ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

VICERRECTORADO DE INVESTIGACIÓN Y VINCULACIÓN CON LA COLECTIVIDAD

UNIDAD DE GESTIÓN DE POSTGRADOS

MAESTRÍA EN INGENIERIA DE SOFTWARE
I PROMOCIÓN

TESIS PREVIA A LA OBTENCIÓN DEL TÍTULO DE MAGISTER EN INGENIERIA DE SOFTWARE

AUTOR: MILTON EDUARDO ESCOBAR SANCHEZ

TEMA: "DISEÑO DE UN MODELO PARA EL DESARROLLO DE PRUEBA DE SOFTWARE A FIN DE ALCANZAR EL NIVEL 2 DE TEST MATURITY MODEL EN LA EMPRESA SIREDCOM EN LA CIUDAD DE QUITO".

DIRECTOR:
ING. WALTER FUERTES PHD.

LATACUNGA MAYO 2014

CERTIFICACIÓN

El trabajo de investigación titulado **"DISEÑO DE UN MODELO PARA EL DESARROLLO DE PRUEBA DE SOFTWARE A FIN DE ALCANZAR EL NIVEL 2 DE TEST MATURITY MODEL EN LA EMPRESA SIREDCOM EN LA CIUDAD DE QUITO".** Elaborado por el Ing. Milton Eduardo Escobar Sánchez, egresado de la Maestría en Ingeniería de Software, I Promoción, ha sido dirigido, corregido y revisado prolijamente en todas sus partes, procurando coherencia con las normas vigentes de la ESPE, por lo que se autoriza al autor para que lo presente y sustente públicamente a las instancias académicas correspondientes.

Latacunga, Mayo del 2014.

Director de Tesis

_____

ING. Walter Fuertes PHD

C.C. 170701770-1

AUTORÍA

DE RESPONSABILIDAD

Yo, Milton Eduardo Escobar Sánchez, declaro bajo juramento que el presente proyecto de grado denominado, **"DISEÑO DE UN MODELO PARA EL DESARROLLO DE PRUEBA DE SOFTWARE A FIN DE ALCANZAR EL NIVEL 2 DE TEST MATURITY MODEL EN LA EMPRESA SIREDCOM EN LA CIUDAD DE QUITO"**, es de mi autoría; y no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento, respetando los derechos de autor de terceros, conforme las citas que constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Latacunga, Mayo del 2014.

_____

Ing. Milton Eduardo Escobar Sánchez

C.C. 171055754-5

AUTORIZACIÓN

Yo, Ing. Milton Eduardo Escobar Sánchez, autorizo a la Universidad de las Fuerzas Armadas "ESPE", la publicación en la biblioteca virtual de la Institución, mi proyecto de titulación llamado, **"DISEÑO DE UN MODELO PARA EL DESARROLLO DE PRUEBA DE SOFTWARE A FIN DE ALCANZAR EL NIVEL 2 DE TEST MATURITY MODEL EN LA EMPRESA SIREDCOM EN LA CIUDAD DE QUITO".**

Latacunga, Mayo del 2014.

_____

Ing. Milton Eduardo Escobar Sánchez

C.C. 171055754-5

DEDICATORIA

A mi Dios, Jesús y la virgencita Guadalupana, quienes siempre me acompañan en cada día de mi vida y me dan su bendición, ellos mis apoyos primordiales en los buenos y malos momentos de mi vida.

A mi Madre quien fue la mentora de mi vida y quien me impulsó para la realización de esta maestría. A ella que supo darme sus cuidados y que sigue siendo la luz que guía mi camino por donde voy, ella que con su ejemplo de sacrificio y abnegación supo sacar adelante a sus hijos haciéndolos profesionales y hombres de bien. Gracias mamá por ser la mejor madre del mundo.

A mi Padre quien supo siempre dar el apoyo incondicional cuando de estudio se trató, a él que con su ejemplo de disciplina, honor y lealtad, supo inculcar en sus hijos la rectitud y la hombría de bien, gracias padre por ser el referente de hacer realidad la frase “si avanzo sígueme, si me detengo empújame y si retrocedo ayúdame”.

AGRADECIMIENTO

Un especial agradecimiento al Sr. PHD. Walter Fuertes, por impartir sus conocimientos de manera generosa durante el desarrollo del presente proyecto de titulación, quién ha depositado su confianza, guía y tiempo en el fruto de mi esfuerzo y dedicación.

Agradezco de todo corazón a mis amados Padres, quienes con su incondicional apoyo, han forjado la senda de mi camino para que pueda realizar mis sueños, y llegar hasta este momento de mi vida; jamás olvidaré todo lo que ustedes han hecho por mí Padres míos.

Agradezco a mis hijos Stefano Alessandro Escobar S y André Eduardo Escobar S por todos los momentos en los cuales no pude estar, porque siempre conocían la frase es que tengo que hacer mi tesis, porque lo único que quiero es que sean mejores que mí.

ÍNDICE DE CONTENIDO

CERTIFICACIÓN I
AUTORÍA II
DE RESPONSABILIDAD II
AUTORIZACIÓN III
DEDICATORIA IV
AGRADECIMIENTO V
ÍNDICE DE CONTENIDO VI
ÍNDICE DE TABLAS. VIII
ÍNDICE DE ILUSTRACIONES IX
RESUMEN X
CAPÍTULO 1 1
GENERALIDADES 1
1.INTRODUCCIÓN. 1
1.1 PLANTEAMIENTO DEL PROBLEMA. 2
1.2 FORMULACIÓN DEL PROBLEMA. 5
1.2.1 Preguntas de investigación. 6
1.2.2 Limitaciones y Supuestos. 7
1.3 OBJETIVO GENERAL 7
1.3.1 Objetivos específicos. 7
1.3.2 Meta 8
1.4 HIPÓTESIS 8
1.4.1 Operacionalización de las variables. 8
1.5 JUSTIFICACIÓN E IMPORTANCIA 11
CAPÍTULO 2 13
MARCO TEÓRICO. 13
2. INTRODUCCIÓN. 13
2.1 ESTADO DEL ARTE. 14
2.1 EVOLUCIÓN HISTÓRICA DE LA INDUSTRIA DE SOFTWARE. 14
2.1.1 Etapa 1: Hasta 1956 - La orientación de las pruebas a la solución de errores. 15
2.1.2 Etapa 2: 1957-1978 - La orientación de las pruebas a la demostración. 16
2.1.3 Etapa 3: 1979-1983 - La orientación de las pruebas a la detección. 17
2.1.4 Etapa 4: 1983-1987 - La orientación de las pruebas a la evaluación. 18
2.1.5 Etapa 5- 1988 - La orientación de las pruebas a la prevención. 21
2.2 MODELOS DE CALIDAD DE SOFTWARE. 23
2.2.1 ISO-9001-2000. 23
2.2.2 ISO/IEC 9126 – Modelo de Calidad 26
2.2.3 Modelo personal – PSP 29
2.3 PARTICIPANTES EN EL PROCESO DE PRUEBAS. 33
2.4 EVALUACIÓN COMPARATIVAS DE LOS MÉTODOS DE PRUEBAS DE SOFTWARE. 35
2.4.1 TMM y TMMI 35
2.4.2 TPI (Test Process Improvement). 39
2.4.3 Test PAI 41
2.4.4 TMAP 42
2.5 DEFINICIONES. 46
2.5.1 Pruebas. 47
2.5.2 Verificación y Validación de una Prueba 48
2.5.3 Error, falta, defecto y falla. 49
2.5.4 Requerimientos y su relación con la Prueba. 51
2.5.5 Elementos de la Prueba. 52
2.6 PRUEBAS DE SOFTWARE. 54
2.6.1 Consideraciones respecto a la Prueba. 54
2.6.2 Requisitos basados en las pruebas. 55
2.6.3 Actitud frente a las pruebas. 55
2.6.4 Clasificaciones de la Prueba. 55
2.7 OTROS TIPOS DE PRUEBAS. 60
2.7.1 Pruebas de Regresión. 60
2.7.2 Prueba de Humo 61

2.8 TÉCNICAS DE PRUEBA. 61
2.8.1 Técnicas de Caja negra. 62
2.8.2 Técnicas de Caja Blanca. 66
2.8.3 Técnicas según quién hace la Prueba 67
2.8.4 Técnicas basadas en la intuición o en la experiencia. 68
2.9 PLANIFICACIÓN DE LA PRUEBA 70
2.9.1 Componentes del plan de pruebas. 74
2.10 CONCLUSIONES. 75
CAPÍTULO 3 82
MODELO FORMAL DE PRUEBAS FUNCIONALES PROPUESTO 82
3. INTRODUCCIÓN 82
3.1 DIAGNOSTICO DEL SISTEMA DE CALIDAD DE LA EMPRESA SIREDCOM. 82
3.1.1 Estándar de calidad de Software a nivel de proceso. 82
3.1.2 Análisis de Resultados del Diagnóstico. 90
3.2 DIAGNÓSTICO DEL NIVEL DE TMMI. 95
3.3 CONCLUSIONES. 102
3.3.1 Conclusiones (9001:2000) 102
3.3.2 Conclusiones (TMMI) 106
3.4 MODELO FORMAL DE PRUEBAS FUNCIONALES PROPUESTO 107
3.4.1 Principales características del Modelo de Pruebas Funcionales Propuesto. 108
3.4.2 Estructura del Modelo de Pruebas Funcionales Propuesto. 113
i. Especificación de la prueba. 115
ii. Planificación 122
iii. Ejecución de las Pruebas. 130
iV. Evaluación y Resultados de la Prueba. 152
3.5 ROLES. 157
3.6 RELACIÓN CON EL CICLO DE VIDA DE DESARROLLO. 157
3.7 DOCUMENTACIÓN GENERADA DEL MODELO DE PRUEBAS FUNCIONAL PROPUESTO 161
3.7.1 Ciclos de Prueba (CP). 164
3.7.2 Mejoramiento del Proceso de Prueba (MPP). 165
3.7.3 Casos de Prueba (CP). 165
3.7.4 Casos de Prueba de Regresión (CPR). 166
3.7.5 Versión Ejecutable del Producto (EJ) 167
3.7.6 Estimación de Tareas (ET) 167
3.7.7 Especificación de Requerimientos (ERS). 167
3.7.8 Informe de Satisfacción del Cliente (ISC) 167
3.7.9 Listado de Pruebas (LP) 167
3.7.10 Matriz de Trazabilidad (MT). 168
3.7.11 Plan de Desarrollo del Producto (PD). 168
3.7.13 Plan de Ejecución del Ciclo (PEC) 170
3.7.14 Proceso de Incidentes (PI) 170
3.7.15 Plan de Pruebas (PP). 170
3.7.16 Plan Pruebas del Ciclo (PPC). 174
3.7.17 Acuerdo Alcanzado (AA). 175
3.7.18 Reporte de Avance del Ciclo de Pruebas (RACP) 175
3.7.19 Reporte de la Gestión de la ConIlustraciónción (RGC). 179
3.7.20 Reporte de Evaluación del Ciclo de Pruebas (RECP) 179
3.7.21 Reporte Final del Proyecto (RFP) 179
3.7.22 Reporte de Incidentes (RI) 180
3.7.23 Reporte de Ejecución (RE) 181
3.7.24 Reporte de Problemas (RP) 181
3.7.25 Resumen de Reunión (RR). 182
3.7.26 Testware (TW) 182
3.8 ROLES. 183
3.8.1 Líder de Proyecto de Prueba. 184
3.8.2 Diseñador de Pruebas 186
3.8.3Tester 187
3.8.4 Cliente. 189
3.8.5 Desarrollador o Contraparte técnica. 190

CAPÍTULO 4.....191
APLICACIÓN DEL MODELO......191
4. INTRODUCCIÓN......191
4.1 PRODUCTO A PROBAR DE LA EMPRESA SIREDCOM. 192
4.2 APLICACIÓN DEL MODELO PROPUESTO.....193
4.2.1 Especificación de la prueba.....193
4.2.2 Planificación.....195
4.2.3 Ejecución de la Prueba.....204
4.2.4 Resultado y Evaluación del Proyecto.....214
4.3 CONCLUSIONES.....221

ÍNDICE DE TABLAS.

