PROCESO DE PRUEBAS PARA PEQUEÑAS ORGANIZACIONES DESARROLLADORAS DE SOFTWARE

MARTHA LUCIA ROJAS MONTES

Director: Ph.D. FRANCISCO JOSÉ PINO CORREA

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Sistemas – Grupo de Investigación IDIS

Línea de Investigación Calidad del Software: Producto y Proceso

Popayán, Septiembre de 2013

Agradecimientos

“A ti Dios Mío, por no abandonarme en mi camino, gracias por ayudarme a levantarme en mis fracasos, por aprender de ellos y principalmente por permitirme realizar el sueño más importante de mi vida.”

A mi familia, por su eterna paciencia, sin esperar nada a cambio han acompañado cada momento de mi vida. Papi espero ser tu orgullo. Mami, aquí tienes mi esfuerzo, tarde pero aquí está, este triunfo es de las dos, gracias por apoyarme.

A mi novio Ing. Luis Carlos Imbachi por estar conmigo en este tiempo tan importante para mí y por su incesante colaboración.

Al Ph.D. Francisco José Pino Correa por su dedicación, tiempo, apoyo y enorme conocimiento para guiarme en este reto.

A Nexura Internacional S.A.S por abrirme las puertas de la empresa, por el tiempo y por su apoyo.

A mis amigos, y educadores, fieles compañeros en este proceso con su ánimo, colaboración y palabras de aliento.

Para finalizar, mi agradecimiento a la Universidad del Cauca, institución que me forjó como persona, brindándome la oportunidad a través del programa de Ingeniería de Sistemas de realizar mis estudios de pregrado.

TABLA DE CONTENIDO

Pág.

LISTA DE FIGURAS

LISTA DE TABLAS

Capítulo 1

1. INTRODUCCIÓN

En la actualidad las pequeñas organizaciones [1] compiten con grandes empresas en condiciones y oportunidades desiguales, que van desde el mas elevado costo de las materias primas, insumos y/o productos, hasta las políticas reglamentarias de las entidades financieras [42] [43]. Sin embargo, la industria en Colombia está conformada en su mayoría por pequeñas empresas [50]. Esto ocurre también para la industria del software [46]. Entonces es importante abordar trabajos que fortalezcan el que hacer de este tipo de empresas con el fin de que puedan tener mayor competitividad [42] [45] [47]. En este sentido, las pequeñas organizaciones software necesitan llevar a cabo sus pruebas de manera planeada y sistemática con el fin de promover el desarrollo de software de calidad en el importante sector económico en el que se desenvuelven para obtener ventajas competitivas.

Las empresas de software, grandes o pequeñas, reconocen hoy en día la importancia de las pruebas de software como un instrumento para asegurar la calidad de los productos desarrollados, de esta manera, se protegen en cierta medida de los riesgos de litigios, sobrecostos y pérdida de reputación ante los clientes [9]. El mercado valora, cada día más, la calidad, por lo tanto, las compañías exigen la disminución de errores y penalizan los retrasos en entregas y las cancelaciones de proyectos.

Esta situación conduce a las pequeñas organizaciones a cambiar su orientación hacia las pruebas, y verlas como un proceso necesario para apoyar la calidad de sus productos. Las empresas dedicadas al desarrollo de software deben aumentar la calidad de sus procesos para evitar los problemas inherentes a sus productos: plazos y presupuestos incumplidos, insatisfacción del usuario, escasa productividad y la baja calidad en el software producido. La ingeniería del software cuenta con la premisa de que “la calidad del producto depende en gran parte de la calidad del proceso” [14] [48], por lo que para proporcionar software de alta calidad son muy importantes los procesos utilizados, lo que motiva un cambio de perspectiva para mejorar la calidad, desde una visión centrada en los procesos más que en los productos. Estos procesos junto a las personas, la organización y los procedimientos utilizados para definirlos, ejecutarlos y mantenerlos, forman la base sobre la cual se sustenta el desarrollo exitoso de productos software de alta calidad, por lo que contribuir hacia su implantación debe ser un propósito fundamental [49].

Definir un proceso de pruebas dentro de una organización puede ser muy valioso en la medida en que se incremente la calidad del producto, se facilite la comprensión y comunicación entre los miembros del equipo, se dé el soporte necesario para mejorar

[1] El término “pequeñas organizaciones” denota en este documento el concepto very small entities – VSEs, el cual hace referencia a una empresa, organización, departamento, grupo o proyecto que cuenta con menos de 25 personas [ISO/IEC 29110]

continuamente el proceso y se proporcione soporte a la ejecución automática de ciertas tareas [10].

En este trabajo de grado se presenta un proceso liviano definido para soportar y guiar las pruebas en pequeñas organizaciones desarrolladoras de software. Con la incorporación de técnicas de pruebas funcionales, se hacen más llevaderas las actividades requeridas para evaluar un software de manera organizada y sistemática, especialmente aquellas que tienen que ver con el diseño y ejecución de las pruebas. Este trabajo ha sido desarrollado a partir del análisis de los procesos de pruebas existentes en los modelos de referencia y en la literatura, y ha sido exitoso tras su ejecución en una pequeña empresa de desarrollo software.

1.1 PLANTEAMIENTO DEL PROBLEMA

1.1.1 Descripción del Problema

La calidad de *software* a través de los años se ha enfocado en ofrecer a las organizaciones, estrategias para el desarrollo de productos *software* que satisfagan las necesidades reales de los clientes, sin embargo, estas estrategias por si solas no han sido suficientes para que dichos productos incorporen la calidad requerida. Productos que no hacen exactamente lo que se espera, que no se utilizan por la dificultad de su manejo, que son imposibles de mantener cuando desaparece la persona o personas que lo desarrollan, que no se terminan nunca, y que son poco seguros, son problemas que aparecen como consecuencia de la falta de calidad en sistemas ya implantados o en producción, y las pérdidas que estos han ocasionado, ha hecho necesaria la actualización de la percepción que se tiene acerca de la calidad del *software* [14]. Por ello las organizaciones deben preocuparse por la calidad de sus productos, con el fin de aumentar el grado de satisfacción y confianza por parte de sus clientes. Una forma para que los productos cumplan la calidad requerida por el cliente es usar técnicas de pruebas apropiadas en cada etapa del ciclo de desarrollo de un producto *software* a través de un proceso de pruebas definido para tal fin.

Actualmente las organizaciones se preocupan de la calidad de los procesos que se siguen para desarrollar *software* basándose en la premisa “con un proceso de calidad se obtiene un producto de calidad” [14], sin embargo, con base en la literatura analizada no se identifica aun un proceso de pruebas para la industria del *software* enfocado al contexto de las pequeñas organizaciones que defina explícitamente las actividades y técnicas de pruebas funcionales que deben utilizarse oportuna y eficazmente en el proceso de desarrollo de un producto *software*. En el desarrollo de *software*, el proceso de pruebas es generalmente la fase más costosa, que puede ser responsable de más del 50 % de los costos de desarrollo [15]. Esto parece ser una gran inversión, sin embargo, el proceso de pruebas podría ser eficiente tan solo con el 20 % de los recursos [16], lo cual supone costos más bajos y productividad más elevada [18]. Cuando las pruebas están relacionadas con conceptos como aseguramiento de la calidad, satisfacción del cliente e imagen pública, el desarrollo de un proceso de pruebas se hace imprescindible, por tanto, no tener un proceso de pruebas adecuado conllevará problemas de calidad, de sobrecostos y afectará el buen nombre de la organización ante los clientes [9]. Entonces

es importante trabajar sobre la definición y la integración de actividades y técnicas[2] de pruebas funcionales que garanticen la calidad del producto *software* en el contexto de la pequeña organización mediante la definición y adopción de un proceso de pruebas adecuado.

A partir de lo anterior el proyecto se plantea el siguiente interrogante: ¿Cuáles serían las actividades y técnicas de prueba funcionales adecuadas para dar soporte al desarrollo de un producto *software* en pequeñas organizaciones? Para dar respuesta a esta pregunta se ha desarrollado el presente trabajo de grado, en el cual se plantea un proceso de pruebas que recoge las actividades mínimas de pruebas que debe seguir una pequeña empresa. Estas actividades se establecen a partir de la comparación entre los procesos de pruebas exitosos en pequeñas organizaciones con la norma ISO/IEC 29119 pruebas de software y se complementa tras la experiencia adquirida en la aplicación del caso de estudio (en el capitulo tres se hace énfasis en este tema). Por otro lado, se presentan las técnicas de pruebas funcionales adecuadas para dar soporte a las actividades de pruebas en la pequeña empresa a partir del análisis hecho a cada una de ellas, la asesoría de un asesor experto en pruebas de software y la aplicación en el caso de estudio. Estas técnicas se incorporan al proceso de pruebas específicamente en las actividades de diseño y ejecución de pruebas con el objeto de ser una ayuda en la organización y sistematización de las pruebas.

1.1.2 Justificación

Las pruebas de *software* son comúnmente realizadas para el control de calidad en los proyectos de desarrollo. Son un componente crítico en el proceso de desarrollo de *software*, es una de las actividades del proceso más difíciles y costosas, y proveen un soporte para desarrollo de *software* de alta calidad [10]. Debido a que son éstas quienes entregan información acerca de la calidad del producto, las pruebas de *software* pueden sacar a la luz la falta de calidad, que se revela en los defectos encontrados. Las pruebas de *software* a través de un proceso sistemático controlado garantiza la fiabilidad de los productos [6]. En un proceso de pruebas es importante conocer exactamente lo que se va a probar (el objeto de prueba), contra lo que se va a comparar (la prueba básica) y como se va a probar (el método y la técnica de prueba) [8]. No tener un proceso de pruebas puede conllevar a altos costos debido al re-trabajo, la pérdida de productividad, pérdida de la reputación con el cliente y la pérdida de competitividad reflejados en la tardía disponibilidad del nuevo producto o en la falta de calidad del mismo [8].

Este proyecto busca ofrecer un proceso de pruebas de software para el contexto de pequeñas organizaciones, que integre actividades y técnicas de pruebas funcionales, con el fin de apoyar la calidad del proceso de desarrollo *software* y sus productos asociados. El proceso que se plantea debe ser de fácil uso y aplicación dado que, en la actualidad, los procesos orientados a las pequeñas organizaciones no involucran las técnicas adecuadas para llevar a cabo las actividades de dichos procesos y son por tanto, difícilmente aplicables a ese perfil de organizaciones.

[2] Técnicas de prueba: conjunto de saberes prácticos o procedimientos para obtener un resultado deseado.

Algunos motivos por los que las pequeñas organizaciones no acogen un proceso de pruebas son el coste, la duración y la complejidad del mismo, que llegan a ser desproporcionados respecto a los recursos disponibles de estas organizaciones [6]. Además, su nivel de madurez, especialmente en lo que respecta a procesos de prueba de *software*, es normalmente muy bajo, y cualquier avance en la madurez de la organización en temas de procesos de prueba que se consiga con las actividades y técnicas de pruebas propuestas, supondrá un avance importante respecto a la situación de partida en la que se encuentra actualmente [5].

1.2 OBJETIVOS

En el presente trabajo de grado se ha propuesto los siguientes objetivos: general y específicos.

1.2.1 Objetivo General

Definir[3] un proceso de pruebas y vincular un conjunto de técnicas apropiadas de pruebas funcionales[4] que apoyen la ejecución del proceso definido en el contexto de las pequeñas organizaciones (VSEs).

1.2.2 Objetivos Específicos

- Definir un proceso de pruebas adaptado a las características de pequeñas organizaciones desarrolladoras de *software* a partir del análisis de los procesos de pruebas existentes en los modelos de referencia de procesos y en la literatura.

- Identificar, analizar y relacionar un conjunto de técnicas de pruebas funcionales que puedan dar soporte a alguna(s) actividades del proceso de prueba definido

- Modelar el proceso definido mediante una herramienta de modelado de procesos.

- Evaluar el proceso y las técnicas de pruebas funcionales mediante un caso de estudio.

1.3 SOLUCION PROPUESTA

Esta tesis propone la definición de un proceso de pruebas de software para el contexto de pequeñas organizaciones, que integre actividades y técnicas de pruebas funcionales, con el fin de apoyar la calidad del proceso de construcción e integración de *software*[5] y sus

[3] Describir y relacionar los elementos de proceso (actividades, tareas, roles y productos de trabajo) para lograr los objetivos establecidos.

[4] Pruebas funcionales: Pruebas de software basadas en los requisitos funcionales [ISO/IEC 29119 Parte 4. Técnicas de pruebas]

[5] Construcción e Integración del software: De acuerdo a la ISO/IEC 12207: i) El propósito del proceso de construcción del software es producir unidades ejecutables de software que reflejen adecuadamente el diseño del software y ii) El propósito del proceso de integración de software es combinar las unidades y componentes de software compatibles con el diseño de software, que demuestran que los requisitos de software funcionales y no funcionales se cumplen en una plataforma operacional equivalente o completa.

productos asociados. En este sentido la propuesta pretende aprovechar la información que aportan los procesos de pruebas encontrados en la literatura y de ellos abstraer las actividades y productos de trabajo que generen un valor agregado a las pruebas. Para apoyar el proceso se relacionan las técnicas funcionales de pruebas, se comparan y se evidencian de forma clara las que más se adaptan a las pequeñas organizaciones por ser de fácil uso y aplicabilidad.

El proceso que se plantea es de fácil uso y aplicación, con lo que espera contribuir con procesos efectivos que fortalezcan la competitividad de las pequeñas empresas, adecuando a sus propias características los procesos y técnicas internacionalmente reconocidos.

1.4 ESTRUCTURA DEL DOCUMENTO

En este documento se encuentran diferentes secciones que contienen la descripción de los conceptos teóricos y la metodología utilizada para el desarrollo del proyecto. A continuación se describe de manera general el contenido de esta monografía y su organización.

En el capítulo 1 se presenta la problemática que motivó el planteamiento de este proyecto, la justificación del desarrollo del mismo, los objetivos que se definieron y la solución propuesta.

En el capítulo 2 se describen las bases teóricas que enmarcan el proyecto, teniendo en cuenta los conceptos básicos en el área de procesos de pruebas de software, las principales investigaciones realizadas alrededor de ésta área, estándares relacionados con el área, modelos que han servido de referencia para procesos de pruebas, procesos de mejora de pruebas y procesos de pruebas exitosos en pequeñas organizaciones desarrolladoras de software que se utilizaron como base para la definición del proceso planteado en este proyecto.

En el capítulo 3, se cubre la definición y desarrollo del proceso de pruebas propuesto, las técnicas de pruebas que apoyan el proceso definido, y finalmente se presenta el modelado del proceso que permite entender fácilmente las actividades e información planteada en el proceso, pues se muestra de una forma organizada y sistemática

En el Capítulo 4, se presenta la aplicación y ajuste del proceso de pruebas en la práctica (caso de estudio real en una pequeña organización desarrolladora de *software*), dando a conocer las actividades de intervención hecha tras las actividades establecidas en el procedimiento de campo del caso de estudio, la recolección de datos y el análisis de los resultados obtenidos.

El capítulo 5 describe las conclusiones que se establecieron a partir de la experiencia adquirida en el desarrollo y aplicación del proyecto y se proponen varias ideas de trabajo futuro para la continuidad del proyecto.

En el capítulo 6 se muestra la bibliografía y documentación empleada en la realización del proyecto.

Capítulo 2

2. ESTADO DEL ARTE

En este capítulo se presentan los trabajos relacionados con el área de procesos de pruebas, estándares que se relacionan con el área, modelos que han servido de referencia para procesos de pruebas, procesos de mejora de pruebas y procesos de pruebas. Se establece una comparación entre los procesos de pruebas, tomando como punto de partida los aspectos más relevantes de cada uno de ellos y de esta forma evidenciar la diferencia con la propuesta a desarrollar en este trabajo de grado. Finalmente se hace una discusión al respecto y se describen los aportes de este proyecto.

Los trabajos relacionados con el área de procesos de pruebas se describen y analizan en este capítulo en cuatro secciones. En la primera sección, se describen algunos modelos de referencia de procesos para la industria del software, ISO/IEC 12207 Procesos del ciclo de vida del software, CMMI y Competisoft, en los cuales se han incorporado procesos de pruebas de acuerdo a las características de cada modelo particular descritos en el literal 2.1.4. En la segunda sección se trata la norma ISO/IEC 29119 Pruebas de Software, está norma proporciona información valiosa al respecto de las pruebas de software (vocabulario, procesos, documentación, técnicas y un modelo de evaluación del proceso de pruebas de software). En esta sección también se muestra la norma ISO/IEC 29110 como norma a consultar por el enfoque que en esta se hace hacia la obtención de procesos en el contexto de las VSEs. En la tercera sección se muestran los procesos de mejora de pruebas, TMMI, TPI, Propuesta para el testeo de Pymes y Minimal test Practice Framework.

En la última sección se hace una breve descripción de los procesos de pruebas aplicados en pequeñas organizaciones desarrolladoras de software, entre los que se encuentran: i)ISO/IEC 2911 Pruebas de Software. Parte 2 – Proceso de pruebas, ii) Software Qualification Testing Process, iii) Proceso de pruebas de software para el modelo de referencia Competisoft, iv) TestPAI y v) Proceso de pruebas para pequeñas empresas: En un escenario Brasileño. Estos procesos aportan al proceso de pruebas definido en este trabajo de grado, actividades exitosas en el contexto de la pequeña empresa.

2.1 TRABAJOS RELACIONADOS

De acuerdo con [3] [4] la calidad del *software* debe propender para que el *software* haga lo que se espera de él, dentro del tiempo y presupuesto establecidos y que permita a las partes involucradas alcanzar los objetivos derivados del negocio; para que se pueda cerciorar dicha calidad en un producto, la ISO ha definido el aseguramiento de la calidad como "Un conjunto de actividades preestablecidas y sistematizadas, aplicadas al sistema de calidad, que han demostrado ser necesarias para dar confianza adecuada que un producto o servicio satisfará los requisitos para la calidad" [4]. Según [4] la definición plantea claramente que el aseguramiento de la calidad del *software* es un proceso que

implica planeación, seguimiento y verificación aplicado durante todo el proceso de desarrollo, garantizando al cliente que la organización dispone de los procedimientos efectivos para asegurar la satisfacción de los requerimientos establecidos.

De acuerdo con [3] el aseguramiento de la calidad de *software* se ha convertido en una necesidad prioritaria para las organizaciones públicas y privadas que desarrollen productos software, ya sea para uso interno o en implementaciones externas para clientes, porque los errores en el *software* repercuten directa o indirectamente en graves consecuencias para la organización. Un error que desde el punto de vista de codificación puede ser relativamente simple de corregir, puede resultar muy difícil y costoso de detectar y puede llegar a tener graves efectos en la organización, por tal razón, las pruebas de *software* son la esencia del aseguramiento de la calidad [3].

Las pruebas de *software* son una de las disciplinas que tienen la capacidad de proporcionar asistencia para mejorar la calidad de los productos de una organización, ya que su objetivo es evaluar cómo el producto cumple con los requisitos de los clientes a través de la ejecución controlada del *software* [1].

De acuerdo con [1], la prueba del *software* es una actividad que se usa para garantizar la fiabilidad de los productos, complementándola con las revisiones y las técnicas de especificación formal y rigurosa. Particularmente, las pruebas funcionales se valen de técnicas especializadas (partición de equivalencias, análisis de valor límite, arreglos ortogonales, entre otras) para verificar mediante el funcionamiento experimental del producto, si este satisface las necesidades solicitadas por los usuarios [7].

La ejecución de las pruebas funcionales de *software* dentro de un proceso sistemático de pruebas, seguramente conllevará a la detección de errores de manera más efectiva, asegurando la calidad desde el primer momento en que el defecto es encontrado y reportado [32]. Asegurar la calidad desde las primeras etapas de desarrollo del producto *software* involucra que los costos del control en las etapas posteriores tiendan a disminuir al tener menos aspectos que controlar, pues, la calidad estaría asegurada en sus bases [2]. La calidad que pueden alcanzar los productos *software*, y en general cualquier producto, está sometida a la manera como se desarrolla cada una de las etapas de la vida del producto, partiendo por la definición de la idea del producto hasta la entrega y mantenimiento del mismo [2]. Así la entrega de calidad a un producto considera actividades tales como: i) aplicación de técnicas formales a lo largo de todo el proceso y ii) pruebas acuciosas en diferentes etapas del desarrollo [2]. Se concluye entonces, de acuerdo con la bibliografía citada, que seguir de manera organizada un proceso de pruebas en cada etapa de desarrollo de un producto *software* es de vital importancia para el éxito del mismo.

A continuación se relaciona de manera resumida las propuestas que hay acerca de procesos de pruebas de *software* que han adelantado otros autores, así como estudios relacionados. Se quiere de esta manera relacionar dichos trabajos con el proceso de pruebas a desarrollar con el fin de establecer un paralelo que nos permita identificar claras diferencias, además de los aportes que estas propuestas puedan brindar al proyecto.

2.1.1 Modelos de Referencia de Procesos para la Industria del Software

A continuación se describen brevemente algunos modelos de referencia de procesos que consideran el proceso de pruebas

2.1.1.1 ISO/IEC 12207 Procesos del ciclo de vida del *Software*

La ISO/IEC 12207 proporciona un conjunto definido de procesos aplicables a todo el ciclo de vida del software que facilitan la comunicación entre compradores, proveedores y otras partes interesadas en el producto s*oftware* [20]. Las actividades y tareas que propone esta norma se desarrollan dentro de cada proceso dando cumplimiento a los objetivos establecidos en cada uno de ellos, son 43 procesos clasificados según su contexto y divididos en dos grandes grupos:

1er. Grupo: Procesos del contexto del sistema

- Procesos de concertación (dos procesos)
- Procesos organizacionales del proyecto (cinco procesos)
- Procesos del proyecto (siete procesos)
- Procesos Técnicos (once procesos)

2do. Grupo: Procesos específicos del software

- Procesos de implementación de *software* (siete procesos)
- Procesos de soporte de *software* (ocho procesos)
- Procesos de reutilización de *Software* (tres procesos)

La Estructura de este estándar ha sido concebida de manera flexible, de tal manera que las organizaciones puedan acogerse a los procesos anteriormente nombrados de acuerdo a sus necesidades de negocio y dominio de aplicación.

En lo que respecta a pruebas, este estándar incluye un proceso llamado: "*Software Qualification Testing Process"* incluido *en* los procesos específicos de software, más exactamente dentro de los siete procesos de implementación de *software* sugeridos por la norma. Este proceso se discutirá en el literal 2.4.1 ubicado en los procesos de pruebas.

2.1.1.2 CMMI

El proyecto CMMI (Capability Maturity Model Integration) fue desarrollado por el instituto de ingeniería de *software* (SEI) con el fin de proveer a la industria de herramientas que mantengan la mejora de los procesos empleados para la construcción de productos y servicios, cubriendo todo el ciclo de vida mediante 22 áreas de procesos. Cada área de proceso está formada de metas y, a su vez, cada meta se cumple a partir de la ejecución de diferentes prácticas establecidas por el modelo. El despliegue e implantación de CMMI implica costos y tiempo, razón por la cual las pequeñas empresas se niegan a usarlo [24] [25], por lo que se hace necesario adecuar este tipo de modelos al contexto de la pequeña empresa. Con relación a las pruebas de *software*, no define explícitamente un área de proceso para tal fin, se describen algunas actividades de pruebas, pero el nivel de

detalle es limitado, especialmente desde el punto de vista del profesional especializado en pruebas [28].

2.1.1.3 Competisoft

El modelo Competisoft (Mejora de Procesos para Fomentar la Competitividad de la Pequeña y Mediana Industria del *Software* de Iberoamérica) se compone de diez procesos incluidos dentro de tres categorías (alta dirección, gerencia y operación), su objetivo primordial es incrementar el nivel de competitividad de las PYMES Iberoamericanas productoras de *software*, mediante la mejora y evaluación continua de sus procesos [27], razón por la cual, dichos procesos están enfocados a entornos pequeños, especialmente en el contexto de América Latina [26]. Competisoft al igual que CMMI no define un proceso exclusivamente de pruebas, trata las pruebas como algo genérico y este proceso necesita mucha atención y una definición más exhaustiva**.**

Modelos como ISO/IEC 12207, CMMI y Competisoft han servido de referencia para la creación de procesos de pruebas como los que se muestran en los literales 2.1.4.1, 2.1.4.2, 2.1.4.3 y 2.1.4.4.

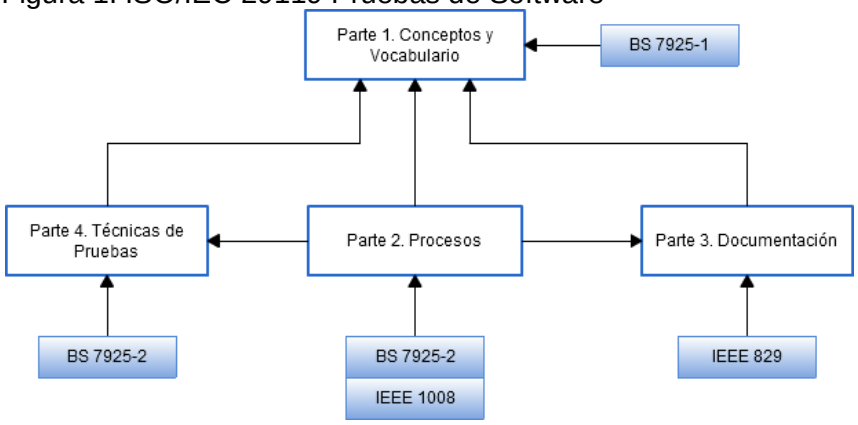
2.1.2 Estándares Relacionados con Pruebas de *Software* y VSEs

Las organizaciones de normalización como ISO (International Organization for Standardization), IEEE (Institute of Electrical and Electronics Engineers), y BSI (British Standards Institution) entre otras, han publicado varias normas relacionadas directa o indirectamente con pruebas de *software* y VSEs. En este trabajo se da una particular importancia a dos de ellas, ISO/IEC 29119 Pruebas de *software* e ISO/IEC 29110 Perfiles de ciclo de vida para pequeñas organizaciones (VSEs).

2.1.2.1 ISO/IEC 29119 Pruebas de Software

Es un proyecto que aun está siendo desarrollado y revisado por profesionales y académicos de todo el mundo, el objetivo de esta norma es recopilar los conceptos y vocabulario (BS 79251), procesos (BS 79252, IEEE 10087), documentación (IEEE 829) y técnicas de pruebas (BS 79252) para formar un solo estándar de pruebas de *software*. En la Figura 1 se muestra la estructura de la norma.

Figura 1. ISO/IEC 29119 Pruebas de Software



Esta norma es una fuente de información muy importante en este proyecto, pues maneja gran información referente a pruebas. En este trabajo de grado se presta un particular interés a las partes dos, tres y cuatro. De la parte dos se utiliza el proceso de pruebas como referente del proceso de pruebas definido en este trabajo de grado, de la parte tres se utiliza la documentación para nutrir parte de los productos de trabajo del proceso definido y la parte cuatro se utiliza como fuente de información para las técnicas de pruebas.

Es preciso aclarar que la ISO/IEC 29119 Pruebas de software, en su parte 4 incluye técnicas de pruebas definidas a nivel general, es decir, puntualiza una lista de técnicas de pruebas y algunos ejemplos, pero no establece ninguna relación con las actividades de pruebas enfocadas hacia pequeñas organizaciones, entonces Integrar técnicas de pruebas al proceso de pruebas y trabajar enfocados hacia las VSEs es la diferencia con esta norma.

2.1.2.2 ISO/IEC 29110 Perfiles de ciclo de vida para pequeñas organizaciones (VSEs).

Este estándar reúne un conjunto de normas y guías de acuerdo con las características y necesidades de las VSEs, este conjunto de normas se conoce como perfiles VSEs. El propósito de un perfil VSE es definir un subconjunto de normas pertinentes al contexto del entorno VSE. Para el uso de los perfiles se accede al documento acordado internacionalmente que incluye las especificaciones de uno o más perfiles ISP. La conformidad con estos perfiles es la forma mediante la que las VSEs muestran y documentan el uso y la comprensión de los estándares internacionales.

Los aportes más significativos que hacen estas normas al proyecto que aquí se plantea son los siguientes:

- La ISO/IEC 29119 es un modelo de referencia para llevar a cabo pruebas de *software* que describe algunas técnicas de pruebas. Este modelo debe ser considerado para definir el proceso de pruebas a desarrollar en este trabajo ya que será un estándar internacional.

- El aporte más significativo de la norma ISO/IEC 29110 a este trabajo es el enfoque que se hace hacia la obtención de procesos en el contexto de las VSEs.

