimización de pruebas exploratorias. Sin embargo, se debe explorar procesos de mejora basado en conocimiento y experiencias que permitan estimar y planear con información histórica.

En la gestión por procesos la mejora y el uso de buenas prácticas es un propósito implícito, siendo la conducta individual (Mäntylä & Itkonen, 2014) y el uso de prácticas definidas en los modelos de referencia de calidad un factor clave en los programas de mejora de procesos (Capote et al., 2013). Una situación de los modelos de referencia existente en pruebas de software es que resuelve el interrogante ¿qué hacer? pero no ¿cómo hacerlo? (Watkins & Mills 2010).

Se puede evidenciar que la tendencia en investigación en el dominio de pruebas de software tiene una mayor influencia hacia las iniciativas enfocadas a los aspectos técnicos y de proceso, dejando a un lado los temas relacionados con la conducta y hábitos de los miembros del equipo de pruebas. También existe la misma orientación en las iniciativas analizadas que abordan los modelos de gestión de conocimiento, porque se excluye la cultura del personal de las capacidades organizacionales.

Particularmente, el actual enfoque de gestión de conocimiento no solo se centra en capturar y almacenar en repositorios el conocimiento, sino enfatiza en considerar el conocimiento como un fenómeno complejo relacionado con aspectos humanos y culturales (Hosseingholizadeh, 2014).

Partiendo de diferentes autores Pawlosky, Castañeda y Fernández Rios, Kim, Crossan, Lane, White, Zapata, Castañeda y Pérez, se concluye que: “el aprendizaje individual es la base fundamental del aprendizaje organizacional”. En consecuencia, existe la necesidad de conocer cómo aprenden los individuos. Siendo uno de los mecanismos más usados en el aprendizaje organizacional las instrucciones o indicaciones. En términos de ingeniería de software se denominan: procesos de software. De acuerdo con Zapata (Zapata, 2005), expresa que entre más claro sea el proceso mayor probabilidad existe de que ocurra un aprendizaje significativo.

En el dominio de pruebas es una necesidad desarrollar iniciativas que estandarice el proceso y estructure una estrategia de gestión de conocimiento a partir del recurso más relevante de la organización, que es el individuo.

Por tal razón, existe una brecha de investigación centrada en la cultura organizacional con respecto a las prácticas y la conducta de los individuos que integran los equipos de pruebas, ya que están alineadas con las tendencias de investigación identificadas por (Galvis & Sánchez, 2013) (Fernández-Sanz & Misra, 2011) (Fernández-Sanz et al., 2009).

Según los autores (Fernández-Sanz & Misra, 2011), la eficacia organizacional, la mejora de la eficiencia y la calidad dependen en gran medida de la aplicación del conocimiento y el uso práctico. En consecuencia, es evidente que cualquier esfuerzo para implementar programas de gestión de conocimiento está alineado a la estrategia organizacionales de mejora de procesos.

La gestión de conocimiento es el principal componente en las iniciativas de mejora de procesos y en el ámbito de la ingeniería de software es útil en la definición y adaptación de procesos de software (Aurum, 2008).

De acuerdo con Zapata, concluye que toda iniciativa de gestión de conocimiento que no incorpore el aprendizaje individual tiene el riesgo de solo realizar gestión de información (Zapata, 2005).

De acuerdo con los autores Pawlosky, Fernández y Castañeada, Kim, Crossan, Lane y White, identifican como la ruta más corta para institucionalizar conocimiento en el marco del aprendizaje organizacional, el enfoque individual por encima del grupal y organizacional en general, dado que el aprendizaje individual incide en los factores: cognitivo y conductual para generar una cultura organizacional para el aprendizaje y por consiguiente en la mejora de los procesos.

Concluyendo que sin aprendizaje individual existe el riesgo que no exista gestión del conocimiento sino sólo gestión de información.

3 Capítulo: Gestión de conocimiento en el marco de la mejora de procesos de pruebas

Este capítulo busca dar solución al objetivo específico 3 de la tesis que corresponde a: determinar los activos de conocimiento basado en las teorías gestión del conocimiento para la mejora del proceso de software, partiendo de los hallazgos encontrados en la revisión de literatura del capítulo 2, se puede establecer que el enfoque para el desarrollo del modelo propuesto se centra en el aprendizaje organizacional y la mejora de procesos teniendo como premisa que las iniciativas de gestión de conocimiento aplicadas a las pruebas de software no abordan estrategias centradas en el individuo. Siendo los miembros del equipo de pruebas elementos fundamentales para el aprendizaje organizacional y la mejora continua. Mediante un proceso definido y medible, se busca que el individuo pueda lograr el aprendizaje basado en sus experiencias e incorporar las mejores prácticas como hábitos a las actividades y compromisos en el entorno de las pruebas de software.

Para lograr este propósito se presentan los aspectos de conocimiento asociados a la mejora de procesos que fundamentan el modelo de gestión de conocimiento centrado en el individuo.

3.1 La gestión de conocimiento y la mejora de procesos de software

De acuerdo con a Capote (Capote et al., 2013), los programas de mejora de procesos de software generan grandes volúmenes de información los cuales deben ser debidamente

gestionados para alcanzar el impacto previsto en la organización. Por lo tanto, la mejora de procesos de software implícitamente se concibe como una actividad intensiva en conocimiento.

De acuerdo con Chugh y sus colegas (Chugh et al., 2015), la gestión de conocimiento en los programas de mejora de procesos, tienen las siguientes características:

- El proceso de creación de conocimiento en el marco de los programas de mejora de procesos ocurre principalmente en el nivel individual y se escalan después en el nivel grupal.
- Las iniciativas de mejora de procesos deben ser definidas a través de métodos.
- Los métodos proporcionan una estructura a los procesos de software, facilitando la comunicación entre los proyectos y la captura de las experiencias relacionadas con el proceso y los ciclos para la retroalimentación.
- Los métodos juegan un papel importante en el aprendizaje organizacional.

Para Santos y sus colegas (Santos et al, 2007), los conocimientos y experiencias se generan tanto a nivel individual como grupal. Para los investigadores (Glazer et al, 2008) la madurez organizacional parte de la capacidad de los individuos para lograr un aprendizaje efectivo.

A nivel personal la mejora de procesos según a Humphrey (Humphrey, 2001), se caracteriza por:

- El desempeño individual está dado por el conocimiento, la disciplina y el compromiso del individuo.
- La adopción de un proceso facilita el cumplimiento de los compromisos.
- La medición y el análisis de los datos derivados del desempeño permite identificar alternativas de mejora.
- Las lecciones aprendidas son fundamentales para la mejora de las prácticas personales.

Las practicas personales asociadas a la mejora de procesos de software, permite que el ingeniero de software defina cómo administrar la calidad de sus productos y cómo hacer compromisos que ellos puedan cumplir (Soto & Reyes, 2010).

La implementación de enfoques individuales en el contexto del software permite elevar los niveles de productividad y asertividad en el desempeño de los profesionales (Vargas & Soto, 2012).

Las estrategias aplicables para mejorar los indicadores de productividad y asertividad en el desarrollo de software aportan al aprendizaje significativo de los individuos (Vargas & Soto, 2012).

En este sentido, el aprendizaje individual es una condición necesaria pero no suficiente para la existencia de aprendizaje organizacional, según Swieringa (Swieringa, 1995). Dado que el aprendizaje individual es la base fundamental para desarrollar los procesos de aprendizaje como el grupal y organizacional (Simon,1991), (Nonaka et al., 1994), (Kim, 1995), (Bain, 1998), (Crossan et al., 1999) (Moreno et al. 2001).

3.2 Integración de Conocimiento y aprendizaje en la mejora de proceso

3.2.1 Conocimiento experiencial

Según la descripción desarrollada por (Mazoni, 2010), el modelo de aprendizaje experiencial de Kolb acorde al marco de los procesos de software está compuesto por los siguientes elementos:

- La Experiencia Concreta. Está asociada con el uso de las prácticas y procesos en un proyecto “el hacer”
- La Observación Reflexiva. Se da una vez finalizado el paso anterior y consiste en el análisis acerca de la experiencia de la aplicación del proceso “Reflexión”.
- La Conceptualización Abstracta. De manera subsiguiente a la reflexión, se generan conceptos, modelos e ideas que sustentan la generación de propuestas y sugerencias para la mejora del proceso “Abstracción”.

- La Experimentación Activa. Consiste en la validación de las nuevas prácticas y procesos, como parte de las mejoras identificadas en el paso anterior “Decidir”

3.2.2 Conocimiento SECI

De acuerdo a la interpretación de autor (Mazoni, 2010), presentada a partir del trabajo realizado por Arent y Norbjerg (Arent & Norbjerg, 2000), sobre el ciclo genérico de mejora de procesos de software en el marco del modelo de creación de conocimiento de Nonaka y Takeuchi (Nonaka & Takeuchi, 1995) describe los siguientes términos:

- Socialización: Ocurre cuando los individuos aplican las prácticas descritas en el proceso de software durante la ejecución de un proyecto.
- Externalización: Una vez aplicado el proceso de software, los individuos explicitan los conocimientos tácitos adquiridos durante el desarrollo de la aplicación del proceso. Esta orienta a producir sugerencias y propuestas de mejora.
- Combinación: Las propuestas de mejoras son incorporadas al proceso previamente analizadas para generar una modificación al proceso de software.
- Internalización: Es conocida como el aprender haciendo y ocurre cuando el individuo se familiariza con el nuevo proceso.

3.2.3 Aprendizaje organizacional desde el enfoque individual

Partiendo de la premisa que el aprendizaje organizacional se soporta en los procesos de aprendizaje individual, se presentan las principales características de este enfoque.

- El aprendizaje individual es un proceso que el individuo genera conocimiento a partir de la interpretación y asimilación de la información tácita y/o explícita (Moreno et al., 2001).
- Según los autores (Moreno et al., 2001), las capacidades cognitivas del individuo se incrementan con el aprendizaje individual.
- El aprendizaje individual influye en la conducta partiendo del hecho que el conocimiento generado modifica las estructuras internas de pensamiento (Fiol & Lyles, 1985), (Stata & Almond, 1989), (Swieringa & Wierdsma, 1992), (Garvin, 1985) y (Kim, 1998).

- Se mejora el comportamiento humano a partir del incremento en el desempeño a partir del reusó del conocimiento experiencial obtenido (Moreno et al., 2001).

A partir del modelo mejorado por Castañeda & Pérez (Castañeda & Pérez, 2005) y de la base de Crossan, Lane y White (Crossan et al., 1999) se incorpora los cuatros procesos del aprendizaje individual en el contexto de la mejora de procesos de software, así:

- Atención: Identificar las prácticas y medidas asociadas al proceso de software.
- La Retención: Concebir y analizar las causas y efectos de la aplicación del proceso para identificar propuestas de mejora.
- Producción: Apropiar las buenas prácticas al proceso de software.
- Motivación: Retroalimentar y evaluar permanente las prácticas para la optimización del proceso.

3.2.4 Estrategia de mejora

De acuerdo con Humphrey (Humphrey, 2001), PSP es un enfoque disciplinado y guiado por un proceso iterativo y escalonado. Este marco orientado al proceso personal se sustenta en los principios de la mejora continua y comprenden los pasos descritos en la figura 11, así:

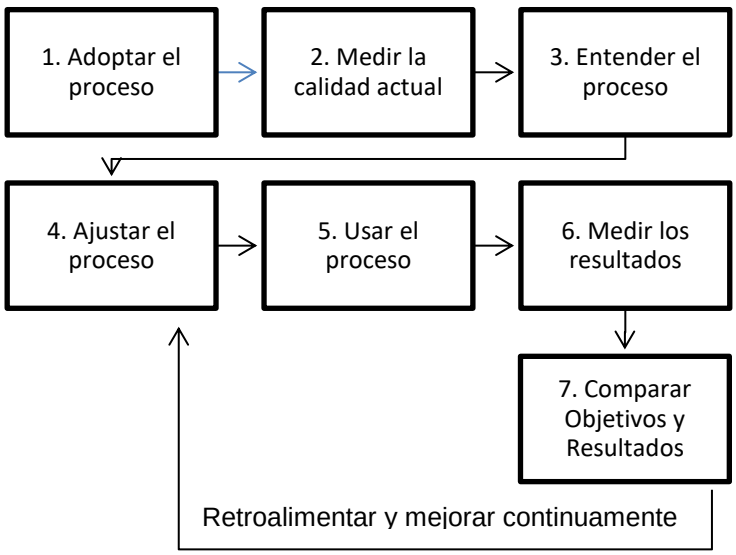


Figura 11. Principios de Mejora Continua
Modificada por los autores a partir (Humphrey, 2001)

La estrategia parte de la definición del objetivo o propósito de calidad.

1. Adoptar el proceso: Aplicar las actividades y prácticas del ciclo de vida del proceso.
2. Medir calidad actual: Acorde con el objetivo de calidad y las métricas establecidas se realiza el registro de los datos requeridos para valorar el desempeño actual del proceso.
3. Entender el proceso: analizar las características y criterios del proceso actual para establecer las posibles alternativas de mejora.
4. Ajustar el proceso: Incorporar las alternativas de mejora identificadas.
5. Usar el proceso: Aplicar el proceso con las mejoras incorporadas.
6. Medir los resultados: Tomar las medidas del nuevo proceso.
7. Comparar objetivos y resultados: Contrastar las metas de calidad y las medidas obtenidas del proceso ajustado para determinar los aciertos y dificultades.