Tabla 1-1. Problemas identificados, Solución, Enfoque y Evidencia de la Solución.....6
Tabla 2.1: Técnicas de Prueba de Software.....62
Tabla 2.2: Componentes del Plan de Prueba.....74
Tabla 2.3 Evaluación histórica de las pruebas de software.....76
Tabla 2.4 Criterios de comparación de modelos más importantes de pruebas de Software.....79
Tabla 2.5 Resultados de Comparación.....81
Tabla 3.1: Lista de Chequeo, con base en la ISO-9001-2000 referente al Sistema de Gestión de Calidad 83
Tabla 3.2: Lista de Chequeo, con base en la ISO-9001-2000 referente a la Responsabilidad de la Dirección. 84
Tabla 3.3: Lista de Chequeo, con base en la ISO-9001-2000 referente a la Gestión de los Recursos. 85
Tabla 3.4: Lista de Chequeo, con base en la ISO-9001-2000 referente a la Realización del Producto. 86
Tabla 3.5: Lista de Chequeo, con base en la ISO-9001-2000 referente a la Medición, Análisis y Mejora. 88
Tabla 3.6: Resultados del diagnóstico de Sistema de Calidad de la Empresa Siredcom.....90
Tabla 3.7: Información de Evaluación de TMMI.....95
Tabla 3.8: Procesos que constituyen el nivel 2 de TMMI.....97
Tabla 3.9 – Especificación de la Prueba.....116
Tabla 3.10 – Planificación.....123
Tabla 3.11 – Diseño de las Pruebas.....133
Tabla 3.12 – Entorno de Pruebas.....138
Tabla 3.13– Ejecución de las Pruebas.....140
Tabla 3.14 – Seguimiento y Control.....146
Tabla 3.15 – Evaluación del Proyecto.....152
Tabla 3.16 – Documentos Generados y Sus Relaciones.....162
Tabla 3.17 – Información adicional para estimación del esfuerzo.....169
Tabla 3.18 – Métricas sobre el ciclo de pruebas.....176
Tabla 3.19 – Métricas finales del proyecto.....180
Tabla 3.20 – Responsabilidades del líder de pruebas.....184
Tabla 3.21 – Responsabilidades del líder de pruebas.....186
Tabla 3.22 – Responsabilidades del tester.....188
Tabla 3.23 – Responsabilidades del cliente.....189
Tabla 3.24 – Responsabilidades del desarrollador.....190
Tabla 4.1 – Alcance de la Especificación de la Prueba.....194
Tabla 4.2 – Esfuerzo de la Especificación de la Prueba.....197
Tabla 4.3- Alcance definido para la fase de Planificación de los Ciclos de Pruebas.....197
Tabla 4.4 - Planificación de las Pruebas.....200
Tabla 4.5 – Esfuerzo de la Planificación de las Prueba.....201
Tabla 4.6 – Alcance del Ciclo 1 (Planificado Versus Real).....202
Tabla 4.7 – Casos de Prueba e Incidentes Encontrados por Funcionalidad.....204
Tabla 4.8 – Incidentes por Tipo (Ciclo 1).....205
Tabla 4.9 – Incidentes según Criticidad (Ciclo 1).....205
Tabla 4.10 – Cronograma Planificado – Cronograma Real (Ciclo 1).....207

Tabla 4.11 – Resumen del Ciclo No 1. 207
Tabla 4.12 – Esfuerzo del Ciclo de Prueba 1. 208
Tabla 4.13 – Ciclo 2 Planificado Versus Real. 209
Tabla 4.14 – Caso de Prueba e Incidentes según funcionalidad. 210
Tabla 4.15– Caso de Prueba de Regresión y sus Incidentes. 211
Tabla 4.16 – Tipos de Incidentes en el Ciclo 2. 211
Tabla 4.17 – Criticidad en Incidentes en el Ciclo 2. 212
Tabla 4.18 – Cronograma Planificado Versus Real para el Ciclo 2. 213
Tabla 4.19 – Resumen de las Pruebas en el Ciclo2. 213
Tabla 4.20 – Esfuerzo en el Ciclo2. 214
Tabla 4.21 – Planificación de las Pruebas Versus la Planificación Real de las Pruebas. 215
Tabla 4.22– Encuesta de Satisfacción del Cliente. 215

ÍNDICE DE ILUSTRACIONES

Ilustración 2.1: Mapa Conceptual sobre la investigación realizada. 13
Ilustración 2.2: Modelo de calidad ISO/IEC 9126. 28
Ilustración 2.3: Estructuras del Conocimiento en PSP 31
Ilustración 2.4 Estructura de TMM. 35
Ilustración 2.5 Estructura de TMMI. 36
Ilustración 2.6 Niveles de TMMI. 37
Ilustración 2.7 Estructura TPI 40
Ilustración 2.8 Áreas Claves de Madurez 40
Ilustración 2.9 Objetivos Y Prácticas Específicas de Test PAI. 42
Ilustración 2.10 Elementos de TMap. 43
Ilustración 2.10 Elementos de TMap. 46
Ilustración 2.12 Estructura de defectos y fallas. 51
Ilustración 2.13 Jerarquía de Planes de Prueba. 72
Ilustración 3.1 Proceso - Sistema de Gestión de la Calidad según ISO 9001:2000 91
Ilustración 3.2 Proceso – Responsabilidad de la Dirección según ISO 9001:2000 92
Ilustración 3.3 Proceso – Gestión de Recursos según ISO 9001:2000 92
Ilustración 3.4 Proceso – Realización del Producto según ISO 9001:2000 93
Ilustración 3.5 Proceso – Gestión de Recursos según ISO 9001:2000 94
Ilustración 3.6 Resultados de la Evaluación del Nivel de TMMI de la Empresa Siredcom 106
Ilustración 3.7 Modelo propuesto para Pruebas Funcionales de Software 113
Ilustración 3.8 Modelo propuesto para Pruebas Funcionales de Software Nivel de Abstracción 2114
Ilustración 3.9 Actividades de Especificación de Prueba. 117
Ilustración 3.10 Actividades de la Planificación de Prueba. 123
Ilustración 3.11 Ejecución de la Prueba. 131
Ilustración 3.12 Actividades de la sub-etapa Diseño de las Pruebas 134
Ilustración 3.13 Actividades de la sub-fase Entorno de Pruebas. 138
Ilustración 3.14 Sub Fase Ejecución de las Pruebas. 141
Ilustración 3.15 Seguimiento y Control 147
Ilustración 3.16 Actividades de la fase de Resultado de la Prueba. 153
Ilustración 3.17 El modelo propuesto en relación al ciclo de vida de desarrollo 1599
Ilustración 4.1 Estados de un Incidente. 198
Ilustración 4.2– Funcionalidades, Casos de Prueba e Incidentes de las Pruebas Realizadas en Cada Ciclo 216
Ilustración 4.3– Funcionalidades, Casos de Prueba e Incidentes de las Pruebas Realizadas en Cada Ciclo 216
Ilustración 4.4 – Incidentes por Tipo según su Ciclo. 217
Ilustración 4.5– Incidentes por Tipo según su Ciclo. 217
Ilustración 4.6 – Incidentes según Importancia. 218

Ilustración 4.7 – Incidentes según Importancia. 218
Ilustración 4.8 – Porcentaje de Incidentes según la Importancia (Alta, Media , Baja) 219
Ilustración 4.9– I 219

RESUMEN

En el Ecuador las aplicaciones de software son cada vez más importantes para las organizaciones. El registro de errores es una parte importante de la ejecución de la prueba, estas actividades pasan casi desapercibidas, debido al desconocimiento e inaplicabilidad de procesos formales. Se utilizó la lista de chequeo, basada en la ISO-9001-2000 lo que nos permitió evaluar el sistema de gestión de calidad de la empresa Siredcom, alcanzado altos índices de inconformidad. Por otro lado en la evaluación realizada dentro del nivel 2 de TMMI, encontrándose que en los procesos involucrados, el número de inconformidades es mayor a las conformidades. Se llega a concluir que el TMMI, está en un nivel 1, ya que no se encuentra definido un proceso que permita distinguirlo del debugging y no se determina de forma adecuada la especificación del software a ser probado. Además se denota que el personal de la empresa no posee la competencia adecuada. Todo lo anterior demanda el diseño de un modelo que permita implantar y ejecutar pruebas funcionales de software, a fin de alcanzar el nivel 2 de Test Maturity Integration como una respuesta eficaz ante las necesidades de la empresa SIREDCOM? La propuesta se fundamenta en el ciclo de prueba y se compone de cuatro partes: Especificación, Planificación, Ciclo y Resultado. A continuación se procedió a la aplicación y validación del modelo propuesto, para lo cual se tomó el producto a ser probado, "Sistema modular de administración educativa" el cual se encuentra en producción, habiendo realizado todas las actividades con excelentes resultados

Palabras Clave: Modelo funcional de pruebas, nivel 2 de Test Maturity Model Integration, modelos de prueba, pruebas de software y tipos de prueba.

ABSTRACT

In Ecuador software applications are becoming increasingly important for organizations. The error log is an important part of the test performance because it is the main business concept; these activities are almost unnoticed due to ignorance and inapplicability of processes. It was used the checklist, based on the ISO-9001-2000 allowing to assess the quality management system of the company Siredcom, achieved high levels of dissatisfaction. Moreover the evaluation applied in level 2 of TMMI, allowed finding that the processes involved and the number of disagreements are more than the conformities. In conclusion the TMMI, is at level 1 because it is not defined a process to distinguish the debugging and it is not determined the software specification to be tested. The tests were performed in the process when coding is complete. Further

denotes that the company's staff lacks the appropriate competence. All this demand designs a model to implement and execute functional test software, to achieve Maturity Model Integration Test Level 2 as an effective response to the needs of the company SIREDCOM? The proposal model is based on the test cycle and consists of four parts: Specification, Planning, Cycle and Result. Then we proceeded to the implementation and validation of the proposed model, for which the product to be tested, "modular system educational administration" which is in production once all the activities were performed with exceptional results.