2.1.3 Procesos de Mejora de Pruebas

2.1.3.1 TMMI

TMMI es un *framework* de mejora de procesos de pruebas, desarrollado por la Fundación TMMI como un complemento a CMMI. En TMMI las pruebas de *software* se aplican en su sentido más amplio con el fin de abarcar todas las actividades relacionadas con la calidad del producto desarrollado [28]. Este *framework* se soporta en cinco niveles de madurez (Iniciación, planificación, definición, gestión y medición y optimización) que definen la evolución de la mejora del proceso de pruebas. TMMI está diseñado para organizaciones

grandes por lo que es difícilmente aplicable a pequeñas organizaciones desarrolladoras de *software*, dado que, el coste de este proceso de mejora es desproporcional respecto a los recursos disponibles por una PYME [6].

2.1.3.2 TPI

TPI (Test Process Improvement) es un modelo de mejora de proceso de pruebas adaptado a su empresa desarrolladora Sogeti. TPI se soporta en 20 áreas claves, las cuales se van evaluando a medida que se evoluciona en el modelo. El estado de las áreas en el proceso de madurez se evalúa de acuerdo a 13 niveles de madurez. Un nivel más alto implica más eficiencia desde el punto de vista de tiempo, de coste y/o de calidad [29]. Al igual que TMMI este modelo de mejora está diseñado para organizaciones grandes por lo que es difícilmente aplicable a pequeñas organizaciones desarrolladoras de *software* por tanto los costos al adquirirlo e implantarlo son desproporciónales respecto a los recursos disponibles por una PYME.

2.1.3.3 Propuesta para el testeo de Pymes

Es una propuesta que se hace con el fin de preparar a las empresas en sus procesos de pruebas antes de aplicar un proceso de mejora, se plantea: i) Pruebas Unitarias por programador y ii) Personal exclusivamente dedicado a las pruebas [5]. Esta propuesta fue aplicada en una PYME, donde se concluye que la aplicación de dichas actividades tuvo un gran éxito tras hacer una comparación del antes y el después de la actividad de prueba [6], pero las actividades planteadas son muy amplias y traen consigo otras tantas actividades implícitas que no se plantean de la mejor manera a quienes quieren apropiarse de este proceso. En [5] se afirma que es de vital importancia adaptar un proceso de pruebas a pequeñas organizaciones debido a que la utilización de metodologías existentes es muy costosa y difícilmente adaptable a estos contextos.

2.1.3.4 Minimal Test Practice Framework

MTPF (Minimal Test Practice Framework) se especializa en mejora de procesos de pruebas para pequeñas empresas que resulta ser útil para las mismas planteándose este como un primer paso a la mejora de procesos de pruebas. Este Framework se estructura en cinco categorías y se nivela en tres fases [7], pero no se identifican las técnicas de pruebas asociadas a las actividades ejercidas en cada una de las categorías planteadas.

Cabe resaltar que las propuestas para mejora de procesos que se encuentran en la bibliografía definen la mejora, pero no muestran el proceso a mejorar ni el proceso mejorado.

2.1.4 Procesos de Pruebas

En este espacio se hace una descripción global de los procesos de pruebas encontrados en la bibliografía consultada, es en el capitulo tres donde se describen con mayor detalle

cada uno de ellos, pues es en ese punto donde se analizan los subprocesos, prácticas, actividades, objetivos, sub-practicas y tareas, que los soportan, para de este modo definir el proceso de pruebas que se propone en este trabajo de grado.

2.1.4.1 ISO/IEC 29119 Pruebas de Software. Parte 2 – Proceso de Pruebas

La ISO/IEC 29119 Pruebas de software en su parte 2 define un proceso de pruebas que intenta abarcar todos los sistemas de *software*, por tanto es muy grande y de difícil aplicabilidad a pequeñas organizaciones desarrolladoras de *software*, en esta situación, el proceso que se planea desarrollar en este trabajo sería una buena opción para pequeñas organizaciones desarrolladoras de *software* que busquen incorporar calidad a sus productos.

2.1.4.1 Software Qualification Testing Process

La ISO/IEC 12207 ejecuta sus actividades de prueba a través del proceso de pruebas (*Software Qualification Testing Process*) cuya finalidad es verificar que el producto de *software* integrado cumple con sus requisitos definidos. En este proceso se definen de manera general tareas comunes de acuerdo con las actividades establecidas [20]. El aporte que hace este proceso al proceso de pruebas planteado en este trabajo, es la forma en que debe adecuarse un proceso de pruebas a todo el ciclo de vida de un producto *software*.

2.1.4.2 Proceso de pruebas de *software* para el modelo de referencia Competisoft

Esta propuesta presenta un proceso de pruebas de *software* que se añade al nivel de operaciones en el modelo Competisoft. El proceso se compone de cinco subprocesos y actividades recomendadas cuyo objetivo es proporcionar orientación a la empresa de desarrollo de *software* en la realización de las pruebas de *software* a través de tareas a realizar durante todo el ciclo de vida de los proyectos. Este proceso se basa en TMMI y TPI, para la madurez de pruebas de integración y la mejora de procesos de prueba respectivamente. En este trabajo, se hace uso de la experiencia de sus autores como una guía para seleccionar y refinar algunas ideas de TMMI y TPI para la definición de los subprocesos y las prácticas aptas para el contexto de las pequeñas y medianas empresas, sin dejar de lado la estructura de Competisoft [23].

2.1.4.3 TestPAI

TestPAI es una propuesta para la mejora de procesos de pruebas compatible con CMMI, es decir, conserva su misma estructura. Los objetivos de esta propuesta son mejorar el proceso de pruebas al mismo tiempo que se produce la mejora en el resto de los procesos de la organización e incorporar herramientas para la mejora de una manera poco costosa y sencilla, por tanto, es fácilmente adaptable a la pequeña y mediana empresa. TestPAI define cinco objetivos (SG) con sus respectivas tareas asociadas (SP), dentro de las cuales se incluyen todas las prácticas que tienen que ver con pruebas (políticas de pruebas, plan de pruebas, casos de prueba, ejecución de pruebas y reporte de resultados). El mayor aporte que hace esta propuesta a nuestro trabajo es que TestPAI trae implícita la definición del proceso a mejorar.

2.1.4.4 Proceso de pruebas para pequeñas empresas: En un escenario Brasileño

Se define un proceso de pruebas para pequeñas empresas que incluye una serie de actividades y tareas relacionadas con la planificación y ejecución de las pruebas. El proceso de pruebas incluye la ejecución de dos actividades: 1. Planificación de la prueba y 2. Ejecución de la prueba [1]. La actividad 1 se compone de seis tareas y la actividad 2 de tres tareas en las cuales se especifican siempre los artefactos de entrada, artefactos de salida, criterios de entrada, criterios de salida, y roles esenciales. Sin embargo, estas tareas no son fácilmente ejecutables, teniendo en cuenta que este proceso se realizó con el fin de dar solución a una serie de falencias encontradas en la aplicación de procesos de pruebas de *software* en pequeñas organizaciones.

2.2 COMPARATIVA DE LAS PROPUESTAS ANALIZADAS VS PROPUESTA A DESARROLLAR

En la literatura se proponen procesos muy grandes o muy pequeños, que no se adecuan a las necesidades de las pequeñas organizaciones desarrolladoras de *software*, estos procesos suelen ser de difícil implantación en una VSE por su complejidad y abstracción ya que muchos de los cuales dan respuesta a la pregunta ¿Qué actividades ejecutar? con respecto al tema de pruebas en una etapa de desarrollo, pero no se abordan las técnicas funcionales que apoyen la ejecución del proceso de pruebas.

Para realizar el análisis de los procesos citados anteriormente, se han definido ocho criterios: Tipo, Licencia, Exclusivo de pruebas, Completamente definido, Actividades, Tareas, Incorpora técnicas de pruebas, Adecuado para VSEs (ver Tabla 1).

En la Tabla 2, se relacionan los procesos revisados y presentados anteriormente tratando los aspectos más relevantes de cada uno de ellos, de tal manera que sea posible hacer una comparación y evidenciar de forma clara la diferencia con el proceso de pruebas planteado en el trabajo que se ha realizado en este proyecto de grado.

Tabla 1. Criterios definidos

Criterio	Definición
Tipo	Alcance: 1. Procesos de todo el ciclo de vida del *software* 2. Proceso exclusivamente de pruebas 3. Mejora de los procesos de todo el ciclo de vida del *software* 4. Mejora del proceso de pruebas 5. Pruebas de Software
Licencia	Propietario: Una propuesta es de tipo propietario si el usuario tiene limitaciones para usarla, modificarla o redistribuirla y requiere permiso de una organización privada. Abierto: Una propuesta es de tipo abierto si el usuario NO tiene limitaciones para usarla o modificarla y se puede redistribuir libremente.
Exclusivo de pruebas	Una propuesta es exclusiva de pruebas si se enfoca hacia todas las prácticas relativas a pruebas.

Completamente Definida	Una propuesta está completamente definida si todas las prácticas descritas en la misma están completamente especificadas.
Incorpora Actividades	Una propuesta que define un proceso que contiene un conjunto de tareas definidas, que se llevan a cabo para cumplir los objetivos fijados.
Incorpora Tareas	Son el nivel de detalle con que se desglosan las actividades.
Incorpora Técnicas de Pruebas	La propuesta incluye procedimientos técnicos y de gestión que ayudan a desarrollar las actividades planteadas.
Adecuada para VSEs	La propuesta es adecuada a VSEs si su implantación en este tipo de organizaciones es relativamente sencilla.

Tabla 2. Evaluación de criterios

Criterios / Propuestas	Tipo	Licencia	Especifica de pruebas	Completamente Definida	Incorpora Actividades	Incorpora Tareas	Incorpora Técnicas de pruebas	Adecuada para VSE'S
ISO/IEC 12207	Procesos de todo el ciclo de vida del *software*	Abierto	NO	SI	SI	SI	NO	NO
CMMI	Mejora de los procesos de todo el ciclo de vida del *software*	Propietario	NO	SI	SI	SI	NO	NO
Competisoft	Mejora de los procesos de todo el ciclo de vida del *software*	Propietario	NO	SI	SI	SI	NO	SI
ISO/IEC 29119	Pruebas de Software	Abierto	SI	NO	SI	SI	SI	NO
ISO/IEC 29110	Procesos de todo el ciclo de vida del *software* de las VSEs	Abierto	NO	SI	SI	SI	NO	SI
TMMI	Mejora del proceso de pruebas	Propietario	SI	SI	SI	NO	NO	NO
TPI	Mejora del proceso de pruebas	Propietario	SI	SI	SI	SI	NO	NO
Propuesta para el testeo de PYMES	Mejora del proceso de pruebas	Abierto	SI	NO	SI	NO	NO	SI

Minimal Test Practice Framework	Mejora del proceso de pruebas	Propietario	SI	NO	SI	SI	NO	SI
ISO/IEC 29119 Parte 2	Proceso exclusivamente de Pruebas	Abierto	SI	NO	SI	SI	NOI	NO
Software Qualification Testing Process	Proceso exclusivamente de pruebas	Abierto	SI	SI	SI	SI	NO	NO
Proceso de pruebas de *Software* para el modelo de referencia de Competisoft	Proceso exclusivamente de pruebas	Propietario	SI	SI	SI	SI	NO	SI
TestPAI	Mejora del proceso de pruebas	Propietario	SI	SI	SI	SI	NO	SI
Proceso de pruebas para pequeñas empresas: En un escenario Brasileño	Proceso exclusivamente de pruebas	Propietario	SI	SI	SI	SI	NO	SI
Mi propuesta	Proceso exclusivamente de pruebas	Abierto	SI	SI	SI	SI	SI	SI

Se puede observar de la tabla anterior, que la literatura hace explícitos muchos modelos que tienen que ver en su gran mayoría con mejora de procesos o procesos que cubren todo el ciclo de vida de un producto *software*, dando un mínimo porcentaje a los procesos de pruebas aun cuando son conscientes de la importancia de las pruebas. Estos trabajos establecen como mejorar un proceso de pruebas pero no muestran explícitamente el proceso de pruebas a mejorar ni el proceso de pruebas obtenido a partir de la mejora realizada, a su vez los procesos que cubren todo el ciclo de vida describen de manera muy general las actividades a realizar durante las pruebas de *software*, estas pasan casi desapercibidas, pues no se invierte el tiempo necesario para crear tareas de prueba que garanticen la calidad de los productos desarrollados.

2.3 DISCUSIÓN

De acuerdo a [52] gran parte de las empresas que desarrollan *software* son pequeñas y no por eso desarrollan productos menos significativos que las grandes empresas, por tanto, para construir sus productos es necesario adaptar buenas prácticas de Ingeniería de *software* de acuerdo a su tamaño y tipo de negocio. Gran parte de los procesos descritos anteriormente, son robustos y complejos, y las VSEs que preparan la utilización de dichos procesos se encuentran con problemas como: la poca disponibilidad de recursos tanto de personal como de infraestructura, resistencia al cambio y la considerable inversión de dinero y esfuerzo [30] [53]. La gran mayoría de este tipo de empresas no están en capacidad de adquirir una solución sofisticada dentro de este campo, pues su aplicación resultaría costosa por la inversión que debería hacerse en tiempo, recursos y dinero [1]. Además, deben esperar largos plazos para observar los resultados y el retorno de la inversión [30]. Dada la imposibilidad de aplicar estos procesos, la manera que tienen las pequeñas empresas para respaldar la confianza de los desarrollos que realizan, son las pruebas empíricas ejecutadas por cada desarrollador, y en el mejor de los casos hacen uso de las pruebas funcionales con técnicas sencillas pero no rigurosamente aplicadas. Se hacen funcionar los productos *software* desarrollados, observando directamente su comportamiento y arreglándolos directamente cada vez que aparece una deficiencia una vez el sistema desarrollado ha sido terminado. Sin embargo, esta no es una buena opción, pueden quedar errores no descubiertos por los desarrolladores en partes de los productos *software* que no hayan sido ejecutadas o pueden encontrarse errores muy graves en la etapa de codificación que afecten significativamente el proyecto y que se deban a la falta de revisión de productos anteriores como el diseño o los requerimientos por ejemplo y la mejor opción no es devolverse, solo se puede usar el ingenio para solucionar el problema (apagar el incendio). Por esta razón de acuerdo a [32] el acompañamiento de las pruebas a medida que se va construyendo el producto *software* es fundamental, se deben aplicar convenientemente pruebas a la especificación de los requerimientos evaluando en estos atributos de calidad, así como también aplicar pruebas a los diseños, pero no olvidar que si se habla de la pequeña empresa, el proceso de pruebas debe tener un peso significativo sobre los procesos de construcción e integración que se lleven a cabo.

De acuerdo a la ISO/IEC 12207: i) El propósito del proceso de construcción del *software* es producir unidades ejecutables de *software* que reflejen adecuadamente el diseño del *software* y ii) el propósito del proceso de integración de *software* es combinar las unidades y componentes de *software* compatibles con el diseño de *software*, que demuestran que

los requisitos de *software* funcionales y no funcionales se cumplen en una plataforma operacional equivalente o completa.

Por tanto estos procesos tienen una importancia significativa en la pequeña empresa, son quienes finalmente entregan un producto terminado al cliente. Sobre el producto terminado recae la calidad de todos los procesos seguidos para su desarrollo. Finalmente la calidad del producto refleja la calidad del proceso [15].

Las soluciones muy simples se quedan cortas en la aplicación de pruebas que proporcionen la calidad que requieren los productos y proyectos de *software*. Normalmente estos pequeños modelos no le dan la importancia que merece el impacto que causan las actividades de pruebas que se van ejecutando a lo largo de los proyectos, dejan a un lado el seguimiento a estas actividades, a los planes de pruebas establecidos y a los planes de acción, por tanto no se dan la oportunidad de mejorar la eficacia y efectividad del proceso que se lleva a cabo.

2.4 APORTES DEL PROYECTO

En la Tabla 2, se puede observar la mayor contribución de este trabajo, la definición de un proceso de pruebas adecuado a las pequeñas organizaciones el cual incorpora técnicas de pruebas al proceso planteado. Se pretende identificar, analizar y relacionar un conjunto de técnicas de pruebas funcionales que apoyen la ejecución del proceso de pruebas definido, de tal manera que permita resolver los dos interrogantes siguientes: ¿Qué actividades de pruebas llevar a cabo en el contexto de la pequeña organización? y ¿Cómo ejecutar algunas actividades mediante las técnicas de pruebas funcionales apropiadas?, El objetivo es apoyar la calidad del proceso de construcción e integración de *software* y sus productos asociados.

Estas técnicas de pruebas funcionales dan el peso que se necesita en el proceso de pruebas sobre los procesos de construcción e integración del *software*. Cada técnica con debilidades y fortalezas ayudará a encontrar un tipo específico de defecto que impide el adecuado cumplimiento de los requisitos funcionales.

En la actualidad se han incrementado las técnicas a utilizar en las pruebas de un producto *software* con el fin de facilitar el aseguramiento de la calidad del mismo, pero sería aun más efectivo el uso de estas técnicas si se tuviera la certeza que se están usando de la mejor manera y en el momento adecuado. Con las técnicas de pruebas funcionales, se pretende satisfacer la funcionalidad especificada que es a menudo el criterio más importante para la aceptación del sistema de información [8]. La elección de la técnica más adecuada para la ejecución de una prueba en un contexto determinado no es una tarea trivial. Conscientes de esta situación, en el proyecto se propone vincular un conjunto de técnicas de pruebas funcionales apropiadas que apoyen la ejecución de algunas actividades del proceso definido, con el fin que este sirva de soporte en la creación de *software* de alta calidad. No se trata de que las pequeñas organizaciones tengan que invertir más tiempo o contratar nuevo personal, lo que se necesita es un proceso que defina ¿Qué hacer? y ¿Cómo hacerlo? mediante actividades, tareas, técnicas y responsabilidades bien definidas.

Los procesos y técnicas de pruebas de software no se aplican de forma correcta, sistemática e integrada en un porcentaje muy importante de los casos [4]. Empresas líderes en servicios de testing y calidad del software, como Software Quality Systems S.A., afirman que la implantación de técnicas y procesos de pruebas dan confianza a la hora de tomar decisiones en este campo. La calidad de productos software está influenciada por los procesos que lo desarrollan, el proceso de pruebas apoya de manera directa la calidad del producto, por ello no se debe descuidar realizar a conciencia procesos organizados de este tipo, los cuales se deben definir con base en: normas técnicas, mejoramiento de la calidad, sistema de métricas y experiencia obtenida en otros proyectos[4].

Capítulo 3

3. PROCESO DE PRUEBAS

Así como la calidad de un producto *software* depende de la calidad del proceso que lo desarrolla, las pruebas de *software* también dependen de la calidad del proceso de pruebas que se use para llevarlas a cabo. En este sentido, la definición de un proceso de pruebas con actividades y responsabilidades claras, que se desarrolle de manera paralela al proceso de desarrollo, es la solución a muchos de los problemas relacionados con el control de la calidad de los productos software.

La hipótesis de este trabajo es definir un proceso de pruebas que contenga las mínimas prácticas relativas a pruebas, que integre de una forma sencilla técnicas de pruebas funcionales. De este modo, será posible desarrollar de forma eficaz las actividades relativas a establecer un proceso de pruebas mientras se desarrollan actividades de mejora para otros procesos de la organización. El proceso de pruebas desarrollado permitirá que todas aquellas pequeñas empresas interesadas en adquirir pruebas de *software*, puedan alcanzar el objetivo marcado sin necesidad de entorpecer las formas de trabajo que se llevan a cabo o utilizar modelos con actividades que no le generan un valor agregado a sus procesos.

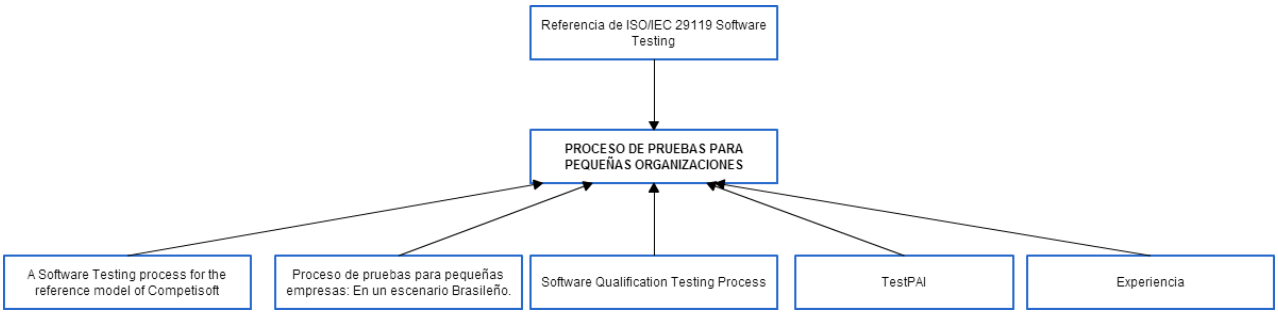
El proceso de pruebas definido en esta investigación es el resultado de: i) la clasificación y análisis de los procesos de pruebas existentes, ii) la identificación, análisis e integración de técnicas de pruebas funcionales y iii) la identificación, observación y análisis de las pruebas de acuerdo a los procesos y formas de trabajo para el desarrollo de software que se llevan a cabo en la pequeña empresa.

Por consiguiente, este capítulo se estructura de la siguiente manera: En la primera sección se analizan los modelos que sirvieron de referencia para definir el proceso de pruebas, a partir de los procesos, subprocesos, prácticas, actividades, objetivos, sub-practicas y tareas que los soportan. En la segunda sección se especifica y describe el proceso de pruebas definido que se compone de 6 fases descritas mediante un patrón de fases que sirve para la documentación de las mismas. Este patrón se compone de dos partes: Definición general de la Fase y Actividades y Tareas. En la tercera sección se describen y analizan las técnicas de pruebas que soportan el proceso de pruebas definido. En la cuarta sección se describen de manera general los niveles de pruebas funcionales en los que se aplican las técnicas de pruebas funcionales. En la quinta sección se presenta el modelado del proceso a través del cual se entienden fácilmente las actividades e información planteada en el proceso, pues se muestra de una forma organizada y sistemática, y finalmente se describen algunas herramientas de pruebas usadas en la aplicación del proceso, las cuales se proponen como un elemento de gran ayuda en la aplicación de algunas actividades de pruebas descritas en el proceso.

3.1 ANÁLISIS DE LOS PROCESOS DE REFERENCIA PARA EL PROCESO DE PRUEBAS

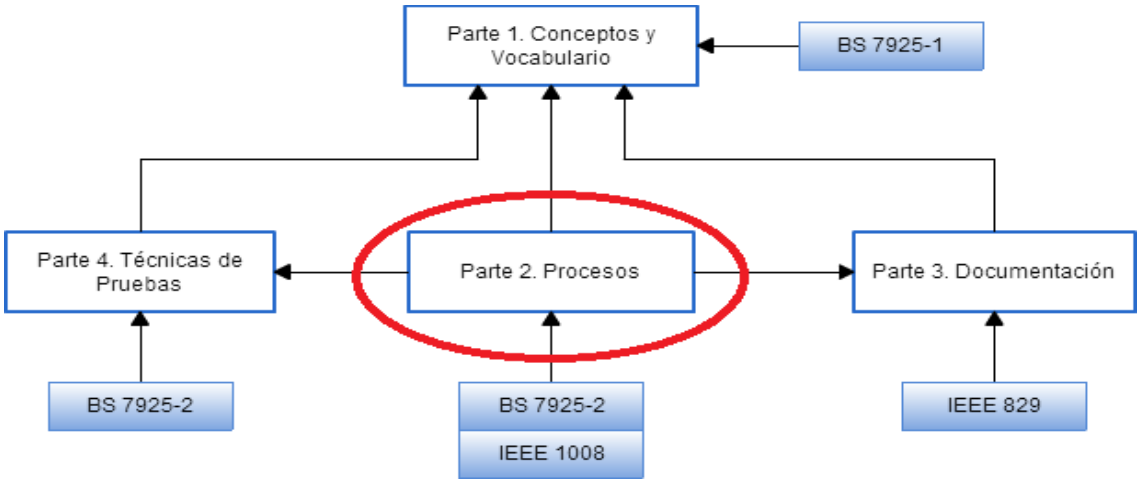
De acuerdo a [82] y [83] la implantación de grandes modelos de referencia como CMMI ha sido una tarea problemática en pequeñas empresas de *software*, siendo una de las principales causas las implicaciones económicas que esto conlleva, aunque en [80] y [81] se afirme que es este uno de los modelos de referencia más difundido dentro de la industria del *software*. En [84] se muestra que aunque algunas empresas pequeñas o medianas han intentado adoptar un modelo como este no han podido, pues son procesos muy completos que usan prácticas complejas a nivel de la pequeña empresa. Lo anterior es solo un ejemplo del problema de la aplicación de estándares y grandes modelos de referencia (como las normas ISO, CMMI, entre otros) a las pequeñas y medianas empresas. Con esto aunque se quiere mostrar que es difícil establecer un estándar o modelo de referencia en la pequeña empresa, no quiere decir que no existan procesos que se adecuen a sus formas de trabajo y puedan ser fácilmente adaptados. En esta situación, el proceso de pruebas definido en este trabajo de grado adecua sus prácticas pensando en la pequeña empresa usando como referencia la parte dos de la ISO/IEC 29119 (ISO/IEC 29119 Pruebas de Software. Parte 2 – Proceso de Pruebas), los procesos de pruebas encontrados en la bibliografía y la experiencia adquirida tras la aplicación del caso de estudio (ver Figura 2).

Figura 2. Fuentes para la definición del proceso de pruebas



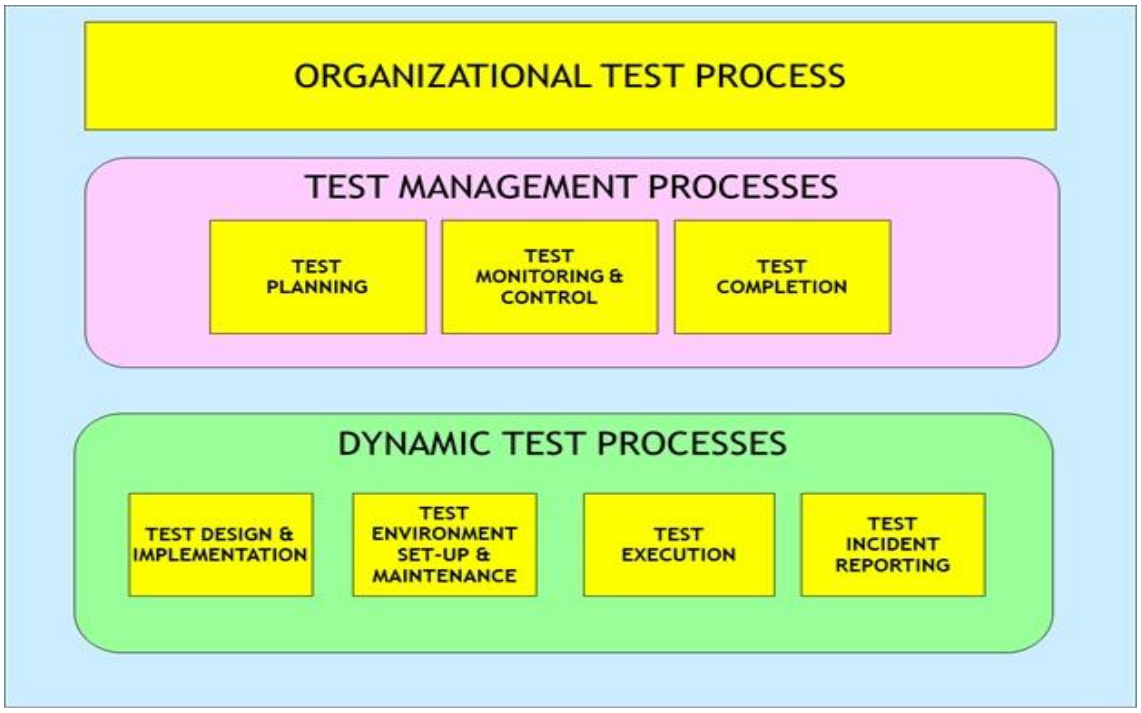
La primera fuente y mas importante es la segunda parte de la norma ISO/IEC 29119 *Software Testing* — Part 2: *Test process* [35], (ver Figura 3). Esta parte de la norma define un proceso de pruebas genérico que busca ser utilizado dentro de cualquier desarrollo de *software* y/o ciclo de vida.