Retroalimentar y mejorar continuamente incorporando los ajustes identificados en el punto anterior.

3.2.5 Visión integradora del aprendizaje individual y la mejora de procesos

A nivel del aprendizaje organizacional, se plantea una estrategia de mejora que integra lo los procesos del aprendizaje individual asociados a las fases de los modelos de creación de conocimiento de Nonaka y Takeuchi (Nonaka & Takeuchi, 1995) y el modelo de aprendizaje basado en experiencia de Kolb (Kolb, 1984) en el contexto de los procesos de software.

Por consiguiente, la tabla 9 describe la relación entre la estrategia de mejora de procesos instanciada para el modelo propuesto de pruebas de software en relación con el aprendizaje individual y los modelos de creación de conocimientos y de aprendizaje experiencial.

Tabla 9. Estrategias de mejora de procesos aplicadas al individuo.

Aprendizaje Individual	Procesos del modelo SECI	Procesos del modelo Experiencial	Pasos de la estrategia de mejora
Atención	Socialización	Experiencia Concreta	1
Retención	Externalización	Observación Reflexiva	2, 3,6 y 7
Producción	Combinación	Conceptualización Abstracta	4
Motivación	Internalización	Experimentación Activa	5

Fuente: Elaboración propia

3.3 Caracterización de los modelos de mejora aplicadas a las pruebas de software

De acuerdo a los modelos de calidad existentes en el ámbito de pruebas de software, estos no describen las actividades y procesos para guiar a los miembros del equipo de pruebas (Esteban, 2012;Watkins & Mills 2010).

Los modelos de mejora de procesos existentes en las pruebas de software se basan en conocimiento explícito. En otras palabras, el enfoque dominante es la codificación del conocimiento. De hecho, ninguno de los modelos hace referencia explícita a la gestión del conocimiento. Sin embargo, instancian técnicas como: repositorios organizacionales, sistemas para gestionar los activos del proceso, registros de lecciones aprendidas, artefactos de conocimiento resultantes de actividades de pruebas de software y conocimiento embebido en los procesos organizacionales.

En la tabla 10 se caracterizan los modelos de mejora más relevantes de las pruebas de software. Partiendo de las actividades de gestión de conocimiento que integran cada modelo y la clasificación según el enfoque de implementación definido por Earl (Ear,2001).

Tabla 10. Modelos de mejora vs. Escuelas de KM

Modelo de mejora de procesos		Escuelas de KM según (Earl, 2001)					
		Sistemas	Cartográficas	Ingenieril	comercial	espacial	Estratégica
Procesos TMMi	Organización de pruebas.	x					
	Programas de formación y entrenamiento.			x			
	Prevención de defectos	x					
	Gestión del ciclo de vida			x			
Practicas TestPAI	Identificar y establecer recursos.	x					
	Establecer programas de entrenamiento.			x			
	Determinar acciones correctivas	x					
Areas Claves TPI	Modelo del Ciclo de Vida			x			
	Métricas			x			
	Funciones de Pruebas y capacitación			x			
	Gestión de defectos	x					
	Informes			x			

Fuente: Elaboración propia

Los modelos TMMi y TPI incluyen el concepto de repositorio del conocimiento dentro de los procesos de organización de pruebas y modelo de ciclo de vida. Las actividades de capacitación y entrenamiento de personal son incluidas en los tres modelos analizados. Los modelos se ubican en las corrientes definidas como Ingenieril y de Sistemas, dado que se soportan en el enfoque de procesos y la naturaleza tecnológica del dominio.

Partiendo de las capacidades de conocimiento definidas por los autores (Gold et al., 2001) se caracterizan las áreas claves que integran los marcos de calidad seleccionados para conocer la orientación a nivel de infraestructura y proceso de KM. La relación entre los procesos de Calidad y las capacidades de KM se muestra en la Tabla 11.

Tabla 11. Procesos de Calidad vs. Capacidades de KM.

Modelo de mejora de procesos para pruebas de software		**Capacidades de KM (Gold et al., 2001).**						
		Tecnología	**Cultura**	**Estructura**	**Adquisición**	**Conversión**	**Aplicación**	**Protección**
Procesos TMMi	Organización de pruebas.	x		x		x		
	Programas de formación y entrenamiento.				x			
	Gestión del ciclo de vida	x		x		x		
	Prevención de defectos	x				x	x	
Practicas TestPAI	Identificar y establecer recursos.	x		x				
	Establecer programas de entrenamiento.				x			
	Determinar acciones correctivas					x		
Areas Claves TPI / TPI NEXT	Modelo del ciclo de vida	x		x		x		
	Métricas					x		
	Funciones de Pruebas y capacitación	x			x			
	Gestión de defectos	x				x	x	
	Informes					x		

Fuente: Elaboración propia

Las capacidades organizacionales de KM de los modelos analizados se acentúan en la capacidad tecnológica a nivel de infraestructura y se soportan en las capacidades: estructura, adquisición y conversión. Esta caracterización de los modelos resulta coherente con la escuela de sistemas, al presentar actividades en los aspectos relacionados con la capacidad de la infraestructura tecnológica y la capacidad del proceso de conversión del conocimiento.

Además, otro elemento importante es que todos los modelos tienen, al menos, un proceso asociado con la estructura organizacional dada el enfoque de procesos que los soporta.

Además, el proceso de aplicación se soporta en los procesos de adquisición y tecnología y está asociado con la prevención de defectos de los modelos TMMi y TPI. También es importa resaltar que ninguno de los modelos de referencia implementa procesos para generar una cultura organizacional de mejora y mecanismos de protección de la información.

4 Capítulo: Modelo y Validación

En este capítulo se presenta el modelo de gestión de conocimiento aplicado a las pruebas de software instanciando un enfoque personal que incorpora el aprendizaje y la mejora continúa dando respuesta al objetivo específico número cuatro y su posterior validación que da respuesta al objetivo específico número cinco.

El modelo aborda la problemática planteada y contempla las brechas existentes identificadas en la revisión de la literatura. En consecuencia, el modelo se fundamenta en los siguientes criterios:

1. El enfoque individual es un elemento relevante para lograr la efectividad de la gestión de conocimiento en el marco de la mejora continua y el aprendizaje en las pruebas de software.
2. La implementación de un proceso definido y medible en el dominio de pruebas de software propicia la gestión de conocimientos y experiencias para incorporar los principios de la mejora continua en las pruebas de software.
3. La estrategia de mejora continua adopta buenas prácticas del marco de procesos TMMi e instancia una implementación escalonada e iterativa basada en el proceso de pruebas.
4. Incluye el uso de las lecciones aprendidas como instrumento en la ejecución de proyectos y no solo a partir de proyectos finalizados.
5. Incorpora procedimientos y artefactos para la gestión del conocimiento y la experiencia.

El modelo propuesto instancia la estructura e intencionalidad del proceso personal de software, en el contexto de pruebas de software implementando las prácticas y principios básicos de la gestión de conocimiento para propiciar el autoaprendizaje del conocimiento a partir de su propio desempeño, como estrategia base para el aprendizaje organizacional. Este modelo se denomina proceso de pruebas personal basado en conocimiento (PTPk).

Para la validación se realizó la valoración del impacto de la aplicación del modelo a través de dos grupos de la asignatura de pruebas de software adscritos al programa de ingeniería en software de una institución educación superior, seleccionando un grupo para la aplicación del modelo denominado grupo experimental y otro de referencia denominado grupo de control.

4.1 Modelo PTPk

El modelo PTPk se centra en gestionar el desempeño de los integrantes del equipo de pruebas introduciendo buenas prácticas al proceso y así generar una cultura de autoaprendizaje y mejora continua.

El PTPk, se centra en las prácticas de trabajo de los probadores a nivel individual, con el propósito de generar un proceso definido y medible que permita adquirir compromisos mesurables y hacer un seguimiento al trabajo. Permitiendo una mejorar continua a partir del análisis de los resultados de cada trabajo y reflexionar para generar mejora en el proceso del proyecto siguiente.

4.1.1 Marco de Conocimiento

El modelo está estructurado en tres niveles de abstracción del conocimiento como se observa en la figura 12. El primer nivel se hace uso del término concepto y habilidad. El concepto se usa para describir los aspectos intelectuales como: la información, los hechos, la terminología y los componentes tecnológicos. El término habilidad se refiere a la capacidad de un probador para aplicar conceptos a la ejecución de una tarea.

El segundo nivel contempla el núcleo de conocimiento que integra los conceptos y las habilidades, y se refiere a la comprensión de un rango de conceptos y experiencias adquiridas durante la ejecución del modelo. Las áreas de conocimiento conforman el área de competencia que es el mayor nivel de abstracción.

El tercer nivel establece el área de competencia que agrupa un conjunto de núcleos de conocimiento relacionados y provee al individuo de la capacidad procedimental e intelectual para el buen desempeño de las pruebas de software.

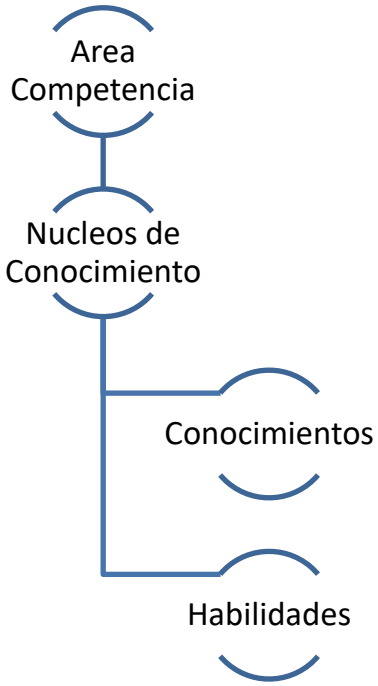


Figura 12. Niveles de abstracción del conocimiento.

Fuente: Elaboración propia

Las áreas de competencias que integran el modelo son:

- Gestión del proceso: Esta área se refiriere al marco operacional del proceso de software personal denominado PSP. En la tabla 12 se describen los componentes del área de competencia de la gestión del proceso.
- Gestión de proyectos: Hace referencia al conjunto de buenas prácticas para gestionar proyectos denominado PMBOOK. En la tabla 13 se describen los componentes del área de competencia de la gestión de proyectos.

- Gestión de pruebas: Esta dimensión se refiere al marco de procesos que integra las pruebas de software descrito en el estándar ISO/IEC 29119. En la tabla 14 se describen los componentes del área de competencia de la gestión de pruebas.
- Mejora de procesos de pruebas: Esta área se refiere al modelo de madurez denominado TMMi. En la tabla 15 se describen los componentes del área de competencia de la mejora de procesos de prueba.

Tabla 12. Área de gestión del proceso

Núcleos de Conocimientos	**Definición**	**Conocimientos y Habilidades**
Proceso Definido	Conocimiento que describe los conceptos y habilidades necesarios para crear, estabilizar, y usar procesos definidos.	(C) Proceso, actividad, fundamentos para definir un proceso, plan, proceso personal, proceso habilitante y operacional, fases del proceso, desarrollo incremental. (H) Uso de procesos de alta calidad.
Componentes del Proceso	Conocimiento que describe los elementos que integran un proceso personal y el marco para organizar el trabajo del proyecto.	(C) Elementos de proceso, guías, plantillas, métricas, estándares para la toma de datos y uso en el proceso técnico. (H) Uso de guías, formas, métricas y estándares de procesos.
Estadística	Conocimientos para analizar e interpretar la planificación y mejora de los procesos de ingeniería de software personal.	(C) Media, varianza, desviación estándar. (H) Análisis y proyecciones.

Fuente: Elaboración propia

Tabla 13. Área de Gestión de proyectos

Núcleos de Conocimientos	Definición	Conocimientos y Habilidades
Gestión de Alcance	Conocimientos necesarios para comprender el tamaño del proyecto.	(C) Estimación, restricciones, supuestos, paquetes de trabajo, esfuerzo, esquema de composición de tareas, requisitos. (H) Estimación y descomposición de tareas.
Gestión de Tiempo	Conocimientos asociados al desarrollo de cronogramas.	(C) Tareas o actividades, técnicas de revisión y evaluación de proyectos (PERT), modelo para trazar ruta crítica (CPM). (H) Secuenciación de tareas, asignación de tiempos y programación de proyectos.

Fuente: Elaboración propia

Tabla 14. Área de Gestión de pruebas

Núcleos de Conocimientos	Definición	Conocimientos y Habilidades
Procesos de Pruebas	Conocimientos necesarios para comprender los procesos y actividades de las pruebas de software.	(C) Estrategia, políticas, fases y actividades de pruebas, entorno de pruebas. (H) Gestión del proceso de pruebas.
Documentación de pruebas	Conocimientos de los elementos y características de los artefactos asociados a las pruebas de software.	(C) Activos del proceso (plan de pruebas, diseño de casos de uso, informes entre otros). (H) Gestión documental de las pruebas de software.

Técnicas de pruebas	Conocimientos que se refieren a la técnicas y aspectos de pruebas.	(C) Tipos de pruebas (caja blanca y caja negra), métodos de pruebas basadas en especificación, estructura y experiencia. (H) Uso de técnicas de pruebas.