Keywords: Functional model test, Test Maturity Model Integration level 2, test models, software testing and types of test.

CAPÍTULO 1

GENERALIDADES

1. INTRODUCCIÓN.

En los últimos tiempos las aplicaciones de software son cada vez más importantes en las actividades de la sociedad, debido al avance en las tecnologías, ahora las aplicaciones son necesarias y complejas en especial para las organizaciones, constituyéndose en la mayoría de los casos en el eje principal del negocio.

El software en la actualidad se lo realiza como una actividad programada, planificada y con un ciclo de vida definido, sin embargo en nuestro medio las pruebas de software pasan casi desapercibidas, lo que ocasiona una serie de errores y fallas en el software que redundan en pérdida de tiempo y dinero para las organizaciones; los equipos de pruebas en la mayoría de organizaciones, no se encuentran capacitados y lo que es peor aún no existen; por lo que se hace necesario el diseño de un modelo para el desarrollo de pruebas software a fin de alcanzar el nivel 2 de TMMI.

1.1 PLANTEAMIENTO DEL PROBLEMA.

El software actualmente se encuentra en casi todos los trabajos y áreas donde se desenvuelve el ser humano, siendo el soporte primordial en las operaciones propias de cada negocio.

De acuerdo con (Iris Pinkster, 2010) el registro de errores (también llamados defectos o problemas) es una parte importante de la ejecución de la prueba, no sólo porque permite que el equipo de pruebas demuestre si el sistema de información depende de la calidad requerida, sino también porque permite a los desarrolladores, reproducir y resolver las fallas.

Basado en (Galin D, 2005) el Aseguramiento de la Calidad del Software (SQA – por sus siglas en inglés) se ha convertido en una cuestión esencial en el desarrollo y mantenimiento de productos.

La calidad del software desde el enfoque de los clientes y usuarios es medida por la cantidad de fallas que se encuentran en el producto final y sobre todo por la gravedad que éstas conllevan para el normal desenvolvimiento de su organización.

En nuestro medio las actividades de pruebas pasan casi desapercibidas, pues no existen pruebas de software que permitan garantizar la calidad del mismo; no hay una orientación clara en la planificación del proyecto, tampoco políticas organizacionales, debido al desconocimiento o inaplicabilidad de procesos formales, aumentando el riesgo de producir software con errores o fallas graves.

Lo anterior requiere de una metodología que concrete actividades y defina responsabilidades, dentro de un modelo de procesos definido. En la Empresa Siredcom, no existe una metodología que permita implantar y ejecutar un proceso de pruebas eficiente, que permita mostrar que el software cumple con sus especificaciones, a través de la inicialización de un proceso de planificación de pruebas que incluye: establecer objetivos, analizar riesgos, delinear estrategias, realizar el diseño de los casos de prueba.

Además según (Mala D., 2010) actualmente, en la industria del software, las pruebas de software constituyen uno de los procesos más importantes, ya que permite asegurar la calidad de los productos software.

Las pruebas que se realizan al producto de software deben ser ejecutadas de manera exhaustiva antes de su entrega y no después, a fin de probar el comportamiento del mismo, tomando en cuenta sus posibles combinaciones en diferentes escenarios y estados que el sistema puede enfrentar. Se debe tomar en cuenta que esto, conlleva gran cantidad de tiempo y dinero, tornándose dicho control en ciertas ocasiones en algo inviable, sin embargo el desarrollo de software se debe transformar de una actividad empírica a una actividad formal (programada, planificada y con un ciclo de vida definido), a fin de evitar la pérdida de tiempo y dinero.

Según (Esteban, 2012) un modo de mejorar las pruebas software, y solventar en gran medida los problemas expuestos anteriormente, consiste en desarrollar

iniciativas de mejora de procesos que impliquen la implantación y mejora de un proceso formal, completo y bien definido de pruebas software. Esto requiere de la existencia de modelos que especifiquen el conjunto de procesos a implementar así como las actividades y tareas que se requieren realizar. Sin embargo, los modelos de referencia de procesos disponibles en la actualidad carecen de un enfoque adecuado o completo que permita a las organizaciones, especialmente, a las pequeñas y medianas empresas (PYMES) llevar a la práctica estas iniciativas en el área de las pruebas software.

La competitividad que existe en la industria del desarrollo de software se ve liderada por aquellas empresas que poseen un modelo basado en metodologías o procedimientos estándares que permitan el planeamiento, especificación y ejecución de pruebas de software, dejando de lado la informalidad e improvisación mediante el uso gradual de CMM o TMM, y sus niveles correspondientes.

De acuerdo con (KANER C., 2002) "El éxito del desarrollo de las pruebas software depende en gran medida de las habilidades, conocimientos, intuición y experiencia de los miembros que integran el equipo de pruebas". En Siredcom, no existe un equipo de pruebas Software que permita desarrollar procesos, lo que hace suponer que el conocimiento juega un papel importante, en la aplicación de métodos y técnicas de prueba, para el éxito de las mismas.

Las competencias aseguran el desempeño de las tareas para el cumplimiento de las actividades y objetivos propuestos por la organización haciendo que ésta sea competitiva; por lo tanto quienes están conformando el equipo de pruebas deberán

tener las competencias necesarias para el desempeño del puesto. Basados en (URQUIZA A., 2011) estas deben incluir, tanto las competencias generales como las competencias técnicas asociadas a cada uno de los roles que participan en el proceso, incluyendo el nivel en que se requiere cada una.

No son suficientes los procesos mencionados para obtener los resultados esperados en beneficio de la organización si no se dispone de personal calificado, si éstos no disponen del conocimiento y habilidades para realizar de forma eficiente las actividades encomendadas. Actualmente en Siredcom no existe un modelo orientado a cumplir con los objetivos propuestos en el nivel 2 de TMM, que cumpla con los requisitos formales; al carecer de éste modelo, no disponemos de una herramienta que permita formar a las personas que integrarán los equipos de prueba, lo que redunda en la falta de éxito en la organización que quiere desarrollar iniciativas de mejora.

1.2 FORMULACIÓN DEL PROBLEMA.

¿Existe la necesidad de disponer del diseño de un modelo para las pruebas funcionales de software, a fin de alcanzar el nivel 2 de Test Maturity para la empresa SIREDCOM?

Tabla 1-1. Problemas identificados, Solución, Enfoque y Evidencia de la Solución.

PROBLEMA	SOLUCIÓN	ENFOQUE	EVIDENCIA
Las actividades de prueba pasan casi desapercibidas	Mejora de Procesos	Estrategia: Procesos	Mejora de productos e incremento de fallos detectados
Los equipos de pruebas no están en capacidad de realizar las tareas en forma adecuada	Capacitación en competencias específicas	Estrategia: Competencias	Capacitación en competencias específicas
Los defectos encontrados en fases avanzadas demandan trabajo adicional, costos y pérdida de tiempo	Seguimiento y Control de los Ciclos de Pruebas	Estrategia: Seguimiento y Control.	Seguimiento y Control de los Ciclos de Pruebas
No existe un modelo que permita implantar y ejecutar un proceso de pruebas funcionales eficiente en la organización	Diseñar un modelo para el desarrollo de prueba de software.	Estrategia: Nivel 2 de Test Maturity Model	Modelo Propuesto

Fuente: Elaboración Propia.

1.2.1 Preguntas de investigación.

¿Cuáles son las metodologías y técnicas actuales que permiten el desarrollo de pruebas funcionales de software?

¿Cómo superar las deficiencias del proceso de pruebas funcionales de software de la Empresa Siredcom basado en el TMM Nivel 2. ?

¿Cuáles son los elementos o fases que integran un modelo para el desarrollo de pruebas funcionales de software?

¿Cuál es el objetivo de alcanzar el nivel 2 del Test Maturity Model TMM?

1.2.2 Limitaciones y Supuestos.

Esta investigación pretende llegar al diseño de un modelo formal para pruebas funcionales de software que permita alcanzar el nivel 2 de TMM, la aplicación y la validación del mismo; no es posible su implantación por la limitación del tiempo y los recursos.

1.3 OBJETIVO GENERAL

Diseñar un modelo de pruebas funcionales independientes de software, para la empresa Siredcom, apoyado por un equipo de pruebas que disponga del recurso humano con competencias, que permita alcanzar el nivel 2 de TMM.

1.3.1 Objetivos específicos.

Analizar las metodologías formales aplicadas a los procesos de pruebas de software que existen actualmente, las ventajas y desventajas que pueden brindar a las pruebas funcionales de software.

Diagnosticar el estado actual del proceso de pruebas de los proyectos desarrollados en la empresa Siredcom.

Diseñar un modelo formal y apropiado al proceso de pruebas funcionales de software, que permita a la Empresa Siredcom alcanzar los objetivos del Nivel 2 de TMM.

1.3.2 Meta.

Diseñar un modelo para pruebas funcionales de software a fin de alcanzar el nivel 2 de test Maturity Model para la empresa Siredcom en la ciudad de Quito.

1.4 HIPÓTESIS.

El diseño de un Modelo para realizar pruebas funcionales independientes de un producto de software permitirá alcanzar el nivel 2 de TMM.

1.4.1 Operacionalización de las variables.

Variables	Dimensiones	Indicadores	Instrumentos
Variable Independiente El diseño de un Modelo para realizar pruebas funcionales independientes de un producto de software. El modelo permite abordar los aspectos técnicos y directivos de madurez para realizar pruebas apoyadas por herramientas y técnicas básicas de prueba.	Especificación de la prueba.	Ep1. Necesidad del cliente Ep2. Revisión de requerimientos Ep3. Análisis de riesgo A1 - Estimación de las tareas A2 – Medición de Esfuerzo	AR-Acta de Reunión AA-Acuerdo alcanzado LP Listado de Prueba Listado de Prueba Priorizado ET- Estimación de Tareas ME- Matriz de Esfuerzo

Continua ⟶

	Planificación de la prueba	PL1 - Exploración del Producto. PL2 - Definición de los Ciclos de Prueba. PL3 - Definición del Testware. PL4 - Planificación de las Pruebas PL5 - Definición del Proceso de Incidentes A1 - Estimación de las tareas A2 – Medición de Esfuerzo	LP - Listado de Pruebas LCP - Listado de Ciclos de Prueba. TW – Testware. PP - Plan de Pruebas. PI - Proceso de Incidentes. ET - Estimación de Tareas. ME - Matriz de Esfuerzo.
	Ejecución de la Prueba	**Diseño de Pruebas:** PL2- Revisión de requerimientos E2 – ConIlustraciónción DP1 – Diseño de los Casos de Prueba DP2 – Validación de los Casos de Prueba DP3 – Asignación de los Casos de Prueba **Seguimiento y Control:** AC1 – Planificación del Ciclo de Pruebas. AC2 - Gestión de la ConIlustraciónción. AC3 - Seguimiento y Control del Ciclo de Pruebas. AC4 - Evaluación del Ciclo de Pruebas. A1 - Estimación de las tareas A2 - Medición de Esfuerzo **ConIlustraciónción del Entorno** E1 - Instalación de Herramientas. E2 – ConIlustraciónción **Ejecución de las Pruebas** EP1 - Pruebas de Humo. EP2 - Ejecución de las Pruebas.	 LP- Listado de Pruebas RI - Reporte de Inconvenientes. CP – Casos de Prueba MT – Matriz de Trazabilidad RR – Resumen de Reunión PE – Plan de Ejecución PPC- Plan de Pruebas del Ciclo RGC - Reporte de la Gestión de la ConIlustraciónción. RACP - Reporte de Avance del Ciclo de Pruebas. RECP - Reporte de Evaluación del Ciclo de Pruebas. ET - Estimación de Tareas ME - Matriz de Esfuerzo RI- Reporte de Inconvenientes Reporte de problemas RE - Reporte de Ejecución