Figura 3. ISO/IEC 29119 Pruebas de software - Parte 2: Proceso de pruebas



Este proceso en general está soportado por tres procesos como se muestra en la Figura 4:

Figura 4. Componentes del proceso de pruebas de la ISO/IEC 29119



Fuente: http://softwaretestingstandard.org/part2.php

Los tres procesos a su vez se dividen en otros procesos los cuales se soportan a través de actividades y tareas. A continuación se hace una descripción de cada uno de ellos:

1. **Proceso de pruebas a nivel organizacional**: En este proceso se establecen las especificaciones[6] de pruebas a nivel organizacional. Con estas especificaciones de pruebas se busca establecer a través de un documento de políticas, políticas de pruebas, el propósito, los objetivos, y el alcance global de las pruebas dentro de la organización. A través de las estrategias de pruebas se define el modo en el que deben llevarse a cabo las pruebas en la organización.

2. **Procesos de Gestión de pruebas:** Este proceso se compone de tres procesos:

   2.1 Proceso de Planeación de Pruebas: En este proceso se desarrolla, registra y comunica el alcance que se dará a las pruebas.
   2.2 Proceso de monitoreo y control de pruebas: El propósito de este proceso es asegurar que las pruebas se realizan de acuerdo al plan de pruebas, políticas de pruebas y estrategias de pruebas.
   2.3 Proceso de Pruebas Terminadas: El propósito de este proceso es asegurar que las pruebas reutilizables quedan disponibles para ser usadas en etapas posteriores, el entorno de pruebas se deja en un estado satisfactorio, que los resultados de las pruebas se registran y comunican a los interesados.

3. **Procesos de pruebas Dinámicas**: Este proceso se compone de cuatro procesos:

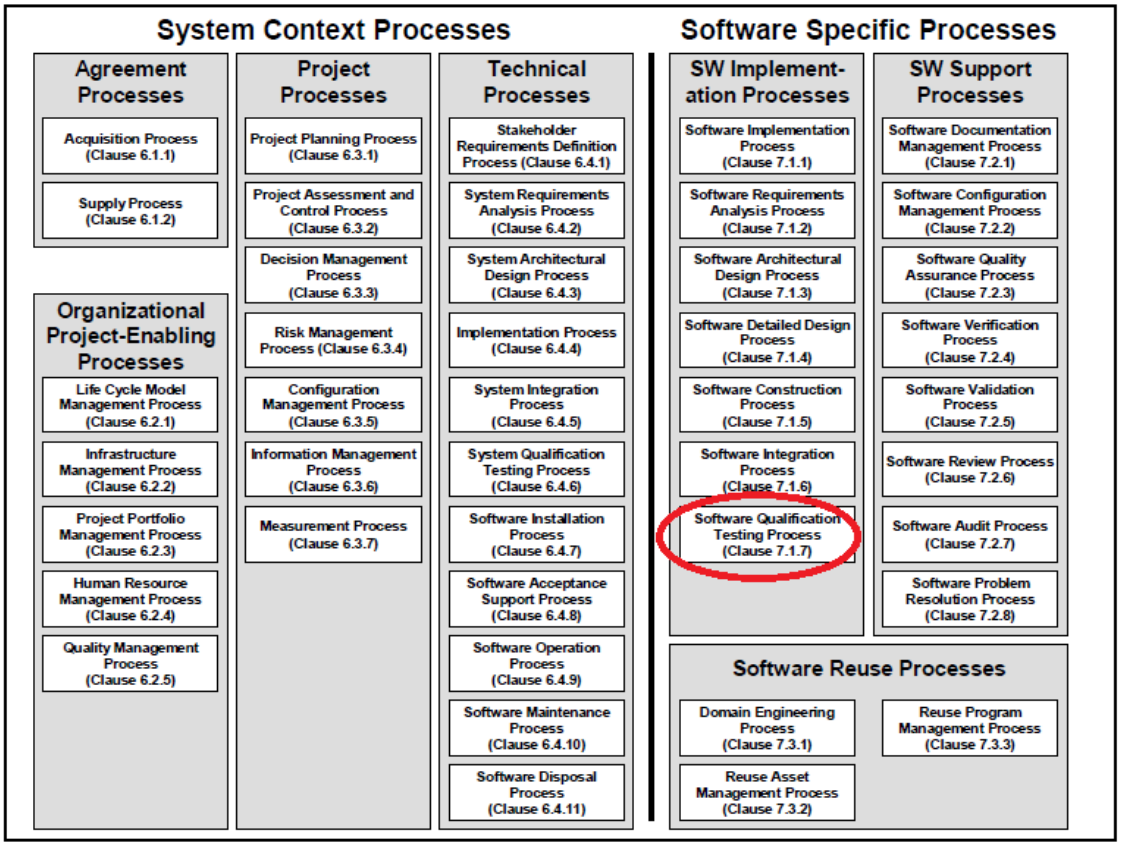
   3.1 Proceso de Diseño e implementación de pruebas: En este proceso se derivan los procedimientos de pruebas que serán ejecutados durante el proceso de ejecución de pruebas.
   3.2 Proceso de configuración del entorno y mantenimiento de pruebas: Este proceso se encarga de establecer y mantener los requisitos del entorno de pruebas.
   3.3 Proceso de ejecución de pruebas: En este proceso se ejecutan los procedimientos de pruebas en el entorno configurado en el proceso anterior, y se registran los resultados.
   3.4 Proceso de reporte de incidentes: Mediante este proceso se da a conocer medidas adicionales identificadas como resultado de la ejecución de los procedimientos de pruebas.

En el Anexo N se ilustran en detalle las Actividades que hacen parte de estos procesos.

La segunda fuente es el *Software Qualification Testing Process* [20] (ver Figura 5). Este proceso hace parte de los procesos de implementación de software incluidos en los procesos específicos de software que establece la ISO/IEC 12207 Systems and software engineering - Software life cycle processes.

[6] Especificaciones de pruebas en la norma ISO/IEC 29119 hace referencia a las políticas y estrategias de pruebas a nivel organizacional.

Figura 5. Software Qualification Testing Process



Fuente: ISO/IEC 12207 International Standard, Systems and *software* engineering - *Software* life cycle processes.

El objetivo de este proceso es confirmar que el producto software integrado cumple con sus requisitos definidos. Como resultado de la implementación de este proceso:

- Se definen los criterios que demuestren que el software integrado cumple con los requisitos del software;
- Se verifica el software integrado usando los criterios definidos;
- Se registran los resultados de pruebas, y
- Se desarrolla una estrategia de regresión para volver a probar el software integrado cuando se realice un cambio en los elementos de software.

Este proceso se compone de una Actividad y 5 tareas:

1. Probar cada elemento software. Se debe probar cada elemento software o elemento de configuración. Esta actividad consta de las siguientes tareas:
   a) El ejecutor deberá llevar a cabo las pruebas evaluando cada requisito del elemento software. Los resultados de las pruebas deben documentarse.
   b) El ejecutor deberá actualizar la documentación para el usuario si es necesario.

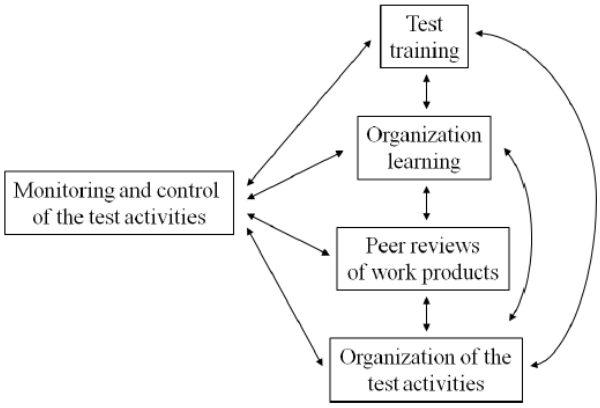
c) El ejecutor deberá evaluar el diseño, código, pruebas, resultados de pruebas, y documentación de usuario, teniendo en cuenta: La cobertura, la conformidad con los resultados esperados, la viabilidad de la integración del sistema, y la viabilidad de la operación y el mantenimiento.
d) El ejecutor deberá soportar auditoría (s), y los resultados de las auditorias deben documentarse.
e) Al completar con éxito las auditorías, el ejecutor deberá actualizar y preparar el producto software entregable, probar el sistema, instalar el software o dar soporte al software en su caso.

La tercera fuente es "*A Software Testing process for the reference model of Competisoft*" [23]. En la Figura 6 se puede observar dentro de los procesos definidos en Competisoft, la ubicación del proceso de pruebas propuesto.

Figura 6. Ubicación del Proceso de pruebas desarrollado para Competisoft dentro de sus procesos

El enfoque propuesto por [23] incluye un modelo de proceso de pruebas basado en el aprendizaje de la organización en lo referente a pruebas. Este proceso se compone de objetivos y actividades distribuidos en 5 subprocesos, como se muestra en la Figura 7.

Figura 7. Subprocesos del proceso de pruebas de software para el modelo de referencia Competisoft



La cuarta fuente es TestPAI [36], un proceso de pruebas soportado a través de prácticas, objetivos y sub-practicas. Por último, la quinta fuente es el "Proceso de pruebas para pequeñas empresas: En un escenario Brasileño" [1].

En el capítulo dos se analizaron los procesos citados anteriormente tomando como referente ocho criterios definidos: Tipo, Licencia, Exclusivo de pruebas, Completamente definido, Actividades, Tareas, Incorpora técnicas de pruebas, Adecuado para VSEs. Lo anterior con el objetivo de comparar y evidenciar de forma clara la diferencia con el proceso de pruebas planteado en el trabajo que se ha realizado en este proyecto de grado. Deduciendo de esto la inexistencia de procesos de pruebas en el contexto de la pequeña empresa que involucren técnicas de pruebas funcionales a sus actividades.

En la tabla que se presenta a continuación se han clasificado en niveles de abstracción los procesos, subprocesos, prácticas, actividades, objetivos, sub-practicas y tareas que soportan los procesos de pruebas, con el objetivo establecer un paralelo que indique las actividades comunes a la pequeña empresa y que deban tenerse en cuenta para la definición del proceso de pruebas planteado en este trabajo de grado.

Tabla 3. Niveles de abstracción de los procesos de pruebas

Proceso	Nivel 1	Nivel 2	Nivel 3	Nivel 4
ISO/IEC 29119	Procesos	Sub-procesos	Actividades	Tareas
Un proceso de pruebas de software para el modelo de referencia Competisoft		Subprocesos	Objetivos	Actividades
TestPAI		Practicas	Objetivos	Sub-practicas
Proceso de pruebas para pequeñas empresas: En un escenario Brasileño.		Actividades	Tareas	

Los procesos de pruebas exitosos en pequeñas organizaciones se comparan nivel a nivel a partir del nivel 2, ya que los procesos pensados para pequeñas organizaciones no llegan hasta un nivel 1 de abstracción. Estas comparaciones se hacen utilizando como referente la norma ISO/IEC 29119 Pruebas de software – parte 2. Proceso de pruebas. Se usa la norma como referente por ser esta utilizable dentro de cualquier ciclo de vida de desarrollo de software.

En el Anexo O se especifica en detalle las comparaciones de Nivel 2, de las cuales se obtiene un resultado como el que se muestra en la Tabla 4.

Tabla 4. Resultados de comparaciones de nivel 2

Nivel de abstracción 2	Proceso a Nivel Organizacional	Proceso de Planificación de pruebas	Proceso de Monitoreo y control de pruebas	Proceso de finalización de pruebas	Proceso de Diseño e implementación de Pruebas	Proceso de Configuración y mantenimiento del entorno de pruebas	Proceso de ejecución de pruebas	Proceso de Notificación de Incidentes
Numero de procesos que tienen este subproceso	**1**	**3**	**1**	**1**	**1**	**0**	**3**	**0**
Total	**2**	**4**	**2**	**2**	**2**	**1**	**4**	**1**

De acuerdo a la tabla anterior donde se toma como referente los sub-procesos del proceso de pruebas definido en ISO/IEC 29119 Pruebas de Software – Parte 2 comparado con los sub-procesos del proceso de pruebas de software para el modelo de referencia Competisoft, Prácticas de TestPAI y Actividades del Proceso de pruebas para pequeñas empresas: En un escenario Brasileño, se concluye que un proceso de pruebas debe tener como mínimo las etapas de Planificación y Ejecución de las pruebas.

En el Anexo P se especifica en detalle las comparaciones de Nivel 3. De estas comparaciones, se obtiene un resultado como el que se muestra en la Tabla 5.

Tabla 5. Resultados de comparaciones de nivel 3

Nivel de abstracción 3	5.2.3.1 Desarrollar especificaciones de prueba Organizacional (OT1)	5.2.3.2 Monitoreo y control del uso de las especificaciones de pruebas	5.2.3.3 Actualización de las especificaciones de pruebas (OT3)	6.2.3.1 Entender el Contexto (TP1)	6.2.3.2 Organizar el Desarrollo del plan de pruebas (TP2)	6.2.3.3 Identificar y analizar los riesgos (TP3)	6.2.3.4 Identificar el enfoques para el tratamiento de los riesgo (TP4)	6.2.3.5 Diseño de las Estrategias para prueba s(TP5)	6.2.3.6 Determinación del Personal y Programación (TP6)	6.2.3.7 Registro del Plan de Pruebas (TP7)	6.2.3.8 Hacer un consenso sobre el Plan de pruebas (TP8)	6.2.3.9 Comunicar y Poner a disposición el Plan de Pruebas (TP9)	6.3.3.1 Configuración (TMC1)	6.3.3.2 Monitoreo (TMC2)	6.3.3.3 Control (TMC3)	6.3.3.4 Informe (TMC4)	6.4.3.1 Archivar las pruebas Activas (TC1)	6.4.3.2 Limpiar el entorno de prueba s (TC2)	6.4.3.3 Identificar las Lecciones Aprendidas (TC3)	6.4.3.4 Informe de Terminación de las pruebas (TC4)	7.2.3.1 Identificación del conjunto de características (TD1)	7.2.3.2 Deducir las condiciones de prueba (TD2)	7.2.3.3 Deducir la cobertura de las prueba (TD3)	7.2.3.4 Derivar los casos de prueba (TD4)	7.2.3.5 Ensamblar los conjuntos de prueba (TD5)	7.2.3.6 Derivar los Procedimientos de prueba (TD6)	7.3.3.1 Establecer el entorno de prueba (ES1)	7.3.3.2 Mantenimiento del Entorno de prueba (ES2)	7.4.3.1 Ejecutar los Procedimiento s de prueba (s) (TE1)	7.4.3.2 Comparar los resultados de las prueba (TE2)	7.4.3.3 Registro de ejecución de pruebas (TE3)	7.5.3.1 Analizar el resultado de las pruebas (s) (IR1)	7.5.3.2 Crear / Actualizar los Reportes de Incidentes (IR2)
Numero de procesos que tienen este subproceso	1			3		1		2	2	2				**2**	2	2								2	1		2		2		1	2	
Total	2			4		2		3	3	3				**3**	3	3								3	2		3		3		2	3	

De acuerdo a la tabla anterior tomando como referente las actividades del proceso de pruebas definido en ISO/IEC 29119 Pruebas de Software – Parte 2 comparado con los objetivos del proceso de pruebas de software para el modelo de referencia Competisoft, objetivos de TestPAI y Tareas del Proceso de pruebas para pequeñas empresas: En un escenario Brasileño, se concluye que un proceso de pruebas debe tener como mínimo las siguientes actividades:

- Entender el Contexto
- Diseño de las Estrategias para pruebas
- Determinación del Personal y Programación
- Registro del Plan de Pruebas
- Monitoreo
- Control
- Informe
- Derivar los casos de prueba
- Establecer el entorno de prueba
- Ejecutar los Procedimiento s de prueba (s)
- Analizar el resultado de las pruebas (s)

En el Anexo Q se especifica en detalle las comparaciones de Nivel 4. De estas comparaciones, se obtiene un resultado como el que se muestra en la Tabla 6. Resultados de comparaciones de nivel 4

Tabla 7. Resultados de comparaciones de nivel 4

Proceso de Referencia: ISO/IEC 29119-2 Procesos de pruebas para pequeñas empresas	(OT1) (b) Crear el borrador de las especificaciones de pruebas	(TP1) Comprensión del contexto en el cual se identifican los elementos que se pondrán a prueba	(TP4) (a) (c)Identificar los medios adecuados para tratar los riesgos y monitorizar los riesgos relacionados con el producto y proyecto	(TP5) (a) (c) Diseñarse una estrategia de pruebas teniendo en cuenta las características a ser probadas, técnicas y tipos de pruebas, análisis de riesgos, limitaciones de tiempo, recursos, herramientas , estimaciones de tiempo y esfuerzo, indicadores y esta debe ser aprobada por los interesados.	(TP6) (a) Identificar los roles y habilidades del personal que llevara a cabo las pruebas y monitorizar su grado de participación. Esto puede requerir necesidades de formación	(TP2) (a) Debe planificarse y programarse las actividades de pruebas	(TP7) (b) Deben registrarse en un plan de pruebas las actividades descritas en la estrategia y receptores de errores	(TD1) (a) Identificación de conjuntos de características	(TD4) (a)(c) Derivar l y registrar os casos de prueba	(TD6) (a) (d) Derivar y registrar los procedimientos de pruebas	(ES1) (a) Preparar el entorno de pruebas y los datos necesarios para ejecutar las pruebas	(TE1) (a) ejecutar los procedimientos de pruebas en el entorno preparado para tal fin	(IR2) (a) Reportar los incidentes encontrados a los interesados	(TMC4) (a) Debe crearse y comunicarse un informe sobre el progreso de las pruebas en informes finales	(TC3) (a) (b) Identificar y registrar las lecciones aprendidas las cuales incluyen el análisis del proceso	(TMC3) © analizar los resultados de las pruebas , Identificar las acciones necesarias para controlar la divergencia entre los valores esperados y obtenidos en pruebas y actuar contra las fuentes de desviación
Tarea cumplida por proceso	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Tarea cumplida por proceso		X		X	X		X				X	X	X			X
Tarea cumplida por proceso			X	X	X		X						X	X	X	X
Total	1	2	2	3	3	1	3	1	1	1	2	2	3	2	2	3

Las actividades a considerar en el proceso de pruebas derivadas del análisis hecho se muestran a continuación:

- (OT1) (b) Crear el borrador de las especificaciones de pruebas
- (TP1) Comprensión del contexto en el cual se identifican los elementos que se pondrán a prueba
- (TP4) (a) (c) Identificar los medios adecuados para tratar los riesgos y monitorizar los riesgos relacionados con el producto y proyecto
- (TP5) (a) (c) Diseñarse una estrategia de pruebas teniendo en cuenta las características a ser probadas, técnicas y tipos de pruebas, análisis de riesgos, limitaciones de tiempo, recursos, herramientas , estimaciones de tiempo y esfuerzo, indicadores y esta debe ser aprobada por los interesados.
- (TP6) (a) Identificar los roles y habilidades del personal que llevara a cabo las pruebas y monitorizar su grado de participación. Esto puede requerir necesidades de formación
- (TP2) (a) Debe planificarse y programarse las actividades de pruebas

- (TP7) (b) Deben registrarse en un plan de pruebas las actividades descritas en la estrategia y receptores de errores
- (TD1) (a) Identificación de conjuntos de características
- (TD4) (a)(c) Derivar l y registrar os casos de prueba
- (ES1) (a) Preparar el entorno de pruebas y los datos necesarios para ejecutar las pruebas
- (TE1) (a) ejecutar los procedimientos de pruebas en el entorno preparado para tal fin
- (IR2) (a) Reportar los incidentes encontrados a los interesados
- (TMC4) (a) Debe crearse y comunicarse un informe sobre el progreso de las pruebas e informes finales
- (TC3) (a) (b) Identificar y registrar las lecciones aprendidas las cuales incluyen el análisis del proceso
- (TMC3) (c) analizar los resultados de las pruebas, Identificar las acciones necesarias para controlar la divergencia entre los valores esperados y obtenidos en pruebas y actuar contra las fuentes de desviación

El proceso de pruebas definido en la ISO/IEC 29119 Pruebas de Software, recoge gran parte de las actividades descritas en los procesos, como se muestra en laTabla 6, con la diferencia de que los procesos de pruebas para pequeñas empresas tienden a enfocar sus actividades hacia el aprendizaje de la organización en pruebas de software.

El proceso de pruebas planteado en este trabajo de grado tiene implícito el objetivo que quieren lograr los procesos descritos anteriormente. Aunque no se describe de manera puntual una actividad que fomente el aprendizaje de la organización, a través del conocimiento que se adquiere al involucrar técnicas puntuales en el proceso de pruebas y utilizarlas en el diario vivir de las pruebas de software, se va adquiriendo además de aprendizaje cierta perspicacia al empezar a intuir los errores que puede arrojar el software, acercándolo a técnicas de mayor nivel como lo es la técnica de “Adivinar errores” convirtiendo de acuerdo a la experiencia a los responsables de las pruebas en personas que "huelen" los defectos.

Las Actividades derivadas a partir del análisis de la Tabla 6, no se describen específicamente como las define la norma ISO/IEC 29119, se hace un hibrido con el fin de indicar en un solo párrafo los ítems comunes, como resultado de la comparación. De ahí, se agrupan las actividades en fases de acuerdo a unas características comunes y como resultado se obtiene el proceso mostrado a continuación:

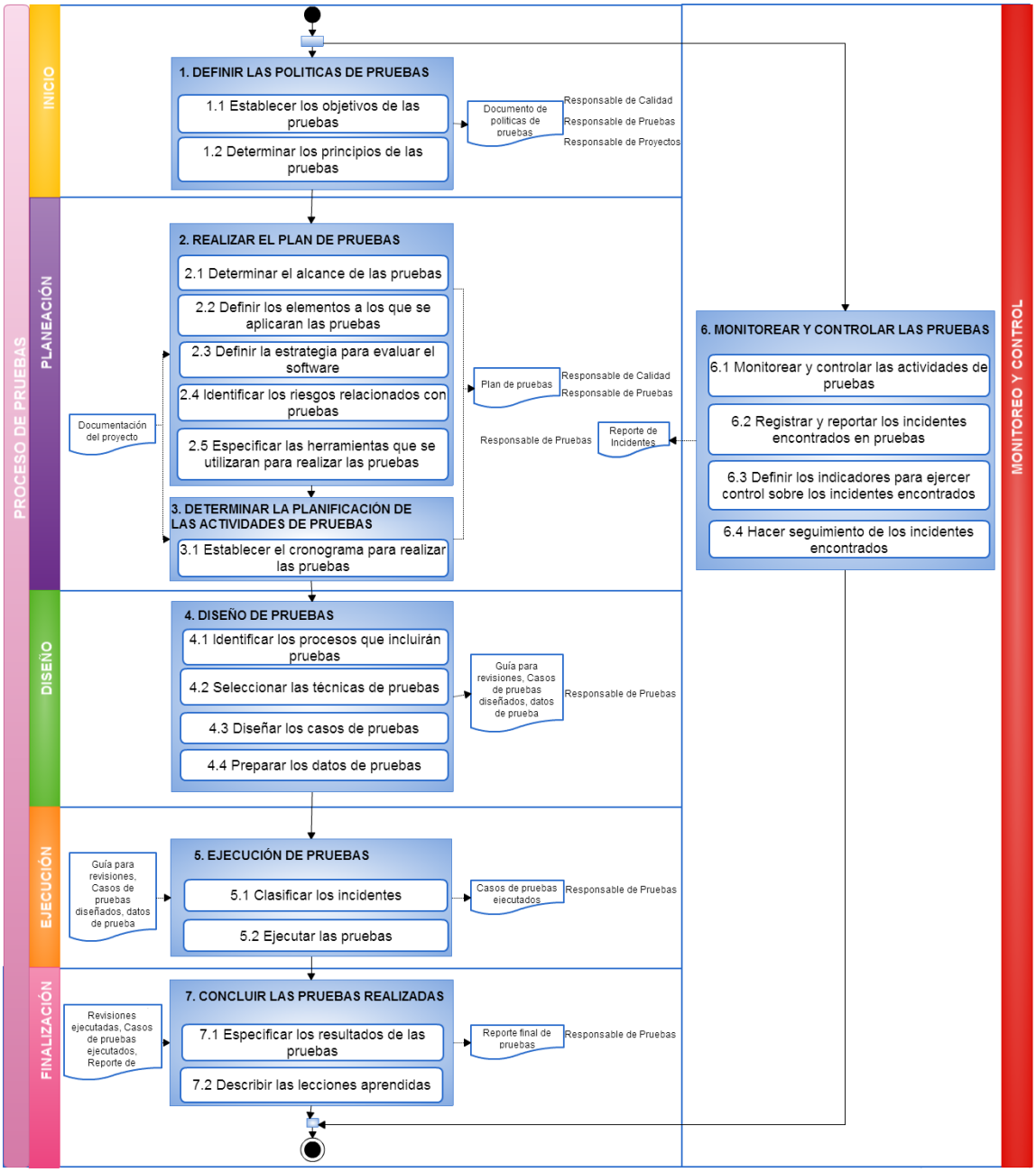
Tabla 8. Actividades Básicas de un proceso de pruebas

	Actividades Básicas
Inicio	• (OT1) (b) Crear el borrador de las especificaciones de pruebas • (TP1) Comprensión del contexto en el cual se identifican los elementos que se pondrán a prueba
Planeación	• (TP4) (a) (c) Identificar los medios adecuados para tratar los riesgos y monitorizar los riesgos relacionados con el producto y proyecto

	• (TP5) (a) (c) Diseñarse una estrategia de pruebas teniendo en cuenta las características a ser probadas, técnicas y tipos de pruebas, análisis de riesgos, limitaciones de tiempo, recursos, herramientas , estimaciones de tiempo y esfuerzo, indicadores y esta debe ser aprobada por los interesados. • (TP6) (a) Identificar los roles y habilidades del personal que llevara a cabo las pruebas y monitorizar su grado de participación. Esto puede requerir necesidades de formación • (TP2) (a) Debe planificarse y programarse las actividades de pruebas • (TP7) (b) Deben registrarse en un plan de pruebas las actividades descritas en la estrategia y receptores de errores • (TD1) (a) Identificación de conjuntos de características
Monitoreo y Control	• (TMC4) (a) Debe crearse y comunicarse un informe sobre el progreso de las pruebas e informes finales • (TC3) (a) (b) Identificar y registrar las lecciones aprendidas las cuales incluyen el análisis del proceso • (TMC3) (c) analizar los resultados de las pruebas, Identificar las acciones necesarias para controlar la divergencia entre los valores esperados y obtenidos en pruebas y actuar contra las fuentes de desviación
Diseño	• (TD4) (a)(c) Derivar l y registrar os casos de prueba
Ejecución	• (ES1) (a) Preparar el entorno de pruebas y los datos necesarios para ejecutar las pruebas • (TE1) (a) ejecutar los procedimientos de pruebas en el entorno preparado para tal fin • (IR2) (a) Reportar los incidentes encontrados a los interesados
Finalización de las pruebas	• Informes Finales

A partir de ahí se plantea inicialmente el siguiente proceso:

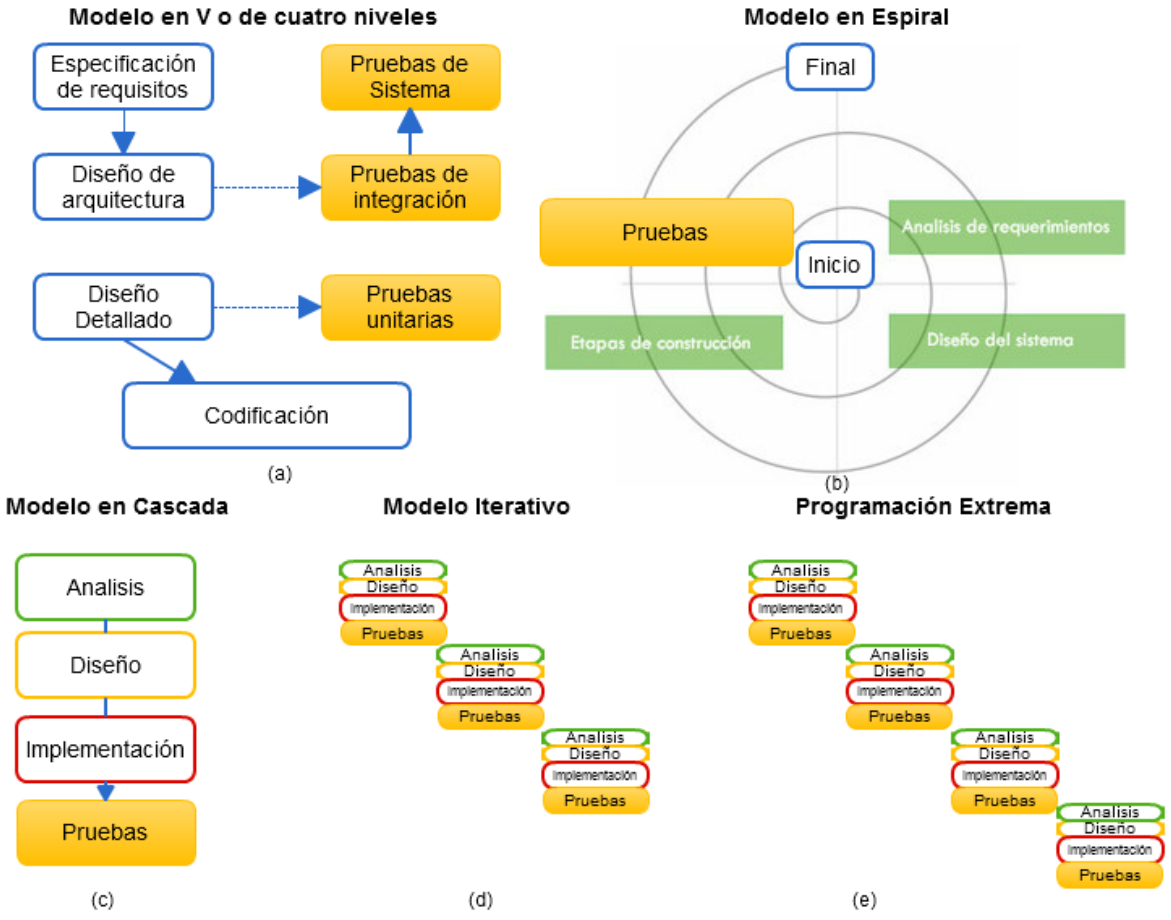
Figura 8. Proceso Inicial de Pruebas



El proceso descrito en la Figura 8, se aplicó en la segunda iteración de pruebas indicada en el caso de estudio. Tras la identificación de las fuentes de desviación a partir del análisis de los informes de hallazgos encontrados en un primer ciclo de ejecución, se vio la necesidad de enfocar el proceso de pruebas hacia un proceso que realmente acompañara el ciclo de vida del proceso de desarrollo, soportando tanto revisiones

formales[7] como pruebas funcionales. Para esto, fue necesario analizar las pruebas desde diferentes frentes de acuerdo a algunas metodologías de desarrollo seguidas por las organizaciones que construyen *software*, buscando la forma de acompañar el proceso de desarrollo no solo a partir de las pruebas funcionales como se muestra en la Tabla 8 si no también de las pruebas estáticas o revisiones formales. Los modelos analizados fueron: modelo en cuatro niveles, modelo en espiral, modelo en cascada, modelo iterativo y programación extrema.