Fuente: Elaboración propia

Tabla 15. Mejora de procesos de pruebas

Núcleos de Conocimientos	Definición	Conocimientos y Habilidades
Área de procesos	Conocimientos para comprender la gestión de procesos de madurez en las pruebas de software.	(C) Marco de procesos, procesos de madurez, objetivos y áreas claves. (H) Gestión del proceso mejora a partir de las áreas claves de proceso.
Practicas	Conocimientos asociados a las buenas prácticas en marco de la mejora de procesos de pruebas.	(C) Prácticas genéricas y específicas. (H) Adaptación e implementación de las buenas prácticas al proceso de pruebas.

Fuente: Elaboración propia

4.1.2 Estructura del modelo

El enfoque individual instancia de manera progresiva e incremental practicas definidas en el TMMi. El modelo implementa prácticas como:

- Establece un plan para cada proyecto.
- Registro de tiempos, incidentes y problemas.

- Análisis de datos para la toma de decisiones en proyectos futuros.
- Definir propósitos de mejora al final de cada proyecto.

El modelo se implementa en cuatro niveles como se observa en la fig. 13.

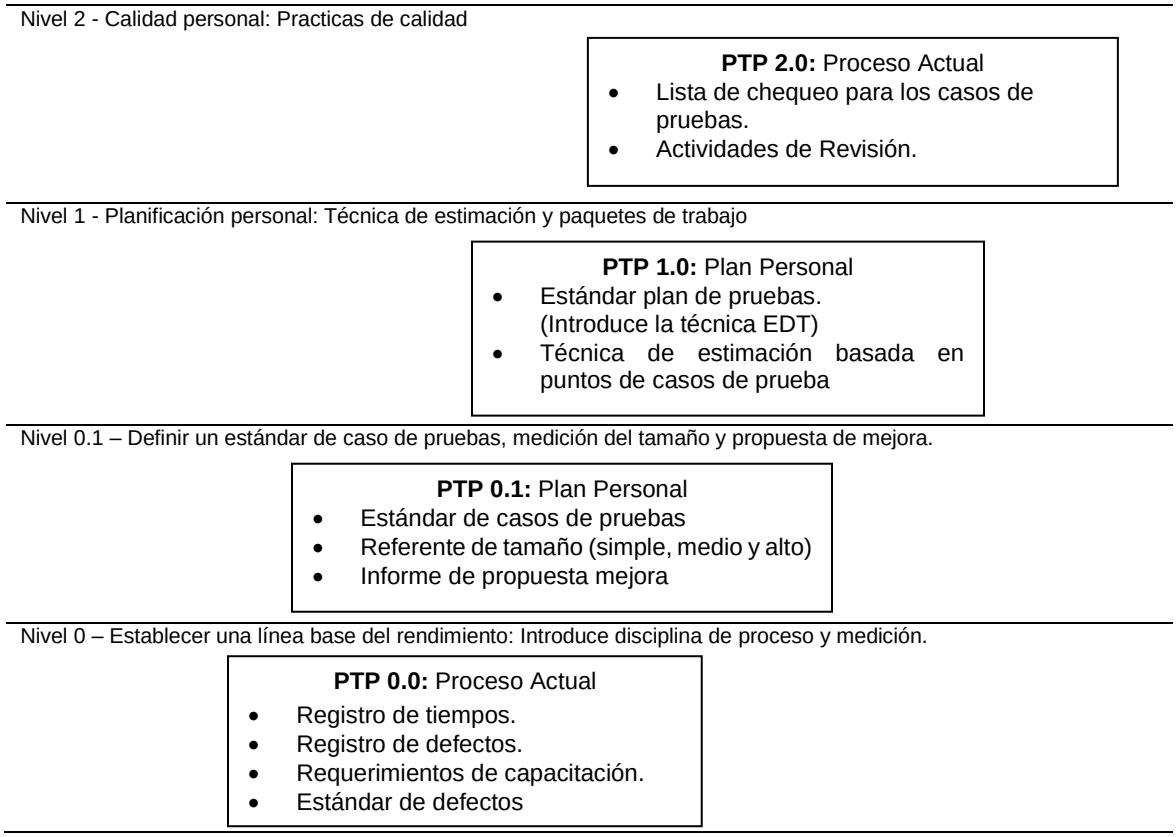


Figura 13. Estructura por niveles del modelo.
Fuente: Elaboración propia

Nivel 0. Establecer una línea base del rendimiento.

En este nivel se inicia el registro de las métricas básicas del proceso de pruebas, como son el registro de tiempos y defectos con base a un estándar de clasificación. Iniciando por el registro de la estimación del tiempo (tiempo planeado expresado en horas hombre) del proyecto acorde a su experiencia.

En el nivel 0, el probador desarrolla un proyecto con casos de pruebas (complejidad simple, media y alta). Determinando una línea base del rendimiento a partir de las medidas registradas como son: tiempo y defectos. También adopta las practicas del proceso para ejecutar el alcance propuesto.

En el segundo nivel, se incluyen los elementos como el estándar de pruebas y la complejidad de los de los mismos.

Para medir el desempeño del proceso de pruebas es necesario establecer un referente de medida. El estándar de casos de prueba permite establecer el tamaño del proyecto. Para homogeneizar el esfuerzo del caso de prueba se propone un estándar que describa los elementos y requisitos para su posterior diseño y ejecución.

Dada la dinámica de evaluación de diferentes aspectos del software se proponen casos de prueba por cada aspecto a evaluar. Se puede afirmar que los casos de prueba tienen diferentes niveles de complejidad acorde con el aspecto de la prueba a evaluar.

Por lo tanto, se propone tres tamaños de casos de prueba a partir de un juicio cualitativo que describe de manera implícita la complejidad, expresada a través de los niveles simple, media o alta. Cada tipificación del caso de prueba derivado por el aspecto de prueba a evaluar y la complejidad, se denominará componente estándar del proceso.

Al final del nivel, se obtiene la duración de la ejecución del proyecto (tiempo ejecutado expresado en minutos) y los productos esperados. Es de anotar, que partir de las medidas recolectadas se puede establecer el desempeño inicial del probador y una línea base del proceso.

En la fase de post mortem se verifica el proceso y registros (tiempos y defectos) y se calcula la complejidad (usando la técnica de puntos de casos de prueba) de las tareas realizadas. Se Identifica las necesidades de conocimiento para afrontar el proceso y realiza el análisis de los tiempos estimados y ejecutados.

Nivel 1. Planificación personal: Técnica de estimación y paquetes de trabajo.

Incluye la técnica de descomposición de tareas para determinar los paquetes de trabajo en la fase de planeación y tener una calendarización más ajustada a los requerimientos del proceso.

Para este nivel, se realiza la estimación basada en los referentes de tamaño de caso de prueba o también llamado componente estándar por tipo de prueba. Para desarrollar la

estimación se toman los datos asociados al esfuerzo y tiempo del proyecto ejecutado en el nivel anterior.

En la fase del post mortem, el probador analiza el registro de los datos recolectados y realiza una propuesta de mejora partiendo de las experiencias adquiridas en el proceso.

Nivel 2. Calidad personal: Practicas de calidad

En este nivel, se adiciona la actividad para realizar las revisiones a los casos de prueba, con el propósito de incrementar la calidad y efectividad del proceso.

Partiendo de los defectos encontrados, se puede establecer predicciones sobre el comportamiento de pruebas para nuevos ciclos. Un propósito fundamental es asegurar el proceso basados en la documentación generada para la orientación de la estrategia de pruebas.

El objetivo principal es la revisión detallada de los casos de prueba. Los efectos de una buena revisión en estos puntos ayudan al probador a incorporar las prácticas de calidad en su conducta y hábitos de prueba.

Nivel 3. Desarrollar el proceso de manera cíclica.

A partir de este nivel, el probador asume las prácticas de un proceso definido y medible, lo cual permite que el individuo pueda establecer compromisos reales y ajustados a su desempeño.

Un proceso definido permite identificar las falencias del proceso y así establecer planes de mejora que fortalezcan su conducta y hábitos frente al proceso de pruebas.

Adoptar un proceso definido le permite al probador aumentar la base de conocimiento e incrementar la madurez para asumir proyectos de mayor complejidad.

4.1.3 Fases del modelo

El modelo define las siguientes fases para el proyecto desarrollado en cada nivel, como se observa en la figura 14. así:

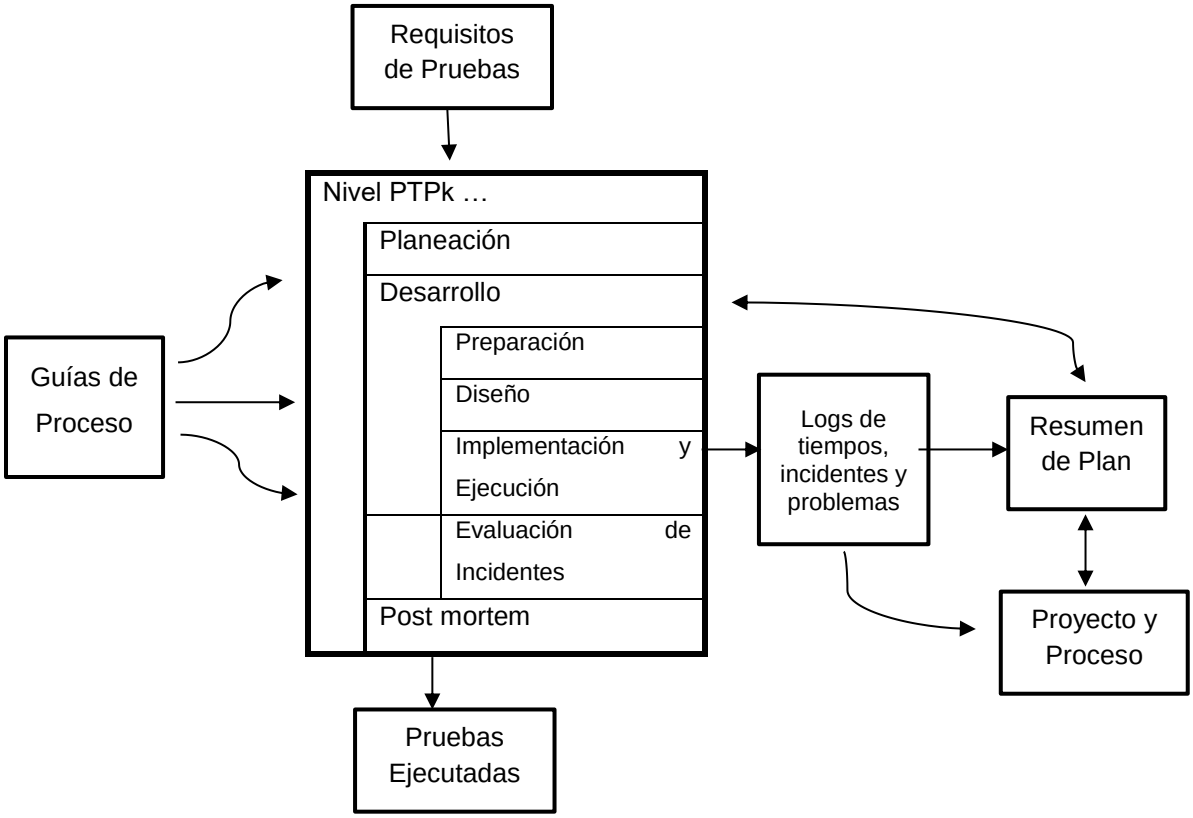


Figura 14. Fases del Modelo
Fuente: Elaboración propia.

- Planeación: Establece la información base para la ejecución y control de las pruebas al inicio del proyecto (ISO, 2013). Siendo la primera aproximación del plan de pruebas maestro y luego se termina de definir en cada nivel o ciclo (Hass, 2014).

  De acuerdo con los autores (Graham et al., 2008), las actividades desarrolladas en el plan son el punto de partida para la ejecución del proyecto independientemente del modelo de ciclo de vida que se adopte.

  La planeación según Vargas y sus colegas (Vargas et al, 2017), tiene como propósito la comprensión de los requisitos de pruebas para definir elementos claves

como: el alcance, los recursos, la estimación, la estrategia y la documentación del proyecto de pruebas acorde con el estándar ISO/IEC 29119.

- Ejecución: Realizar las pruebas de las versiones del software según el ciclo. Simultáneamente preparar el entorno y gestionar los incidentes derivados del proceso según plan propuesto.

  Para las etapas de la ejecución del proyecto, se instancian el nivel de procesos dinámicos de las pruebas establecidos en el estándar ISO/IEC 29119 (ISO, 2013), como son:
  - Preparación
  - Diseño
  - Implementación y ejecución
  - Evaluación de Incidentes.

  Partiendo de la dinámica iterativa del proceso de pruebas, se incluye una etapa denominada preparación para articular los posibles ciclos y el plan desarrollado en la fase de planeación.

- Post mortem: Analizar los resultados respecto al software probado y al proceso utilizado, con el propósito de proponer alternativas de mejora. El propósito es identificar las buenas prácticas para incorporarlas al proceso.