Continua ⟶

		EP3 - Testing Exploratorio. EP4 - Reporte de Incidentes. EP5 - Validación de los Incidentes. EP6 – Verificación de las Correcciones.	RI - Reporte de Incidentes RE - Reporte de Ejecución
	Evaluación y Resultado de la Prueba	RP1- Determinar la Satisfacción del Cliente. RP2- Reporte de Finalización del Proyecto. RP3- Archivar el Proyecto RP4- Mejoramiento del Proceso de Prueba A1 - Estimación de las tareas A2 - Medición de Esfuerzo	ISC – Informe de Satisfacción del Cliente. RFP – Reporte Final del Proyecto AP – Archivo del Proyecto MPP – Mejora del Proceso de Prueba. ET - Estimación de Tareas ME - Matriz de Esfuerzo
Variable Dependiente Alcanzar el nivel 2 del Test Maturity, Model TMM**.** Es la fase de definición, cuyo objetivo principal es verificar que el software satisfaga los requisitos especificados dentro de un modelo de madurez centrado en el Proceso de Pruebas.	Desarrollar pruebas y definir Objetivos de depuración Iniciar un proceso de planificación de pruebas Institucionalizar métodos y técnicas básicos de pruebas	Objetivos, las tareas, las actividades, y las herramientas, responsabilidades Localizar defectos correcciones nuevas pruebas Definir los objetivos, analizar los riesgos y las estrategias, y desarrollar las especificaciones del diseño de pruebas y los casos de pruebas asignación de recursos Planificación de las actividades de pruebas, y las responsabilidades para hacer pruebas de unidad, de integración, del sistema, y los criterios de aceptación de conformidad con los resultados. Estrategias de pruebas de caja blanca y de caja negra, Matrices de validación de requerimientos División de las pruebas de ejecución basadas en **sub-fases**	Documento con los objetivos y políticas de pruebas Establecer políticas de planificación de pruebas en la organización las Cuáles deben ser respaldadas por la alta dirección. Debe elaborarse una plantilla de planificación y especificación de pruebas la cual debe ser entregada y puesta a disposición de los gerentes de proyecto y desarrolladores. Se debe implementar un mecanismo para integrar los requisitos del usuario al plan de

Continua ⟶

		como pruebas de unidad, pruebas de integración, pruebas del sistema, y pruebas de aceptación.	pruebas.

Fuente: Elaboración Propia.

1.5 JUSTIFICACIÓN E IMPORTANCIA

En la actualidad, las empresas en nuestro medio no acostumbran a llevar procesos de prueba en el desarrollo de productos de software por lo que es necesario establecerlos de manera formal y técnica, la administración y estandarización de un modelo que permita el aseguramiento de la calidad y con esto la mejora continua en las empresas.

La investigación del tema es necesaria, ya que la presencia de fallas o errores que han sucedido hasta la actualidad en Siredcom, dificulta el mantenimiento y la confiabilidad de sus clientes, afectando el funcionamiento de sus sistemas y lo que es peor podría redundar en el paro de las operaciones del negocio.

La importancia del tema a investigar está relacionada con un problema actual que afecta a la mayoría de empresas del país, al no contar con productos de software, libre de fallas y errores, al no aplicar estándares de calidad en el área de la Ingeniería de software, para que quienes dependen de las tecnologías en sus trabajo diario, cuenten con aseguramiento de calidad y mejora continua en sus productos.

La finalidad de esta investigación, es proponer el diseño de un modelo para pruebas funcionales de software a fin de alcanzar el nivel 2 TMM, que permita en forma sistemática la mejora de la calidad de las aplicaciones finales que brinda la

empresa a través del proceso de pruebas, el mismo que estará reflejado en tiempo, confiabilidad y costo.

CAPÍTULO 2

MARCO TEÓRICO.

2. INTRODUCCIÓN.

Este capítulo se inicia con un mapa conceptual como se observa en la Ilustración 2.1 donde se puede observar la organización de la investigación, definiéndose dentro del estado del arte: la evolución histórica de la industria del software, los modelos de calidad de software, los participantes del proceso y la evaluación comparativa de los modelos más importantes; definiciones, pruebas de software, otros tipos de pruebas, técnicas de prueba y planificación de las pruebas enfocadas a la funcionalidad del producto a probar.

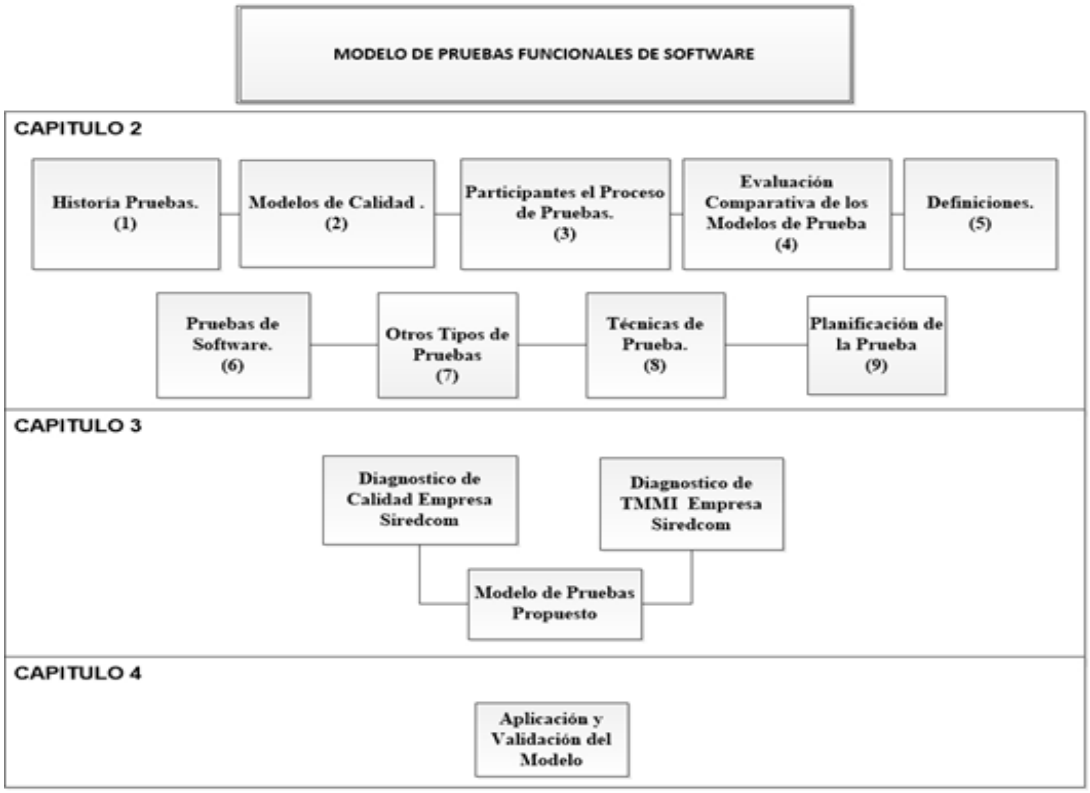


Ilustración 2.1: Mapa Conceptual sobre la investigación realizada.
Fuente: Elaboración Propia.

2.1 Estado del Arte.

La investigación se la realizó basándose en un mapa conceptual que permitió llegar a la consecución del presente tema de tesis como se observa en la Ilustración 2.1.

2.1 Evolución histórica de la industria de software.

De acuerdo con (Shewhart, 1931) la calidad es un aspecto de suma importancia en el ciclo de vida de desarrollo de software, trabajo en el mejoramiento de los procesos, utilizando control estadístico de la calidad, como un problema de variación, permitiendo la eliminación temprana de errores y sus respectivas causas.

Según (G. J. Myers, 1979) se promovió que la ingeniería del software separe las disciplinas fundamentales del desarrollo del software de la verificación y validación, concepto que se vio reforzado gracias a la técnica de pruebas constituyendo el núcleo del aseguramiento de la calidad.

Basado en (Myers, 2004) la prueba es el proceso de ejecución de un programa con la intención de encontrar errores ; la verdadera definición de lo que constituye la prueba de software y la comprensión de la misma permitirá que los testers tengan éxito en su esfuerzo al buscar errores

W. Edwards Deming, refina los principios de Shewhart y de acuerdo con (Deming, 1986) se establece el ciclo Deming que fue utilizado en la mejora continua de la calidad dentro de la empresa (Planear, Hacer, Verificar y actuar). Deming fue

un crítico de la gestión estadounidense, en ese contexto presenta 14 puntos que constituyen la teoría de su gestión basada en principios como la calidad y liderazgo.

Fundamentado en (W. C. Hetzel & Hetzel, 1988) el Dr. Dave Gelperin y el Dr. William C. Hetzel, realizaron una clasificación basada en los modelos de pruebas más influyentes de la época, sin embargo, los modelos se presentan en forma evolutiva, acumulando así conceptos conforme se desarrollan.

2.1.1 Etapa 1: Hasta 1956 - La orientación de las pruebas a la solución de errores.

Durante los comienzos de las Tecnologías de la Información (TI), es notoria la aparición de problemas relacionados con el software desarrollado, éstos fueron relegados en beneficio del hardware, la programación era vista como una actividad que debía ser desarrollada y probada por el mismo programador.

Fundamentado en (Turing, 1950) se escribe el primer artículo basado en pruebas del software, que hoy en día se podrían denominar "pruebas de corrección", definió el desarrollo de una prueba con el fin de demostrar el comportamiento inteligente de una aplicación, estableció que su funcionamiento y de un sistema de referencia, en este caso el ser humano, debería resultar indistinguible para un tercero, el cual asumiría el papel del probador.

2.1.2 Etapa 2: 1957-1978 - La orientación de las pruebas a la demostración.

Del libro "Digital Computer Programming" de (McCracken, 1957) se determina que la verificación de los programas perseguía dos objetivos fundamentales:

i. Asegurar el funcionamiento de la aplicación.
ii. Asegurar que la aplicación resolvía el problema para el que había sido planteada.

Durante este período, se produjo un notable incremento de los desarrollos de software en términos de cantidad, coste y complejidad, por lo cual, se comenzó a dar mayor énfasis a la "mejora de las pruebas", y detectar problemas antes de la entrega de los productos.

En esta etapa, debido a que llevar a cabo las pruebas suponía un esfuerzo comprendido de entre un 30% y un 50% según (B. Beizer) como mínimo del coste total de un desarrollo software. De acuerdo con (Ince, 1987) se concluyó que si el proceso de pruebas pudiese ser automatizado, se reduciría significativamente su costo.