Figura 9. Metodologías de desarrollo vs Pruebas de *Software*



En la Figura 9, se puede observar cómo varía el lugar que ocupan las pruebas en cada metodología, por ejemplo:

[7]De acuerdo a [32] las revisiones formales también conocidas como inspecciones de software "son un método de análisis estático para verificar y validar un producto software manualmente"

I. En el modelo de ciclo de vida en V (a) para cada fase de desarrollo, existe una fase correspondiente o paralela de pruebas.
II. En el modelo en espiral (b) se planifica una funcionalidad, se implanta y se prueba.
III. En el modelo en cascada (c) el inicio de las pruebas solo sucede cuando se ha terminado la codificación, donde los elementos ya programados, se ensamblan para componer el sistema y se comprueba que funciona correctamente antes de pasar a la siguiente fase.
IV. El modelo iterativo (d) es una derivación del ciclo de vida en cascada, que busca reducir el riesgo que surge entre las necesidades del usuario y el producto final, donde la funcionalidad planificada para el sistema se alcanza de forma gradual en varios ciclos de desarrollo y prueba.
V. En programación extrema (e) en lugar de planificar, analizar y diseñar a medio o largo plazo, se hacen todas estas actividades simultáneamente a lo largo de todo el desarrollo.

De estas metodologías y el tratamiento que dan a la etapa de pruebas se concluye que:

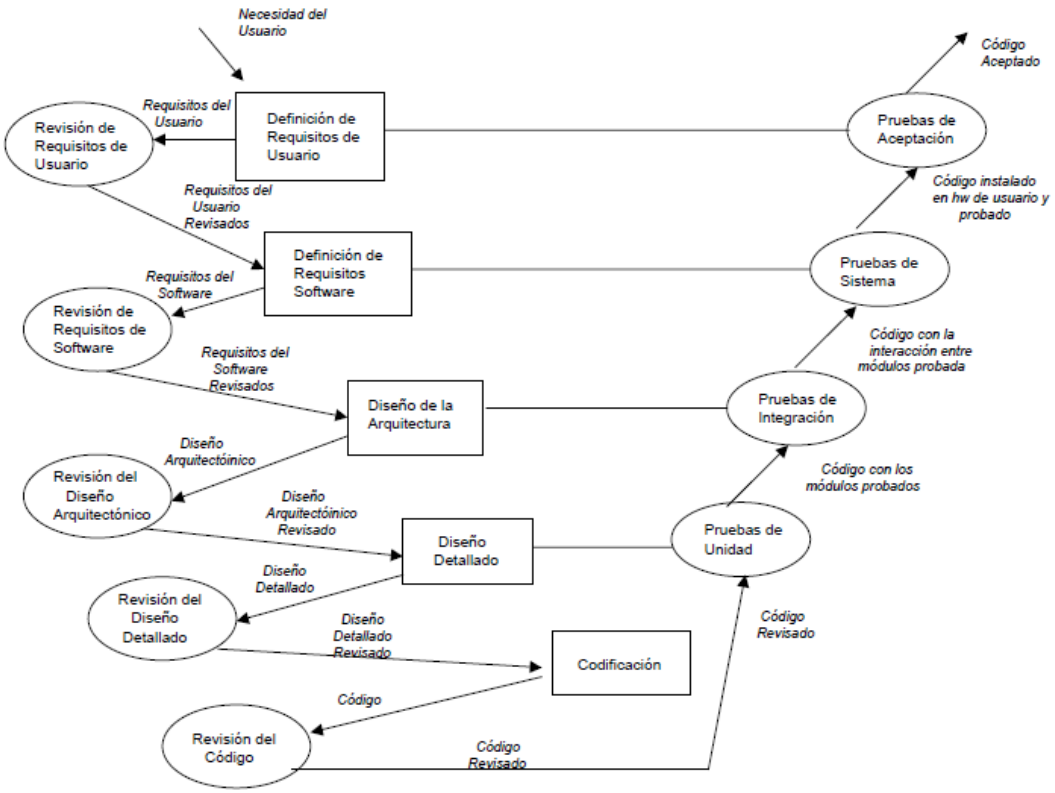
- El modelo en V es igual de estricto al modelo en cascada tradicional, las pruebas solo empiezan cuando se ha terminado el desarrollo, la diferencia está en que en este modelo son identificables los tipos de pruebas que se deben aplicar según sea la fase de desarrollo en que se encuentre. De acuerdo a [63] esto “es útil de cara a la planificación de los proyectos de *software*, pero no muestra el proceso incremental del desarrollo” y finalmente termina con los mismos problemas del modelo en cascada.

- Las pruebas al final del desarrollo como lo muestra el modelo en cascada no son las más adecuadas dado que cualquier error de una fase inicial encontrado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado (re-trabajo), aumentando (dependiendo del fallo encontrado) los costos del desarrollo.

- La fase de pruebas no puede considerarse solo hasta el final del desarrollo. Unos requisitos ambiguos son difíciles de entender (cuando no es desarrollado por quien obtiene el requisito) y difíciles de probar en el mismo caso. Por tanto una organización en crecimiento no puede acogerse a estas formas de trabajo. Lo mismo pasa con una arquitectura no modular, o un código mal documentado.

En [32] se muestra un modelo en V desde la perspectiva de las revisiones, donde se hace un paralelo entre cada artefacto del ciclo de vida del desarrollo de un producto software y su respectiva revisión (ver Figura 10).

Figura 10. Modelo en V de evaluación de Software



Fuente: [32] Natalia Juristo, A. M. M., Sira Vegas (2006). "TÉCNICAS DE EVALUACIÓN DE SOFTWARE."

Esta fuente de información ayuda a ampliar la visión del proceso de pruebas y a partir de esto definir un proceso con un mayor alcance en cada una de sus actividades de acuerdo a las necesidades de las pequeñas organizaciones.

A lo largo de este trabajo de investigación se ha hecho énfasis en que el costo de corregir un error en las últimas fases del desarrollo es mucho más alto que hacerlo en las fases iniciales, mas sin embargo, y como se muestra en la Figura 9, la mayoría de las metodologías de la ingeniería de *software* tradicional no tienen en cuenta esto, por lo que el concepto de diseñar y ejecutar de manera eficiente las pruebas en todo el ciclo de vida del desarrollo, parece no ser atendido por todos. De acuerdo a [72] la calidad del *software* no puede recaer sobre las pruebas hechas al final de la construcción del mismo, la calidad la generan todos los procesos de la ingeniería del *software*, “las pruebas solo sirven para confirmar y comprobar si el *software* es de calidad”. De ahí la necesidad de desarrollar el tipo de investigación planteada en esta tesis de pregrado, donde se busca y selecciona las mejores actividades de pruebas que sirvan de estrategia aplicable en las pequeñas empresas de desarrollo de *software* actual.

Con el fundamento anterior, las pruebas deben dejar de verse como una fase aislada dentro del ciclo de vida, y se debe contemplar como un proceso paralelo al proceso de desarrollo con actividades claras para llevar a cabo revisiones y pruebas, aplicadas en varios momentos del desarrollo y no sólo al final y únicamente sobre el código. Aunque es claro que el proceso de desarrollo y el proceso de pruebas son dos cosas distintas existe una correspondencia que de alguna manera los une y los invita a trabajar juntos por un mismo propósito, lograr *software* de excelente calidad [69] [68]. Se debería por tanto diseñar casos de prueba para cada una de las fases del ciclo de vida, es decir, debería tenerse casos de prueba desde los requisitos hasta el desarrollo, sustentados en el hecho de que el número de casos de prueba se va incrementando en la medida en que se avanza a lo largo de las fases hasta llegar a la entrega del producto, con lo que se asegura que si el cliente detecta anomalías, estas serán mínimas [68].

En el proceso de pruebas planteado en este trabajo de grado, se propone hacer un acompañamiento al proceso de desarrollo (especialmente en pequeñas organizaciones), con el fin de obtener productos y procesos de excelente calidad, detectando el mayor número de errores en el *software*; para ello, acoge las actividades de pruebas apropiadas a la pequeña empresa de acuerdo a la Tabla 8 y se amplía el alcance de dichas actividades, de tal forma que no solo haga énfasis en las actividades desde el punto de vista de pruebas funcionales si no también apropie las revisiones más específicamente las revisiones formales para que de esto se genere un informe que refleje el acto de la revisión.

En el numeral 3.3 del presente documento, se exponen las técnicas de pruebas funcionales, las cuales al integrarse y ejecutarse de acuerdo a las actividades y tareas planteadas en el proceso definido hacen de este un proceso de fácil uso y aplicación, sobre todo en el contexto de las pequeñas organizaciones desarrolladoras de *software*.

3.2 DESCRIPCIÓN DEL PROCESO DE PRUEBAS

Cualquiera que sea el proceso utilizado y aplicado al desarrollo del *software*, y casi independientemente de él, siempre se aplica un modelo de ciclo de vida. Como se mostro anteriormente, es normal que para estas metodologías las pruebas se consideren parte del ciclo de vida del desarrollo de cualquier aplicación (requerimientos, análisis y diseño, implementación, pruebas y despliegue); pero, se debe tener en cuenta que a su vez, las pruebas tienen su propio ciclo de vida y, dependiendo de la organización, este ciclo puede tener más o menos fases.

En este trabajo de grado se propone un proceso de pruebas soportado en 6 fases: i) Inicio, ii) Planeación, iii) Diseño, iv) Ejecución, v) Monitoreo y control las pruebas y vi) Finalización. Este proceso no hace parte del ciclo de vida del software como una fase más, en lugar de esto se propone el acompañamiento a este ciclo con un proceso que soporta tanto revisiones formales como pruebas funcionales, mediante las cuales se puede evaluar el producto no solo después del desarrollo sino también en etapas previas.

Cada fase contiene actividades, tareas, productos de trabajo y roles. Una actividad es la suma de tareas específicas, que normalmente se aplican en una secuencia ordenada hasta cierto punto (Las actividades del proceso se muestran como secuenciales, pero en la práctica algunas actividades deben visitarse nuevamente), y que se deben realizar para elaborar los productos de trabajo del proceso. Los productos de trabajo establecidos incluyen informes y documentación generados por el proceso de pruebas. Los roles encargados de realizar las actividades de pruebas descritas por el proceso propuesto, deben desempeñarlos un equipo independiente al que creó el *software*, ya que al hacer esta tarea los mismos desarrolladores, inconscientemente tratarán de demostrar que el *software* funciona y no que no lo hace, por lo que la prueba puede tener menos éxito.

El proceso de pruebas se describirá siguiendo los siguientes lineamientos tomados en cuenta de [85]:

Definición general del proceso

Propósito	**Objetivos generales medibles y resultados esperados de la implantación efectiva del proceso.**
Objetivos	Objetivos específicos cuya finalidad es asegurar el cumplimiento del propósito del proceso.
Descripción	Descripción general de las actividades y productos que componen el flujo de trabajo del proceso.
Productos de Trabajo	Productos generados durante el flujo de trabajo del proceso de pruebas. En algunas tareas se especifica particularmente las entradas y las salidas de cada una de ellas, no quiere decir eso que una salida necesariamente sea un producto de trabajo del proceso de pruebas.
Responsabilidad y autoridad	Responsabilidad es el rol principal del responsable de la ejecución del proceso. Autoridad es el rol responsable por validar la ejecución del proceso y el cumplimiento de su propósito.
Diagrama de flujo de trabajo	Diagrama de actividades, donde se especifican las actividades del flujo de trabajo, las tareas, los roles, y las salidas de las actividades.

Patrón para las fases

El patrón de fases es un esquema de elementos que servirá para la documentación de las fases del proceso. Está constituido por dos partes: Definición general de la Fase y Actividades.

En la definición general de la fase se identifica su nombre y acrónimo, propósito, descripción general de sus actividades, roles, entradas y salidas.

En las actividades y tareas se identifican los roles, se describen las actividades en detalle (Tareas), entradas y salidas.

- **Definición general de la Fase**

Fase	**Nombre de la fase**		Abreviatura	**Abreviatura de la técnica**
Descripción	Descripción de la fase.			
Propósito	Objetivos generales y resultados esperados de la implantación efectiva de la fase.			
Descripción de las Actividades	Descripción general de las actividades y productos que componen el flujo de trabajo de la fase.			
Roles involucrados	**Abreviatura**	**Nombre**		
	Abreviatura del rol	Nombre del rol		
Entradas	Nombre del producto o recurso			
Salidas	Nombre del producto o recurso			

- **Actividades y Tareas**

Rol	Descripción
(Abreviatura de la Fase)A1. Nombre de la Actividad	
Entradas	**(opcional)**
	(Abreviatura de la Fase)A1.1 Descripción de la tarea 1.
	(Abreviatura de la Fase)A1.1 Descripción de la tarea 2.
Salidas	**(opcional)**
(Abreviatura de la Fase)A2. Nombre de la Actividad	
Entradas	**(opcional)**
Salidas	**(opcional)**

Una vez descrita la forma cómo se mostrará el proceso de pruebas desarrollado en este trabajo de grado, a continuación se presenta de manera detallada el mismo siguiendo el patrón descrito previamente.

3.2.1 Definición General del Proceso de Pruebas

PROPÓSITO	**El propósito del proceso es incorporar actividades y técnicas de pruebas que sean fácilmente aplicables a pequeñas organizaciones desarrolladoras de *Software* y que conduzcan a la obtención de productos de alta calidad.**
OBJETIVOS	El proceso de pruebas define los siguientes objetivos: • Asegurar la calidad de los productos desarrollados. • Lograr casos de pruebas exitosos tras la aplicación de las técnicas de pruebas funcionales que dan soporte a algunas de las actividades descritas por el proceso de pruebas.
DESCRIPCIÓN GENERAL	El proceso de pruebas se compone de las fases de inicio, planificación, diseño, ejecución, monitoreo y control: • Fase de Inicio: ✓ **FIA1. Análisis de la información:** Se recoge la información del proyecto al que se aplicarán las pruebas, con el objetivo de identificar los elementos que se van a probar en el Software en reposo[8] y en el Software ejecutable. Esto permite establecer una visión global de las pruebas en el sistema y en ese sentido prepararse para establecer un plan de pruebas que sirva como soporte para el proceso de pruebas que se llevará a cabo. A partir de esto, es posible encontrar información de pruebas en la organización que aporte contenido reutilizable. • Fase de Planeación: ✓ **FPA1. Planear las Pruebas:** Establece las decisiones sobre qué es lo más importante para lograr el éxito de las pruebas, definiendo un plan de pruebas, con los siguientes elementos: ➢ Estrategia de pruebas: - Elementos priorizados - Alcance

[8] Software en reposo: En este trabajo de grado "software en reposo" hace referencia a requerimientos, modelos, y demás artefactos que hacen parte del desarrollo software y que pueden ser evaluados sin que para ello se haya construido el producto.

	- Riesgos - Tipos y técnicas de pruebas - Roles - Indicadores de pruebas - Estimaciones de tiempo y esfuerzo - Herramientas para pruebas - Atributos de calidad a probar - Cronograma • Fase de Diseño: ✓ **FDA1. Análisis detallado de los elementos a probar:** Se recoge información detallada de los elementos a probar, para obtener diseños más precisos que permitan encontrar un mayor número de errores. ✓ **FDA2. Seleccionar y validar las técnicas de pruebas:** Seleccionar y validar las técnicas de pruebas más apropiadas para elaborar el conjunto de artefactos que evalúen los elementos descritos en la actividad anterior. ✓ **FDA3. Definir los artefactos a diseñar y sus convenciones:** Definir los artefactos a diseñar y sus convenciones, de tal manera que se conozca que componentes utilizar según sea el elemento a probar. ✓ **FDA4. Diseñar las pruebas:** El diseño de las pruebas se basa en la creación de casos de prueba y/o listas de chequeo cuya ejecución permita observar posibles síntomas de defectos. • Fase de Ejecución ✓ **FEA1. Ejecutar las pruebas:** En esta actividad se prepara y organiza el entorno especifico para las pruebas estáticas y otro para las pruebas dinámicas, ya que cada tipo de prueba requerirá un entorno con unas características especificas para que su ejecución sea lo más efectiva posible. Se establecen las condiciones bajo las cuales un caso de prueba y/o lista de chequeo pasa o falla, y finalmente se ejecutan las pruebas estáticas y/o dinámicas, y se reportan los hallazgos encontrados. • Fase de Monitoreo y Control ✓ **FMCA1.Gestionar informes de seguimiento:** En

	esta actividad se analizan los indicadores de producto y proceso para emitir juicios acerca del progreso de las pruebas y la calidad del producto. ✓ **FMCA2. Ejecutar planes de acción:** En esta actividad se ejecutan los planes de acción producto del análisis de los informes de seguimiento, y se verifica que estas acciones se hayan ejecutado de manera exitosa.
PRODUCTOS DE TRABAJO	Los productos de trabajo del proceso se definen a partir de entradas y salidas, en algunas tareas se definen entradas particulares y en otras con las entradas de cada actividad es suficiente para llevar a cabo las tareas, No necesariamente una salida de cada actividad es un producto de trabajo del proceso. Los productos de trabajo del proceso son: • Plan de pruebas • Casos de pruebas y/o listas de chequeo • Informes de hallazgos • Informes de seguimiento Ver Plantillas en el Anexo B.
RESPONSABILIDAD Y AUTORIDAD	Responsables: • Responsable(s) de pruebas Autoridad: • Líder de pruebas
Diagrama de flujo de trabajo	
(ver Anexo A) **Nota:** Las actividades del proceso se muestran como secuenciales, pero en la práctica algunas actividades deben visitarse nuevamente	

3.2.2 Patrón para las Fases

El proceso de pruebas se compone de un conjunto de fases, gran parte de estas secuenciales y una transversal al proceso. Cada proyecto tiene un inicio y un final definidos, los entregables específicos y las actividades que se llevan a cabo entre éstos variarán ampliamente de acuerdo con el proyecto. El ciclo de vida proporciona el marco de referencia básico para dirigir el proyecto, independientemente del trabajo específico involucrado.

3.2.2.1 Fase de Inicio

Fase	**FASE DE INICIO**	Abreviatura	**FI**
Propósito	El propósito de la fase de inicio es identificar y clasificar los elementos a probar analizando la información requerida para tal fin.		
Descripción de las actividades	La fase de inicio se compone de la siguiente actividad: • **Análisis de la información:** Se recoge la información del proyecto al que se aplicaran las pruebas, con el objetivo de identificar los elementos que se van a probar en el *software* en reposo y en el *software* ejecutable. Esto permite establecer una visión global de las pruebas en el sistema y en ese sentido prepararse para establecer un plan de pruebas que sirva como soporte para el proceso de pruebas que se llevara a cabo. A partir de esto, es posible encontrar información de pruebas en la organización que aporte contenido reutilizable.		
Roles involucrados	**Abreviatura**	**Nombre**	
	RPruebas	Responsable(s) de Pruebas	
	RProyectos	Responsable(s) de Proyectos	
Entradas	Información de pruebas en la organización Información del proyecto		
Salidas	Elementos a probar identificados y clasificados Información de pruebas reutilizable		

Actividades y tareas

Rol	Descripción
FIA1. Análisis de la información	
Entradas	Información referente a pruebas en la organización Información del Proyecto al que se aplicará el proceso de pruebas
RPruebas	**FIA1.1 Recoger la información** • Ubicar las fuentes[9] de información • Usar una o varias técnicas para recoger información
RPruebas **RProyectos**	**FIA1.2 Identificar los elementos a los que se aplicarán las pruebas** • Clasificar los elementos en: ✓ Elementos para revisiones formales ✓ Elementos para pruebas funcionales
RPruebas	**FIA1.3 Identificar información de pruebas reutilizable (opcional)**

[9] Las fuentes de información son instrumentos necesarios para la búsqueda y acceso a la información Ej. Repositorios de información, Personas, entre otros.

	• Ubicar los repositorios de información • Identificar artefactos de pruebas • Identificar la información que sea reutilizable de acuerdo a las necesidades del proyecto **Nota:** En caso de que la organización no cuente con información u artefactos de pruebas se empezara a levantar información con la ejecución de este proceso.
Salidas	Elementos a probar identificados y clasificados Información de pruebas reutilizable
Técnicas	Técnicas para recoger información: • Revisión de documentos • Entrevistas • Cuestionarios • Observación

3.2.2.2 Fase de Planeación

Fase	**FASE DE PLANEACIÓN**	Abreviatura	**FP**
Propósito	Esta fase tiene como propósito identificar la estrategia de pruebas necesaria para definir el plan de pruebas que guía la evaluación de los elementos que se pondrán a prueba en el proyecto.		
Descripción de las actividades	La Fase de Planeación se compone de la siguiente actividad: • **Planear las pruebas:** Establece las decisiones sobre qué es lo más importante para lograr el éxito de las pruebas, definiendo un plan de pruebas, con los siguientes elementos: ✓ Estrategia de pruebas: - Elementos priorizados - Alcance - Riesgos - Tipos de pruebas - Roles - Indicadores de pruebas - Estimaciones de tiempo y esfuerzo - Herramientas para pruebas - Atributos de calidad a probar - Cronograma		
Roles involucrados	**Abreviatura**	**Nombre**	
	RPruebas	Responsable(s) de Pruebas	
Entradas	Elementos a probar identificados y clasificados		

	Información de pruebas reutilizable
Salidas	Plan de pruebas

Actividades y Tareas

Rol	Descripción
FPA1. Planear las Pruebas	
Entradas	Elementos a probar identificados y clasificados Información de pruebas reutilizable
RPruebas	**FPA 1.1 Definir la estrategia para probar el software** • Priorizar los elementos a prueba • Definir el alcance • Identificar los riesgos y su tratamiento • Definir los tipos y técnicas de pruebas • Identificar los roles del personal que llevara a cabo las pruebas • Definir indicadores de pruebas • Estimaciones de tiempo y esfuerzo • Especificar las herramientas a utilizar • Determinar los atributos de calidad a probar
RPruebas	**FPA 1.2 Planificar y Programar las actividades de pruebas** • Analizar el orden de las actividades • Analizar la duración de las actividades • Identificar y asignar recursos • Generar el cronograma
RPruebas	**FPA1.3 Condensar la información en el plan de pruebas** • Registrar los resultados de la tarea FIA1.2 Definir el alcance de las pruebas. • Registrar los resultados de la tarea FPA 1.1 Definir la estrategia para probar el software. • Registrar los resultados de la tarea FPA 1.2 Planificar y Programar las actividades de pruebas.
Salidas	Plan de pruebas

3.2.2.3 Fase de Diseño

Fase	**FASE DE DISEÑO**	Abreviatura	**FD**
Propósito	La fase de diseño tiene como propósito construir los artefactos[10] necesarios para ejecutar las pruebas en el proyecto.		
Descripción de	La fase de diseño se compone de las siguientes actividades:		

[10] En la fase de diseño los artefactos de pruebas hacen referencia a casos de prueba en lo relacionado a pruebas dinámicas y listas de chequeo en cuanto a pruebas estáticas.

las actividades	• Análisis detallado de los elementos a probar, para obtener diseños más precisos que permitan encontrar un mayor número de errores. • Seleccionar y validar las técnicas de pruebas más apropiadas para elaborar el conjunto de artefactos que evalúen los elementos descritos en la actividad anterior. • Definir los artefactos a diseñar y sus convenciones, de tal manera que se conozca que componente utilizar según sea el elemento a probar. • Diseñar las pruebas: el diseño de las pruebas se basa en la creación de casos de prueba y/o listas de chequeo cuya ejecución permita observar posibles síntomas de defectos. Los casos de prueba se diseñan se acuerdo a la técnicas de pruebas funcional seleccionada.	
Roles involucrados	**Abreviatura**	**Nombre**
	RPruebas	Responsable(s) de Pruebas
Entradas	Información de los elementos a probar Plan de pruebas	
Salidas	Casos de prueba diseñados Listas de chequeo	

Actividades y Tareas

Rol	Descripción
FDA1. Análisis detallado de los elementos a probar	
Entradas	Información de los elementos a probar Plan de pruebas
RPruebas	**FDA1.1 Recoger información detallada** • Ubicar los elementos a prueba • Buscar información detallada del elemento • Identificar las variables y los valores del elemento a prueba (para el caso de pruebas Funcionales) • Identificar los atributos de calidad a revisar (Para revisiones)
Salidas	Información detallada de los elementos a probar
FDA2. Seleccionar y validar las técnicas de pruebas	
Entradas	Información detallada de los elementos a probar
RPruebas	**FDA2.1 Identificar las técnicas apropiadas** • Analizar las variables y valores (para el caso de pruebas Funcionales) • Analizar los atributos de calidad a revisar (Para las listas de chequeo) • De acuerdo al análisis hecho identificar la técnica más apropiada.