La estructura del modelo provee para cada paso un ejercicio con el propósito de desarrollar las practicas asociadas a cada nivel del modelo, instanciado los siguientes elementos:

- Una guía de proceso que describe las entradas, actividades y salidas de cada ejercicio propuesto. (anexo A)
- Formatos de registro de tiempos y defectos. (anexo B y C)
- Estándar de tipos de defectos. (anexo D)
- El enunciado de ejercicios denominados requerimientos del proceso. (anexo E)
- Un formato denominado plan resumen. (anexo F)
- Listas de Chequeo. (anexo G)

Las métricas definidas para evaluar el desempeño y progreso del proceso se presentan en la tabla 16:

Tabla 16. Métricas

Métrica	Formula
Eficiencia del proceso	Número de casos de pruebas ejecutados / Número total de incidentes encontrados
Incidencias por caso de prueba	Número total de incidentes / Número de casos de prueba ejecutados
Índice de asertividad de la programación del Tiempo.	Número de tiempo ejecutado / Número de tiempo planeado,
Tasa de reparación de problemas	Numero de defectos corregidos / Número de defectos encontrados atribuibles al proceso de prueba

Fuente: Elaboración propia

4.1.4 Método de estimación

De acuerdo con los autores Arahna y Borba (2007, 2009), el caso de prueba es un elemento fundamental para la estimación de manera individual y teniendo como base la suite de pruebas.

En consecuencia, se establece el caso de prueba como base para medir la complejidad y el esfuerzo en el modelo PTPk. Teniendo como premisa las siguientes características: la estrategia individual y el desarrollo de actividades netamente de pruebas de software, que desarrolla el modelo.

El modelo tiene como propósito generar estimados basados en el desempeño personal, para ello, se debe calcular el promedio de los estimados históricos de los proyectos o ejercicios desarrollados por el probador. Por tal razón, se retoma el método denominado estimado basado en componentes estándares (PROBE).

Para Humphery (Humphrey, 1995), el método PROBE es usado para estimar el tamaño del software soportado en los datos históricos generados en el marco de proceso PSP.

Para el modelo PTPk, la estimación de los proyectos personales debe articular los casos de prueba y la gestión de datos históricos. En este contexto de combinación de técnicas de estimación con el método PROBE, se puede referenciar la experiencia desarrollada por Ochoa y sus colegas (Ochoa et al., 2007). En la cual proponen un método que integra el método PROBE y la técnica de estimación puntos de función, para la estimación de proyectos de software en el contexto de las pequeñas y medianas empresas.

Siguiendo un esquema similar al presentado por Ochoa y sus colegas (Ochoa et al., 2007), se plantea la articulación de la técnica TCPA y el método PROBE, que describen el esfuerzo asociado a los casos de prueba y los componentes estándares del modelo PTPk.

Antes de realizar estimaciones basados en esta combinación de métodos, se requiere determinar los componentes estándares, su esfuerzo y recolectar información histórica.

Cada individuo tiene distintos tipos de componentes estándares desarrollados y para obtenerlos se debe hacer levantamiento de los proyectos desarrollados previamente. Para tal efecto se plantean los siguientes pasos:

1. Determinar el esfuerzo (TCP) asociado a los componentes estándares como lo muestra la tabla 17.

Tabla 17. Puntos de Casos de Prueba

Tipos de Prueba	Simple	Medio	Alto
Funcional	6	8	12
Usabilidad	6.9	9.2	13.8
Interfaces	7.9	10.6	15.9
Base de Datos	9.1	12.2	18.3
Rendimiento (manual)	10.5	14.0	21.0
Seguridad	12.1	16.1	24.1
Algoritmos de cálculo	13.9	18.5	27.8

Fuente: Elaboración propia

2. Establecer a partir de los componentes estándares (casos de prueba) el esfuerzo en términos de TCPs de los proyectos ejecutados. En la tabla 18 se presenta el esfuerzo de los proyectos.

Tabla 18. Esfuerzo expresado TCP por proyecto ejecutado

Casos de Prueba	Simple		Medio		Alto		Total
	Peso	Cantidad	Peso	Cantidad	Peso	Cantidad	
Funcional	6		8		12		
Usabilidad	6.9		9.2		13.8		
Interfaces	7.9		10.6		15.9		
Base de Datos	9.1		12.2		18.3		
Rendimiento (manual)	10.5		14.0		21.0		
Seguridad	12.1		16.1		24.1		
Algoritmos de cálculo	13.9		18.5		27.8		
Total por Proyecto Ejecutado							

Fuente: Elaboración propia

3. Crear una base histórica con el consolidado de las variables tiempo (expresado en horas hombre) y esfuerzo (expresado en TCPs) por proyecto ejecutado. Tabla 19.

Tabla 19. Histórico de proyectos

Proyectos	Tiempo (HH)	Esfuerzo (TCP)
Proyecto 1		
Proyecto 2		
Total		

Fuente: Elaboración propia

4. Calcular el valor de las variables esfuerzo y tiempo, partiendo del consolidado de los proyectos históricos (PH), así:

   HH x TCPs = Total del tiempo (PH) / Total esfuerzo(PH)

5. El método PROBE permite realizar estimaciones teniendo en cuenta los siguientes pasos:
   - Estimados basados en el esfuerzo de los componentes estándares.
   - Uso de una base histórica de proyectos similares.
   - Calcular el tiempo del nuevo proyecto.

   Por lo tanto, a partir del peso determinado en los componentes estándares (Casos de Prueba) expresado en TCPs. Infiere la cantidad mínima (MIN), máxima (MAX) y estimada (EST) de componentes que integran el nuevo proyecto. Con estos datos calcula el valor ajustado(VA) usando la fórmula de regresión lineal, que a continuación se expone:

   $$VA = (4 * EST + MAX + MIN) / 6$$

   El total de TCPs de cada componente es el producto de las variables, peso en TCPs (calculado anteriormente) por el valor ajustado VA. En la tabla 20, se presenta cada elemento para calcular el esfuerzo total del proyecto expresado en TCPs.

Tabla 20. Esfuerzo total del proyecto expresado en TCPs

Componente estándar por tipo de Prueba	Tamaño	Peso TCPs	Cantidad de Componente			va	Total TCPs
			min	max	est		
Funcional	simple	6					
	medio	8					
	alto	12					
Usabilidad	simple	6.9					
	medio	9.2					
	alto	13.8					

Interfaces	simple	7.9					
	medio	10.6					
	alto	15.9					
Base de Datos	simple	9.1					
	medio	12.2					
	alto	18.3					
Rendimiento (manual)	simple	10.5					
	medio	14					
	Alto	21					
Seguridad	simple	12.1					
	medio	16.1					
	alto	24.1					
Algoritmos de cálculo	simple	13.9					
	medio	18.5					
	alto	27.8					
Total TCPs del proyecto							

Fuente: Elaboración propia

Una vez calculado el esfuerzo estimado del proyecto a desarrollar, se debe seleccionar de la base histórica los proyectos con características similares (dominio, metodología de desarrollo, tecnología entre otros) al nuevo proyecto, para determinar el promedio de la variable tiempo por esfuerzo (HH x TCPs).

Para calcular el tiempo del nuevo proyecto se multiplica el promedio tiempo por esfuerzo (HH x TCPs) de los proyectos seleccionados por el total TCPs del nuevo proyecto.

HH Proyecto Nuevo= (HH x TCPs) x Total TCPs del proyecto.

4.2 Validación

El propósito de realizar la validación del modelo denominado: proceso individual de pruebas propuesto en esta tesis doctoral es demostrar que el individuo:

- Adopta un proceso definido y medible para conocer su desempeño y asumir compromisos.
- Identifica las falencias y dificultades en su desempeño para proponer planes de mejora.
- Aumenta la base de conocimiento e incrementa la madurez para asumir proyectos de mayor complejidad.

Para ello, se evidencia que a través del modelo el individuo por un lado adquiere habilidades y experiencias para potenciar el desempeño en los proyectos de pruebas; y, por otro, la apropiación de buenas prácticas en la conducta y hábitos del probador.

Para valorar el impacto del modelo se plantea una validación experimental aplicada en el ámbito académico, teniendo dos grupos de referencia. Un grupo de control y otro de prueba. Los grupos seleccionados pertenecen a la asignatura pruebas de software del sexto nivel adscritos al programa ingeniería en software de una institución de educación superior (Anexo I).

Los grupos de control y experimental cuenta con una cantidad similar de estudiantes, 21 y 18 respectivamente. Para el desarrollo de la validación se aplican 4 proyectos sobre pruebas de software funcionales en ambos grupos. La intervención inicia a partir de la 10 semana del semestre académico con el propósito que los grupos aborden los temas fundamentales de asignatura y esenciales para la aplicación del modelo PTPk.

Para tal efecto, en el grupo experimental se inicia con la explicación de la estructura e instrumentos que soporta el modelo.

Para el grupo de control, los estudiantes solo realizan el registro del tiempo planeado y ejecutado para el desarrollo cada ejercicio.

El desarrollo de los ejercicios propuestos (anexo E) se realizan de manera simultánea por los estudiantes de los dos grupos.

Grupo Experimental

El primer ejercicio es desarrollado por lo estudiantes siguiendo el proceso descrito en el modelo PTPk nivel 0, dando inicio con el registro del tiempo a la fase de planeación.

Para el desarrollo del proceso es necesario que el estudiante comprenda y analice el alcance de la propuesta e indague sobre el problema en caso de que existan inquietudes. Una vez aclarado el alcance del ejercicio, el estudiante realiza la estimación de la solución, incluyendo el tiempo dedicado a la fase de planeación y registra el tiempo en minutos. Con esta acción se marca el cierre de la primera fase y se actualiza el formato de tiempos.

Una vez iniciado el desarrollo, el estudiante debe registrar la hora de inicio para la fase y realizar las siguientes etapas de manera secuencial:

- Preparar y refinar las condiciones y escenarios requeridos para el diseño de los casos de prueba.
- Diseñar los casos de prueba acorde con el formato respectivo (anexo H).
- Implementar los casos de pruebas en la herramienta de software y recrear los datos requeridos en el caso que la prueba sea automatizada y por último se ejecutan.
- Evaluar los resultados obtenido y generar el reporte de los resultados.

Es de anotar que durante la ejecución de las etapas se debe registrar de forma simultánea los defectos y tiempos del proceso.

En la fase de post mortem, se verifica la documentación del proceso.

Los datos obtenidos del nivel 0, se presentan en la tabla 21.

Tabla 21. Datos obtenidos del Nivel 0

Estudiante	Tiempo en minutos		Defectos
	Planeado	Ejecutado	
1	45	112	6
2	60	118	6
3	45	120	5
4	45	123	7
5	60	180	8

6	180	147	10
7	90	126	9
8	90	140	4
9	90	165	9
10	60	123	6
11	60	160	11
12	90	161	6
13	100	172	12
14	90	182	11
15	60	168	12
16	150	125	2
17	90	100	2
18	120	135	5

Fuente: Elaboración propia.

Se puede observar que los estudiantes no tienen un método apropiado para realizar estimaciones. Dado que existen amplias diferencias entre los tiempos planeados y los ejecutados. La tasa de defectos frente al proceso es alta porque presentan deficiencias el conocimiento y uso de la técnica de prueba.

El segundo ejercicio se ejecuta acorde con los lineamientos del modelo PTPk 0.1, El estudiante continúa realizando la estimación a priori, previa lectura y análisis del enunciado del problema propuesto.

Para esta planeación el estudiante debe analizar cualitativamente los posibles casos de prueba teniendo como parámetro los niveles complejidad (simple, medio y alto). Lo cual permite una descomposición del trabajo a través de componentes estándares de tamaño para una estimación más aproximada a la real.

En este nivel continua con el registro de las métricas como son: el registro de tiempos y defectos (igual al realizado en el nivel PTPk 0).

Para la fase de desarrollo, el estudiante ejecuta las etapas siguiendo la guía proceso. En esta fase el estudiante incorpora el estándar de caso de pruebas para su diseño e implementación.

En la fase de post mortem se verifica el proceso y los registros de tiempos y defectos. Cada estudiante a partir de las falencias reconocidas en su proceso propone una mejora para incorporarla en el siguiente nivel.

Entre los hallazgos de la implementación se pueden identificar las siguientes falencias:

- Falta de conocimiento en la técnica aplicada.
- Dificultad al estimar la complejidad del caso de prueba.
- Inconsistencias en la información asociada al caso de prueba.
- Falta de análisis en los escenarios y condiciones previstos para la prueba.

Al final de proyecto, se obtiene el tiempo total de ejecución y el registro de las métricas descritas en el modelo.

Con la finalización del segundo ejercicio se actualiza los promedios de los componentes estándares de complejidad acorde con el desempeño de cada estudiante.

Tabla 22. Datos obtenidos del Nivel 0.1

Estudiante	Tiempo en minutos		Defectos	Casos de Prueba
	Planeado	Ejecutado		
1	120	142	4	15
2	150	130	6	13
3	110	127	7	10
4	120	112	4	12
5	120	146	5	14
6	150	165	4	11
7	150	171	6	15
8	120	115	3	14
9	100	124	4	13
10	120	135	5	11
11	150	163	6	12
12	120	155	4	12
13	150	142	6	11
14	150	133	6	13
15	120	137	7	15
16	125	132	6	15
17	120	110	3	14
18	150	126	2	15

Fuente: Elaboración propia.