Según (G. J. Myers, 1979) determinó que al construir los caso de prueba estos producían sobre los datos a utilizarse un volumen grande de posibilidades, razón por la cual un programa al probarlo se volvía infinito.

Basados en (William E. Howden, 1976) y (William E Howden, 1978) donde se pública la primera aproximación teórica hacia como diseñar métodos sistemáticos que puedan ser utilizados para construir pruebas funcionales dos años después establece dos aproximaciones a las pruebas de software, una teórica y otra empírica.

2.1.3 Etapa 3: 1979-1983 - La orientación de las pruebas a la detección.

Según (G. J. Myers, 1979) determinó que el modelo orientado a la detección y definió éstas como "El proceso de ejecutar un programa con la intención de encontrar errores". El objetivo es demostrar que si un programa falla, los datos de prueba deberían tener una alta probabilidad de detectarlo.

Tanto en el libro de Myers como en otras publicaciones aparecidas en este periodo: "Software Project Verification and Validation" de (Deutsch, 1981) y "Software Testing and Validation Techniques" de (HOWDEN.W.E., 1981), se analizó el uso de las pruebas en el desarrollo del software y su empleo en las técnicas de revisión. Otros artículos como "Life-Cycle Software Validation" de (William E. Howden, 1982), contienen las primeras descripciones de las aproximaciones a la detección combinada de fallos.

En 1982, surge una nueva metodología orientada a la automatización de casos de prueba denominada "aleatoria". Publicada en "Automatic generation of random self-checking test cases" de (Bird & Munoz, 1983), resultando ésta una técnica muy simple que crea datos aleatorios para cualquier caso que se quiera probar, lo cual presenta un bajo índice de cobertura.

2.1.4 Etapa 4: 1983-1987 - La orientación de las pruebas a la evaluación.

Según (Shorten, 1985) en 1983, el "Institute for Computer Sciences and Technology of the National Bureau of Standars" publica una línea de investigación especialmente orientada a los sistemas de procesamiento de información federal (FIPS). En la cual se describe una metodología que integra el análisis, la revisión y las actividades de prueba con el fin de ofrecer una evaluación del producto a lo largo del ciclo de vida del software, estableciendo la necesidad de obtener en cada fase del ciclo de vida un conjunto de actividades y de productos asociados. Esta guía de desarrollo recomienda tres conjuntos de técnicas de evaluación; la básica, la comprensiva y la crítica. Dichas técnicas presentan un desarrollo evolutivo, de tal forma que cada una de ellas contiene lo desarrollado por su predecesora.

FIPS no fue el único modelo que surgió en el periodo del 1983-1987. En este periodo fueron creados dos estándares relativos al desempeño de buenas prácticas a lo largo del desarrollo de las pruebas en la ingeniería del software.

En 1979 según (G. J. Myers, 1979), dado el desarrollo de las pruebas hasta ese momento, promovió que la Ingeniería del Software se separe de las disciplinas fundamentales del desarrollo del software de la verificación y validación, disciplina que abarcaría las pruebas del software.

Ese mismo año, un grupo del comité técnico de ingeniería del software del IEEE comenzó a trabajar en un estándar para la documentación de las pruebas del software. Como resultado, el estándar ANSI/IEEE STD 829-1983 que fue publicado en 1983 especificando el contenido y el formato de ocho documentos estándares. En los cuales se tuvo especial consideración con los aspectos relativos a la modularidad, cohesión, acoplamiento, usabilidad y facilidad de revisión de las pruebas.

Las principales diferencias existentes entre la propuesta ofrecida en el estándar ANSI/IEEE STD 829-1983 y las prácticas realizadas por aquel entonces, residían en las especificaciones de la planificación y el diseño de las pruebas. Un plan de pruebas según establece este estándar, basa sus objetivos en función de: la identificación de riesgos, el establecimiento de una estrategia general, la definición de una estructura para sus tareas, la distribución de los recursos y sus responsabilidades, el calendario de desarrollo y los posibles contratiempos que puedan surgir. Además, propone la identificación y la descripción de los casos específicos y los procedimientos de prueba con el fin de diferenciar las especificaciones dadas.

Muchos de los planes de pruebas realizados hasta entonces incluían tanto tareas de planificación como de diseño, lo cual retrasaba considerablemente la planificación, limitando de esta forma la elección de estrategias. En la actualidad según (Kaner, Bach, & Pettichord, 2008) se utiliza el estándar (IEEE, IEEE Standard for Software and System Test Documentation, 2008) el que habla sobre los documentos de prueba.

Un año más tarde, en 1984 se aprobó la creación de un organismo de investigación que desarrolle modelos de mejora para los problemas en el desarrollo de los sistemas de software, y evaluar la capacidad de respuesta y fiabilidad de las compañías que suministran software al Departamento de Defensa (DoD), dando lugar al Instituto de Ingeniería del Software (SEI), en él se comenzó un año más tarde a trabajar en un marco de madurez de procesos que permitiese evaluar a las empresas productoras de software.

Años más tarde, un segundo grupo perteneciente al IEEE comenzó a desarrollar un estándar basado en las pruebas unitarias. Como resultado, el estándar ANSI/IEEE STD 1008-1987 fue publicado en 1987, en él se especifican las fases, actividades, tareas y documentos que comprenden las pruebas unitarias, continuando la línea que establecía un fuerte paralelismo entre el desarrollo del software y el de las pruebas. Este estándar enfatiza la necesidad de diseñar un conjunto de pruebas, además establece la necesidad de realizar dos tipos de documentos a la hora de implementar las pruebas unitarias: una especificación del diseño de pruebas y un informe del resumen de las mismas. Estos requisitos fomentan el diseño de las pruebas y representan su principal diferencia con las prácticas hasta el momento más comunes, ya que, normalmente, las pruebas unitarias pasaban desapercibidas.

Un año después de comenzarse a desarrollar el estándar orientado a las pruebas unitarias, en 1985 sus autores introdujeron aquel proceso a lo largo de los distintos niveles de prueba existentes, dando como resultado una metodología conocida como "El proceso de evaluación y de pruebas sistemáticas" (STEP) de acuerdo con (W. C. Hetzel & Hetzel, 1988). Esta metodología definió un sistema de

tareas de pruebas, productos y roles con el fin de dar consistencia y salvar costos a la hora de alcanzar los objetivos propuestos en las pruebas. STEP se basa en un modelo de prevención del ciclo de vida paralelo al desarrollo de software, estableciendo una secuencia de actividades como la planificación, el análisis, el diseño, la implementación, la ejecución y el mantenimiento de las pruebas.

2.1.5 Etapa 5- 1988 - La orientación de las pruebas a la prevención.

(Gelperin & Hetzel, 1988). Publica su artículo “The growth of Software Testing” en el que hablan de la trayectoria que ha seguido el proceso de pruebas.

(Watts S. Humphrey, Snyder, & Willis, 1991). Extendieron los principios desarrollados por Deming para su aplicación a la industria del software a través de sus trabajos en IBM.

En la década de los 90, es publicado el libro "Software Testing Techniques" de (B. Beizer) el cual hace establece un catálogo de técnicas de prueba. Beizer comenta que "el acto de diseñar pruebas es una de las técnicas más efectivas a la hora de prevenir defectos".

En ese mismo año según (Korel, 1990) se introduce otra técnica basada en la automatización de los casos de prueba con la publicación de la metodología orientada al objetivo, en la que se establece como el proceso de generar datos de prueba con el fin de alcanzar un determinado estado sin importar el camino que recorran estos. (B. Hetzel, 1991) estableció el proceso de pruebas como la

planificación, el diseño, la implementación y la ejecución de las pruebas y sus entornos. Incorporando todas las ideas que hasta la fecha habían considerado a las pruebas como un proceso gestionado, es decir, un ciclo de vida implicado con las pruebas.

Pero 1991 fue un año especialmente significativo ya que también vio la luz el proyecto comenzado años más tarde por el SEI, el Modelo de Madurez de las Capacidades para el Software (SW-CMM), Capability Maturity Model for Software).

El desarrollo de SW-CMM ha significado un hito importante tanto en el desarrollo de software como en el área de verificación y validación de productos software. Transformándose CMM IntegrationSM (CMMI) cuya primera versión fue publicada en agosto del año 2000 y, actualmente, se encuentra disponible en su versión 1.3. El Objetivo de CMMI residía en solucionar los problemas existentes con los distintos CMM. Además se ha convertido en estándar de facto dentro de las organizaciones de desarrollo de software a nivel mundial empleándose como mecanismo para la mejora de procesos.

En relación al área de verificación y validación, se han desarrollado distintos modelos de referencia específicamente dirigidos a incrementar la calidad tanto del producto como del proceso de pruebas. Entre estos modelos, se pueden destacar: el Modelo de Madurez de Pruebas (TMM), el Modelo de Madurez de Pruebas Integrado (TMMI), el Proceso de Mejora de Pruebas (TPI) y una metodología para gestión de pruebas TMap (Test Management approach).

2.2 Modelos de calidad de Software.

Aquí se presentan el modelo ISO-9001-2000 para evaluar el sistema de gestión de calidad de una empresa de software, un modelo relacionado con la calidad como es la ISO/IEC 9126, y el modelo personal PSP.

2.2.1 ISO-9001-2000.

De acuerdo con (Sevilla, 2014) es una norma internacional que determina los requerimientos de un sistema de gestión de calidad de una organización, enfocándose a que los productos satisfagan los requisitos de los clientes y con esto su satisfacción. Para que una organización posea un sistema de gestión de calidad esta debe tomar en consideración lo siguiente:

i) Identificar los procesos que se requieren.
ii) Secuencia e interacción de los procesos.
iii) Métodos que permitan operación y control del manejo eficaz de los procesos.
iv) Disponibilidad de recursos e información como apoyo de los procesos.
v) Seguimiento, medición y análisis de los procesos.
vi) Implementar acciones para cumplir la planificación.
vii) Mejora continua de los procesos.

Esta norma analiza el sistema de calidad desde las siguientes áreas y sub áreas:

a) Responsabilidad de la dirección.

a.1) Compromiso de la dirección.

Establecer o evidenciar un compromiso con el sistema de calidad y su mejora continua.

a.2) Enfocándose al cliente.

La dirección se enfoca a asegurar que los requisitos de los clientes se realicen a fin de satisfacerlo.

a.3) Política de la calidad.

La dirección se compromete con la política de calidad y determina si es la adecuada para la organización según los objetivos de calidad, siendo de ser necesaria su modificación.

a.4) Planificación.

El Nivel Directivo debe asegurarse de que la integridad del sistema de calidad se mantiene, a través de la planificación, para asegurar que los requisitos y objetivos se cumplan, independientemente de los cambios que se realicen.

a.5) Responsabilidad, autoridad y comunicación.

La alta dirección debe verificar que existen responsabilidades asignadas en el sistema de calidad dentro de la organización, así como determinar la existencia de procesos de comunicación adecuados.

a.6) Revisión por la dirección.