	Técnicas de pruebas funcionales	Partición equivalente
		Análisis de valor limite
		Tablas de decisión
		Arreglos Ortogonales (Herramienta de ayuda allpairs)
Salidas	Técnicas	
FDA3. Definir los artefactos a diseñar y sus convenciones		
RPruebas	**FDA3.1 Definir los artefactos que soportaran la ejecución de las pruebas** • Identificar el artefacto a utilizar para las pruebas. En caso de revisiones formales se sugiere el uso de listas de chequeo y para pruebas dinámicas el uso de casos de prueba. **Nota:** Para esto se proporcionan plantillas como anexos al documento.	
RPruebas	**FDA3.2 Definir la estructura del artefacto** • Identificar los campos que tendrá el artefacto (Identificación, descripción, prerrequisitos, actores, responsables, resultado esperado, resultado obtenido) • Identificar estados (Ej. En construcción, Exitoso, Fallido, Frenado, Pendiente de ejecución)	
Salidas	Plantillas para casos de prueba y/o listas de chequeo	
FDA4. Diseñar las pruebas		
Entradas	Técnicas	
RPruebas	**FDA4.1 Construir los artefactos** • Registrar información básica solicitada en la plantilla definida • Diseñar Listas de chequeo: ✓ Enumerar los ítems de chequeo ✓ Descripción del ítem ✓ Descripción que permita evaluar el atributo de calidad para cada ítem (puede ser una pregunta) • Diseñar Casos de prueba: ✓ Describir el proceso o escenario que debe llevarse a cabo para ejecutar la prueba (uso de la técnica, ver numeral 3.3.1) ✓ Indicar las precondiciones para ejecutar la prueba (si las hay) ✓ Indicar los pasos necesarios para ejecutar la prueba ✓ Indicar el resultado esperado ✓ Indicar el estado del caso de prueba	
Rpruebas	**FDA4.2 Listar los datos necesarios para las pruebas**	
Salidas	Casos de prueba y/o Listas de chequeo	

3.2.2.4 Fase de Ejecución

Fase	**FASE DE EJECUCIÓN**	Abreviatura	**FE**
Propósito	El propósito de esta fase es ejecutar la información contenida en los artefactos diseñados previamente, y a partir de esto reportar los hallazgos encontrados.		
Descripción de las actividades	• **FEA1. Ejecutar las pruebas** En esta actividad se preparar y organiza el entorno especifico para las revisiones y otro para las pruebas dinámicas, ya que cada tipo de prueba requerirá un entorno con unas características especificas para que su ejecución sea lo más efectiva posible. Se establecen las condiciones bajo las cuales un caso de prueba y/o lista de chequeo pasa o falla, y finalmente se ejecutan las pruebas estáticas y/o dinámicas.		
Roles involucrados	**Abreviatura**	**Nombre**	
	RPruebas	Responsable(s) de Pruebas	
Entradas	Casos de prueba diseñados Listas de chequeo		
Salidas	Casos de prueba ejecutados Listas de chequeo diligenciadas Reporte de hallazgos		

Actividades y Tareas

Rol	Descripción
FEA1. Ejecutar las pruebas	
Entradas	Casos de prueba y/o Listas de chequeo
RPruebas **RProyectos**	**FEA1.1 Preparar y organizar el entorno en el que se ejecutaran las pruebas** • Organizar el entorno de pruebas ya sea documentación en caso de pruebas estáticas o producto *software* en caso de pruebas dinámicas. • Preparar el entorno con los datos necesarios para ejecutar las pruebas (Documentos en caso de pruebas estáticas o datos en la base de datos para pruebas dinámicas)
RPruebas **RProyectos**	**FEA1.2 Identificar condiciones de paso o fallo de las pruebas** **Nota:** Las condiciones de paso o fallo de las pruebas están dados por los indicadores. (En la plantilla de Plan de pruebas se facilitan indicadores de proceso y producto)
RPruebas	**FEA1.3 Ejecutar los casos de prueba o listas de chequeo**

	• Contrastar la información de los casos de prueba o listas de chequeo con la documentación y/o producto software • Evaluar el elemento a prueba. En el caso de listas de chequeo se puede evaluar diciendo si para cada ítem es aceptable o no la condición evaluada. Para casos de prueba se compara el valor esperado con el valor obtenido y se emite un juicio al respecto.
RPruebas	**FEA1.4 Registrar los hallazgos** ✓ El juicio emitido tras ejecutar los artefactos puede ser el hallazgo de un incidente que debe ser registrado en un documento ✓ Identificar la forma de registrar el incidente. La forma en que se escribe un hallazgo encontrado juega un papel importante en la comunicación entre los interesados, por tanto se debe llegar a un acuerdo en el lenguaje con el fin de que todos se entiendan. **Nota:** Por ejemplo para hallazgos en producto software desarrollado se puede tomar como referente la siguiente estructura: ✓ **Descripción:** Campo para definir ¿Dónde?, ¿Cuándo?, ¿Qué hallazgo se encontró?, ¿Que se espera del sistema tras la acción?, si es necesario se pueden adjuntar imágenes para ilustrar el hallazgo ✓ **Pasos para reproducir el hallazgo:** Campo para enumerar el número de pasos necesarios para llegar al hallazgo. Paso1->pasoN **Ejemplo:** ✓ **Descripción:** En el campo “correo electrónico” del formulario en el que se crea un usuario (modulo Gestión de usuarios), cuando se ingresa un dato que no conserva la estructura de un correo electrónico, se permite crear el usuario, se espera un mensaje de error que indique que el formato ingresado en ese campo no se corresponde con un formato correcto. ✓ Pasos para reproducir el hallazgo: Panel de administración -> Gestión de usuarios -> crear un usuario nuevo.
RPruebas	**FEA1.5 Clasificar los hallazgos** • Ordenar los hallazgos con características comunes. Ej. Para revisiones (hallazgos de Forma, hallazgos de fondo), para pruebas dinámicas (hallazgos de interfaz, hallazgos bloqueantes y hallazgos funcionales). Este ítem puede hacer parte de la estrategia de pruebas, mediante el cual se establezcan las prioridades para solución de hallazgos. • Definir estados para el hallazgo. Los estados permiten establecer reglas de comunicación entre los interesados. Por ejemplo, un

	hallazgo puede pasar por los siguientes estados: - Reportado - Aceptado/Rechazado - Resuelto - Confirmado por pruebas - Cerrado
RPruebas	**FEA1.6 Reportar los hallazgos** **Nota:** Debe acudirse al Responsable del proyecto, para que de las indicaciones de la persona encargada de dar respuesta a los hallazgos encontrados.
Salidas	Casos de prueba ejecutados Listas de chequeo diligenciadas Reporte de hallazgos

3.2.2.5 Fase de Monitoreo y Control

Fase	**FASE DE MONITOREO Y CONTROL**	Abreviatura	**FMC**
Propósito	El propósito de la fase de monitoreo y control es asegurar que las pruebas se llevan a cabo de acuerdo a lo planeado.		
Descripción de las actividades	**FMCA1.Gestionar informes de seguimiento** En esta actividad se analizan los indicadores de producto y proceso para emitir juicios acerca del progreso de las pruebas y la calidad del producto. **FMCA2. Ejecutar planes de acción** En esta actividad se ejecutan los planes de acción producto del análisis de los informes de seguimiento, y se verifica que estas acciones se hayan ejecutado de manera exitosa.		
Roles involucrados	**Abreviatura**	**Nombre**	
	RCalidad	Responsable(s) de Calidad	
	RPruebas	Responsable(s) de Pruebas	
	RProyectos	Responsable(s) de Proyectos	
	EDesarrollo	Equipo de desarrollo	
	DOrganización	Directivos de la organización	
Entradas	Información de Pruebas		
Salidas	Informes de seguimiento		

Actividades y Tareas

Rol	Descripción
FMCA1. Gestionar informes de seguimiento	
Entradas	Información de pruebas
RPruebas	**FMCA1.1 Revisar indicadores** • Ubicar las fuentes de información • Extraer la información requerida de acuerdo a la fórmula del

	indicador • Interpretar los resultados **Nota**: En caso de que no se establezca un indicador se analiza la información de acuerdo a los criterios de la organización para el paso/fallo de una prueba, en caso de que el informe este asociado a la fase de ejecución
RPruebas	**FMCA1.1 Crear el informe** • Registrar la información requerida por el informe
RPruebas	**FMCA1.2 Comunicar el informe de seguimiento** • Establecer la periodicidad de los informes • Enviar el informe a los interesados
Salidas	Informes de seguimiento
FMCA2. Ejecutar planes de acción	
Entradas	Planes de acción
RPruebas **RProyectos** **RCalidad** **EDesarrollo** **DOrganización**	**FMCA2.1 Realizar acciones correctivas** • Llevar a cabo acciones correctivas en caso de que los resultados de los indicadores se desvíen considerablemente de los resultados esperados
RPruebas **RProyectos**	**FMCA2.2 Verificar que las acciones correctivas se hayan realizado**
Salidas	Acciones correctivas efectuadas

3.2.2.6 Fase de finalización

Fase	**FASE DE FINALIZACIÓN**	Abreviatura	**FF**
Propósito	El propósito de la fase de finalización es concluir las pruebas realizadas y analizar los resultados obtenidos en el proceso.		
Descripción de las actividades	**FFA1. Concluir las pruebas:** Mediante esta actividad se analizan los resultados de las pruebas y mediante los indicadores se determina el nivel de calidad del *software*.		
Roles involucrados	**Abreviatura**	**Nombre**	
	RPruebas	Responsable(s) de Pruebas	
Entradas	Información de todo el proceso de pruebas		
Salidas	Informe final		

Actividades y Tareas

Rol	Descripción
FFA1. Concluir las pruebas	
Entradas	Información de todo el proceso de pruebas
RPruebas	**FFA1.1 Especificar los resultados de las pruebas**

	• Ubicar las fuentes de información • Extraer la información necesaria para formular un resultado • Analizar indicadores finales • Registrar los resultados en el informe final • Comunicar los resultados a los interesados
Salidas	Informe final

3.2 TÉCNICAS PARA EL DISEÑO DE PRUEBAS

Como se ha especificado a lo largo de este trabajo de grado, los casos de prueba revelan información que permite concluir si un desarrollo *software* funciona correctamente o no. Ahora bien, las técnicas son los métodos que guían al diseñador de pruebas en la creación de dichos casos, partiendo de las posibles entradas que se requieren para ejecutar el software que se pretende probar [32][37]. Con lo anterior, se busca encontrar un porcentaje significativo del total de errores que tenga el elemento a prueba, sin que para ello se tenga que llevar a cabo un gran número de ciclos de pruebas. Este proceso determina la efectividad final de las pruebas y la calidad del producto que se construye. Por tanto, determinar la mejor técnica para una tarea específica no es algo trivial, sobre todo porque es difícil encontrar las mejores combinaciones de entradas del sistema que tengan una alta probabilidad de hallar errores dentro de un gran conjunto de entradas que no se pueden probar en su totalidad, de ahí que para crear buenos casos de prueba no solo es necesario conocer cada técnica si no que al igual que desarrollar software, se requiere de inteligencia y creatividad, por lo que necesariamente esta actividad debe realizarse por personas con buen conocimiento técnico de las pruebas y del dominio del producto *software* objetivo [68]. Lo más importante es estructurar, diseñar y aplicar eficientemente un conjunto de casos de prueba. Este diseño es importante partiendo del principio expuesto por [71] que denota que no son posibles las pruebas exhaustivas, por lo que la mejor estrategia es crear casos de prueba tan completos y eficientes como sea posible.

La documentación que existe alrededor de las técnicas de pruebas es muy amplia, algunos autores las describen en mayor o menor detalle, y las ilustran a su manera sin salirse de la esencia de cada una de ellas. Por ser los casos de prueba uno de los elementos más importantes de la prueba, y debido a que existen diferentes enfoques para crearlos, cada propuesta describe estrategias propias para diseñarlos y aplicarlos. En la bibliografía se encuentra gran variedad de técnicas, de acuerdo a un estudio consignado en [37] estas técnicas se agrupan en siete familias: técnicas funcionales, aleatorias, de flujo de control, de flujo de datos, de mutación, de minimización, de regresión, entre otras. Estas técnicas en la mayoría de los casos son erróneamente clasificadas en términos de caja negra y caja blanca, se dice erróneo debido a que algunas de ellas se categorizan única y exclusivamente como de caja blanca y en realidad tienen también una amplia aplicabilidad en enfoque de caja negra. De acuerdo a [58] es difícil hacer una clasificación homogénea de todas las técnicas, dado que utilizan diferentes tipos de información y características, y están destinadas a encontrar diferentes tipos de errores que son inherentes a la técnica y el modelo en que están basadas.

El Glosario estándar de la IEEE [70] define las pruebas de caja Negra (también llamadas pruebas funcionales) como las pruebas que ignoran el mecanismo interno de un sistema o componente y se centran únicamente en las salidas generadas en respuesta a las entradas y las condiciones de ejecución seleccionadas, y las pruebas de caja blanca (también llamadas pruebas estructurales) son las pruebas que tienen en cuenta el mecanismo interno de un sistema o componente.

3.3.1 Técnicas Funcionales

La familia de técnicas objeto principal y alcance de esta tesis de pregrado son las técnicas funcionales, pues son las que se relacionan directamente con la actividad de diseño de pruebas previamente establecida dentro del proceso de pruebas definido en este documento. Las pruebas funcionales surgieron en los años 80, y de acuerdo a [68] con estas pruebas se buscaba incrementar la calidad del *software* haciendo de las pruebas un proceso continuo y paralelo al ciclo de vida de los proyectos de desarrollo. Las pruebas funcionales son aquellas que se aplican a un producto final (llámese funcionalidad o modulo), y permiten detectar en qué puntos el producto no cumple sus especificaciones, es decir, gracias a este enfoque de pruebas se puede comprobar su funcionalidad. Según [56] la ejecución de las pruebas funcionales consiste en estructurar, diseñar y aplicar un conjunto de casos de prueba con el objetivo de verificar si el comportamiento que presenta el *software* cumple o no con lo esperado de acuerdo a su especificación, para lo que es necesario tener en cuenta el punto de vista del usuario. De acuerdo a [37] “Esta familia de técnicas se basa en las especificaciones del programa para generar los casos de prueba. Dividen el dominio de posibles entradas al sistema en subdominios, y generan los casos de prueba eligiendo uno o más elementos (depende de la técnica en particular dentro de la familia) del subdominio”.

A lo largo de la historia de las pruebas de *software* se han propuesto técnicas de prueba que a la vez han dado origen a investigaciones que han culminado con diferentes formas de aplicarlas. Al comienzo de todo se estructuraron técnicas que seguían el enfoque de pruebas funcionales pero hacía falta cómo ponerlas en práctica, por lo que algunos investigadores *como Boris Beizer, Roger Pressman, Ian Sommerville, Glenford Myers*, entre otros, se dieron a la tarea de realizar proyectos de investigación que condujeron a las primeras propuestas de aplicación de esas técnicas. Por Ejemplo *Boris Beizer* [56] separa las técnicas en funcionales y estructurales y mediante una serie de ejemplos demuestra cómo utilizar estas técnicas para validar los requisitos del *software* a prueba.

A continuación se describen un conjunto de técnicas de pruebas funcionales que pueden dar soporte a alguna(s) actividades del proceso de pruebas definido, se analizan y se relacionan al proceso indicando al lector el momento apropiado para usarlas. Se pueden ver otras técnicas analizadas en el Anexo R.

3.3.1.1 Partición equivalente

Normalmente los datos de entrada y salida de un software también llamados parámetros, se pueden agrupar en diferentes clases de acuerdo a las características comunes que los identifiquen de manera particular. Habitualmente, los datos pertenecientes a un mismo

grupo se comportan de manera similar. Por tanto, de producirse alguna falla a causa de uno de ellos, y sin que sea necesario utilizarlos todos, se esperaría que los demás también produjeran la misma falla. De ahí que, el software debería comportarse de idéntica forma para todos los datos que comparten las mismas características particulares. Debido a este comportamiento equivalente, estas clases reciben el nombre de particiones de equivalencia, clases de equivalencia o dominios [56]. Alrededor de esta técnica, varios autores dan su propia opinión al respecto:

- *Tim Koomen, Leo van der Aals, Bart Broekman y Michiel Vroon*, en su libro "*Tmap Next for result-driven testing*" [57] reconocen está técnica como "Clases de equivalencia" para lo que indican que: " La aplicación de las clases de equivalencia permite definir un rango para el valor de un parámetro que está dividido por medio de clases, en el que el comportamiento del sistema es similar (equivalente)".

- *Roger Pressman* define está técnica como: "La *partición equivalente* es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba" [55].

- *Ian Sommerville* en [54] indica que: "Las particiones de equivalencia son conjuntos de datos en donde todos los miembros de los conjuntos deberían ser procesados de forma equivalente".

- *Boris Beizer* afirma que: "La partición equivalente divide el dominio de entrada de un programa en clases. Para cada una de estas clases de equivalencia, el conjunto de datos debe ser tratado de la misma forma por el módulo bajo prueba y debe producir la misma respuesta. Los casos de prueba deben ser diseñados para que las entradas se encuentren dentro de estas clases de equivalencia".

- *Glenford Myers* indica que: "Dividir el dominio de entrada de un programa en un número finito de clases de equivalencia que puede asumir razonablemente (pero, por supuesto que no, estar absolutamente seguro) de que una prueba de un valor representativo de cada clase es equivalente a una prueba de cualquier otro valor".

Como es de notar, se brindan definiciones homogéneas alrededor de este tema, de lo que se concluye que finalmente el propósito de esta técnica, es dividir en particiones equivalentes las entradas y salidas válidas y no validas del sistema (o elemento) que se quiere probar, de tal manera que, el comportamiento del sistema sea el mismo para cualquier elemento contenido en una partición particular.

De la aplicación de esta técnica se espera reducir la cantidad de casos de prueba que se podrían generar a lo largo del ciclo de vida del software. Para la partición equivalente el diseño de casos de prueba equivale a evaluar en las clases de equivalencia los parámetros del elemento a probar, con el objetivo de definir casos de prueba que descubran clases de errores, reduciendo el número de casos requeridos para cubrir las condiciones de entrada [55].

De acuerdo a la bibliografía, para diseñar casos de prueba de manera eficiente bajo la utilización de esta técnica, se recomienda seguir los siguientes pasos:

1. Identificar las clases de equivalencia
2. Identificar los casos de prueba

3.3.1.1.1 Identificar las clases de equivalencia

De acuerdo a [60] el problema está en identificar las clases de equivalencia, para lo que no existe una regla general, sin embargo, existe mucha documentación al respecto, donde cada autor desde su investigación o experiencia facilita guías a través de las cuales se logra la identificación de dichas clases. Una manera sencilla y práctica de identificar las clases de equivalencia se muestra en [58] “Recetas para el diseño de casos de testeo”, donde a través de una serie de instrucciones, se indica cómo identificar clases de equivalencia utilizando los ingredientes y utensilios adecuados para ello. De acuerdo a esto, los pasos apropiados para obtener las clases de equivalencia son:

1. Identificar todas las funciones del software a probar.
2. Determinar las entradas y salidas de cada función.
3. Examinar los tipos de valores a los que pertenecen las entradas.
4. Dividir las entradas en conjuntos de acuerdo a los tipos de valores en los que se identifican. A estos conjuntos son a los que se les denomina clases de equivalencia. Cada clase puede ser válida (con datos permitidos por la función) o no valida (con datos no permitidos por la función).

Es importante, de acuerdo a este artículo, verificar que cada elemento que compone las clases de equivalencia identificadas, son equivalentes, es decir, cada uno de ellos cumple con los siguientes ítems:

- Tienen las mismas características.
- Si uno de ellos produce un fallo los otros probablemente también.
- Si uno de ellos no produce ningún fallo los otros probablemente tampoco.

A esta técnica, resulta conveniente agregar una serie de recomendaciones citadas por autores como *Boris Beizer, Glenford Myers, Roger Pressman*, entre otros:

- Si una condición de entrada especifica un rango, se definen una clase de equivalencia válida y dos no válidas [55].
- Si una condición de entrada requiere un valor específico, se definen una clase de equivalencia válida y dos no válidas [55].
- Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y otra no válida [55].
- Si una condición de entrada es booleana, se definen una clase de equivalencia válida y otra no válida [55].

3.3.1.1.2 Identificar los casos de prueba

Identificar un caso de prueba radica en lograr una combinación de elementos de entrada para los que se espera obtener un resultado específico. Para dar este paso, es necesario

escoger un representante de cada clase de equivalencia y combinarlos. Por lo tanto la identificación de casos de prueba se compone de dos pasos:

1. Escoger un representante de cada clase: De acuerdo a [58] para identificar los casos de prueba es necesario seleccionar un valor que represente cada clase equivalente, si se verificó que todos los elementos que componen las clases son equivalentes, cualquier valor que se escoja representara a todos los demás.

2. Generar los casos de prueba a partir de los representantes escogidos: De acuerdo a la bibliografía las mejores combinaciones se logran cuando se combinan elementos de clases validas con elementos de clases validas y elementos de clases validas con elementos de clases no validas, para lo que se especifica la salida deseada de acuerdo a cada combinación particular.

La finalidad de esta técnica es concluir con casos de prueba óptimos, por lo que es necesario considerar tantas condiciones de entrada como sea posible.

¿Cuándo utilizar esta técnica?

En [58] se establece que esta técnica resulta útil cuando se cuenta con entradas que se pueden dividir en conjunto con características similares y se conocen exactamente los resultados que deben arrojar tras ejecutar las funciones individuales del software a probar. Puede utilizarse por ejemplo a nivel de código fuente, para probar subprogramas, en las pruebas a formularios en los que cada campo es una variable de entrada con su propio dominio; etc.

3.3.1.2 Análisis en el valor limite

La experiencia en las pruebas de software de acuerdo a [32] y [60], ha mostrado a través del tiempo, que analizar el software en los puntos de cambio de las clases de equivalencia, es decir en los límites, da mejores resultados, esto bajo la consideración de que existe una tendencia a fallar precisamente cuando el software trabaja con elementos que se hallan en los márgenes de las clases de equivalencia, que normalmente son dos, uno justo abajo y otro justo encima. Esta técnica se denomina Análisis de valor límite y establece una guía en la creación de casos de prueba en el "borde" de las clases de equivalencia, razón por la cual muchos autores de la bibliografía encontrada afirman que esta técnica complementa la partición equivalente, pues gracias a ella fácilmente se puede identificar el mejor valor equivalente con la mayor probabilidad de encontrar un error. Boris Beizer al rededor de esta técnica afirma que " Los Bugs acechan en las esquinas y se congregan en las fronteras" [61].

Roger Pressman indica que: “El análisis de valores límite es una técnica de diseño de casos de prueba que complementa la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el análisis de valor limite lleva a la selección de casos de prueba en las “aristas” de la clase” [55].

Glenford Meyers dice que: “Las condiciones de frontera son aquellas situaciones directamente en, sobre y debajo de los bordes de las clases de equivalencia para los valores de entrada y salida del software” [59]. El análisis de valor límite de acuerdo a este autor requiere un cierto grado de creatividad y un cierto grado de especialización hacia el problema en cuestión. Sin embargo, en [55] se plantean algunas pautas generales para tener en cuenta:

1. “Si una condición de entrada especifica un rango limitado por los valores a y b, los casos de prueba deben diseñarse con esos valores, además de los que se encuentran apenas abajo y arriba de ellos”
2. “Si una condición de entrada especifica diversos valores, deben desarrollarse casos de prueba que ejerciten los números mínimo y máximo. También se prueban los valores ubicados apenas arriba y debajo de estos máximos y mínimos.”
3. “Aplicar las directrices 1 y 2 a las condiciones de salida.”
4. “Si la estructura interna de datos del programa tiene límites prescritos debe diseñarse un caso de prueba para ejercitar los límites de la estructura de datos”.

¿Cuándo utilizar esta técnica?

Varios autores infieren que esta técnica es un complemento a la técnica de partición equivalente. En esta última, para deducir los casos de prueba es necesario escoger un representante de cada clase de equivalencia y combinarlos. De ahí, que el análisis del valor límite complementa la partición equivalente, pues ayuda a escoger no solo el mejor representante de la clase si no el valor con mayor tendencia a causar un error.

3.3.1.3 Tablas de decisión

De acuerdo a [63] los diagramas de estado utilizados al aplicar la técnica de pruebas de transición de estados dejan de ser útiles cuando el número de estados es demasiado alto, pues se convierte en un diagrama gráficamente intratable. De ahí que utilizar la técnica de tablas de decisión resulta mucho más práctico. De acuerdo a [56] la información presentada en estas tablas es la misma que se presenta en los diagramas de estados, a continuación un ejemplo de estructura de tabla:

Figura 11. Ejemplo de tabla para técnica de tablas de decisión

	Regla 1	Regla 2	Regla 3
Condiciones			
Acciones			

1. Las condiciones representan las condiciones de entrada (causas). Cada condición se expresa como un valor booleano (verdadero o falso) para una entrada, o combinación de ellas.

2. Las acciones son los eventos que se generan dependiendo de las condiciones de entrada. Cada evento se expresa como una expresión booleana que representa un resultado, o una combinación de resultados.
3. Cada regla (o columna) se convierte en un caso de prueba [56], donde cada una de ellas define la relación entre las causas y los efectos.

Según [57] esta técnica es muy completa, pues permite analizar minuciosamente el funcionamiento del elemento a prueba gracias a las condiciones y tablas de decisión que utiliza como base fundamental para el diseño de casos de prueba. El objetivo de esta técnica de acuerdo a [57] no es hacer la mayor combinación de caminos funcionales si no hacer la cobertura completa de las condiciones. De ahí la importancia de identificar las reglas (Condiciones de prueba) y acciones (Resultados esperados).

¿Cuándo utilizar está técnica?

Esta técnica es utilizada para registrar reglas del negocio complejas basadas en un conjunto de condiciones, las cuales deben ser implementadas en el sistema.

3.3.1.4 Arreglos Ortogonales

Está técnica es también conocida como “El método del Dr. Genichi Taguchi para el diseño”. La técnica de arreglos ortogonales es usada para deducir las mejores combinaciones de un conjunto de valores con el objetivo de lograr un diseño óptimo [86]. En el caso de las pruebas de software, las mejores combinaciones logran diseños óptimos con un mínimo número de casos de prueba. A continuación se enseña la manera en que deben ubicarse las variables y valores seleccionados para analizarlos a partir de esta técnica.

Tabla 9. Organización de variables y valores para la técnica de arreglos ortogonales

Variable1	Variable2	Variable3	Variable4	Variable5	Variable6	Variable7
Valor1	Valor5	Valor9	Valor13	Valor17	Valor21	Valor25
Valor2	Valor6	Valor10	Valor14	Valor18	Valor22	Valor26
Valor3	Valor7	Valor11	Valor15	Valor19	Valor23	Valor27
Valor4	Valor8	Valor12	Valor16	Valor20	Valor124	Valor28

¿Cuándo utilizar está técnica?

Esta técnica se utiliza cuando los valores que toman las variables son tantos que no se pueden tratar por tablas de decisión. Para utilizar está técnica se aconseja el uso de la herramienta allpairs. Esta herramienta permite ingresar por línea de comando un documento de texto que contiene todas las variables con sus posibles valores y a partir de estas entradas arroja la mejor muestra de todas las posibles combinaciones. A partir de allí es responsabilidad de quien diseña la prueba analizar la muestra y plantear los escenarios de pruebas para crear los casos de prueba.

3.3.1.5 Escenarios de pruebas

El objetivo de esta técnica es diseñar casos de prueba a partir de escenarios[11] del sistema a prueba. Un requerimiento del sistema puede tener diferentes escenarios, por tanto, un requerimiento puede tener varios casos de prueba.

De acuerdo a [62] las pruebas basadas en escenarios definen los casos de prueba de acuerdo a los flujos de trabajo entre usuarios y el sistema o entre el elemento a prueba con otros sistemas, para lo que se hace necesario cubrir como mínimo el escenario normal del sistema a prueba, y de ahí en adelante abordar el número de escenarios que se deseen cubrir y los flujos alternativos que se pueden presentar en la secuencia normal de trabajo.

En [62] se dan algunas pautas para derivar casos de prueba bajo esta técnica:

1. Se debe seleccionar el escenario al cual se le diseñara el caso(s) de prueba.
2. Se deben identificar las entradas necesarias para ejecutar la trayectoria que cubrirá el caso de prueba.
3. Se debe determinar el resultado que se espera del caso de prueba tras la inserción en el sistema de la entrada (s) establecida en el paso anterior.
4. Se deben repetir estos pasos hasta alcanzar el nivel requerido de cobertura[12] de la prueba de escenarios.

¿Cuándo utilizar esta técnica?

De acuerdo a [62] las pruebas basadas en escenarios normalmente se utilizan para realizar "pruebas de extremo a extremo", por ejemplo, durante las pruebas del sistema o pruebas de aceptación del usuario. El diseño de estas pruebas puede valerse de la herramienta allpairs usada en la técnica de arreglos ortogonales. Por medio de esta herramienta se pueden deducir los mejores escenarios cuando se tienen variables con muchos valores y es pesado hacer las combinaciones de los mismos.

3.3.1.5.1 Casos de prueba desde los casos de uso

De acuerdo a [62] los casos de prueba a partir de casos de uso son un ejemplo de técnicas basadas en escenarios, donde cada caso de prueba se modela como un caso de uso. Según [67] los escenarios suelen confundirse con los casos de uso, siendo que estos últimos representan las funciones de todo el sistema, y en cambio, los escenarios ejemplifican el uso del sistema, los escenarios son el flujo que sigue un caso de uso durante su ejecución, de ahí que, de un caso de uso se pueden derivar muchos escenarios.

[11] Un escenario es la descripción de un comportamiento específico del sistema.
[12] Número de escenarios verificado por las pruebas. Básicamente la cobertura se define a partir de dos preguntas: ¿Qué se selecciona? y ¿Qué se deja por fuera?