Se puede observar que las estimaciones basadas en la complejidad del tamaño permiten que la diferencia entre los tiempos ejecutados y los planeados es más cercana. La tasa de

defectos frente al proceso es inferior dado que los estudiantes adquieren practica en el diseño y ejecución de los casos de prueba asociados a las técnicas de pruebas aplicadas.

Para el tercer ejercicio del modelo PTPk 1.0, la estudiante continua con las practicas adquiridas en los niveles anteriores y focaliza en el desarrollo de la propuesta de mejora e incluye en su proceso la técnica de descomposición de paquetes de trabajo denominada (WBS por sus siglas en ingles) para facilitar el análisis en la fase de planeación y desarrollar una calendarización más ajustada a los requerimientos del proceso.

En la fase de planeación se incluye los elementos básicos de un plan de pruebas, para que el estudiante adopte un estándar. También se realiza la estimación basada en los referentes de tamaño de caso de prueba. Para la estimación se toman los datos asociados al esfuerzo y tiempo de los proyectos ejecutados en los niveles anteriores.

Para la fase de ejecución se mantiene el cumplimiento de las etapas previstas en el proceso.

En la fase de post mortem, la estudiante continua con la verificación del proceso y los registros de tiempos y defectos. Cada estudiante analiza su proceso con respecto al anterior y genera una propuesta de mejora basada en las dificultades del proceso ejecutado.

Con la finalización del tercer ejercicio se optimizan los tiempos y las practicas aplicadas por cada estudiante en su proceso.

En el nivel se obtienen los siguientes datos:

Tabla 23. Datos obtenidos del Nivel 1.

Estudiante	Tiempo en minutos		Defectos	Casos de Prueba
	Planeado	Ejecutado		
1	124	135	4	20
2	115	122	6	18
3	112	125	4	24
4	132	140	3	22
5	126	118	5	20
6	142	122	6	18
7	128	105	6	16
8	115	87	4	22
9	119	126	5	18

10	112	121	4	20
11	145	130	4	21
12	130	138	6	22
13	137	129	4	19
14	125	135	4	20
15	142	123	4	18
16	142	137	6	22
17	135	126	4	18
18	124	142	6	24

Fuente: Elaboración propia.

Frente a los datos recolectados se puede evidenciar que las diferencias se reducen entre los tiempos planeados y ejecutados. Teniendo en cuenta que el cálculo del tiempo planeado responde al promedio del rendimiento calculado por cada estudiante. Lo cual refleja que el uso de históricos contribuye a estimaciones más ajustadas a la realidad. También se puede observar que la tasa de defectos es menor frente al desarrollo del proceso porque los estudiantes han adquirido mayor destreza en el diseño y ejecución de casos de prueba.

En el desarrollo del cuarto ejercicio que corresponde al PTPk 2.0, se introduce al proceso definido la etapa denominada revisión de caso de prueba basado en la lista de chequeo definida para tal fin. Es de anotar que se deben tener en cuenta los diferentes tipos de prueba para el desarrollo de esta actividad, para el caso de la validación solo se tienen en cuenta las pruebas funcionales.

El estudiante desarrolla el proceso con las practicas incorporadas en los niveles anteriores y presta mayor atención al análisis de los defectos encontrados para generar planes de mejora al equipo de desarrollo del producto y para su proceso establece predicciones sobre el comportamiento del componente sometido a pruebas.

Se obtienen los siguientes datos:

Tabla 24. Datos obtenidos del Nivel 2.

Estudiante	**Tiempo en minutos**		**Defectos**	**Casos de Prueba**
	Planeado	**Ejecutado**		
1	145	155	2	28
2	135	126	2	26
3	134	141	3	24

4	127	126	1	28
5	135	142	4	28
6	123	116	3	26
7	130	136	3	25
8	129	141	4	26
9	135	146	3	26
10	145	138	3	28
11	146	135	4	24
12	140	133	2	26
13	135	138	1	28
14	142	136	2	26
15	135	142	0	28
16	145	140	1	28
17	130	141	1	28
18	125	138	2	28

Fuente: Elaboración propia.

Los datos obtenidos reflejan que las estimaciones son mas cercanas a los tiempos de ejecución. También se puede observar que los defectos frente al proceso se disminuyen y los encontrados son atribuibles al proceso de desarrollo. La inclusión de la revisión al diseño de casos de prueba permite validar el cumplimiento de la condiciones y criterios de cobertura de la prueba. El nivel de productividad del probador aumenta.

Grupo de control

Los estudiantes no siguen un proceso adecuado para el desarrollo de las pruebas porque la documentación generada no establece métricas básicas del proceso.

De acuerdo con los entregables generados los estudiantes mantienen las mismas dificultades encontradas en los cuatros ejercicios desarrollados.

La estimación de los estudiantes no es consistente frente al tiempo ejecutado porque no tienen una medida base para el desarrollo del proceso y la falta de datos históricos no permiten medir su propio desempeño.

5 Capitulo: Contribuciones de la Tesis

La principal contribución de esta tesis de doctorado fue la generación de conocimientos conducentes en la construcción de un modelo gestión de conocimiento aplicado al proceso de pruebas bajo el enfoque individual.

El conocimiento generado, de evidente calidad científica, es original e inédito; estructurado sobre la base de una rigurosa metodología que permitió superar la frontera del conocimiento actual en el tema correspondiente. La contribución principal constituye un aporte significativo al avance del área de la Ingeniería de Software y gestión de conocimiento.

Otras contribuciones producto del desarrollo de la presente tesis, son:

- La caracterización de las experiencias asociadas a la gestión de conocimiento y las pruebas de software.
- El método de estimación para procesos individuales de pruebas de software en el marco de la gestión de conocimiento.
- La definición de los flujos de proceso basado en las áreas y núcleos de competencia que soportan el modelo.
- El diseño del modelo basado en el principio de aprendizaje individual y la mejora continua.
- Las plantillas definidas para apoyar cada una de las etapas que soportan el desarrollo de las actividades asociadas al proceso.

A continuación, se relaciona la producción intelectual (Publicaciones y productos) en el ámbito de la tesis:

5.1 Artículos en Revistas Internacionales

- Soto, Dario, Jiménez, B. & Reyes, A. (2017). Application of Knowledge Management to the software testing process. Revista ESPACIOS Vol. 38, No. 43.

5.2 Artículos en Revistas Nacionales

- Soto, Dario, Jiménez, B. & Reyes, A. (2017). Aplicación de la Gestión de Conocimiento al proceso de pruebas de software. Revista de Ingenierías de la Universidad San Buenaventura. Vol. 8, No. 2.
- Soto, Dario, Jiménez, B. & Reyes, A. (2015). Un modelo de gestión de conocimiento para la mejora del proceso de pruebas de software. Revista de Investigaciones de la Universidad del Quindío. Vol. 27 fasc.1.

5.3 Trabajos Completos en Eventos Internacionales

- Soto, Dario, Jiménez, B. & Reyes, A. (2017, septiembre). A Knowledge Management Model for Improving the Software Test Process. 18th European Conference on Knowledge Management ECKM. (pp.192-203). España, Barcelona.
- Soto, D., Jiménez, B. & Reyes, A. (2017, Julio). Integrando a las pruebas de software la gestión de Conocimiento. 16th Iberoamerican Conference on Systems, Cybernetics and Informatics CISCI. (p. 72 -83). Estados Unidos, Orlando.
- Villamizar, K., Soto, D, Giraldo, J. & Jiménez, J. (2016, junio). Modelo de pruebas en proyectos BI. 14th International Multi-Conference For Engineering, Education Caribbean Conference For Engineering And Technology LACCEI. (p.1-9). Costa Rica, San José.
- Vargas, F., Soto, D., Giraldo, J. & Durango, C. (2016, abril). Representación en el núcleo SEMAT de la norma ISO 29119-2 para identificar patrones en pruebas de software. 4to. Congreso Internacional de Investigación e Innovación en Ingeniería de Software CONISOFT (p. 40 -47). México, Puebla.

5.4 Pasantías Nacionales e Internacionales

- Grupo Sistemas de Información

Instituto de informática
Universidad Federal de Rio Grande do Sur
Brasil, Porto Alegre, (2015)

5.5 Trabajos Completos en Eventos Nacionales

- Conferencia: “La gestión de Conocimiento en las organizaciones de software”. Universidad Francisco de Paula Santander. Cúcuta, Norte de Santander. Octubre 2014.

5.6 Capítulos de Libro

- Giraldo, J., Soto, D., & Villamizar, K. (2017). Proceso de pruebas en proyectos para obtener conocimiento utilizando inteligencia de negocios. En Investigación e Innovación en Ingeniería de Software, Volume 1. Eds F. A. Vargas, D. E. Soto & W. Perdomo.
- Vargas, F., Soto, D., Giraldo, J. & Durango, C. (2017). A Representation Based in SEMAT Kernel of the Test Planning Process According to ISO/IEC/IEEE 29119-2 Standard. En Software engineering : methods, modeling, and teaching, Volume 4. Eds C. M. Zapata, C. E. Durango & W. Perdomo.

5.7 Orientaciones

Asesorías de tesis de grado:

- Modelo para la caracterización de las lecciones aprendidas en proyectos de educación virtual integrando el contexto, la problemática y las buenas prácticas. Programa de maestría en Ingeniería Informática. Universidad Nacional de Colombia. Sede Medellín. (En proceso).
- Guía para la implementación de las pruebas de carga en el contexto de arquitectura de microservicios orientada al sector de las PYMES del software. Programa de Maestría en Gestión de las Tecnologías de la Información. Tecnológico de Antioquia Institución Universitaria. (En proceso).

- Sistema de información para la gestión del conocimiento tácito de los procesos misionales. Programa académico de Ingeniería en Software de la Tecnológico de Antioquia Institución Universitaria. (2015).
- Desarrollo de Aplicativo que soporta el proceso de pruebas de software bajo el marco del estándar ISO:29119. Programa académico de Ingeniería en Software de la Tecnológico de Antioquia Institución Universitaria. (2014).

5.8 Proyectos y Otros.

- Certificación como: Knowledge Manager.
  Knowledge Research Institute Inc.
  Arlington, Texas USA. (2014).
- Arquitectura para la construcción de una herramienta para gestión de producción y distribución de conocimiento. (2014 -2015).
  Financiado por el Tecnológico de Antioquia I.U.

6 Conclusiones y trabajos futuros

6.1 Conclusiones

Uno de los principales propósitos de articular la gestión de conocimiento y las pruebas de software, es el desarrollo de capacidades para que las prácticas de mejora continua se conviertan en una rutina organizacional, es decir, se ejecute de manera sistemática y permanente. Las iniciativas de KM, mejora de procesos y aprendizaje organizacional, coinciden en factores como el proceso, la tecnología y las personas. Siendo el individuo el componente esencial en la implementación de iniciativas de gestión de conocimiento. En este contexto, la cultura organizacional es la que se encarga de modelar los comportamientos y crear un significado organizacional, ya que, mediante supuestos básicos adapta a toda la organización a acostumbrarse al aprendizaje y la mejora de procesos a partir de una estrategia que gestione el conocimiento desde el plano individual y garantice el uso e institucionalización de las buenas prácticas en el plano grupal.

Un faltante en los marcos de mejora de pruebas de software a nivel individual son la cultura y las prácticas de calidad, evidenciadas en la caracterización realizada a partir de las escuelas de Earl (2001) y las capacidades estructurales de Gold (2001); el modelo PTPk incorpora la gestión de conocimiento desde los procesos de aprendizaje individual y la mejora continua para subsanar este faltante.

De acuerdo con la revisión de la literatura, se puede concluir que los estudios se enfocan en el contexto corporativo sin abordan estrategias orientadas al individuo. Siendo para el modelo PTPk, el principio fundamental en el desarrollo y gestión de las prácticas de pruebas de software en el marco del aprendizaje experiencial basado en los criterios las escuelas tecnocráticas y conductuales de la gestión de conocimiento.

El modelo incorpora la estructura e intencionalidad del marco PSP incorporando parcialmente prácticas del marco de referencia TMMI. A través de un proceso definido que integra las etapas del estándar ISO 29119.

La estrategia de mejora continua desarrollada por el modelo instancia un esquema incremental y escalonado a través de tres niveles: iniciando por la adopción del proceso para lograr la mejora en el proceso de planeación y en última instancia la incorporación de buenas prácticas de calidad basado en un proceso de aprendizaje que usa un enfoque inductivo.

Con el registro de tiempos y problemas, se propone identificar el rendimiento del probador y acciones de mejora en cada nuevo proceso que impactan en el desarrollo del nuevo ciclo de prueba.

El método de estimación desarrollado para el modelo articula los puntos de casos de prueba y el método PROBE, para gestionar los datos asociados a la estimación de los proyectos ejecutados en cada uno de los niveles.

Los elementos que integran el modelo influyen en el comportamiento del individuo y propician la adopción de las buenas prácticas.

La estimación y la productividad en el proceso de pruebas de software no es solo un producto de la intuición del probador, sino un proceso basado en datos recopilados previamente de experiencias similares y especialmente de sus propias experiencias.

En el campo académico, la productividad aumenta a medida que el estudiante incorpora el ejercicio de buenas prácticas y el conocimiento de las herramientas asociadas con el entorno de evaluación.