La alta dirección revisará el sistema de gestión de la calidad en intervalos planificados a fin de que éste sea eficaz y se lo pueda mejorar.

b) Gestión de los recursos.

b.1) Provisión de recursos.

La organización debe entregar los recursos necesarios para la implementación o mantención del sistema de gestión de calidad.

b.2) Recursos Humanos.

El personal que trabaja con el sistema de gestión de calidad, debe tener las competencias, habilidades y experiencia adecuada para producir un producto adecuado.

b.3) Infraestructura.

La organización debe establecer una infraestructura adecuada que permita cumplir con los requisitos del producto y satisfacción del cliente (edificios, espacio de trabajo, software, hardware y servicios de apoyo).

b.4) Ambiente de trabajo.

La organización debe establecer un adecuado ambiente de trabajo que permita llegar a tener un producto de acuerdo a los requisitos del mismo.

c) Realización del Producto.

c.1) Planificación de la realización del producto.

La organización debe establecer procesos y una adecuada planificación para realizar el producto.

c.2) Procesos relacionados con el cliente.

La organización debe verificar los requisitos del producto antes de tener una aceptación formal ante el cliente (contratos, ofertas, etc). Además debe buscar la forma de tener una comunicación adecuada con los clientes.

c.3) Diseño y desarrollo.

La organización debe realizar la planificación y control del diseño del producto (revisión, verificación, control de cambios).

d) Medición, análisis y mejora.

d.1) Generalidades.

La organización debe realizar la planificación y establecer procesos de seguimiento, de análisis y de mejora de la realización del producto a través del sistema de gestión de calidad.

d.2) Seguimiento y medición.

La organización debe realizar un seguimiento de la satisfacción del cliente con respecto al cumplimiento de los requisitos del producto elaborado a través de métodos relacionados.

2.2.2 ISO/IEC 9126 – Modelo de Calidad

Según (Padayachee, Kotze, & van Der Merwe, 2010) es un modelo de calidad enfocado al software tomando en consideración tres aspectos: calidad interna, calidad externa y calidad de uso. La calidad interna y externa está formada de seis características (funcionalidad, confiabilidad, utilidad, eficacia, capacidad de mantenimiento y portabilidad) y la calidad de uso está determinada por cuatro características (eficacia, productividad, seguridad física y satisfacción). A continuación se describen las características:

a) Calidad Interna y Externa:

a.1) Funcionalidad: Capacidad del software de cumplir con sus funcionalidades (requerimientos funcionales).

a.2) Confiabilidad: Capacidad del software para mantener nivel de prestaciones, durante un tiempo y condiciones establecidas.

a.3) Usabilidad: Capacidad del producto software para ser comprendido, utilizado e interesante por el usuario según condiciones establecidas.

a.4) Eficiencia: Capacidad del producto software para tener un desempeño adecuado en relación con los recursos para su uso, según condiciones establecidas.

a.5) Mantenibilidad: Capacidad del producto software para ser modificado (adaptarse a nuevos ambientes y requerimientos, especificaciones).

a.6) Portabilidad: Capacidad del producto software para ser transportado de un ambiente a otro.

b) Calidad de Uso:

b.1) Eficacia: Capacidad del producto software para alcanzar de manera exacta y completa los objetivos requeridos por el usuario.

b.2) Productividad: Capacidad del producto software para utilizar adecuadamente los recursos a fin de mantener una eficacia adecuada, según condiciones específicas de uso.

b.3) Seguridad Física: Capacidad del producto software para afrontar y mitigar riesgos que puedan causar un daño a personas, a la razón de ser del negocio, al software según condiciones específicas de uso.

b.4) Satisfacción: Capacidad del producto software para satisfacer las necesidades de los clientes o usuarios según condiciones específicas de uso.

Como se observa en la Ilustración 2.2 el modelo permite que al mejorar la calidad del proceso, se mejore la calidad del producto y con esto la calidad de su uso, así también si se evalúa la calidad en su uso, mejorara el producto y con esto se mejorará el proceso.

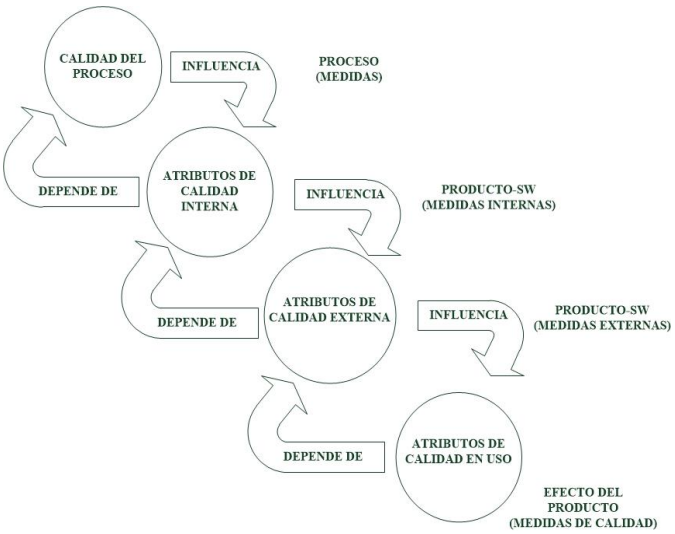


Ilustración 2.2: Modelo de calidad ISO/IEC 9126
Fuente: Elaboración Propia.

Para medir la calidad del producto software (sus características) es necesario utilizar métricas que permitan evaluarlas. Existen tres tipos de métricas:

i. Métricas Internas: Estas se aplican en las fases de diseño y codificación (software no ejecutable) permiten asegurar calidad externa y calidad en su uso.

ii. Métricas externas: Estas se aplican al producto software, basándose en su comportamiento a través de pruebas, utilización y observación al ser este ejecutado.

iii. Métricas de calidad en el uso: Estas se aplican a resolver necesidades del usuario como eficacia, productividad, seguridad y satisfacción (enfocándose a su usabilidad).

2.2.3 Modelo personal – PSP

Basado en (Ampuero & López Trujillo, 2010) para poder desarrollar software de calidad, no solo se requiere de un proceso formal de desarrollo o pruebas de software si no que es necesario un personal capacitado y disciplinado que lleve a cabo las tareas de una manera profesiones (eficiente y eficazmente), siendo un aspecto importante PSP(proceso de software personal) en el mejoramiento de procesos a través de un personal disciplinado y competente.

Según (Watts S Humphrey, Madrid, 2001.) determina que la Ingeniería de Software enfocado como un trabajo es: "la entrega de productos de alta calidad de acuerdo con un costo y un cronograma fijado", para lo cual es importante tomar en consideración lo siguiente planificar , llevar a cabo lo planificado de la mejor manera posible a fin de tener productos de calidad. De acuerdo con (Ampuero & López Trujillo, 2010) los ingenieros de software tienen inconvenientes para llevar a acabo lo planificado de una manera disciplinada por lo siguiente:

i. La ingeniería de software no se enfocó a realizar su trabajo de una manera disciplinada por parte de su personal.
ii. Por ser el desarrollo de software algo que no se realiza a gran escala, razón por la cual no se ve a la disciplina como un factor determinante.
iii. Todo trabajo requiere de estándares para ser disciplinado, la producción de software necesita de una formación en los distintos roles de sus responsabilidades.

Siendo trascendental que los ingenieros de software usen, conozcan y apliquen métodos disciplinados que les permitan realizar de mejor forma su trabajo individual (planificar, registrar tiempos, defectos, medir la calidad del producto).

Fundamentados en ("The Personal Software Process. Technical Report CMU/SEI-2000-TR-022," 2000) el diseño del PSP se basó en los siguientes principios de planificación y mejora de la calidad:

i. Al ser un ingeniero un individuo diferente a los demás, este debe ser eficiente planificando su trabajo según su propia trayectoria.
ii. Los ingenieros deben utilizar procesos definidos y cuantificables para poder realizar una mejora continua.
iii. La calidad de un producto de software es responsabilidad del ingeniero quien lo construyó.
iv. Si se detectan defectos y estos se corrigen a tiempo generarán un menor esfuerzo.
v. Es mejor evitar defectos que luego tener que detectarlos y corregirlos.

vi. La mejor forma de trabajar es hacerlo bien esto resulta más efectivo, rápido y económico.

Además según (Pomeroy-Huff M., 2005) como se observa en la Ilustración 2.3 el área del conocimiento está conformada por conceptos y habilidades , conformando competencias según el área en la que se aplique.

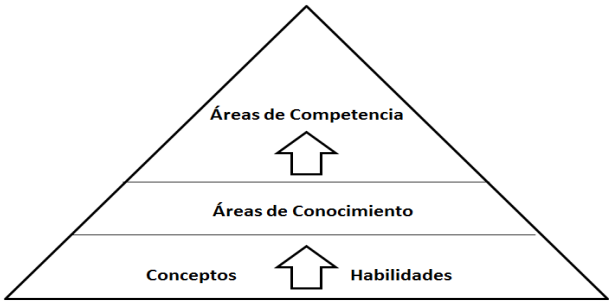


Ilustración 2.3: Estructura del Conocimiento en PSP.
Fuente: Elaboración Propia.

A continuación se detallan de manera general cada uno de los elementos del conocimiento de PSP:

a. Áreas de Competencia.

   Son los fundamentos conceptuales y habilidades correspondientes que un ingeniero de software necesita para realizar una actividad propia de la profesión.

b. Áreas del conocimiento.

   Es la experiencia (comprensión y capacidad) adquirida por un ingeniero de software en un área de competencia especifica.

c. Concepto.

   Conocimiento basado en un principio dentro de área profesional determinada

d. Habilidad.

   Destreza que se va adquiriendo mientras un ingeniero de software se entrena o pasa una experiencia referente a cualquiera de las áreas de conocimiento que este requiera.

Además PSP distribuye en siete áreas de competencia y estas se describen a continuación:

i. Conocimiento Fundacional.

   Competencia cuyo objetivo son los conceptos básicos de mejora del proceso, utiliza métodos estadísticos y es donde se construye PSP.

ii. Conceptos Básicos.

   Competencia cuyo objetivo son los principios fundamentales donde se construye PSP.

iii. Medición y Estimación de Tamaño.

   Su objetivo es definir tamaño y conceptos de estimación (datos históricos).

iv. Realización y seguimiento de los planes del proyecto.

   Su objetivo es utilizar la estimación para planificar y dar seguimiento a un proyecto de software (agenda con sus respectivas tareas y responsabilidades).

v. Planificación y Seguimiento de la Calidad.

Su objetivo es producir productos que satisfagan las necesidades de los clientes o usuarios (medir logros alcanzados) y sean de calidad.

vi. Diseño del software.

Competencia cuyo objetivo es que PSP agregue el diseño y su verificación

vii. Extensiones y personalización del proceso.

Competencia cuyo objetivo es la modificación que se debe realizar a PSP cuando se pasa de un programa pequeño a uno grande.