3.4 NIVELES DE PRUEBAS FUNCIONALES

De acuerdo a la bibliografía encontrada a lo largo de esta investigación, se ha encontrado que las pruebas se dividen en niveles, de acuerdo a la cobertura que se quiera lograr se dividen en pruebas unitarias, de integración y de sistema. Las pruebas unitarias son las que se llevan a cabo en funciones, procedimientos o subprogramas de manera individual [8][72][87]. En lugar de probar todo el desarrollo, se prueban primero partes pequeñas del mismo. Las pruebas de integración al igual que las unitarias tampoco tienen como objetivo probar todo el *software* de una vez, en su lugar, dan un paso más allá de las pruebas unitarias y prueban las mismas partes del software probadas en las pruebas unitarias pero esta vez en conjunto [8][72]. Las pruebas del sistema se llevan a cabo sobre el sistema completo y de acuerdo a [68] este nivel se considera normalmente como el más apropiado para comparar al sistema con sus requisitos no funcionales como seguridad, velocidad, exactitud y confiabilidad, además de las interconexiones externas con otras aplicaciones, utilidades, dispositivos hardware o con el sistema operativo.

3.5 MODELADO DEL PROCESO

El modelado permite entender fácilmente las actividades e información planteada en el proceso, pues se muestra de una forma organizada y sistemática, sirviendo esto como un referente centralizado y de fácil acceso a la organización, lo que supone la disposición de un repositorio de conocimiento sobre el proceso en un formato estandarizado. Esto con el fin de que el proceso no solamente quede definido si no que también se haga un uso efectivo de él, y se establezca como práctica formal de pruebas. El propósito de modelar el proceso de pruebas radica en que dicho proceso sea un activo organizacional de tal manera que si se llegara a desvincular el creador del mismo, este no se lleve consigo todo el conocimiento del proceso y su adaptación.

El proceso de pruebas propuesto en este trabajo de grado se ha modelado en la herramienta *Eclipse Process Framework Composer (EPF Composer*) la cual proporciona un marco de trabajo conceptual que provee los elementos necesarios para modelar, documentar, presentar, publicar y gestionar los diferentes componentes del proceso de pruebas.

Actualmente el proceso configurado contiene los elementos esenciales: actividades, tareas, recursos, productos y actores que el proceso requiere para alcanzar los objetivos planteados. De modo que se puede navegar por el proceso publicado y localizar con facilidad estos elementos y sus relaciones a través del modelo en un lenguaje de fácil comprensión para los ingenieros de *software*. Se pueden identificar por el estereotipo o ícono el tipo de elemento lo que visualmente brinda mucha información. A continuación se presenta en detalle la descripción del modelo en *EPFComposer*:

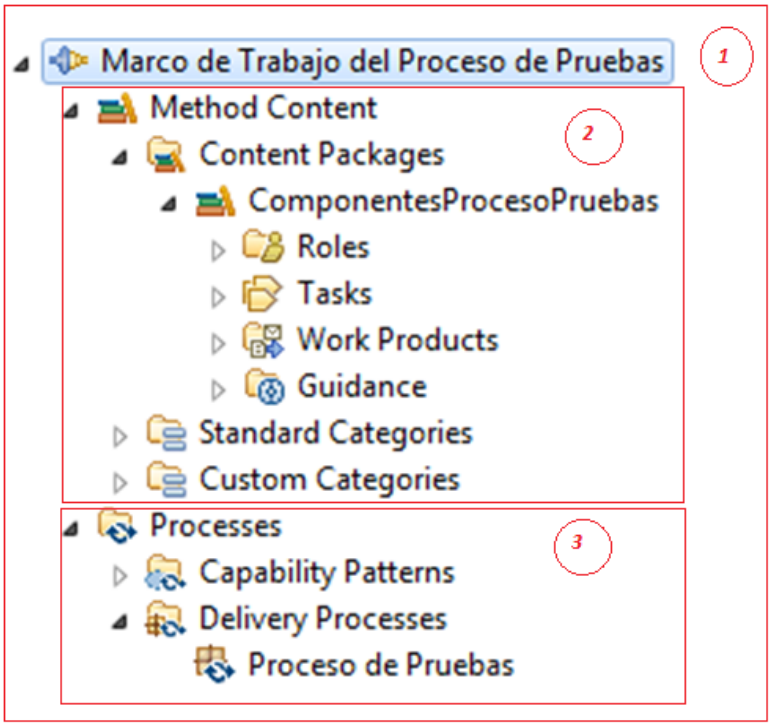
3.5.1 Descripción del modelo

EPFComposer permite representar el proceso de pruebas a partir de tres componentes: roles, productos de trabajo y tareas (ver Figura 12, ítem 2 Componentes proceso de pruebas). Para dar forma a todo esto a través de la herramienta se creó un marco de trabajo (ver Figura 12, ítem 1 Marco de trabajo del proceso de pruebas) que permite:

1. Crear el método que contiene los elementos que componen el proceso (Roles, Tareas (*Tasks*) y Productos de trabajo (*Work Products*)) y

2. Combinar los elementos creados en el método (ComponentesProcesoPruebas) para obtener el proceso (ver Figura 12, ítem 3 Proceso de pruebas).

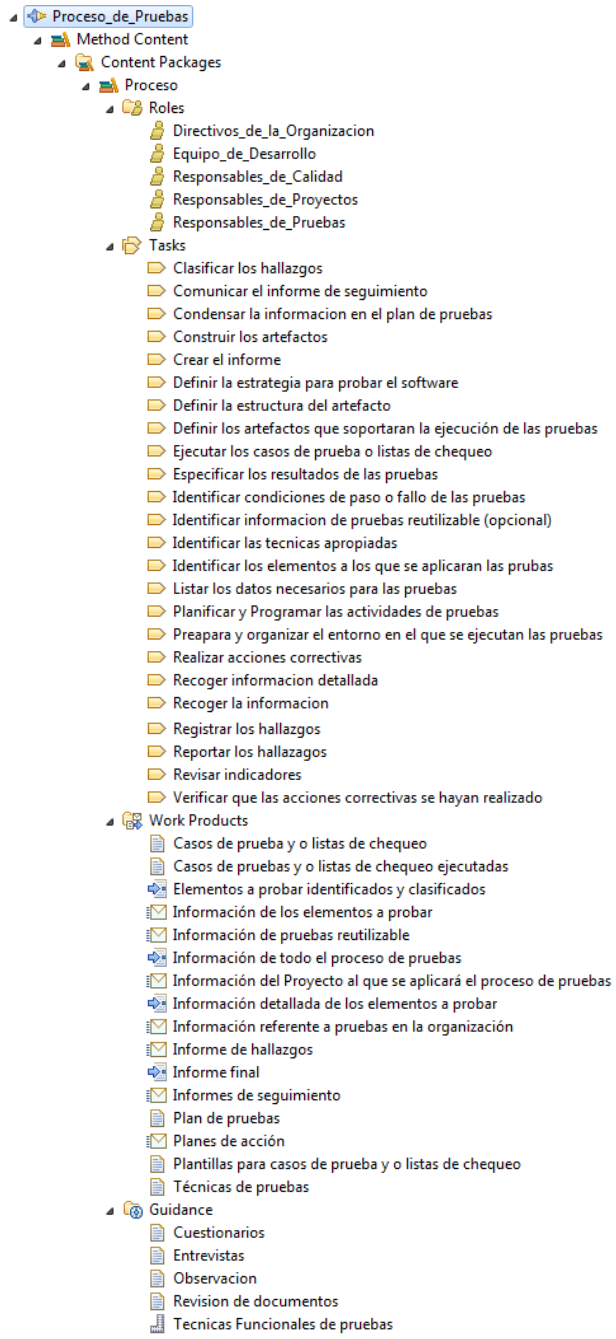
Figura 12. Marco de trabajo del proceso de pruebas



De la anterior figura, las tareas ubicadas dentro de los componentes del proceso representan lo que debe hacerse, los roles representan quien lo hace y los productos de trabajo representan las entradas para ejecutar las tareas y las salidas que estas producen.

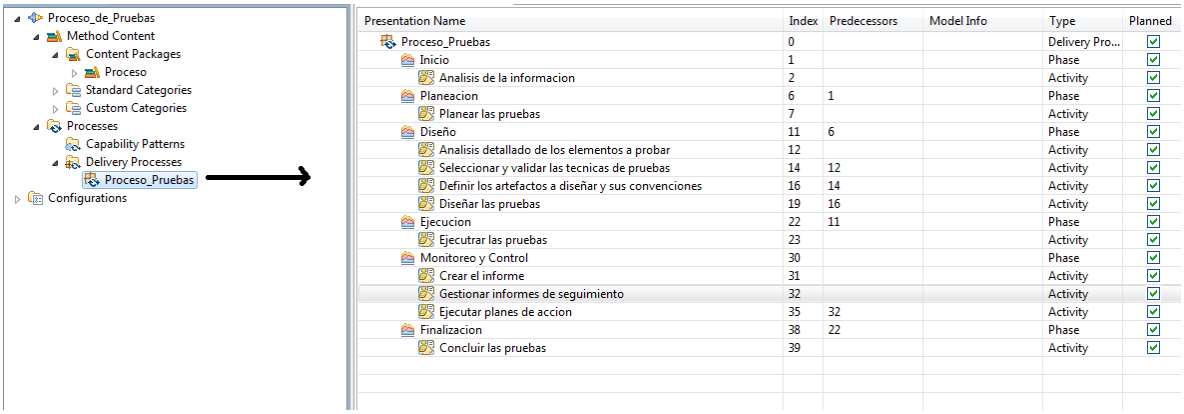
Las tareas, los roles y los productos de trabajo, son los mismos descritos en el numeral 3.2 Descripción del proceso de pruebas (ver Figura 13).

Figura 13. Tareas, roles y productos de trabajo



Una vez poblado el *Method Content* (componentes del proceso de pruebas) con los Content Elements (elementos de contenido), es decir, los elementos básicos, se combinan para obtener el proceso de pruebas, estructurado a través de un (Work BreakDown Structure, WBS). En la figura 5 se puede observar la estructura de desglose definida en el proceso de pruebas.

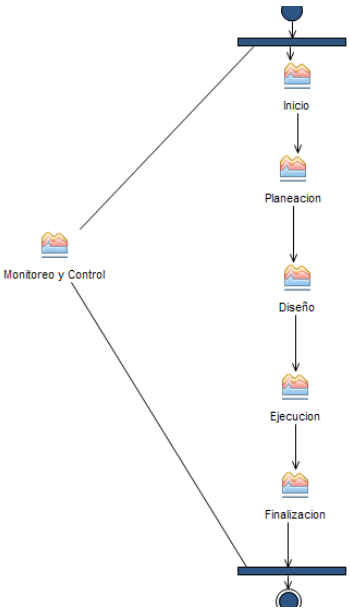
Figura 14. Estructura de desglose



Presentation Name	Index	Predecessors	Model Info	Type	Planned
Proceso_Pruebas	0			Delivery Pro...	☑
Inicio	1			Phase	☑
Analisis de la informacion	2			Activity	☑
Planeacion	6	1		Phase	☑
Planear las pruebas	7			Activity	☑
Diseño	11	6		Phase	☑
Analisis detallado de los elementos a probar	12			Activity	☑
Seleccionar y validar las tecnicas de pruebas	14	12		Activity	☑
Definir los artefactos a diseñar y sus convenciones	16	14		Activity	☑
Diseñar las pruebas	19	16		Activity	☑
Ejecucion	22	11		Phase	☑
Ejecutar las pruebas	23			Activity	☑
Monitoreo y Control	30			Phase	☑
Crear el informe	31			Activity	☑
Gestionar informes de seguimiento	32			Activity	☑
Ejecutar planes de accion	35	32		Activity	☑
Finalizacion	38	22		Phase	☑
Concluir las pruebas	39			Activity	☑

A partir del modelado, se generaron diferentes diagramas. El diagrama general del proceso se observa en la Figura 15 los demás diagramas generados pueden verse en el Anexo S.

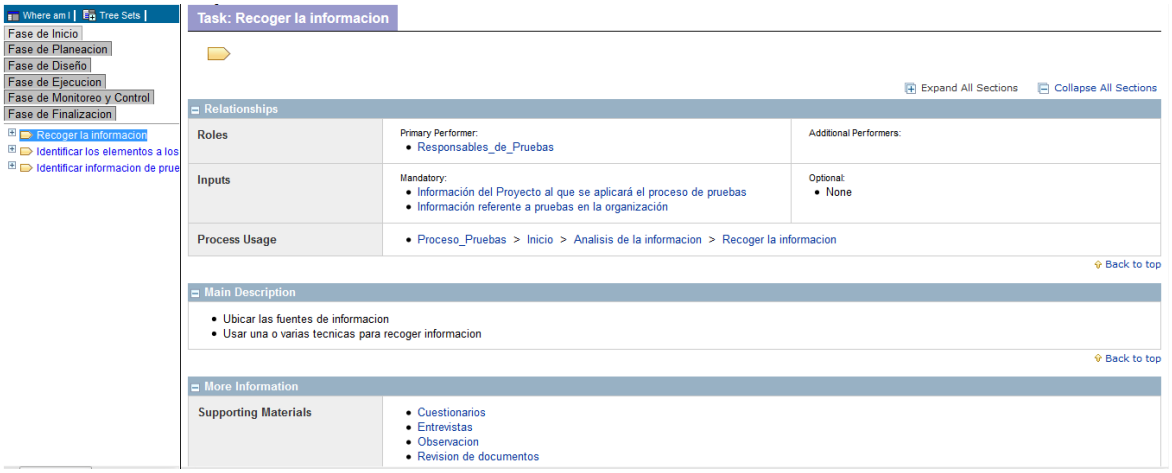
Figura 15. *Diagrama general del proceso*



Finalmente se publica el proceso modelado teniendo en cuenta la estructura de las fases:

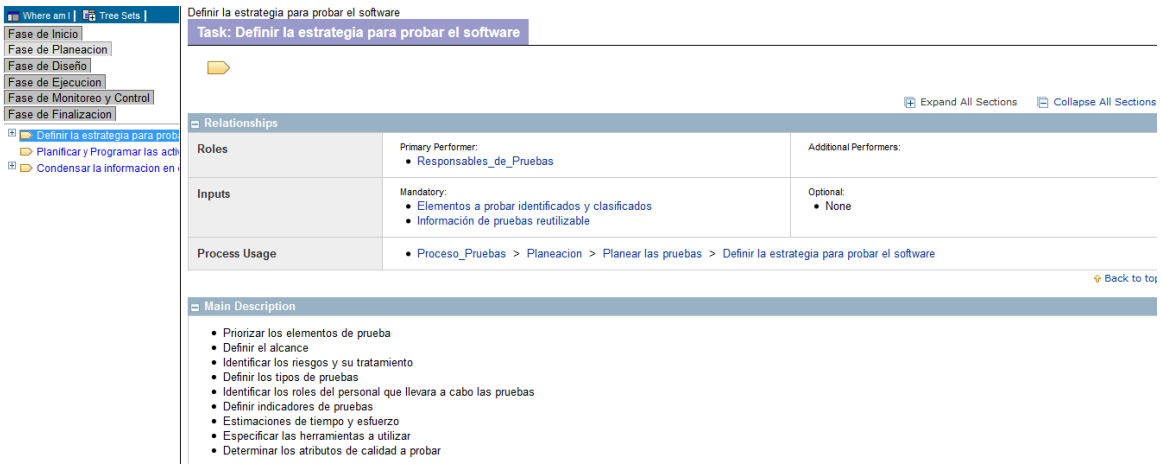
El proceso inicialmente muestra la Fase de Inicio:

Figura 16. Proceso modelado en EPF y publicado



El lector se puede desplazar a través de las fases, observando su contenido:

Figura 17. Fase de planeación en el proceso modelado



Ver todas las fases publicadas en el Anexo C.

3.6 HERRAMIENTAS PARA SOPORTAR EL PROCESO

Como se ha indicado a lo largo de este proyecto de investigación, las pequeñas organizaciones se enfrentan a diversos problemas con la calidad de los productos *software* que desarrollan. A pesar de eso, aunque existen numerosas herramientas para automatizar la aplicación de las pruebas desde cualquier punto de vista planificación, diseño, ejecución y monitoreo, estas se continúan realizando manualmente. En un estudio hecho en [73] se afirma que en la mayoría de propuestas encontradas no se cuenta con un cuerpo de conocimiento que guarde los datos sobre la efectividad, nivel de cobertura y replicación de los casos de prueba, ya que estos se registran en formatos poco estructurados, lo que impide que la evaluación del conjunto se sistematice.

De acuerdo a [54] las herramientas para pruebas ofrecen una serie de facilidades y su uso reduce de manera significativa los costos de las pruebas. De acuerdo a esto y a la realimentación obtenida después de aplicar el proceso, se vio la necesidad de apoyar las actividades del mismo mediante herramientas *software*. En este sentido el proceso puede ser soportado por diferentes herramientas con el fin de apoyar una mejor ejecución del mismo, a continuación se presentan algunos ejemplos de herramientas de libre aplicación que han sido utilizadas con éxito en la implantación del proceso de pruebas que aquí se presenta:

- Las fases de Planeación y Monitoreo y control pueden ser apoyadas por herramientas que brinden soporte en la gestión de las actividades de pruebas. Ejemplo de esto es Libre Plan.

  - ✓ **Libre plan,** es una herramienta colaborativa que permite planificar, monitorizar y controlar varios proyectos simultáneamente, mediante una interfaz *web*. Esta herramienta fue creada pensando en escenarios reales de las empresas en los que múltiples recursos (que participan en varios proyectos a la vez) y proyectos interactúan para llevar a cabo el trabajo diario de la organización*. LibrePlan es open source* mediante AGPL y puede ser descargado, instalado y adaptado a las necesidades de la organización sin tener que pagar ningún coste de licencia, proporcionando un amplio abanico de servicios *web* para importar y exportar información.

- La Fase de Ejecución puede ser apoyada por herramientas que ayuden a sistematizar el seguimiento de hallazgos, Ejemplo de esto es:

  - ✓ **Mantis**, es un sistema de libre aplicación para el seguimiento de errores. Está escrito en el lenguaje de programación PHP y trabaja con *MySQL, MS SQL* y bases de datos *PostgreSQL* y un servidor *web*. *MantisBT* se ha instalado en *Windows*, *Linux, Mac OS, OS / 2*, y otros. Casi cualquier navegador web debe ser capaz de funcionar como un cliente. Se distribuye bajo los términos de la Licencia Pública General de *GNU (GPL).* Esta herramienta permite crear Proyectos y subdivisiones del mismo y asociar a cada uno de ellos usuarios con diferentes roles y permisos. Permite configurar un formulario para el reporte de hallazgos de acuerdo a las necesidades de la organización. Esta herramienta ha sido aplicada para la actividad del proceso

- La Fase de Diseño puede ser apoyada por herramientas que soporten el diseño de casos de prueba y que brinden ayuda en la optimización de los mismos. Ejemplo de esto son:

  - ✓ **TestLink**: Licenciado bajo las condiciones de la licencia GNU GPL, para el mantenimiento del repositorio de casos de prueba y el seguimiento y control de la ejecución de planes de prueba. Permite mantener una visibilidad objetiva del estado del proceso de pruebas.

  - ✓ **ALLPAIRS** (Test Case Generation Tool): Esta herramienta retorna las posibles combinaciones que pueden resultar de un conjunto de valores de entrada. Esta herramienta es de código libre y abierto y se ofrecen bajo la licencia GPL 2.0.

Estas herramientas son un apoyo importante al proceso, pues proporcionan información útil a las personas que intervienen en él, permitiendo tomar decisiones y ejercer acciones sobre los proyectos no solo en lo que respecta a pruebas si no que también puede ayudar desde otros aspectos (desarrollo, diseño).

Capítulo 4

4. APLICACIÓN DEL CASO DE ESTUDIO

En este trabajo de grado se ha utilizado el método de casos de estudio para conducir la aplicación del proceso de pruebas en el contexto real de una pequeña empresa que desarrolla *software* llamada *Nexura Internacional S.A.S*. Este caso de estudio ha seguido la plantilla protocolo para planeación de casos de estudio descritos en [78]. A continuación se describe el caso de estudio en términos de: antecedentes, diseño, sujetos de investigación y unidad de análisis, procedimiento de campo, recolección de datos, intervención y análisis.

4.1 ANTECEDENTES

Nexura Internacional S.A.S., es una empresa de desarrollo *software* que comenzó su recorrido en noviembre de 2001 como pequeña empresa con una nómina inicial de dos personas en la ciudad de Cali. Desde su inicio centró su actividad en la producción y mantenimiento de sistemas de *software*; iniciando una trayectoria de desarrollo empresarial en el que progresivamente se ha ido embarcando en proyectos de mayor complejidad técnica y volumen, especializándose en ofrecer servicios y soluciones de *Gobierno en Línea*, los cuales se encuentran desarrollados e instalados en más de cien entidades en Colombia. La incursión de esta empresa en el mercado ha sido a través del desarrollo de productos software en grandes proyectos del sector gobierno, sin embargo la experiencia ganada prestando este tipo de productos, la ha llevado a extender su campo de aplicación y ofrecer productos como: Portales Liferay, Portales sobre PHP, Portales Sharepoint, Portales Semánticos, Email de alta capacidad, Hosting de misión crítica, Recaudo del Impuesto de Registro y Proyectos en Consorcio o UT e Interventorías [76].

En enero de 2009 sus propietarios consideraron apropiado abrir una sede en la ciudad de Bogotá (Colombia), pues su amplia experiencia en portales de *Gobierno en Línea*, los hacía altamente competitivos para licitar proyectos en organizaciones del sector gobierno. A partir de ese año *Nexura* se divide en dos sedes: una ubicada en la ciudad de Cali (Colombia), donde se encuentra el componente de desarrollo y la otra ubicada en la ciudad de Bogotá (Colombia), con el componente de entrega y mercadeo. La nómina de la empresa supera las 50 personas (de las cuales 15 están involucrados en proyectos de desarrollo y mantenimiento), la empresa cuenta con amplias instalaciones, una sala de reuniones, un set de oficinas para los administrativos, otro para los profesionales encargados del servicio al cliente quienes ofrecen los primeros servicios de comunicaciones y las oficinas de los encargados de proyectos (desarrolladores, líderes de proyectos, y actualmente un ingeniero de pruebas). Tras este crecimiento se adopta una estructura organizativa con cuatro unidades de negocio que permite, ser más ágiles en las demandas de los clientes y el mercado (Gestión estratégica, Gestión de calidad, Gestión administrativa y Gestión humana). Esto simplifica los procesos internos, la organización de los recursos, los esfuerzos, así como focaliza el centro de atención en las necesidades y particularidades del cliente, producto o servicio.

Actualmente, *Nexura* está consolidada como la empresa TIC con el mayor número de implementaciones de portales de Gobierno en Línea en Colombia. De esta forma ofrece a sus clientes la solvencia y capacidades propias de una gran consultora internacional, combinadas con la confianza y cercanía de una empresa local. Para lo que ha tenido que incrementar su productividad de tal forma que pueda mantenerse en el mercado actual. Este crecimiento la ha obligado a analizar los procesos utilizados a lo largo de su historia, mejorarlos o corregirlos cuando ha sido necesario, e incorporar nuevos procesos que son indispensables en el desarrollo de *software* y que aun no se tenían formalmente establecidos. Ejemplo de esto es el proceso de pruebas.

De otro lado, la pregunta de investigación principal (PP) y las preguntas de investigación adicionales (PA) que dirigen el caso de estudio, se encuentran en (ver Tabla 10)

Tabla 10. Preguntas de investigación del caso de estudio

Preguntas de Investigación (Principal y Adicionales)	
PP	¿Es el proceso de pruebas definido en este trabajo de grado, el adecuado para llevar a cabo las pruebas en una pequeña empresa?
PA1	¿Las técnicas sugeridas son las apropiadas para el uso de pequeñas organizaciones?
PA2	¿El cómo llevar a cabo algunas actividades de pruebas, le da un valor agregado al proceso de pruebas y a la calidad de los productos desarrollados?

4.2 DISEÑO

Tomando en cuenta el enfoque presentado en [79], el tipo de diseño del caso de estudio para este trabajo es simple holístico, ya que el proceso de pruebas ha sido aplicado en el contexto de una pequeña empresa desarrolladora de *software* utilizando como unidad de análisis uno de sus proyectos de desarrollo. El objeto de estudio es el proceso de pruebas desarrollado el cual se ha aplicado en el proyecto: “Portal para la Superintendencia Financiera de Colombia”. Por otro lado, las medidas usadas para indagar sobre las preguntas de investigación son: (i) el esfuerzo de las actividades realizadas para llevar a cabo las tareas de pruebas seleccionadas por la empresa (estas actividades están descritas en el proceso de pruebas Nexura ver Anexo M.1), y (ii) el número de errores encontrados antes y después de la ejecución del proceso de pruebas.

4.3 SUJETOS DE INVESTIGACIÓN Y UNIDAD DE ANÁLISIS

El criterio para la selección del caso de estudio fue: una pequeña empresa comprometida con la realización de un proceso de pruebas en uno de sus proyectos en un lapso de al menos seis meses. Así que para este caso de estudio se trabajó con la empresa Nexura

Internacional S.A.S., ubicada en la ciudad de Cali (Colombia), aplicando el proceso de pruebas definido en este trabajo de grado en el portal desarrollado para la Superintendencia Financiera de Colombia. Durante la actividad de asignación del proyecto piloto descrita en el procedimiento de campo que rige las actividades del caso de estudio (Ver Figura 18), se consensuó que para llevar a cabo el primer ciclo de pruebas, el responsable del proceso debía tener en cuenta las siguientes directrices:

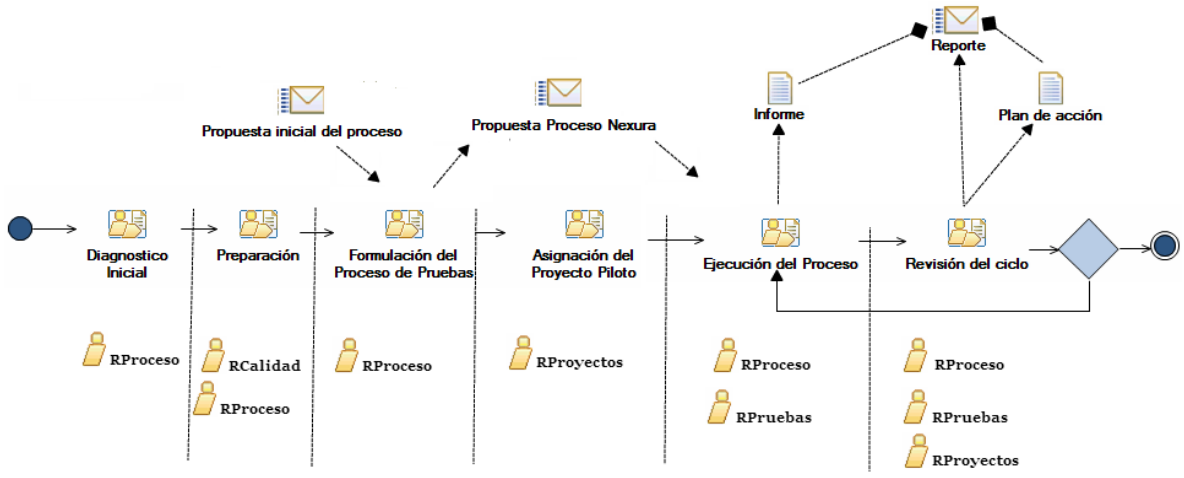
- Seleccionar la etapa crítica del proyecto escogido, con el fin de alinearse a las necesidades inmediatas de la organización. Es decir la primera fase a abordar serian los módulos desarrollados del proyecto "Portal para la Superintendencia Financiera de Colombia".

- Utilizar la información de las plantillas sugeridas por el proceso de pruebas general, para mejorar las existentes sin que esto diera lugar a re-procesos en la generación de información con respecto a pruebas que se tenía a la fecha.