6.2 Trabajos futuros

Como trabajos futuros se proponen:

El desarrollo de un proceso grupal que retome las practicas incorporadas en el modelo PTPk, con el propósito de gestionar los conocimientos y experiencias adquiridas en el proceso personal.

Validar el modelo propuesto en diferentes contextos y tipos de prueba para refinar las actividades asociadas a cada uno de los niveles.

Incorporar en el plano grupal la estrategia de riesgos que incorpora el proceso de pruebas para mejorar la selección de técnicas.

Desarrollar una solución de software que incorpore la estructura y prácticas del proceso PTPk, para facilitar el uso de la practicas y la gestión de métricas.

7 Anexos.

Anexo A – Guía de Proceso.

Anexo B – Registro de tiempos.

Anexo C – Registro de defectos.

Anexo D – Tipificación de defectos.

Anexo E – Ejercicios.

Anexo F – Resumen del plan de proyecto.

Anexo G – Lista de Chequeo.

Anexo H – Plantilla de caso de prueba.

Anexo I – Constancia de Validación.

8 Bibliografía

Aaen, I., Arent, J., Mathiassen, L., & Ngwenyama, O. (2001). A conceptual MAP of software process improvement. Scandinavian journal of information systems.

Abdullah, R., Eri, Z. D., & Talib, A. M. (2011, December). A model of knowledge management system in managing knowledge of software testing environment. In Software Engineering (MySEC), 2011 5th Malaysian Conference in (pp. 229-233)

Afzal, W., Alone, S., Glocksien, K., & Torkar, R. (2016). Software test process improvement approaches: A systematic literature review and an industrial case study. Journal of Systems and Software, 111, 1-33.

Andrade, J., Ares, J., Martínez, M. A., Pazos, J., Rodríguez, S., Romera, J., & Suárez, S. (2013). An architectural model for software testing lesson learned systems. Information and Software Technology, 55(1), 18-34.

Aranha, E., & Borba, P. (2009). Estimating manual test execution effort and capacity based on execution points. International Journal of Computers & Applications, 31(3), 167–172.

Aranha, E., Borba, P., & Lima, J. (2006). Model Simulation for Test Execution Capacity Estimation. 17th International Symposium on Software Reliability Engineering, 231–236.

Arent, J., & Norbjerg, J. (2000, January). Software process improvement as organizational knowledge creation: a multiple case analysis. In System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on (pp. 11-pp). IEEE.

Argyris, C., & Schon, D. A. (1996). Organiational learning II. Addison Wesley.

Aristegui, J. L. (2010). Los casos de prueba en la prueba del software. Lámpsakos, (3), 27-34.

Aurum, A., Daneshgar, F., & Ward, J. (2008). Investigating Knowledge Management practices in software development organisations - An Australian experience. Information and Software Technology, 50(6), 511–533.

Bain, A. (1998). Social defenses against organizational learning. Human relations, 51(3), 413-429.

Bath, G., & Veenendaal, E. V. (2014). Improving the test process: implementing improvement and change: a study guide for the ISTQB expert level module.

Beer, A., & Ramler, R. (2008, September). The role of experience in software testing practice. In Software Engineering and Advanced Applications, 2008. SEAA'08. 34th Euromicro Conference (pp. 258-265). IEEE.

Borade, J. G., & Khalkar, V. R. (2013). Software project effort and cost estimation techniques. International Journal of Advanced Research in Computer Science and Software Engineering, 3(8).

Bourke, P., & Fairley, R. E. SWEBOK V3. 0-Guide to the Software Engineering Body of Knowledge.

Bruegge, B., Dutoit, A. H., Hirales, R. G., López, M. R. C., & González, M. A. D. (2002). Ingeniería de software orientado a objetos. Pearson educación.

Call, D. (2005). Knowledge management–not rocket science. Journal of Knowledge management, 9(2), 19-30.

Capote, J., Llantén, C. J., Pardo, C., & Collazos, C. (2013). Gestión del conocimiento en un programa de mejora de procesos de software en MiPyMEs: KMSPI Model. Revista Facultad de Ingeniería, (50), 205-216.

Castañeda, D. I., & Fernández Ríos, M. (2007). Validación de una escala de niveles y condiciones de aprendizaje organizacional. Universitas Psychologica.

Castañeda, D. I., & Pérez-Acosta, A. M. ¿Cómo ocurre el aprendizaje individual en el aprendizaje organizacional? Una revisión del modelo Crossan, Lane y White. Publicado por Revista Interamericana de Psicología Ocupacional, 2005, 24, 1-15.

Chaudhary, P., & Yadav, C. S. (2012). An Approach for calculating the effort needed on testing Projects. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 1(1), pp-35.

Checkland, P., & Holwell, S. E. (2006). Data, capta, information and knowledge. Introducing Information Management: the business approach, 47-55.

Chugh, M., Chugh, N., & Punia, D. K. (2015, May). Evaluation and analysis of knowledge management best practices in software process improvement a multicase experience. In Advances in Computing and Communication Engineering (ICACCE), 2015 Second International Conference on (pp. 661-666). IEEE.

Crossan, M. M., Lane, H. W., & White, R. E. (1999). An organizational learning framework: From intuition to institution. Academy of management review, 24(3), 522-537.

De Souza, E. F., de Almeida Falbo, R., & Vijaykumar, N. L. (2015). Knowledge management initiatives in software testing: A mapping study. Information and Software Technology, 57, 378-391.

Del Moral, A., Pazos, J., Rodríguez Fernández, E., Rodríguez-Patón, A., Suárez, S. (2007). Gestión del conocimiento, Madrid, Paraninfo.

DeMarco, T. (1986). Controlling software projects: Management, measurement, and estimates. Prentice Hall PTR.

Desai, A., & Shah, S. (2011, February). Knowledge management and software testing. In Proceedings of the International Conference & Workshop on Emerging Trends in Technology (pp. 767-770). ACM

Drucker, Peter. 2002. La Gerencia en la Sociedad Futura. Bogotá: Editorial Norma.

Drucker, Peter. 2003. La Gestión del Conocimiento. Madrid: Ediciones Deusto.

Drucker, Peter. 2004. La Sociedad Post Capitalista. Bogotá: Editorial Norma.

Earl, M. (2001). Knowledge management strategies: Toward a taxonomy. Journal of management information systems, 18(1), 215-233.

Elberzhager, F., Münch, J., & Assmann, D. (2014). Analyzing the relationships between inspections and testing to provide a software testing focus. Information and Software Technology, 56(7), 793-806.

Erdogan, G., Li, Y., Runde, R. K., Seehusen, F., & Stølen, K. (2014). Approaches for the combined use of risk analysis and testing: A systematic literature review. International Journal on Software Tools for Technology Transfer, 16(5), 627–642.

Esteban, A. S. (2012). Marco metodológico para la mejora de las actividades de verificación y validación de productos software (Doctoral dissertation, Universidad Carlos III de Madrid).

Felderer, M., & Ramler, R. (2016). Risk orientation in software testing processes of small and medium enterprises: an exploratory and comparative study. Software Quality Journal, 24(3), 519-548.

Felderer, M., & Schieferdecker, I. (2014). A taxonomy of risk-based testing. International Journal on Software Tools for Technology Transfer, 16(5), 559–568.

Fernández-Sanz L., Villalba M.T., Hilera J.R., Lacuesta R. (2009) Factors with Negative Influence on Software Testing Practice in Spain: A Survey. In: O'Connor R.V., Baddoo N., Cuadrago Gallego J., Rejas Muslera R., Smolander K., Messnarz R. (eds) Software Process Improvement. EuroSPI 2009. Communications in Computer and Information Science, vol 42. Springer, Berlin, Heidelber

Fernández-Sanz, L., & Misra, S. (2011). Influence of human factors in software quality and productivity. Computational Science and Its Applications-ICCSA 2011, 257-269.

Ferrer, J., Chicano, F., & Alba, E. (2013). Estimating software testing complexity. Information and Software Technology, 55(12), 2125-2139.

Fiol, C. M., & Lyles, M. A. (1985). Organizational learning. Academy of management review, 10(4), 803-813.

Galvis-Lista, E., & Sánchez-Torres, J. M. (2013). A critical review of knowledge management in software process reference models. JISTEM-Journal of Information Systems and Technology Management, 10(2), 323-338.

Garcia, C., Dávila, A., & Pessoa, M. (2014, November). Test process models: Systematic literature review. In International Conference on Software Process Improvement and Capability Determination (pp. 84-93). Springer International Publishing.

Garousi, V., & Zhi, J. (2013). A survey of software testing practices in Canada. Journal of Systems and Software, 86(5), 1354-1376.

Garvin, D. A. (1985). Building a learning organization. Org Dev & Trng, 6E (Iae), 274.

Garvin, D. A. (2000). Crear una organización que aprende. Harvard Business Review, 51-89.

Gates, B. (1999). Business @ the Speed of Thought Display: Using a Digital Nervous System.

Gélvez, H. A. P. (2010). Contribución a la gestión de los procesos de pruebas de software y servicios.

Glazer, H., Dalton, J., Anderson, D., Konrad, M. D., & Shrum, S. (2008). CMMI or agile: why not embrace both!

.

Graham, D., Van Veenendaal, E., Evans, I., & Black, R. (2008). Foundations of software testing: ISTQB certification. London: Cengage Learning EMEA.

Gupta, J. N., & Sharma, S. K. (Eds.). (2004). Creating knowledge based organizations. Igi Global.

Harrison, W. (2002, December). A software engineering lessons learned repository. In Software Engineering Workshop, 2002. Proceedings. 27th Annual NASA Goddard/IEEE (pp. 139-143). IEEE.

Heiskanen H., Maunumaa M., Katara M. (2012) A Test Process Improvement Model for Automated Test Generation. In: Dieste O., Jedlitschka A., Juristo N. (eds) Product-Focused Software Process Improvement. PROFES 2012. Lecture Notes in Computer Science, vol 7343. Springer, Berlin, Heidelberg

Hessen, J., Gaos, J., & Romero, F. (1970). Teoría del conocimiento (pp. 25-86). Espasa-Calpe.

Hosseingholizadeh, R. (2014, October). Managing the knowledge lifecycle: A integrated knowledge management process model. In Computer and Knowledge Engineering (ICCKE), 2014 4th International eConference on (pp. 102-110). IEEE.

Humphrey, W. S. (2001). Introducción al proceso software personal sm. Addison Wesley.

IEEE Std 1059 (1993), IEEE Guide for Software Verification and Validation Plans. Consultado 15 Agosto del 2017. https://standards.ieee.org/findstds/standard/1059-1993.html.

ISO, (2005). Norma internacional ISO 9000 - Sistemas de gestión de la calidad - Fundamentos y vocabulario. Ginebra: ISO 9000.

ISO, I. (2010). IEEE, Systems and Software Engineering--Vocabulary. IEEE computer society, Piscataway, NJ.

ISO, I. (2013). IEEE, The International Software Testing Standard, 29119.

Itkonen, J., Mantyla, M. & Lassenius, C. (2013). The role of the tester's knowledge in exploratory software testing, IEEE Transactions on Software Engineering, vol. 39,pp. 707–724

Janczarek, P., & Sosnowski, J. (2015). Investigating software testing and maintenance reports: Case study. Information and Software Technology, 58, 272-288.

Janjic, W., & Atkinson, C. (2013, May). Utilizing software reuse experience for automated test recommendation. In Automation of Software Test (AST), 2013 8th International Workshop on (pp. 100-106). IEEE.

Jennex, M. E., & Olfman, L. (2004). Organizational memory. In Handbook on Knowledge Management 1 (pp. 207-234). Springer Berlin Heidelberg.

Jones, C. (2008). Applied Software Measurement. Global Analysis of Productivity and Quality. New York, USA: McGraw-Hill.

Kaner, C., Bach, J., & Pettichord, B. (2008). Lessons learned in software testing. John Wiley & Sons.

Kasurinen, J., Taipale, O., & Smolander, K. (2009, December). Analysis of problems in testing practices. In Software Engineering Conference, 2009. APSEC'09. Asia-Pacific (pp. 309-315). IEEE.

Kerkhof, C., Ende, J. V. D., & Bogenrieder, I. (2003). Knowledge management in the professional organization: a model with application to CMG software testing. Knowledge and Process Management, 10(2), 77-84.

Kim, D. H. (1998). The link between individual and organizational learning. The strategic management of intellectual capital, 41-62.

King, W. R. (2001). Strategies for creating a learning organization. Information Systems Management, 18(1), 12-20.

Kolb, D. (1984). Experiential education: Experience as the source of learning and development. Englewood Cliffs, NJ.

Kolb, D. A. (2014). Experiential learning: Experience as the source of learning and development. FT press.

Krüger, K.(2006, Octubre). El concepto de sociedad del conocimiento. En: Biblio 3W Revista bibliográfica de geografía y ciencias sociales. Vol. 11, no. 683.

Li, X., & Zhang, W. (2012, April). Ontology-based testing platform for reusing. In Internet Computing for Science and Engineering (ICICSE), 2012 Sixth International Conference on (pp. 86-89). IEEE.

Liu, Y., Wu, J., Liu, X., & Gu, G. (2009, July). Investigation of knowledge management methods in software testing process. In Information Technology and Computer Science, 2009. ITCS 2009. International Conference on (Vol. 2, pp. 90-94). IEEE.