2.3 Participantes en el proceso de pruebas.

A continuación se exponen los roles y participantes que intervienen en un modelo de pruebas funcionales. Basados en (Board, 2005) se establecen dos roles: líder y tester, ambos requieren manejar una buena comunicación interpersonal de defectos, progreso y riesgos del software a probar de una manera positiva. El líder de las pruebas planifica, realiza un seguimiento y control de las actividades, sus tareas son:

i. Determinar o revisar la política y estrategias de pruebas.
ii. Planificar estrategias de las pruebas (coordina con los involucrados).
iii. Planifica la integración.
iv. Planifica las pruebas según los riesgos asociados además estimar tiempo, esfuerzo, costo, recursos ciclos y seguimiento de incidentes.

v. Específica, prepara, implementa, monitoriza y controla la ejecución de las pruebas.
vi. Establecer resultados (informes) y progreso de las pruebas.
vii. Modificar la planificación de las pruebas según resultados obtenidos.
viii. Administrar la gestión de la configuración (determinar trazabilidad).
ix. Establecer métricas que establezcan el progreso de las pruebas y la calidad del producto.
x. Decir que y como se van a automatizar las pruebas.
xi. Determinar herramientas de apoyo a las pruebas.
xii. Capacitación para los testers en herramientas de apoyo.
xiii. Determinar la aplicación del ambiente de pruebas.
xiv. Realizar informes de las pruebas realizadas.

Las tareas de tester se detallan a continuación:

i. Apoyar en la ejecución de los planes de prueba.
ii. Analizar y examinar los requerimientos (especificaciones) del producto versus necesidades del cliente.
iii. Elabora las pruebas (especificaciones)
iv. Instala y configura el ambiente de pruebas.
v. Adquiere los datos de prueba.
vi. Implementar, ejecutar y documentar los resultados de las pruebas
vii. Evaluar resultados y documentar inconvenientes.
viii. Determinar herramientas para administrar o monitorear las pruebas.
ix. Automatizar las pruebas.
x. Inspeccionar pruebas realizadas por otros testers.

2.4 Evaluación Comparativas de los métodos de pruebas de software.

A continuación se realiza una descripción y análisis de modelos especializados de pruebas de software más importantes que existen.

2.4.1 TMM y TMMI

Según (Informática, 2008) TMM (Test Maturity Model) es un modelo de referencia centrado en el Proceso de Pruebas. La mejora del proceso de pruebas es soportada por único área de proceso que define un conjunto de niveles y objetivos de madurez. Propone 5 niveles y describe una estructura interna implementada en cada uno de ellos tal y como se ve en la Ilustración 2.4

Ilustración 2.4: Estructura de TMM.
Fuente: Revista Española de Innovación, Calidad e Ingeniería de Software[1]

Basados en (Informática, 2008) TMMI (Test Maturity Model Integration) es un modelo detallado para mejorar el proceso de prueba, es la evolución de TMM

[1] Tomado de la página: http://www.ati.es/IMG/pdf/SanzVol4Num4.pdf

(Test Maturity Model) y están basados en CMMI, posee cinco niveles de madurez específicos para pruebas, como se ve en la Ilustración 2.5, su aplicación ha dado buenos resultados sobre la calidad del producto, razón por la cual su utilización va en aumento.

Ilustración 2.5 Estructura de TMMI.
Fuente: Revista Española de Innovación, Calidad e Ingeniería de Software[2]

De acuerdo con (Informática, 2008) TMMI (Test Maturity Model Integration) es un modelo que permite mejorar el proceso de prueba, es un avance a TMM (Test Maturity Model), utiliza cinco niveles de madurez, su aplicación ha dado buenos resultados sobre la calidad del producto, razón por la cual su utilización va en aumento, como se observa en la Ilustración 2.6.

[2] Tomado de la página: http://www.ati.es/IMG/pdf/SanzVol4Num4.pdf

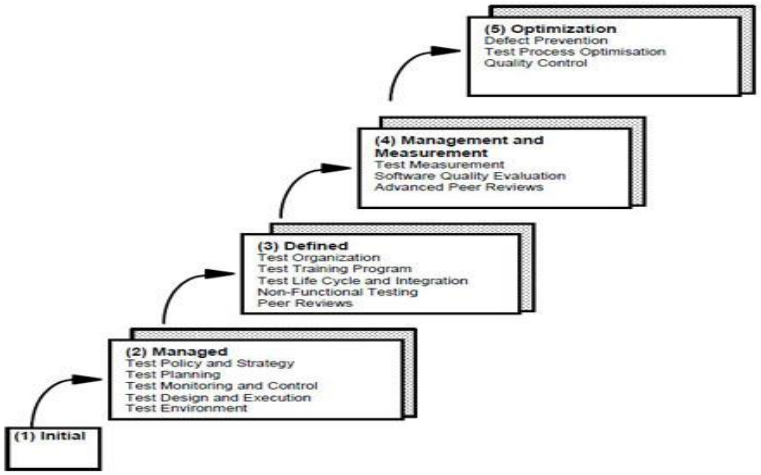


Ilustración 2.6 Niveles de TMMI.
Fuente: The TMMI Foundation [3]

a. Niveles TMMI.

a.1 Nivel 1 (Iniciación).

En este nivel, las pruebas se realizan de manera informal no se tiene definido un proceso adecuado, se realizan cuando la etapa de construcción se ha finalizado, detectando errores a través de la depuración es decir no se contempla un análisis de calidad y de riesgos del producto, lo cual se refleja en el incumplimiento de tiempos de entrega e inconsistencia con los requerimientos del usuario.

a.2 Nivel 2 (Planificación).

En este nivel, las pruebas se realizan de manera formal, se posee definido un proceso adecuado que permite su administración, diferenciándose de la depuración. Se determinan estrategias, un plan de pruebas basado en un análisis de riesgo previo, mediante el uso de técnicas de gestión, siendo este documento el que dice que, cuándo, cómo y quién probará. Además se hace un seguimiento y control de lo

[3] Tomado de la página: http://www.tmmi.org/images/tmmilevels.jpg

planificado, a fin de tomar correcciones de ser necesario. Aquí se confirma que el producto cumple con sus requisitos. Sus procesos son:

i. Política y Estrategia de Pruebas.
ii. Planificación de Pruebas.
iii. Seguimiento y Control de Pruebas.
iv. Diseño (pruebas funcionales) y Ejecución de Pruebas.
v. Entorno de Pruebas.

a.3 Nivel 3. Definición.

En este nivel, ya no se consideran las pruebas como algo posterior a la construcción del producto, si no que por el contrario está presente en el ciclo de vida de desarrollo, se realiza la planificación de manera anticipada mediante un plan de pruebas. Se mejoran los procesos de pruebas, se crea un organismo de pruebas y la capacitación correspondiente para su personal. Además se establecen normas, procesos más detallados y sus procedimientos correspondientes. Sus procesos son:

i. Organización de Pruebas.
ii. Programa de Formación para las Pruebas.
iii. Ciclo de vida de las Pruebas e Integración.
iv. Pruebas No Funcionales.
v. Revisión de pares.

a.4 Nivel 4 (Gestión y Medición).

En este nivel, el proceso de pruebas está definido y es capaz de ser medido, se utiliza estadística y se hacen auditorias como forma de controlar su calidad, siendo parte importante en la gestión. Se integran pruebas dinámicas y pruebas estáticas, se evalúan en función de atributos de calidad (fiabilidad, facilidad de uso, mantenimiento) relacionando productos probados y de trabajo en todo el ciclo de vida. Sus procesos son:

i. Medidas de Pruebas.
ii. Evaluación de la Calidad de productos.
iii. Revisión avanzada de pares.

a.5 Nivel 5 Optimización.

En este nivel el proceso está completamente definido, llegándose a controlar costos y eficacia, mediante la prevención de defectos y control de calidad además se plantean métodos y técnicas de mejora. Siendo fundamental poseer un procedimiento para la selección y evaluación de herramientas especializadas, que apoyan a todo lo relacionado con las pruebas. Sus procesos son:

i. Prevención de defectos.
ii. Optimización del Proceso de Pruebas.
iii. Control de Calidad.
iv.

2.4.2 TPI (Test Process Improvement).

De acuerdo con (Sogeti, 2010) TPI es un modelo de mejora de procesos de pruebas basado en Sogeti siendo su estructura la que muestra en la Ilustración 2.7, considera herramientas, diseño, técnicas y reportes de test.

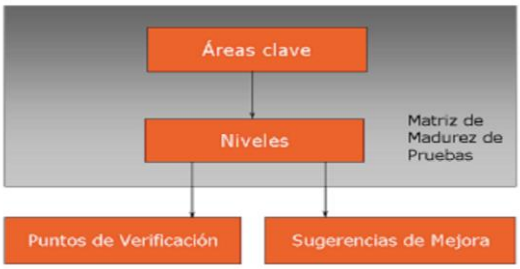


Ilustración 2.7 Estructura TPI
Fuente: El Grupo Sogeti [4]

Basado en (Wallmüller, 2007) el proceso de pruebas posee 20 áreas claves que permiten determinar su madurez y su mejoramiento como se muestra en la Ilustración 2.8, considerando su escala: Ad-hoc (escala 0), Controlado (escala 1-5), Eficiente (escala 6-10) y Optimización (escala de 11 a 13). Los niveles (A, B, C, D) es una manera de establecer el grado de madurez para cada área, siendo A el nivel más bajo y D el más alto. Esta matriz permite identificar que partes del proceso son fuertes o débiles y así de esta manera mejorar.

	Scale													
Key area	0	1	2	3	4	5	6	7	8	9	10	11	12	13
		Controlled					Efficient					Optimizing		
Test Strategy		A					B				C		D	
Life Cycle Model		A			B									
Moment of Involvement			A				B				C		D	
Estimating and Planning				A							B			
Test Specification Techniques		A		B										
Static Test Techniques					A		B							
Metrics						A			B			C		D
Test Tools					A			B			C			
Test Environment				A				B						C
Office Environment				A										
Commitment and Motivation		A				B						C		
Test Functions and Training				A			B				C			
Scope of Methodology					A						B			C
Communication			A		B							C		
Reporting		A			B		C					D		
Defect Management		A				B		C						
Testware Management			A			B				C				D
Test Process Management		A		B								C		
Evaluation							A			B				
Low-level testing					A		B		C					

Ilustración 2.8 Áreas Claves de Madurez
Fuente: Áreas Claves de Madurez [5]

4 Tomado de la página: www.es.sogeti.com/Global/Fichas%20comerciales/Ficha_TPI.pdf
5 Tomado de la página: www.itq.ch/pdf/tpi/2007_TPI_Intro_eV10.pdf

2.4.3 Test PAI

Basados en (Informática, 2008) es un modelo de pruebas que trabaja con CMMI permitiendo mejorar procesos propios de la organización y el proceso de pruebas. Se encuentra organizado en objetivos y prácticas específicas como se observa en la Ilustración 2.9, los objetivos son:

i. Establecer objetivos y políticas asociadas a las pruebas.
ii. Planificar pruebas realizando y manteniendo un plan de pruebas que considere el alcance, los riesgos, estrategias y recursos asociados.
iii. Especificación de casos de prueba que permitan realizar pruebas eficientes y eficaces a través de detalles no contemplados en el plan de pruebas como son entradas, salidas, resultados
iv. Ejecución exitosa de las Pruebas, es asegurar el entorno de las pruebas para una correcta ejecución de las mismas.
v. Análisis y Reporte de los Resultados analiza y evalúa el proceso de pruebas a fin de mejorarlo.