4.4 PROCEDIMIENTO DE CAMPO

Es fundamental y de vital importancia el procedimiento de campo que se siga en el caso de estudio para implantar y ejecutar el proceso de pruebas propuesto en este trabajo de grado, de allí que es necesario analizar el contexto de la organización y adaptar el proceso a sus prácticas para obtener el éxito deseado. El procedimiento que rige las actividades de campo del caso de estudio se encuentra en (ver Figura 18)

Figura 18. Procedimiento de campo que rige las actividades del caso de estudio



En el procedimiento para llevar a cabo el proceso de pruebas en la organización hay 4 roles: Responsable del proceso (RProceso), Responsable de calidad (RCalidad), Responsable de proyectos (RProyectos) y Responsable de pruebas (RPruebas). El procedimiento consta de 7 actividades: Diagnostico inicial, Preparación, Formulación del proceso de pruebas, asignación del proyecto piloto, ejecución del proceso y Revisión del ciclo (las actividades de Ejecución del proceso y revisión del ciclo forman la iteración del

ciclo de pruebas). Los productos de trabajo son: la propuesta del proceso Nexura y el reporte (formado por el informe de ciclo de pruebas y el plan de acción). El informe se obtiene de la ejecución del proceso, por tanto se nutre de todos sus productos de trabajo. Estos productos de trabajo se encuentran en el anexo M.

4.5 INTERVENCIÓN

Se evaluó la situación actual de la organización con respecto a las prácticas de pruebas, modificando intencionalmente algunas de ellas con el fin de reducir la no calidad de los productos desarrollados a partir de la institucionalización de un proceso de pruebas con actividades, técnicas, productos de trabajo y roles bien definidos. Inicialmente se analizó la información recogida de archivos y diálogos establecidos con los participantes de la organización, lo que permitió juzgar mejor lo que estaba sucediendo. Mientras esto pasaba se preparaba la organización para acoger un proceso de pruebas. A partir de esto, se formuló el proceso tomando como base el proceso de pruebas general definido en este trabajo de grado, se asignó un proyecto piloto en el cual se ejecuto el proceso y finalmente se hicieron revisiones por ciclo de pruebas, las cuales permitieron tomar acciones frente a las fallas evidenciadas.

A continuación se presenta de manera detallada cada una de las actividades establecidas en el procedimiento de campo del caso de estudio.

4.5.1 Diagnostico Inicial

Se realizó una valoración inicial que permitió, primero, conocer los procesos generales de la empresa a fin de comprender su actividad, interpretarla, analizarla y abstraer la información relevante al proceso de pruebas, y segundo, estar al tanto del estado de las prácticas de pruebas llevadas a cabo en la organización, al mismo tiempo que se conoció la percepción de los individuos frente a las mismas. Con esto, se logró visualizar la capacidad de adherencia a las prácticas del proceso de pruebas definido en este trabajo de grado, y determinar con base en ello, un punto de partida para el plan de fortalecimiento de prácticas y/o definición de un proceso de pruebas en Nexura.

La empresa cuenta con un mapa de procesos establecidos desde diferentes puntos de vista: Gestión administrativa, Gestión de Calidad, Gestión de Mercadeo, Gestión de Proyectos, Gestión de Servicios, Gestión Estratégica, Gestión Humana (ver Figura 19)

Figura 19. Mapa de Procesos *Nexura*

Fuente http://www.nexura.com/calidad/publicaciones.php?id=37040

Como se observa en la Figura 19, Nexura tiene establecido un mapa de procesos en el cual se plasman las formas de operar de la organización:

1. El objetivo del proceso de Gestión administrativa es brindar soporte desde la administración en lo referente a adquisiciones, infraestructura, ambiente laboral y ambiente financiero a la organización (Elaboración de presupuestos de gastos mensuales, Registros y movimientos en el programa contable, Programación y liquidación de servicios por concepto de mensajería y aseo, manejo de proveedores, cartera, entre otros).

2. El objetivo del proceso de Gestión de Calidad es gestionar la mejora continua de la calidad en los procesos de la organización. El proceso de gestión de calidad inicia con la identificación de las necesidades de calidad de cada proceso y finaliza con el seguimiento a las acciones de mejora (acciones correctivas y preventivas, control de producto y/o servicio, control de registros, control de documentos y auditorías internas de calidad).

3. El proceso de Gestión de Innovación promueve innovaciones en *Nexura* con el fin de establecerla como una cultura dentro de las actividades y responsabilidades de cada colaborador, con el ánimo de potencializar sus capacidades intelectuales y su creatividad.

4. El proceso de Gestión de Mercadeo es el encargado de entender las necesidades del mercado para detectar las oportunidades latentes y conseguir las contrataciones.

5. El proceso de Gestión de Proyectos es el encargado de garantizar la correcta planeación, ejecución y entrega de los proyectos, en el tiempo adecuado y con la calidad esperada de acuerdo al presupuesto asignado, este proceso inicia con la percepción del contrato aprobado por el cliente y finaliza con el acta de cierre del proyecto firmada por las partes.

6. En el proceso de Gestión de Servicios se gestionan los niveles de satisfacción, fortaleciendo la comunicación directa con sus clientes y garantizando la solución de inquietudes oportunamente, este proceso inicia con la recepción de las necesidades del cliente y termina con la implementación de acciones de mejora.

7. El proceso de Gestión Estratégica garantiza la efectiva planeación estratégica y direccionamiento de la empresa, este proceso inicia con la planeación estratégica y finaliza con la toma de acciones correctivas y preventivas.

8. El proceso de Gestión Humana es el encargado de brindar un talento humano calificado que apoye todos los procesos de la organización. Este proceso inicia con el reclutamiento de personal y finaliza con la toma de acciones correctivas y preventivas.

De los procesos mencionados, se fijó la atención sobre gestión de calidad y gestión de proyectos, la caracterización de estos procesos se puede apreciar en detalle en los Anexos E y F respectivamente. En el proceso de Gestión de Calidad no se evidenció un grupo especializado en probar los productos software y tampoco un plan de calidad que integrara prácticas de pruebas, validación, verificación, auditoría y revisiones sobre los artefactos *software* (de la línea base) y procesos *software*. Bajo esta consideración no se podía analizar la ejecución de dicho plan con el objetivo de establecer acciones correctivas y de mejora basadas en datos tomados desde los proyectos.

En cambio, los productos de pruebas se ubicaban en el proceso de Gestión de Proyectos, (ver Figura 20) se señala exactamente el punto en el que se hace referencia a los productos generados tras efectuar la actividad de control de Calidad.

Figura 20. Actividad de control de calidad en el proceso de Gestión de proyectos

Proveedor	Entradas	PHVA	Actividades	Responsable Ejecución	Salida	Cliente
Gestión de Mercadeo	- Contrato - Presupuesto - Project charter	P	Planificar el proyecto	Director del proyecto	- Project charter - Cronograma de trabajo y presupuesto - Plan de proyecto	Equipo del proyecto
- Director del proyecto - Cliente	- Plan de proyecto - Cronograma de trabajo y presupuesto	P	Requerimientos y diseño	Profesional de requerimientos	- Documentos de requerimientos y diseño - Plan de proyecto actualizado	Profesional de desarrollo
- Profesional de requerimientos - Proveedores externos	- Plan de proyecto - Cronograma de actividades - Presupuesto - Documentos de requerimientos, diseño e instructivos	H	Desarrollo de la solución	Profesional de desarrollo	- Documentos de requerimientos y diseño actualizados - Entregables del proyecto	- Cliente - Profesional de control de calidad
- Profesional de desarrollo	- Plan de proyecto - Documento de requerimientos y diseño - Entregables del proyecto	H	Control de calidad	Profesional de control de calidad	- Planes de pruebas - Informe de reporte de errores - Entregables revisados	Profesional de entrega y capacitación
Profesional de control de calidad	- Plan de proyecto - Entregables del proyecto - Documentos de requerimientos y diseño - Contrato - Entregables revisados por control de calidad	H	Entrega y capacitación	Profesional de entrega y capacitación	- Manuales - Actas de aceptación de los entregables - Actas de reunión	- Cliente - Servicio al cliente - Gestión administrativa

Fuente http://nexura.com/calidad/documentos.php?id=118

Como es de notar se llevaban a cabo prácticas básicas, definidas específicamente para las pruebas del producto, para lo que se realizaban planes de pruebas, informes de reporte de errores y entregables revisados (estos productos cuentan con plantillas auto-contenidas de las cuales se muestra un ejemplo en los anexos G, H, I), pero, ¿Se estaban haciendo de forma adecuada, garantizando la calidad del producto? De indagar en la empresa sobre esta pregunta se dedujo lo siguiente:

1. No se hacia la distinción entre plan de pruebas y diseño de pruebas, el plan de pruebas era reemplazado por formatos en los que se diseñaban casos de pruebas, por tanto, las pruebas de software iniciaban sin planificar. En [72] se hace énfasis en la necesidad de planificar las pruebas “antes de desarrollar y ejecutar las pruebas se tiene que llevar un proceso estratégico, es decir la forma en cómo se organizará todo el desarrollo de las pruebas, las formas de ejecución, los recursos que se necesitarán, los tiempos así como de todas las herramientas y técnicas que se necesiten para el adecuado proceso”. Por esta razón, se tuvo que rediseñar el plan de pruebas e informar a los involucrados de la diferencia entre planificar y diseñar pruebas. La planificación sería el medio que permitiera mejorar la toma de decisiones para lograr el fin buscado tras el diseño y ejecución de las pruebas.

2. El reporte de errores era adecuado, pero, no establecía una priorización para la solución de incidencias, se dejaba al libre albedrio quienes tenían que solucionarlos, la implementación de dichas soluciones. Además, los desarrolladores expresaron su inconformismo con la forma como se les reportaban los errores, ya que no se usaba siempre el mismo documento para reporte de errores, se tenían diferentes versiones de este documento, unas más actualizadas que otras, lo que les retrasaba el trabajo, porque en muchas ocasiones se repetían los errores y tenían que disponer de un tiempo para contestar correos y aclarar que ya estaban solucionados.

3. La Guía de pruebas parciales era desconocida por los implicados en los proyectos, por tanto no se utilizaba y no generaba ningún valor agregado a la calidad del producto.

Tras la revisión de estos tres productos, se evidenció que la dificultad para organizar, planificar y ejecutar todo lo referente a pruebas, era uno de los problemas más latentes en esta organización. Para precisar aun más la información obtenida, se realizaron entrevistas a personas con conocimientos relevantes sobre el proceso de Gestión de Proyectos, líderes de proyectos y desarrolladores, obteniendo de estas informaciones más real sobre las prácticas de pruebas en la organización. De estas entrevistas se obtuvo la siguiente información:

- Los encargados de realizar las pruebas de *software* eran los mismos desarrolladores, quienes contaban con pocos fundamentos teóricos y total desconocimiento de la existencia de procesos de pruebas y técnicas de pruebas formales. Las pruebas por tanto se aplicaban de forma empírica de acuerdo a la intuición de cada desarrollador y como una fase más del ciclo de vida del producto, posterior a la finalización de cada desarrollo. Se registraban los casos de prueba en formatos poco estructurados y se reportaban los errores utilizando herramientas colaborativas como google docs, pero no en todos los casos se hacía uso estricto del formato, pues en diferentes ocasiones se quedaba corto a la hora de incluir imágenes o pantallazos del error. Por tal razón llegaban todo tipo de documentos, generando un desorden en el reporte de errores e inconformidad por parte de los desarrolladores.

- No se hacia un seguimiento estricto del error hasta su solución. Una vez que cada proyecto se daba por concluido, la información del error se quedaba olvidada o en el peor de los casos se eliminaba. De lo anterior se pudo inferir que no se contaba con repositorios de información referentes a incidente, y la información referente a pruebas era muy variable, no conservaba una estructura ordenada ya que cada desarrollador o líder del proyecto hacia lo referente a pruebas como mejor lo creía conveniente.

Nexura por tanto al igual que muchas empresas del sector productivo informático en Colombia, enfrentaba grandes problemas como: i) El desconocimiento de la importancia que tiene el proceso de pruebas sobre la calidad del producto y ii) La construcción de software de forma artesanal, empírica y caótica; originándose de esto mala calidad en sus productos, tiempos de desarrollo inadecuados (re-trabajo por falta de calidad), los costos de producción mal estimados y por tanto pérdidas económicas, las actividades de operación y mantenimiento del software difíciles y desde luego, la insatisfacción de los clientes y usuarios finales.

4.5.1.1 Preparación

Inicialmente fue fundamental que la empresa entrara en una cultura de la calidad, donde las pruebas se vieran como un medio para alcanzarla, pues, la calidad no solo dependía del responsable del proceso de pruebas, no era su responsabilidad solucionar los incidentes encontrados (esto se salía del objetivo real de un proceso de pruebas. La verdadera responsabilidad del encargado de las pruebas radicaba en notificar los

incidentes, y proporcionar la información necesaria para que el analista, desarrollador, arquitecto o a quien correspondiera solucionar la incidencia, efectivamente le agregara calidad al producto solucionando los problemas reportados. Para esto se hicieron charlas con el equipo de desarrollo, haciendo énfasis en la importancia de la comunicación entre los miembros del equipo de tal manera que se viera al responsable de pruebas como parte del equipo de trabajo, se incentivo a aportar ideas en pro de mejorar continuamente el proceso de pruebas en la organización, lo anterior siguiendo lo que en [10] se afirma: Definir un proceso de pruebas dentro de una organización puede ser muy valioso en la medida en que se incremente la calidad del producto, se facilite la comprensión y comunicación entre los miembros del equipo, se dé el soporte necesario para mejorar continuamente el proceso y se proporcione soporte a la ejecución automática de ciertas tareas.

4.5.1.2 Formulación del Proceso

Fue necesario adaptar el proceso de pruebas general propuesto al contexto de la empresa con el fin de que se pudiera aplicar de manera adecuada y fácil en los proyectos de desarrollo de la empresa. Esto bajo la premisa de que ninguna organización seguiría un proceso que le implicara solamente llenar documentación y que no le aportara ningún valor agregado.

Cabe notar que los procesos han sufrido cambios después tras las ejecuciones de los ciclos de pruebas, tanto el proceso general definido en este trabajo de grado como el proceso adaptado a los intereses de la organización.

El proceso que surge como resultado de la adaptación del proceso de pruebas genérico a las formas de trabajo de la organización se muestra en el Anexo M, que contiene los Productos del Procedimiento de Campo. Como se muestra el proceso de pruebas Nexura acogió un 100% de las actividades del proceso de pruebas general, como actividades propias del proceso. En el documento que muestra el proceso de pruebas en la organización se establecen objetivos y políticas necesarias para guiar el proceso de pruebas (ver Anexo J), el cual está sujeto a cambios, según sea necesario de acuerdo a la ejecución del proceso. La diferencia entre el proceso de pruebas definido en este trabajo de grado y el proceso acogido por Nexura está en que la organización decidió dejar explicitas algunas tareas definidas en la Actividad FEA1 Ejecutar las pruebas. En esta fase se añadieron las actividades que involucran de manera específica a los desarrolladores y al líder de desarrollo, de tal manera que formaran parte activa del proceso de pruebas.

Tabla 11. Porcentaje de actividades acogidas por el proceso de pruebas *Nexura*

Fases	Actividades del proceso General	Actividades acogidas por el proceso de pruebas Nexura	Porcentaje de actividades acogidas por Nexura
FASE DE INICIO	Análisis de la información	X	100%
FASE DE PLANEACIÓN	Planear las Pruebas	X	
FASE DE DISEÑO	Análisis detallado de los elementos a probar	X	
	Seleccionar y validar las técnicas de pruebas	X	
	Definir los artefactos a diseñar y sus convenciones	X	
	Diseñar las pruebas	X	
FASE DE EJECUCIÓN	Ejecutar las pruebas	X	
FASE DE MONITOREO Y CONTROL	Gestionar informes de seguimiento	X	
	Ejecutar planes de acción	X	
FASE DE FINALIZACIÓN DE LAS PRUEBAS	Concluir las pruebas	X	

Las actividades adicionales al proceso de pruebas *Nexura* en la fase de Ejecución son:

- Se dividió la ejecución de pruebas estáticas y dinámicas ya que en la organización toman rumbos diferentes dentro de esta misma fase.
- Realizar pruebas mínimas
- Recibir el producto en versión Beta en entorno de pruebas
- Devolver versión *Beta* del producto
- Reportar hallazgos (esta es una tarea incluida en FEA1)
- Solucionar los hallazgos

En la fase de finalización se adicionó la actividad: Enviar el informe final al director del Proyecto.

La empresa acogió un 100% de los productos de trabajo sugeridos de acuerdo al proceso de pruebas general, algunos de ellos con variantes mínimas, (ver Tabla 12).

Tabla 12. Porcentaje de productos de trabajo acogidos por el proceso de pruebas *Nexura*

Productos de trabajo acogidos por el proceso de pruebas Nexura		**Porcentaje de productos de trabajo acogidos por Nexura**
Producto de Trabajo General		**100%**
Plan de Pruebas	X	
	X	
	X	
	X	
	X	
	X	
Casos de prueba y/o Listas de chequeo	X	
Informes de Hallazgos	X	
Informes de Seguimiento	X	

4.5.1.3 Asignación del proyecto piloto.

Para validar el proceso de pruebas fue necesario buscar un proyecto piloto. La alta gerencia creyó conveniente empezar por el proyecto más grande que habían adquirido hasta la fecha. Este proyecto ya llevaba avanzada gran parte del desarrollo y contaba con algunas prácticas sencillas de pruebas. El proyecto "Portal Web Superintendencia Financiera de Colombia" con fecha máxima para salir a producción el 30 de Julio de 2013, se estaba desarrollando en lenguaje de programación Java con Base de datos Oracle y soporte de porlets. El objetivo de este proyecto era reestructurar el portal que tenía en producción la Superintendencia Financiera de Colombia, dando cumplimiento a los lineamientos de gobierno en línea y a las necesidades particulares de la organización. Este proyecto de acuerdo al documento técnico se dividía en requerimientos de tipo funcional y no funcional con los cuales debía cumplir Nexura para terminar exitosamente con las metas establecidas. Los requerimientos hacían referencia a:

- Versión móvil del sitio
- Sitio en multi-idioma
- Lenguaje HTML
- Semántica
- Publicación Distribuida
- Diseños del sitio web
- Formularios
- Portlets
- Flexibilidad
- Miga de pan
- Niveles de navegabilidad
- Buscadores
- Requerimientos GEL
- Uniformidad de diseño

- Portabilidad
- Información financiera
- Participación democrática
- Información compartida
- FAQ
- Encuestas
- Institucionalidad
- Descargas
- Personalización por el usuario
- Página de inicio modificable
- Glosario
- Mapa del sitio
- Micrositio de niños
- Componentes de acceso
- Requerimientos adicionales
- Calendario
- Sindicación de contenido (RSS)
- MashUp (Aplicaciones Web Híbridas)
- Podcasting
- Streaming
- Redes Sociales
- Audiovisual
- Seguridad
- Monitoreo
- Accesibilidad
- Administración de usuarios
- Requerimientos adicionales
- Integración LDAP y
- Algunas Generalidades de la Solución

4.5.1.4 Ejecución del proceso

La ejecución del proceso se llevó a cabo por ciclos (ver Anexo T), En el primer ciclo solo se ejecutan pruebas dinámicas a través del proceso de pruebas desde la fase de diseño hasta la fase de ejecución, es decir, se diseñaron y ejecutaron pruebas sobre los módulos desarrollados, esto, porque se necesitaba hacer una entrega rápida, lo que llevaría finalmente a demostrar con resultados reales la necesidad de un proceso que acompañara el ciclo de vida de la construcción de los productos. El primer proceso definido en la organización tomó como base el primer proceso definido en este trabajo de grado (Figura 8). A partir del análisis de los resultados, se mejora tanto el proceso definido inicialmente como el proceso de la organización. Por tanto la ejecución del proceso de pruebas Nexura definitivo se lleva a cabo solo desde el segundo ciclo de pruebas.

4.5.1.5 Revisión del ciclo

Se sometía cada ciclo de prueba a revisión con el objetivo de concluir su efectividad. Tras cada revisión se generaban informes que después de ser analizados permitían establecer hallazgos sobre fallas latentes en los desarrollos, a partir de los cuales se levantaban acciones correctivas o preventivas según el caso. Las acciones levantadas pueden observarse en el Plan de acción contenido en el Anexo M, donde se muestran los registros de las mismas a lo largo de la ejecución del proceso de pruebas en la organización hasta que logro establecerse e institucionalizarse*.*

4.6 RECOLECCIÓN DE DATOS

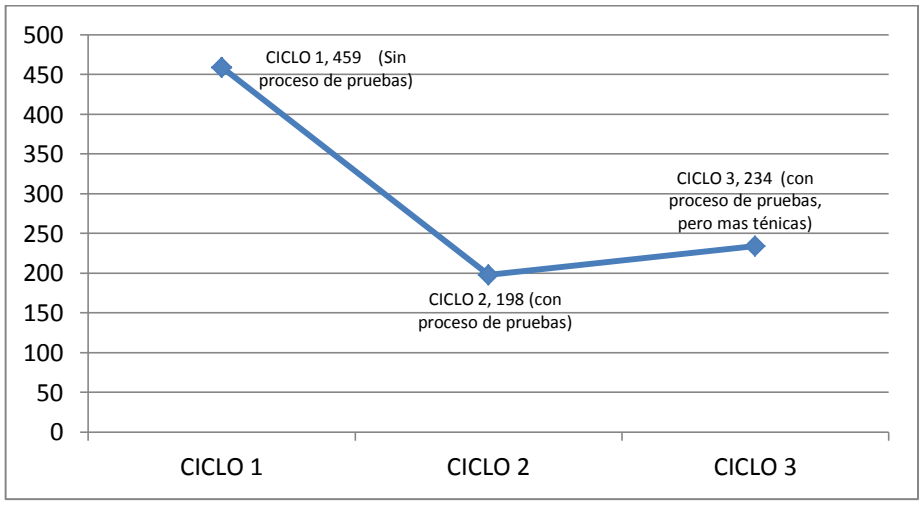
La recolección de datos se hizo mediante la utilización de las plantillas auto-contenidas de los productos de trabajo de: i) El procedimiento de campo que rige las actividades del caso de estudio (Ver Anexo M), ii) El proceso de pruebas de Nexura (Ver algunos ejemplos en el Anexo D) y iii) El cronograma de las actividades que rigieron el caso de estudio (ver Anexo T). A partir de la información registrada en los productos de trabajo mencionados, se ha consolidado: i) el esfuerzo de las actividades realizadas para llevar a cabo las tareas de pruebas seleccionadas por la empresa (ver Tabla 13), y (ii) el número de errores encontrados antes y después de la ejecución del proceso de pruebas (ver Tabla 14).

Tabla 13. Esfuerzo involucrado en cada ciclo de pruebas ejecutado en la empresa

Esfuerzo (Horas x 1 persona)														
Diagnostico, preparación, formulación del proceso, Asignación del proyecto y contextualización												**D**	**H**	**T**
												38	**9**	**343**
CICLO 1	**Actividades**	**D**	**H**	**T**	**CICLO 2**	**Actividades**	**D**	**H**	**T**	**CICLO 3**	**Actividades**	**D**	**H**	**T**
	Diseño	17	9	153		Inicio	2	9	18		Inicio	3	9	27
	Ejecución	14	9	126		Planificación	3	9	27		Planificación	2	9	18
	Informe	1	9	9		Diseño	5	9	45		Diseño	10	9	90
	Plan Acción	4	9	36		Ejecución	10	9	90		Ejecución	9	9	81
	P.Regresión	15	9	135		MonitoreoyControl	2	9	18		MonitoreoyControl	2	9	18
	TOTAL			**459**		**TOTAL**			**198**		**TOTAL**			**234**
Duración del ciclo (semanas)				**9**		**Duración del ciclo (semanas)**			**5**		**Duración del ciclo (semanas)**			**5**

D: Días H: Horas T: Total Horas

Figura 21. Grafica de esfuerzo por ciclo de pruebas ejecutado en la empresa

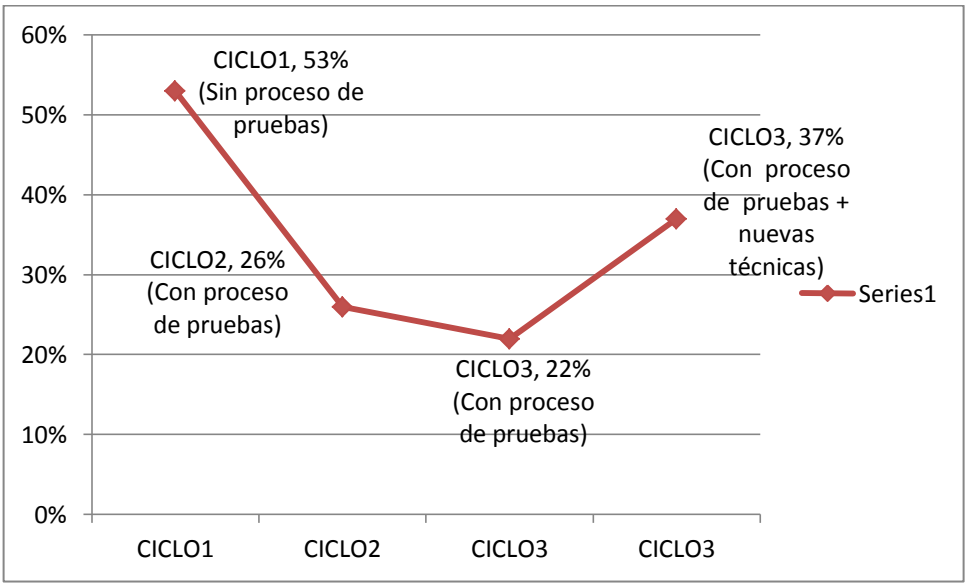


La información relacionada con el esfuerzo al llevar a cabo los ciclos de pruebas se obtuvo después de sintetizar los datos que se tenían individualmente sobre cada una de las actividades que rigieron el caso de estudio con respecto al procedimiento de campo y las actividades del proceso de pruebas de Nexura. La Fase de finalización no se ha llevado a cabo aun por qué no se ha terminado el proyecto.

Tabla 14. Porcentaje de errores encontrados tras la ejecución de cada ciclo de prueba

Ciclos	CP	ERRORES	PORCENTAJE
Antes de la ejecución del proceso de pruebas			
CICLO 1	293	156	53%
Después de la ejecución del proceso de pruebas			
CICLO 2	101	26	26%
CICLO 3	195	43	22%
Incluyendo técnicas de pruebas diferentes			
CICLO 3	270	100	37%

Figura 22. Grafico de porcentaje de errores por ciclo de prueba



La información relacionada con el porcentaje de errores antes y después de la ejecución del proceso de pruebas se obtuvo al sintetizar los datos que se tenían individualmente sobre los casos de prueba ejecutados y el número de errores de cada modulo del portal desarrollado para la *Superintendencia Financiera de Colombia*.

4.7 ANÁLISIS

De la Tabla 13 se puede extraer el esfuerzo invertido por semana en horas/persona para llevar a cabo cada ciclo de pruebas en la empresa. Como se puede observar, en el primer ciclo se ejecutaron 5 actividades (algunas de ellas no se ven como actividades propias del proceso de pruebas): Diseño, Ejecución, Informe, Plan de acción y Pruebas de regresión. La organización venía haciendo entrega de prototipos operativos de cada modulo, con el objetivo de que los funcionarios de la Superintendencia (usuarios finales) opinaran acerca del producto que iban a usar y de esta manera se refinaran los requerimientos. Cuando se asignó este proyecto al responsable del proceso de pruebas, quedaba muy poco tiempo para ejecutar dicho proceso desde la fase de inicio y con todas las actividades (se estaba a una semana de entregar un módulo), razón por la cual y como se muestra en la Tabla 13 se empezó a ejecutar el primer ciclo de pruebas desde la fase de diseño (sin planificar y sin establecer unas políticas de pruebas). El diseño y la ejecución se hicieron sobre la marcha de la entrega de los módulos, reportando de estos cifras alarmantes como se muestra en el Plan de acción contenido en el Anexo M, algunos módulos se entregaban con funcionalidades incompletas, otros superaban el 100% de errores esperados y algunos módulos ni siquiera cumplían con las funcionalidades básicas. Tras observarse esta situación, se creó una no conformidad y se alertó a los implicados quienes tomaron acciones correctivas al respecto. Inicialmente se realizó una reunión, en la cual algunos desarrolladores encontraron culpable a las pruebas realizadas de ver errores donde realmente no los había, razón por la cual se decidió realizar una clasificación de errores, está puede observarse en el Plan de acción contenido en el Anexo M, acción correctiva 2. Al realizar esta actividad, se volvió a citar a reunión y se demostró que aunque era

necesaria la clasificación, este no era el problema de fondo pues seguían siendo muchos los hallazgos. Se decidió entonces hacer un ranking de errores y buscar las causas y posterior a esto las soluciones. De los errores especificados y mostrados a los implicados, lo que más inquietaba era: i) Los desarrolladores, el líder de desarrollo, el analista y el responsable de las pruebas manejaban versiones diferentes de los casos de uso que especificaban los requerimientos, ii) El líder de desarrollo, el analista y el responsable de las pruebas entendían los requerimientos de maneras diferentes y finalmente el desarrollador no sabía qué hacer, y iii) Los desarrolladores desconocían muchas cosas del CMS base de la organización.