Lopez, M., Cabrales, F., y Schmal, R. (2005, Mayo). Gestión del Conocimiento: Una revisión teórica y su asociación con la universidad. En: Panorama Socioeconómico. No. 30.

Mäntylä, M. V., & Itkonen, J. (2014). How are software defects found? The role of implicit defect detection, individual responsibility, documents, and knowledge. Information and Software Technology, 56(12), 1597-1612.

Mäntylä, M. V., Itkonen, J., & Iivonen, J. (2012). Who tested my software? Testing as an organizationally cross-cutting activity. Software Quality Journal, 20(1), 145-172.

Mazoni, G. M. (2010). Modelo para la gestión del conocimiento y la experiencia integrada a las prácticas y procesos de desarrollo software.

Moreno, P., Comorera, V., Balbastre, F., & Vivas, S. (2001). Aprendizaje organizativo y creación de conocimiento: un modelo dinámico integrador de ambas corrientes. In XI Congreso Nacional de ACEDE.

Myers, G. J., Badgett, T., & Sandler, C. (2015). The Art of Software Testing, Third. Practice Book for the Paper-based GRE® revised General Test (PDF),.

Myers, P.(1996). Knowledge Management and Organizational Desing. Washington: Butterwoth-Heinemann

Nageswaran, S. (2001). Test effort estimation using use case points. Quality Week, 1–6.

Narváez Barrera, M. X. (2015). V2soft: Guía metodológica para el proceso de validación y verificación de requerimientos para el usuario final (Bachelor's thesis).

Natali, A. C. C., da Rocha, A. R. C., Travassos, G. H., & Mian, P. G. (2004). Integrating verification and validation techniques knowledge into software engineering environments. Proceedings of 4as Jornadas Ibeoamericanas de Ingeniería del Software e Ingeniería del Conocimiento, JIISIC, 4, 419-430.

Newman, B. D. (1996). Knowledge Management vs Knowledge Engineering. The Knowledge Management Theory Papers.

Nguyen, T. D., Guo, H., Naguib, R. N., & Wickramasinghe, N. (2011). A view of 21st century healthcare industry and software quality improvement practices. International Journal of Networking and Virtual Organisations, 9(2), 155-168.

Nogeste, K., & Walker, D. H. (2006). Using knowledge management to revise software-testing processes. Journal of Workplace Learning, 18(1), 6-27.

Nonaka, I., & Takeuchi, H. (1995). The knowledge-creating company: How Japanese companies create the dynamics of innovation. Oxford university press.

Nonaka, I. and N. Konno (1998). “The Concept of ba: ”Building a Foundation for Knowledge Creation,” California Management Review , 40-3, pp.40-54.

Nonaka, I., & Toyama, R. (2003). The knowledge-creating theory revisited: knowledge creation as a synthesizing process. Knowledge management research & practice, 1(1), 2-10.

Nonaka, I. (2008). The knowledge-creating company. Harvard Business Review Press.

Nonaka, I., Byosiere, P., Borucki, C. C., & Konno, N. (1994). Organizational knowledge creation theory: a first comprehensive test. International Business Review, 3(4), 337-351.

North K. & ROQUE, R. (2008). Gestión del conocimiento: una guía práctica hacia la empresa inteligente. Reino Unido (Escocia - Inglaterra): Libros en red Ochoa, S. F., Pino, J. A., & Andrade, D. (2007). Estrategias para Estimar el Esfuerzo de Desarrollo de Proyectos Web en Escenarios Inmaduros. ÍNDICE Pág., 2905548, 125.

Ossaba, L. A. P., & Isaza, J. E. V. (2012). Gestión de conocimiento: la solución para disminuir el reproceso en las pruebas de software. Ingenierías USBmed, 3(2), 48-53.

Pall, G. A. (1987). Quality management. Prentice Hall PTR.

Pawlowsky, P. (2001). The treatment of organizational learning in management science. Handbook of organizational learning and knowledge, 61-88.

Perona Ossaba, L. Á., & Velásquez Isaza, J. E. (2012). Gestión de conocimiento: la solución para disminuir el reproceso en las pruebas de software.

Perry, W. E. (2007). Effective methods for software testing: Includes complete guidelines, Checklists, and Templates. John Wiley & Sons.

Pohs, W., Thiel, G., & Earley, S. (2001). Practical knowledge management: The lotus knowledge discovery system. IBM Press.

Pomeroy-Huff, M., Cannon, R., Chick, T., Mullaney, J., & Nichols W (2009). The Personal Software Process-SM (PSP-SM) Body of Knowledge, Version 2.0 (No. CMU/SEI-2009-SR-018).

Presman, R. (2002). Ingeniería de Software. Un Enfoque Práctico.(6 ta edición).

Reyes, A., & Zarama, R. (1998). The process of embodying distinctions—A re-construction of the process of learning. Cybernetics & Human Knowing, 5(3), 19-33.

Rodrigues, A., Pinheiro, P. R., & Albuquerque, A. (2010, September). The definiton of a testing process to small-sized companies: the Brazilian scenario. In Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the (pp. 298-303). IEEE.

Rodríguez Antón, J. M., & Trujillo Reyes, J. C. (2007). ¿ Las universidades son organizaciones que aprenden adecuadamente?. Universia Business Review, (15).

Rodriguez, D. (2006) Modelos para la creación y gestión del conocimiento: una aproximación teórica. En: Educar.Vol. 36, p. 25-39.

Rus, I., & Lindvall, M. (2002). Knowledge management in software engineering. IEEE software, 19(3), 26.

Salazar, A. M. B. (2015). Un análisis descriptivo de la relación entre el proceso de aprendizaje organizacional y las ventajas competitivas. Horizontes Empresariales, 6(2), 49-62.

Santos, G., Montoni, M., Figueiredo, S., & Rocha, A. (2007). SPI-KM-Lessons learned from applying a software process improvement strategy supported by knowledge management. Product-Focused Software Process Improvement, 81-95.

Senge, P. M. (2005). La quinta disciplina en la práctica. Ediciones Granica SA
.

Simon, H. A. (1991). Bounded rationality and organizational learning. Organization science, 2(1), 125-134.

Soto, D. & Reyes A. (2010). INTRODUCIENDO PSP (PROCESOS PERSONAL DE SOFTWARE) EN EL AULA. Revista Colombiana de Tecnologías de Avanzada, 2(16), 1-5.

Soto, D. , Reyes, A. X. , & Jimenez, J. (2017a). Aplicación de la Gestión de Conocimiento al proceso de pruebas de software. Ingenierías USBMed, 8(2), 6-13.

Soto, D., Jimenez, J., Reyes, A. (2017b). A knowledge management model for improving the software test process. In Proceedgins of the 18th European Conference on Knowledge Management (pp. 922-930).

Soto, E., Sauquet, A., Gore, E., Vogel, E., Cardenas, J. y Soler, C. (2006). Gestión y conocimiento: en organizaciones que aprenden. México: Thompson.

Stata, R., & Almond, P. (1989). Organizational learning: The key to management innovation. The training and development sourcebook, 2, 31-42.

Swieringa, J. (1995). La organización que aprende. Addison-Wesley Iberoamericana.

Swieringa, J., & Wierdsma, A. (1992). Becoming a learning organization.

Torres Ricaurte, D. M. Un método para medir la productividad del equipo de pruebas en la estimación del esfuerzo de pruebas de software (Doctoral dissertation, Universidad Nacional de Colombia, Medellín).

Vargas, F. A., & Soto, D. E. (2012). Productividad y asertividad. Metodología aplicada desde el aula. Memorias, 10(17), 104-112.

Vargas, F., Soto, D., Giraldo, J. & Durango, C. (2017). A Representation Based in SEMAT Kernel of the Test Planning Process According to ISO/IEC/IEEE 29119-2 Standard. En Software engineering : methods, modeling, and teaching, Volume 4. Eds C. M. Zapata, C. E. Durango & W. Perdomo.

Veenendaal, E. van, & Dekkers, T. (1999). Test point analysis: a method for test estimation. In R. J. Kusters, A. Cowderoy, F. J. Heemstra, & E. V. Veenendaal (Eds.), Project control for software quality (pp. 45–59). Maastricht, the Netherlands: Shaker Publishing BV.

Vegas, S., & Basili, V. (2005). A characterisation schema for software testing techniques. Empirical Software Engineering, 10(4), 437-466.

Vegas, S., Juristo, N., & Basili, V. (2006). Packaging experiences for improving testing technique selection. Journal of Systems and Software, 79(11), 1606-1618.

Villamizar, K., Soto, D, Giraldo, J. & Jimenez, J. (2016). Modelo de pruebas en proyectos BI. En las memorias de la Conferencia Iberoamericana en Sistemas, Cibernética e Informática CISCI 2016. (1-9).Scopus.

Watkins, J., & Mills, S. (2010). Testing IT: an off-the-shelf software testing process. Cambridge University Press.

Wei, O. K., & Ying, T. M. (2007, December). Knowledge management approach in mobile software system testing. In Industrial Engineering and Engineering Management, 2007 IEEE International Conference on (pp. 2120-2123). IEEE.

Xiaochun, Z., Bo, Z., Li, H., Junbo, C., & Lu, C. (2008). An ExperienceBased Approach for Test Execution Effort Estimation. The 9th International Conference for Young Computer Scientists, 1193–1198.

Xue-Mei, L., Guochang, G., Yong-Po, L., & Ji, W. (2009, March). Research and implementation of knowledge management methods in software testing process. In Computer Science and Information Engineering, 2009 WRI World Congress on (Vol. 7, pp. 739-743). IEEE.

Xu-Xiang, L., & Zhang, W. N. (2010, December). The PDCA-based software testing improvement framework. In Apperceiving Computing and Intelligence Analysis (ICACIA), 2010 International Conference on (pp. 490-494). IEEE.

Zapata, D. I. C. (2005). El aprendizaje individual en la gestión del conocimiento. Revista GTI, 4(9), 29-35.

Anexo A

Propósito		Guiar el desarrollo de las pruebas de software
Criterios de entrada		- Descripción del problema - Formato: Resumen del plan de proyecto. - Formato: Defecto y Tiempo. - Estándares: Tipificación de defectos.
Paso	**Actividades**	**Descripción**
1	Planeación	- Comienza el registro de tiempo en tu log de tiempo para la fase de Planeación. - Generar u obtener documentación del proyecto (requisitos de prueba). - Comprender los requisitos. - Establecer el alcance y tareas a desarrollar. - Estimar el tiempo requerido para el desarrollo. - Completa el formato: Resumen del Plan de proyecto. - Completa el registro de tu tiempo en el formato de tiempo.
2	Desarrollo	- Iniciar el registro de tiempo en tu formato de tiempo para la fase. - Preparar identifica las condiciones, los objetivos y el alcance de las pruebas. - Diseñar los casos de prueba - Implementar y ejecutar las pruebas diseñadas. - Evalúa las diferencias y genera el reporte de los resultados. - Completa el registro de tu tiempo en el formato de tiempo
3	Postmortem	Completa el formato "Resumen del plan de proyecto" añadiendo los datos reales de tiempo, defectos y tamaño.

Criterios de Salida	- Pruebas realizadas - Formato "Resumen del plan de proyecto" completado, que incluye datos estimados y ejecutados. - Formatos de tiempo y defectos correctamente diligenciados.

Anexo B.

REGISTRO DE TIEMPO

PROYECTO	FECHA	FASE/ETAPA	INICIO	FIN	INTERRUPCION	TIEMPO	COMENTARIO

SEVERIDAD

Severidad corresponde al nivel de impacto del defecto en la funcionalidad del sistema

Codigo	Tipo	Descripcion
1	Bloqueante	El defecto impide que el sistema o aplicación cumpla con un requerimiento, implica la pérdida del servicio e impiden continuar con la operación del sistema y la ejecución de las pruebas.
2	Crítico	Operaciones esenciales son interrumpidas, compromete la seguridad del sistema o de las personas que usan el sistema. Se dividen en crítico mayor o critico menor
3	Menor	El defecto impide que una función de menor impacto para el sistema el cumplimiento de los requerimientos del usuario. El defecto no afecta el servicio.
4	Cosmético	Defectos originados por la mala ortografía, textos desalineados o elementos en pantalla mal situados o incumplimiento en el "look and feel" de la organización

Según (Wagner, 2008)

PRIORIDAD

Prioridad: Establece el grado de impacto del defecto sobre su operación

Codigo	Nivel	Descripcion
100	Urgente	El defecto afecta a una función importante o crítica de la aplicación y afectará el cronograma de las pruebas. Se requiere la máxima prioridad para el análisis y solución.
200	Medio	El defecto no tiene una afectación importante sobre el cronograma de las pruebas.
300	Baja	Este defecto no tiene impacto sobre el sistema que se encuentra en pruebas, ni sobre el cronograma de pruebas.

Anexo C.

REGISTRO DE DEFECTO

PROYECTO	CASO DE PRUEBA	ETAPA	NUMERO DEF.	TIPO DEFECTO	SEVERIDAD	PRIORIDAD	GENERADO POR OTRO DEFECTO	COMENTARIO

Anexo D.

TIPIFICACION DE DEFECTOS

Código	Tipo	Descripción
10	Código	El defecto corresponde a problemas con el desarrollo del sistema, por ejemplo: Error en la lógica del código, error en variables erróneas o no inicializadas, error por código fuente incompleto, error por codificación errada de los parámetros, errores en cálculos, desviación del diseño.
20	Datos	Error generado por los datos de prueba, como por ejemplo información incompleta, mala interpretación de los resultados. Testing El defecto corresponde a un error en la ejecución de pruebas, por ejemplo: Caso de prueba incompleto, incorrecto o faltante, mala ejecución en el plan de pruebas, registro de defectos que no son defectos
30	Testing	El defecto corresponde a un error en la ejecución de pruebas, por ejemplo: Caso de prueba incompleto, incorrecto o faltante, mala ejecución en el plan de pruebas, registro de defectos que no son defectos.
40	Infraestructura	El defecto corresponde a un error de hardware o de software del sistema, como por ejemplo: Error del servidor, fallos del sistema operativo, error de interfaces.
50	Requerimiento	Error en la documentación del requerimiento, porque está especificado incorrectamente, o el requerimiento es ambiguo, o presenta conflicto con otros requerimientos, o no especifica funcionalidades.
60	Parámetros	El defecto corresponde a error en la parametrización en el ambiente de pruebas, como por ejemplo: Error de parámetros mal configurados, o falta de configuración de parámetros.

Según (Wagner, 2008)

Anexo E.

PROYECTO 1
Se desarrollo una aplicación bancaria, donde el usuario puede conectarse al banco por Internet y realizar una serie de operaciones bancarias. Una vez accedido al banco con las consiguientes medidas de seguridad (clave de acceso y demás), la información de entrada del procedimiento que gestiona las operaciones concretas a realizar por el usuario requiere la siguiente entrada: • Código del banco. En blanco o número de tres dígitos. En este último caso, el primero de los tiene que ser mayor que 1. • Código de sucursal. Un número de cuatro dígitos. El primero de ellos mayor de 0. • Número de cuenta. Número de cinco dígitos. • Clave personal. Valor alfanumérico de cinco posiciones. • Orden. Este valor se introducirá según la orden que se desee realizar. Puede estar en blanco o ser una de las dos cadenas siguientes: o Talonario o Movimientos En el primer caso el usuario recibirá un talonario de cheques, mientras que en el segundo recibirá los movimientos del mes en curso. Si este código está en blanco, el usuario recibirá los dos documentos.
CRITERIO DE FINALIZACIÓN DE LA PRUEBA
El criterio de finalización de la prueba es que se logra una cobertura de partición del **100**% de equivalencia y todos los casos de prueba deben dar como resultado un estado de **aprobación** en la ejecución".

PROYECTO 2
Un programa toma como entrada un archivo con la información de sus empleados y contiene los siguientes campos: • Número de identificación: es un campo de números enteros positivos de 3 dígitos (excluido el 000). • Apellidos y Nombres: es un campo alfanumérico de 25 caracteres. • Meses: es un campo que indica el número de meses que lleva trabajando el empleado. Es un entero positivo (incluye el 000) de 3 dígitos. • Directivo: es un campo de un solo carácter que puede ser «S» para indicar que el empleado es un directivo y «N» para indicar que no lo es. El programa asigna una prima (que se imprime en un listado) a cada empleado según las normas siguientes: • P1 a los directivos con, al menos, 12 meses de antigüedad • P2 a los no directivos con, al menos, 12 meses de antigüedad • P3 a los directivos sin un mínimo de 12 meses de antigüedad • P4 a los no directivos sin un mínimo de 12 meses de antigüedad
CRITERIO DE FINALIZACIÓN DE LA PRUEBA
El criterio de finalización de la prueba es que se logra una cobertura de partición del **100**% de **equivalencia, valores límites.** Todos los casos de prueba deben dar como resultado un estado de **aprobación** en la ejecución.

PROYECTO 3

Un hotel desea verificar un módulo de su aplicación que utiliza para calcular la cuenta de los clientes que se han alojado. Los datos que posee son:

- DNI-Cliente: es un atributo de números enteros positivos de no más de 8 dígitos y no menos de 7.
- Tipo-Cliente: es un atributo enumerado que solo puede contener uno de los valores siguientes: "regular" o "VIP".
- Tipo-Habitación: es un atributo enumerado que solo puede contener uno de los valores siguientes: "simple", "doble", "matrimonial".
- Fecha-Ingreso: es un atributo de tipo DATE con el formato siguiente: (DD/MM/AAAA)
- Cantidad-Noches: es un atributo de números enteros positivos que no puede superar el valor 30.

El módulo para probar calcula el monto que debe pagar el cliente por su estancia en el hotel, en base al tipo de cliente, tipo de habitación, fecha de ingreso y cantidad de noches. La siguiente lista muestra las posibles tarifas, descuentos que puede recibir el cliente, y opciones de pago:

- Habitación Simple: $150.
- Habitación Doble: $ 250.
- Habitación Matrimonial: $ 300.
- Clientes VIP (*): descuento del 10%.
- Permanencia mayor a 15 días (*): descuento del 30%.
- Cuenta inferior y hasta $1000: se debe pagar de contado.
- Cuenta superior a $1000: se puede pagar con tarjeta de crédito.
- Clientes VIP: pueden pagar con cheque.

Nota: Los descuentos no son acumulables.

CRITERIO DE FINALIZACIÓN DE LA PRUEBA

El criterio de finalización de la prueba es que se logra una cobertura de partición del **100**% de **equivalencia, valores límites y tablas de decisión**. Todos los casos de prueba deben dar como resultado un estado de **aprobación** en la ejecución.

Anexo F.

RESUMEN DEL PLAN DEL PROYECTO Programa:				Fecha:
Descripción:				
Resumen	**Plan**	**Real**	**Hasta la fecha**	
Minutos/LOC				
CP				
Defectos/CP				
Rendimiento				
Valoración/Fallo				

Tiempo por Fase (min.)	**Plan**	**Real**	**Hasta la fecha**	**% Hasta la fecha**
Planificación				
Desarrollo				
Postmorten				
Total				
Tiempo máximo				
Tiempo mínimo				

Defectos Introducidos	**Plan**	**Actual**	**Hasta la fecha**	**% Hasta la fecha**	**Def./Hora**
Planificación					
Desarrollo					
Total					

Anexo G.

LISTA DE CHEQUEO DE CASOS DE PRUEBA

ATRIBUTO	LISTA DE CHEQUEO	SI	NO
Calidad	Correcto. Es apropiado para los probadores y el entorno		
	Exacto. Su descripción se puede probar		
	Económico. Tiene sólo los pasos o los campos necesarios para su propósito		
	Confiable y repetible. Se obtiene el mismo resultado sin importa qué se pruebe		
	Rastreable. Se sabe qué requisito se prueba		
	Medible. Retorna al estado de la prueba sin valores en su estado		
Estructura y capacidad de prueba	Tiene nombre y número		
	Tiene un propósito declarado que incluye qué requisito se está probando		
	Tiene una descripción del método de prueba		
	Especifica la información de configuración —entorno, datos, pre-requisitos de prueba, seguridad—		
	Tiene acciones y resultados esperados		
	Guarda el estado de las pruebas, como informes o capturas de pantalla		
	Mantiene el entorno de pruebas limpio		
	No supera los 15 pasos		
	La matriz no demora más de 20 minutos para probarse		
	El script automatizado tiene propósitos, entradas y resultados esperados		
	La configuración ofrece alternativas a los pre-requisitos de la prueba cuando es posible		
	El escenario de aplicación se relaciona con otras pruebas		
Administración de la configuración	Emplea convenciones de nomenclatura y numeración		
	Guarda en formatos especificados los tipos de archivo		
	Su versión coincide con el software bajo prueba		
	Incluye objetos de prueba necesarios para el caso, tales como bases de datos		
	Almacena con acceso controlado		
	Realiza copias de seguridad en red		
	Archiva por fuera del sitio		

(Dustin, 2003)

Anexo H.

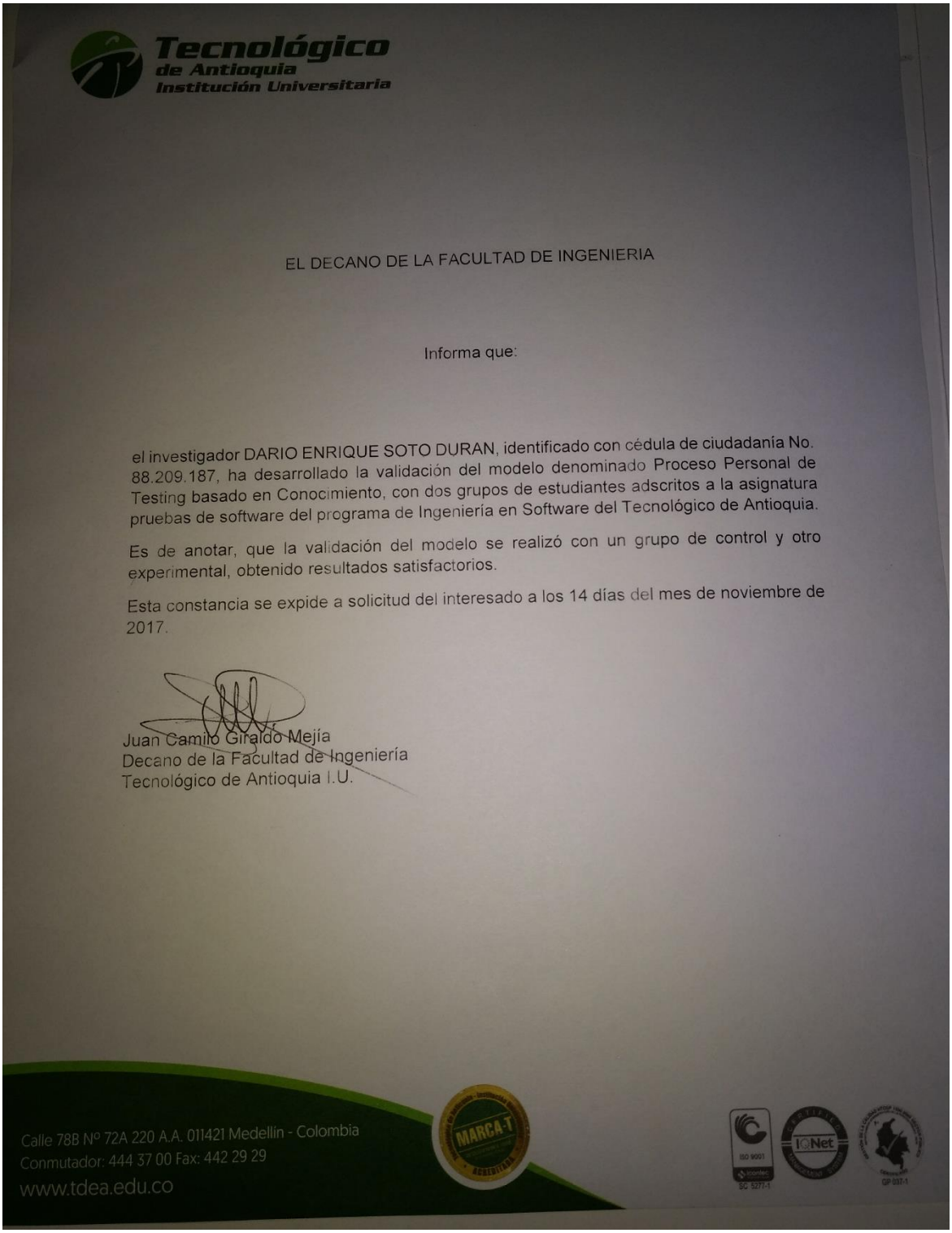
PLANTILLA DE CASO DE PRUEBA-FUNCIONAL

Proyecto No.:		**Página No.:**	
Nombre del Proyecto:			
Caso No.:		**Ejecución No**	
Nombre del Caso:		**Nombre**	
		Estado de la prueba:	
Módulo/Función/Código de la Unidad bajo prueba:		**Requisito No.**	
		Nombre:	
Escrito por:		**Ejecutado por:**	
Fecha:		**Fecha:**	
Descripción del caso de prueba (propósito y método):			
Configuración de la prueba para (H/W, S/W, N/W, datos, pre-requisitos de prueba, seguridad y tiempo):			

Paso	Acción	Resultados esperados	Pasado/Fallido
1			
2			
…			

(Aristegui, 2010)

Anexo I.



Tecnológico de Antioquia
Institución Universitaria

EL DECANO DE LA FACULTAD DE INGENIERIA

Informa que:

el investigador DARIO ENRIQUE SOTO DURAN, identificado con cédula de ciudadanía No. 88.209.187, ha desarrollado la validación del modelo denominado Proceso Personal de Testing basado en Conocimiento, con dos grupos de estudiantes adscritos a la asignatura pruebas de software del programa de Ingeniería en Software del Tecnológico de Antioquia.

Es de anotar, que la validación del modelo se realizó con un grupo de control y otro experimental, obtenido resultados satisfactorios.

Esta constancia se expide a solicitud del interesado a los 14 días del mes de noviembre de 2017.

Juan Camilo Giraldo Mejía
Decano de la Facultad de Ingeniería
Tecnológico de Antioquia I.U.

Calle 78B Nº 72A 220 A.A. 011421 Medellín - Colombia
Conmutador: 444 37 00 Fax: 442 29 29
www.tdea.edu.co