Argumentados en estos hallazgos entre otras soluciones (ver Anexo M), se vio la necesidad de establecer pruebas de *software* desde los requerimientos, de tal manera que se eliminaran ambigüedades y se optimizaran los desarrollos. Es a partir de esto y como se observa en el Anexo T donde empieza realmente la ejecución del proceso de pruebas.

En los ciclos 2 y 3 de pruebas, se ejecutaron 5 de las 6 fases del proceso: Inicio, Planificación, Diseño, Ejecución, y Monitoreo y control. La fase de Finalización no se ha llevado a cabo porque el proyecto no ha finalizado (Se encuentra en estado de reposo mientras se arreglan algunas cosas de fondo, que tienen que ver con la arquitectura y prácticas de programación). Al finalizar el ciclo 2 es cuando se entendió la necesidad latente de la organización de apropiar el proceso de pruebas a todos sus proyectos, lo anterior gracias a los resultados del número de errores encontrados antes de la ejecución del proceso y después como se muestra en la Tabla 14. A partir de allí se ve el proceso de pruebas como una herramienta fuerte para la toma de decisiones y acciones frente a posibles fallas desde el proceso de Gestión de Proyectos y construcción del producto, pues el acompañamiento mutuo de pruebas y construcción, permitió mejoras en los tiempos y en las entregas de los módulos (respondiendo así a la pregunta de investigación principal PP).

Basados en la premisa de que "*El objetivo de las pruebas es la detección de defectos en el software y no demostrar que el software no tiene defectos*", para el ciclo 3 de pruebas se acogieron mas técnicas de pruebas de las especificadas en el proceso general (hasta el momento solo se había aplicado dos de ellas), con el objetivo de no tender a demostrar que el software no tenia defectos. El primero hace referencia a la ejecución de los casos de prueba que existían antes de empezar el proceso (Casos de pruebas que ya tenía la organización con respecto a los módulos citados en el ciclo 3) y tras su ejecución se obtuvieron resultados muy similares a los del ciclo 2, de ahí que surgiera la iniciativa de mejorar los diseños con mas técnicas de pruebas y de esta manera descubrir que las técnicas de pruebas sugeridas en el proceso de pruebas general eran las apropiadas para una pequeña organización, pues, su diseño no implico más de ocho días para cuatro módulos, lo que demuestra que son sencillas y de fácil uso. Con lo anterior se contestan las preguntas de investigación PA1 y PA2 pues la incorporación de más técnicas al proceso de pruebas le aportó más precisión a la calidad del producto. En su lugar lo que sucedía era que no se habían encontrado más errores bajo los métodos utilizados.

En el anexo T se puede observar que gran parte del esfuerzo invertido en el ciclo uno se enfatizó en las pruebas de regresión, comprobando que los cambios sobre los módulos, no introducían más errores en otros módulos (por el número alarmante de errores

encontrados). Estas pruebas se realizaban cada vez que se hacia un cambio en el sistema, tanto para corregir un error como para realizar una mejora. Para la realización de las mismas, se incluía la repetición de los casos de pruebas realizados con anterioridad y que estaban directamente relacionados con la parte del sistema modificada.

Como resultado, se llegó a consolidar una forma de trabajo de la cual *Nexura* sacó provecho y la extendió a otros procesos. El proceso de levantamiento de requerimientos surgió tras la intervención que se hizo en el proyecto del portal de la *Superintendencia Financiera de Colombia*, ver Anexos K y L. Además de contribuir en la calidad de los procesos y en la implementación del producto, el proceso sirvió para identificar la necesidad de establecer una buena especificación de requisitos para obtener una buena calidad en el *software*.

De ahí que los beneficios de las pruebas no solo se vieron en el área de desarrollo de proyectos y en el ahorro de costos, sino que también se dejó notar en el resto de las áreas de negocio y especialmente en la satisfacción del cliente final.

Actualmente el proceso de pruebas se ha fortalecido tras la incorporación de herramientas de pruebas para las diferentes fases que lo componen. Los casos de pruebas derivados de la fase de diseño se fortalecieron con la apropiación de la herramienta allpairs, mediante esta herramienta se diseñan casos de pruebas cuando se tienen funcionalidades complejas, para lo que permite seleccionar variables con una gran cantidad de valores.

Capítulo 5

5. CONCLUSIONES

En este trabajo de grado se ha presentado un proceso de pruebas para pequeñas organizaciones, construido a partir de: i) la clasificación y análisis de los procesos de pruebas existentes, ii) la comparación de los procesos de pruebas enfocados hacia la pequeña empresa con la norma para pruebas de software ISO/IEC 29119, iii) la identificación, análisis e integración de técnicas de pruebas funcionales, iv) la identificación, observación y análisis de las pruebas de acuerdo a los procesos y formas de trabajo para el desarrollo de software que se llevan a cabo en la pequeña empresa y v) la experiencia vivida tras la aplicación de un caso de estudio en una pequeña empresa de la ciudad de Cali. De la aplicación del proceso se puede concluir que es adecuado a las pequeñas organizaciones que estén interesadas en incorporar pruebas de software desde las primeras etapas del ciclo de vida y no encuentren como hacerlo, pues las actividades del proceso, específicamente las actividades de diseño hacen de este una guía para la creación de artefactos que permiten probar el software al tiempo que se usan técnicas exitosas y utilizadas ampliamente en el área de las pruebas de software.

En esta parte final se presentan las conclusiones propias del trabajo de investigación, las conclusiones de la experiencia desarrollada en la empresa piloto y por último se ofrece un compendio de recomendaciones así como sugerencias sobre posibles actividades futuras a desarrollar

CONCLUSIONES DEL TRABAJO DE INVESTIGACIÓN

- El aporte innovador de este trabajo de grado se ve reflejado en la adecuación y adaptación de las técnicas de pruebas funcionales a las actividades de la fase de diseño del proceso definido, y el ajuste de las actividades del proceso a las revisiones, también conocidas como pruebas estáticas.

- Las tareas derivadas de las actividades del proceso definido permiten clarificar no solo el que hacer en un proyecto que requiere revisiones y pruebas funcionales de software sino también visualizar el cómo hacerlo (especialmente en el diseño de las pruebas), diferencia importante entre los procesos de pruebas encontrados en la literatura y definidos para el contexto de las pequeñas organizaciones.

- Una de las mayores barreras a las que se enfrenta la implantación de un proceso en la pequeña empresa es la importancia que le den los directivos de la organización, debido a que por lo general es reacia a invertir recursos en actividades que no muestren una ganancia económica notable a corto plazo.
- Es importante tener en cuenta que la pequeña empresa necesita procesos con la mínima documentación posible y que deriven productos de trabajo que les generen un

valor agregado. En lo posible que estos productos sean reutilizables para agilizar el proceso en los proyectos.

- Las técnicas de pruebas funcionales son fácilmente reutilizables y adaptables a los deferentes niveles de pruebas unitarias, integración y de sistema, esto es importante ya que según *Beize*r en [56] no se conoce un modelo que defina cómo se debe aplicar un nivel determinado, ni cómo se debe asumir que alguno de ellos sea más importante que el otro.

- Las organizaciones pequeñas reciben con gran aprecio la incorporación de herramientas libres en sus procesos, de ahí que es importante trabajar en investigar y adaptar las herramientas existentes a las formas de trabajo de una organización.

CONCLUSIONES DE LA EXPERIENCIA DESARROLLADA EN LA EMPRESA

- A través de esta investigación enmarcada en la Línea de Investigación Calidad del *software*: Producto y Proceso del Grupo de Investigación y Desarrollo en Ingeniería de *Software* (IDIS), se ha aportado un granito de arena a los procesos de *Nexura*, encontrando oportunidades para crecer, incluso en épocas de crisis, adaptando el conocimiento adquirido en la ingeniería del *software* a las formas de trabajo de esta empresa, y aplicando modelos, métodos, técnicas, prácticas y demás artefactos asociados a los procesos relacionados con las pruebas de software definidos por organizaciones internacionales.

- Ha sido un trabajo difícil pero se ha actuado de manera adecuada, pensando siempre en aprovechar la intención de la empresa en incorporar pruebas como un proceso, pues, aunque cada vez se era más consientes de la falta de pruebas dentro del ciclo de vida del producto *software*, no resultaba tan fácil cambiar la forma en que está organización probaba sus productos y además la falta de talento humano para ejercer esta actividad. Para esto fue fundamental la comprensión y compromiso por parte de la dirección de Nexura, a quienes se les ha demostrado que el proceso a mediano plazo supone un ahorro en tiempos de desarrollo y por tanto en dinero pero inicialmente implica una inversión económica dividida en tiempo y talento humano.

- La aplicación del proceso de pruebas fue un éxito para los sucesos y esquemas del proyecto y de la organización, gracias a esto, se ha adquirido un proyecto en convenio especial de cooperación con Fedesoft , el Sena y Green SQA, para la apropiación de prácticas innovadoras de testing y de reconocimiento internacional en la industria de SW & TI. Mediante este proyecto se busca fortalecer el proceso de pruebas formando un equipo que soporte todos los proyectos, ya que hasta ahora solo se ha contado con el apoyo de la persona responsable de ejecutar el proceso de pruebas del que aquí se ha hablado.

- Ciertamente el proceso actual de pruebas de *Software* en *Nexura* aun se encuentra en un nivel bajo de capacidad, en relación con otras empresas dedicadas al desarrollo de *software*, sin embargo en muy poco tiempo se ha logrado posicionar al proceso de pruebas en la cadena de valor del desarrollo de *software* para minimizar el riesgo de

errores o fallas antes de que los sistemas sean utilizados por los usuarios o clientes finales. Por un lado, el proceso de Gestión de Proyectos cumple su compromiso de alcance, plazo, coste y calidad, por otro, mejora la transición de la fase de desarrollo a la de puesta en producción. Cuando los proyectos llegan a la fase final, el sistema está estable, y ofrece una mejor respuesta a las necesidades funcionales y a los aspectos técnicos, lo que provoca la satisfacción de los usuarios finales y un incremento en la confianza de los responsables de la explotación del nuevo *software*.

RECOMENDACIONES Y TRABAJO FUTURO

La comunicación en un proceso de pruebas es fundamental, sobre todo cuando se quiere implantar en una organización que no ha incorporado este tipo de prácticas. Generar un buen ambiente en el equipo de trabajo es fundamental en el éxito de la apropiación de un proceso "nuevo" en una organización. Dicha comunicación va desde detalles tan pequeños como la descripción de un hallazgo o la clasificación del mismo, por tanto se recomienda que para este tipo de procesos se establezcan formas de lenguaje comunes que ayuden al éxito del proyecto y a la calidad del producto desarrollado.

Actualmente no se ha encontrado un sistema que permita monitorear todos los artefactos del flujo de trabajo necesario para realizar un óptimo proceso de pruebas, sin embargo, existen muchas herramientas de software libre que hace una tarea específica (Ej. Testlink y Mantis) con las cuales se puedan controlar o supervisa varias de las fases para dicho proceso. Como trabajo futuro se propone la adaptación e integración de algunas de estas herramientas de tal forma que en su conjunto faciliten al responsable de pruebas obtener indicadores a partir de los cuales se pueda tomar decisiones sobre el proceso y sobre la calidad del producto. Lo que implica investigación sobre las herramientas de pruebas y desarrollo.

En muy pocos artículos de la bibliografía se encuentra documentación acerca del estudio de revisiones formales, el tema fuerte son las pruebas dinámicas, esto se debe a que para que un software sea aceptable como mínimo debe cumplir con la funcionalidad, pero se ha demostrado la importancia de hacer pruebas antes de la construcción del software por esta razón es importante ahondar en este tema.

El área de técnicas de pruebas es muy amplia, por lo que se sugiere a futuros investigadores abordar estas técnicas desde el punto de vista de los procesos, ya que las pequeñas organizaciones los acogen muy bien.

El caso de estudio se aplicó para los requerimientos y para funcionalidades desarrolladas. Por esto, es importante ahondar y llevar a cabo más experiencias empíricas alrededor de su aplicabilidad y extensibilidad a los demás artefactos que componen el ciclo de vida de un software.

Capítulo 6

6. REFERENCIAS BIBLIOGRÁFICAS

[1] A. Rodrigues, P. R. Pinherio y A. Albuquerque, “The definition of a testing process to small-sized companies: the Brazilian scenario,” 2010 Seventh International Conference on the Quality of Information and Communications Technology, pp 298- 303, 2010.

[2] L. Monsalve, “Calidad de los Productos Software,” Revista ingeniería Informática, pp. 1-4, 1998.

[3] J.A Peña, N.J Ontiveros y S.A Chavarría, “Software de Calidad,” Hypatia Revista de Divulgación Científico - Tecnológica del Estado de Morelos México, pp. 1-2, 2010.

[4] S.E Rojas Salamanca y J.J Borja Parra, “Calidad del software: Camino hacia una verdadera industria del software,” Revista Escuela De Administración de Negocios, pp. 38-57, 1999.

[5] Tanja E.J Vos, J. Sánchez Sánchez y M. Mannise, “Mejorando el testeo en las Pyme ¿Cómo empezar?,” Los proceedings de las V Jornadas sobre Testeo de *Software*, pp. 71-80, 2008.

[6] A. de Rojas, Tanja E.J Vos y B. Marín, “Experiencias de una PYME en la mejora de procesos de pruebas,” REICIS Revista Española de Innovación, Calidad e Ingeniería del *software*, vol. 5, pp. 63-69, 2009.

[7] D. Karlström, P. Runeson, y S. Nordén, “A Minimal Test Practice Framework for Emerging *Software* Organizations,” *Software* Testing, Verification and Reliability, vol.15, pp. 145-166, 2005.

[8] T. Koomen, L. Van Der Aalst, Bart Broekman y M. Vroon, TMap Next: for result-driven testing, UTN Publishers, 2007, pp. 752.

[9] J. Kasurinen. “Elaborating *Software* Test Processes and Strategies,” Third International Conference on *Software* Testing, Verification and Validation, pp. 1- 4, 2010.

[10] I. Burnstein, T. Suwanassart y R. Carlson. “Developing a testing maturity model for *software* test process evaluation and improvement,” pp. 581-589, 1996.

[11] J. Cangussu, R.A Decarlo y A.P Mathur. “A State Variable Model for the *Software* Test Process,” Citeseer, pp. 10, 2000.

[12] J.S Collofello, Z. Yang, J.D Tvedt, D. Merrill y I. Rus. “Modeling Software Testing Processes,” IEEE, pp. 289-293, 1996.

[13] L. Xin-ke y Y. Xiao-hui. “A Goal-driven Measurement Model for *Software* Testing Process,” IEEE, vol. 4, pp. 8-12, 2009.

[14] J. Verdugo M. Piattini D. Mellado, M. Rodríguez y E. Fernández-Medina. “Evaluación de la Calidad y Seguridad en Productos *Software*”, pp. 1-12.

[15] E. Kit, “*Software* Testing in the Real World: Improving the Process,” ACM Press/Addison-Wesley Publishing Co , 1995.

[16] L. Huang y B. Boehm, “How Much Software Quality Investment Is Enough: A Value-Based Approach, ” Software IEEE , vol. 5, pp. 88-95, 2006

[17] H. Do and G. Rothermel, “An Empirical Study of Regression Testing Techniques Incorporating Context and Lifetime Factors and Improved Cost-Benefit Models, ” Proc. 14th ACM SIGSOFT international symposium on Foundations of *software* engineering, pp. 141-151, 2006

[18] H. J. Harrington. El coste de la mala calidad, Ediciones Díaz de Santos. 1990

[19] S. Reid, “ISO/IEC 29119 *Software* Testing”. Disponible en: http://www.*software*testingstandard.org/

[20] . ISO/IEC 12207 International Standard, Systems and *software* engineering - *Software* life cycle processes. Ref. Nr. ISO/IEC FDIS 12207:2007(E), pp. 142, 2007

[21] A. Ruiz Mendarozqueta. “CMMI ¿Evolución o nuevo modelo?,” 2002.

[22] A. Sanz y J. Saldaña. “TestPAI: Un área de proceso de pruebas integrada con CMMI,” REICIS Revista Española de Innovación, Calidad e Ingeniería del *Software*, vol. 4, pp. 6-20, diciembre 2008.

[23] P. Cruz, R. Villarroel y F. Mancilla**. “**A *Software* Testing process for the reference model of Competisoft,” IEEE, pp. 9, 2010.

[24] J. Brodman y D. Johnson, “What small business and small organizations say about the CMM: experience report, ” ICSE’94: Proceedings of the 16th international conference on *Software* engineering, pp. 331–340, 1994.

[25] M. Staples, M. Niazi, R. Jeffery, A. Abrahams, P. Byatt y R. Murphy. “An exploratory study of why organizations do not adopt CMMI, ” Journal of Systems and *Software*, vol. 80, pp. 883–895, 2007.

[26] H. Oktaba, F. Garcia, M. Piattini, F. Ruiz, F.J. Pino y C. Alquicira, “*Software* Process Improvement: The Competisoft Project, ” IEEE Computer, vol. 40, pp. 21–28, 2007.

[27] COMPETISOFT, Mejora de Procesos para Fomentar la Competitividad de la Pequeña y Mediana Industria del *Software* de Iberoamérica. Versión 2. Diciembre 2006.

[28] E, van Veenendaal, R. Hendriks, J. van de Laar y B. Bouwers. “Test Process Improvement using TMM(i), ” Testing Experience: The Magazine for Professional Testers, vol. 3, pp. 21-25, 2008.

[29] Sogeti. “TPI Test Process Improvement,” Mejora Del Proceso de Testing, 2010.

[30] S. Sánchez Melchor. "Una Revisión y Comparativa de Modelos de Procesos de Pruebas". 2010

[31] O., J. L. A. (2010). "LOS CASOS DE PRUEBA EN LA PRUEBA DEL SOFTWARE." Revista Digital Lámpsakos No. 3: 27-34.

[32] Natalia Juristo, A. M. M., Sira Vegas (2006). "TÉCNICAS DE EVALUACIÓN DE SOFTWARE."

[33] Maria Marta Sandoval Carvajal, M. A. G. V. "LA TRAZABILIDAD EN EL PROCESO DE REQUERIMIENTOS DE SOFTWARE."

[34] Cortés, O. H. G. (2004). "Aplicación práctica del diseño de pruebas de software a nivel de programación." Revista de la Facultad de Ingenieria, Universidad ICESI **2**: 83 - 128.

[35] (2012). "ISO 29119-2 Systems and Software Engineering — Software Testing — Part 2: Test process." 51.

[36] "TestPAI, Un Área de Proceso de Ingenieria de Nivel de Madurez 3." Disponible en: http://sel.inf.uc3m.es/asanz/testpai/testpai_pa_fullversion_esp.pdf

[37] HERNÁNDEZ, S. V. (2012). ESQUEMA DE CARACTERIZACIÓN PARA LA SELECCIÓN DE TÉCNICAS DE PRUEBAS DE SOFTWARE. Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software. Madrid, UNIVERSIDAD POLITÉCNICA DE MADRID**:** 424.

[38] Edgar serna Montoya, F. A. I. (2010). "Análisis crítico a las propuestas para generar casos de prueba desde los casos de uso para las pruebas funcionales." Revista Avances en Sistemas e Informática Vol. 7: 105 – 111

[39] ZAMBRANO, A. N. C. (2005). HERRAMIENTA PARA EL ANÁLISIS DE REQUERIMIENTOS DENTRO DE LA PEQUEÑA EMPRESA DESARROLLADORA DE SOFTWARE EN BOGOTÁ. FACULTAD DE INGENIERÍA - CARRERA DE INGENIERIA DE SISTEMAS. Bogotá D.C, PONTIFICIA UNIVERSIDAD JAVERIANA**:** 147.

[40][SWEBOK, 04] IEEE Computer Society Professional Practices Committee. “Guide to the Software Engineering Body of Knowledge – 2004 Version”. IEEE Computer Society, Los Alamitos (CA), USA, 2004.

[41] (2012). "ISO 29119-3 Systems and Software Engineering — Software Testing — Part 3: Test Documentation." 122.

[42] Susana Mojica Arcos, J.J.O.R., Edwin Hernando Silva Urazán *Globalización ¿Impacto positivo o negativo para América Latina?* Universidad Militar Nueva Granada.

[43] Maribel Pèrez Sànchez, J.C.R., *Cluster una apuesta al crecimiento.* Universidad Militar Nueva Granada, 2008.

[44] García, M.L.S., *Una propuesta para la determinación de la competitividad en la pyme latinoamericana.* Revista científica Pensamiento y Gestión, Jul-Dic 2012. 33: p. 32.

[45] Pinzon, C.V., *Las TIC, pieza clave para resolver problemas de competitividad nacional.* Universidad Militar Nueva Granada, 2012.

[46] Rodríguez, K. H. (2009). COLOMBIA: DESAFÍOS DE UNA INDUSTRIA EN FORMACIÓN. DESAFÍOS Y OPORTUNIDADES DE LA INDUSTRIA DEL SOFTWARE EN AMÉRICA LATINA. P. B. T. y. F. S. Marques**:** 139 - 170.

[47] Marques, P. B. T. y. F. S. (2009). AMÉRICA LATINA EN LA INDUSTRIA GLOBAL DE SOFTWARE Y SERVICIOS:UNA VISIÓN DE CONJUNTO. DESAFÍOS Y OPORTUNIDADES DE LA INDUSTRIA DEL SOFTWARE EN AMÉRICA LATINA. P. B. T. y. F. S. Marques**:** 249 - 292.

[48] FUGGETTA, Alfonso, CONRADI, Reidar. *Improving Software Process Improvement*. Dipartimento di Elettronica e Informazione. Politecnico di Milano. Julio 2.002.

[49] RUIZ G, Francisco. "*MANTIS: Definición de un Entorno para la Gestión del Mantenimiento de Software*". Tesis Doctoral. Departamento de Informática. Universidad de Castilla – La Mancha. Junio de 2.003.

[50] GOMEZ, E.P.J., PROCESO DE INTERNACIONALIZACION DE LAS PYMES COLOMBIANAS E INCIDENCIA DEL TLC CON ESTADOS UNIDOS. UNIVERSIDAD DE BARCELONA, 2007: p. 19.

[51] Francisco J. Pino, F. G., Mario Piattini, Hanna Oktaba (12 de Julio de 2006). "Revisión Sistemática de Mejora de Procesos Software en Pequeñas y Medianas Empresas de Software." CYTED (Ciencia y Técnologia para el desarrollo): 110.

[52] Fayad, M.E., M. Laitinen, and R.P. Ward, Software Engineering in the Small. Communications of the ACM, 2000. 43(3): p. 115-118.

[53] José A. Calvo-Manzano, J. G., Mario Piattini, Francisco J. Pino, Jesús Salillas, José Luis Sánchez (2008). "Perfiles del ciclo de vida del software para pequeñas empresas: los informes técnicos ISO/IEC 29110" Revista Española de Innovación, Calidad e Ingeniería del Software 4: 14.

[54] Sommerville, I. (2005). INGENIERIA DEL SOFTWARE. Madrid, PEARSON EDUCACIÓN. Séptima edición: 712.

[55] Pressman, R. (2002). Ingenieria del Software, Un Enfoque Practico, McGraw-Hill Companies.

[56] Boris Beizer, Black-Box Testing: Techniques for Functional Testing of Software and Systems, Wiley (May 1995).

[57] Tim Koomen, L. v. d. A., Bart Broekman, Michiel Vroon (2007). 14. Test desing techniques. TMap Next for result-driven testing, UTN Publishers, 's-Hertogenbosch.

[58] Sánchez, T. V. y. J. (2008). "Recetas para el Diseño de Casos de Testeo." Procedings de las IV Jornadas sobre Testeo de Software: 18.

[59] Glenford Myers. The Art of Software Testing. John Wiley and Sons, 2004.

[60] Mateo, P. R. "Calidad de Casos de Prueba." Calidad y Medición de Sistemas de Información: 19.

[61] B. Beizer, Software Testing Techniques. London: International Thompson Computer Press, 1990.

[62] (2012). "ISO 29119-4 CD-2 Systems and Software Engineering — Software Testing — Part 4: Test Techniques." 130.

[63] Torío, J. M. (2004). Estrategias de prueba de líneas de producto de sistemas de tiempo real especificados con diagramas de estados jerárquicos, Universidad Politécnica de Madrid**:** 223.

[64] Federico Leonardo Toledo, B. P. L., Macario Polo Usaola "Técnicas de prueba basadas en modelos para Procesos de Negocio." 6.

[65] Dondeti, S. N. y. J. (2012). "BLACK BOX AND WHITE BOX TESTING TECHNIQUES–ALiterature REVIEW." International Journal of Embedded Systems and Applications (IJESA) **2**: 29-50.

[66] Toledo, F. (2012). "CTweb para diseñar pruebas con técnica de Máquinas de Estado." from http://blog.abstracta.com.uy/2013/04/ctweb-para-disenar-pruebas-con-tecnica.html.

[67] Ridao, M. (2001). Uso de Patrones en el Proceso de Construcción de Escenarios, Tesis Maestría en Ingeniería de Software.

[68] M, E. S. (2011). Análisis y comparación de las propuestas recientes para diseñar casos de prueba desde los casos de uso orientados a verificar los aspectos funcionales del software. Facultad de Minas, Escuela de Sistemas, Universidad Nacional de Colombia Medellín**:** 110.

[69] McGregor, J. D., Sykes, D., (2001) A practical Guide to Testing Object-Oriented Software, Addison-Wesley, Massachussets

[70] IEEE, "IEEE Standard 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology," 1990.

[71] [DAV95] Davis, A., 201 Principles of Software Development, McGraw-Hill, 1995.

[72] Tenorio, R. R. (2010). Las pruebas de software y su importancia en las organizaciones. Facultad de Contaduría y Administración, Universidad Veracruzana, Región Xalapa**:** 104.

[73] Baudry, B., Fleurey F., Jezequel J-M. & Traon Y. L. (2002). "Automatic test cases optimization using a bacteriological adaptation model: Application to .NET components". 17th IEEE International Conference on Automated Software Engineering, ASE'02. Edinburgh, UK, pp. 253-257.

[74] Haumer, P. (2007). Eclipse process framework composer. Eclipse Foundation.

[75] Francisco Ruiz, J. V. (2008). Guía de Uso de SPEM 2 con EPF Composer Departamento de Tecnologías y Sistemas de Información - Grupo Alarcos, Universidad de Castilla-La Mancha**:** 93.

[76] ."Nexura". Disponible en: http://www.nexura.com.

[77] Francisco Oyarce Valderrama, R. V. A. (2012). "Propuesta de Mejora del Proceso de Testing en Pequeñas Empresas." III Congreso Internacional de Computación e Informática del Norte de Chile: 8.

[78] Brereton, P., B. Kitchenham, D. Budgen, and Z. Li. Using a protocol template for case study planning. 2008. Evaluation and assessment in SE. Bari, Italia. pp. 1-8.
[79] Yin, R.K., Case Study Research: Design and Methods. 2003, Newbury Park, Sage Publications.

[80] SEI, CMMi® for Development. SEI, Carnegie Mellon University, 2006.

[81] Paulk, M., Weber, C., Curtis, B. y Chrisis, M. The Capability Maturity Model. Addison-Wesley. 1995.

[82] J. Brodman and D. Johnson, What small business and small organizations say about the CMM: experience report, ICSE'94: Proceedings of the 16th international conference on Software engineering, pp. 331–340, IEEE Computer Society Press, 1994.

[83] M. Staples, M. Niazi, R. Jeffery, A. Abrahams, P. Byatt and R. Murphy, An exploratory study of why organizations do not adopt CMMI, Journal of Systems and Software, Vol. 80, Issue 6, pp. 883–895, 2007.

[84] F. G. Wilkie, D. McFall and F. McCaffery, An evaluation of CMMI process areas for small- to medium-sized software development organisations, Software Process: Improvement and Practice, Vol. 10 , Issue 2, pp. 189–201, 2005.

[85] (Diciembre 2008). "COMPETISOFT (Mejora de Procesos para Fomentar la Competitividad de la Pequeña y Mediana Industria del Software de Iberoamérica) Versión 1.0." 169.

[86] Phadke, M.S., Quality Engineering Using Robust Design, Prentice Hall, 1989.

[87] Macario Polo Usaola, B. P. L., Pedro Reales (2012). Técnicas combinatorias de mutación para testing de sistemas software.