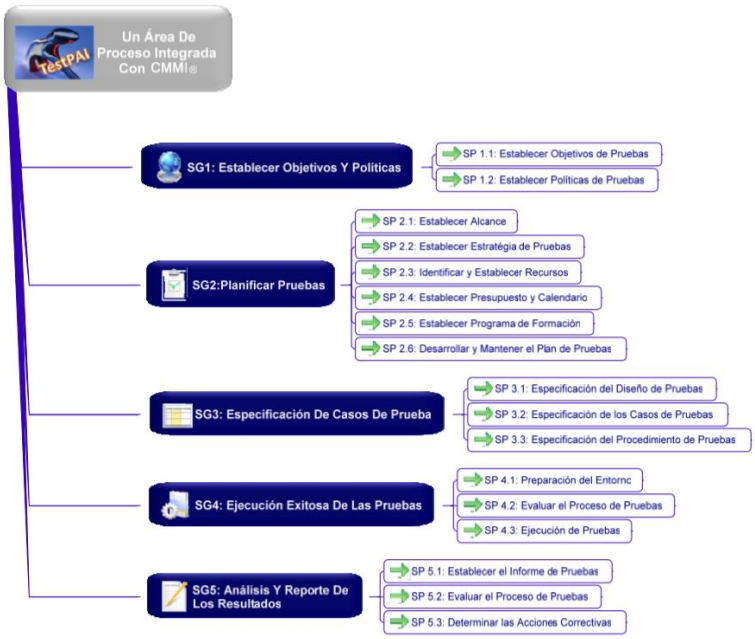


Ilustración 2.9 Objetivos Y Prácticas Específicas de Test PAI.
Fuente: Áreas Claves de Madurez [6]

2.4.4 TMAP

Según (Sogeti, 2010) es una metodología para pruebas de software (planificación, elaboración y control), sus objetivos son: Hallar fallas importantes en menor tiempo y costo, optimar la calidad, conocer la calidad y riesgos del producto. Posee 4 partes como se observa en la Ilustración 2.10 a continuación se las describe de manera general:

A. Business-driven Test Management.

6 Tomado de la página: http://www.ati.es/IMG/pdf/SanzVol4Num4.pdf

A.1 Características:

- Gestionar riesgos, tiempo, costo y resultados.
- Seguimiento y control medido a través de objetivos.
- Administración de test fundamentado en riesgos.

A.2 Pasos a seguir como guía:

1. Formular la misión y recolectar objetivos de las pruebas.
2. Determinar las categorías de riesgo.
3. Determinar cobertura alta/baja.
4. Se realiza una estimación global para las pruebas y la Planificación creada.
5. Asignar técnicas de pruebas.
6. Proporcionar al cliente información sobre el proceso de test.

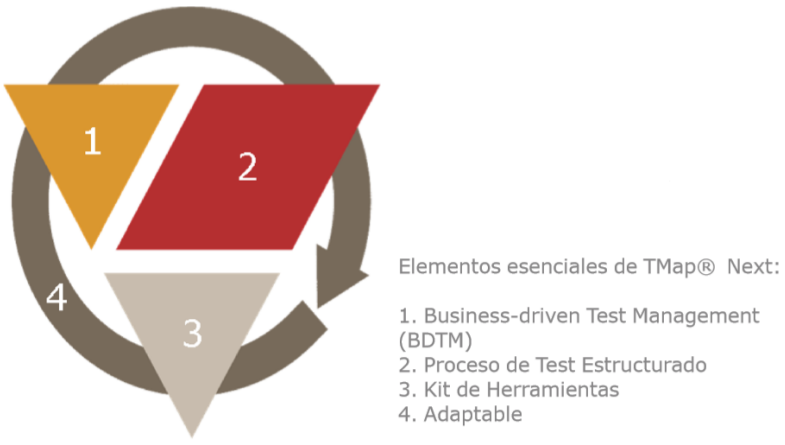


Ilustración 2.10 Elementos de TMap.
Fuente: TMap® Next - Aproximación a la gestión de pruebas [7]

7 Tomado de la página: www.es.sogeti.com/Global/Fichas%20comerciales/Ficha_TMap.pdf

B. Proceso de Pruebas Estructurado está formado de:

i. Plan Maestro de Pruebas es donde se realiza toda la planificación y el control del proceso que permita encontrar fallas en un tiempo y costo mínimo.

ii. Prueba de Aceptación y Sistema está basado en el modelo de ciclo de vida como se observa en la Ilustración 2.11, pudiendo ser utilizado en cualquier etapa de las pruebas, determinando responsabilidades y fases con sus correspondientes actividades y entregables.

B.1 Tests de Desarrollo, está basado en el modelo de ciclo de vida como se observa en la Ilustración 2.11, siendo un referente para el trabajo que los programadores realizan.

B.2 Procesos de soporte, se enfoca a que es más eficiente centralizar el soporte de las pruebas, para lo cual existen dos tipos de formas de organizarse:

i. Organización de pruebas permanente conocida como centro de experiencia de prueba (TEC).

ii. La organización de pruebas permanente conocida como fábrica de prueba (TF).

C. Kit de herramientas está enfocada a :

C.1 La infraestructura (¿dónde? ¿con qué?), se determina el entorno de las pruebas DTAP (desarrollo, test, aceptación y producción) y sus condiciones. Utilizan herramientas de pruebas a fin de mejorar productividad, calidad entre otros aspectos.

C.2 Las técnicas a usarse (¿cómo?), estas incluyen:

i. Estimación de Test
ii. Gestión de Defectos
iii. Creación de Métricas
iv. Análisis de Riesgos del Producto
v. Diseño de Test
vi. Evaluación del Producto

C.3 Y que organización (¿quién?) específica roeles, responsabilidades y tareas correspondientes, teniendo que el tester poseer competencias específicas.

D. Adaptable está formado de:

D.1 Respuesta a los cambios se refiere a que el Plan Maestro de pruebas se adapta a los cambios requeridos por los clientes en sus fases iniciales.

D.2 Re utilización de productos y procesos, se basa en ser rápidos en la utilización a través del uso de checklists, plantillas entre otros. Siendo importante determinar que actividades permiten la reutilización y conservar objetivos (testware).

D.3 Aprendizaje desde la experiencia, es poseer un histórico que permita utilizar lo aprendido anteriormente, permitiendo al proceso mejorar y evolucionar siendo parte importante las métricas.

D.4 Probar antes de usar, se utilizan ensayos de pruebas un enfoque muy importante en TMap, para adecuarse a si a los inconvenientes.

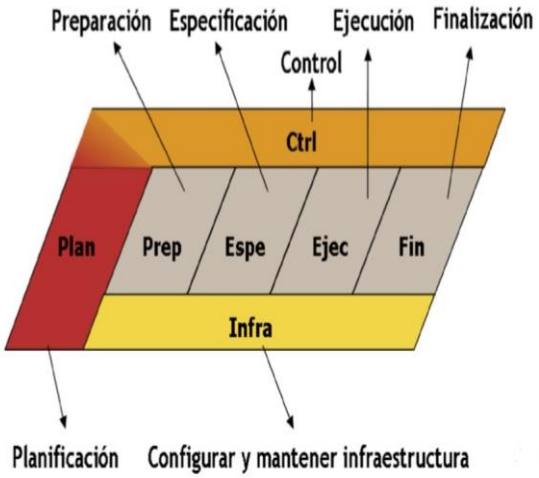


Ilustración 2.11 Elementos de TMap.
Fuente: TMap® Next - Aproximación a la gestión de pruebas [8]

2.5 Definiciones.

Para poder entender las Pruebas de Software, es necesario saber de sus términos básicos como son: prueba (testing), verificación, validación, error, defecto y falla. Además se discuten los requerimientos de software y su involucramiento con las pruebas.

8 Tomado de la página: www.es.sogeti.com/Global/Fichas%20comerciales/Ficha_TMap.pdf

2.5.1 Pruebas.

De acuerdo con (DAICH, PRICE, RAGLAND, & DAWOOD, 1994) existen dos maneras de realizar pruebas de software de una manera dinámica o estática. Estáticamente al verificar el producto sin ejecutarlo, utilizando mecanismos automatizados y manuales. Este trabajo se basó en la (SWEBOK, 2004) donde se utiliza una verificación dinámica (conjunto finito de casos de prueba vs el comportamiento esperado). Según la (SWEBOK, 2004) se definen los siguientes términos :

Prueba es una actividad que permite la evaluación de la calidad de un producto (en su totalidad o en una parte), siendo su objetivo el mejoramiento del mismo mediante la búsqueda de defectos e inconvenientes.

Prueba de Software es la verificación dinámica del comportamiento de un producto versus el comportamiento deseado mediante el uso finito de casos de prueba.

Dinámica significa que las pruebas siempre ejecutan las entradas (valores) del programa o programas a ser evaluados.

Finito significa que no se puede probar todo, por lo que se debe delimitar el número de casos de prueba de acuerdo a los recursos, horarios y requisitos de las pruebas.

Comportamiento Esperado significa determinar si los resultados obtenidos después de la ejecución del programa son aceptables o no, tomando en consideración los intereses del usuario (validación), las especificaciones del producto (requisitos implícitos o expectativas razonables y su correspondiente verificación).

2.5.2 Verificación y Validación de una Prueba

Los procesos de verificación y validación (actividad o etapa que pertenece al proceso de construcción del software) son sumamente importantes (distintos el uno del otro) para determinar la calidad de un producto. Según la (IEEE, 1990b) la definición de verificación y validación de una prueba de software es:

La verificación es el proceso de evaluación de un sistema o componente de software para determinar si los productos de una fase de desarrollo dado, cumplen los requisitos impuestos al inicio de esa fase.

La validación es el proceso de evaluación de un sistema o componente de software durante o al final del ciclo de desarrollo con el fin de determinar si satisface los requisitos especificados.

Un producto software de acuerdo con (Sommerville., 1995) debe ser verificado y validado en cada una de las fases de desarrollo, usando documentación adquirida en etapas anteriores, siendo sus definiciones:

La verificación es ratificar que el programa respeta su especificación.

La validación es ratificar que el programa satisface las expectativas del usuario. Otros autores definen los conceptos de verificación y validación. De acuerdo con (Abadia, 2010):

La verificación es el intento de encontrar defectos ejecutando un programa en un ambiente de prueba (simulado).

La validación es el intento de encontrar defectos ejecutando un programa en un ambiente real. Basado en (Bahill, Bharathan, & Curlee, 1995):

La verificación consiste en hacer chequeos que certifiquen que el sistema funciona de acuerdo con las especificaciones.

La validación consiste en asegurar que el sistema hace lo que se supone debe hacer, es decir que la especificación este correcta.

Basados en (Puello, 2013):

La Verificación es asegurar que los productos de trabajo seleccionados cumplen sus requerimientos especificados.

La Validación es demostrar que un producto o componente de producto se ajusta a su uso previsto cuando se sitúa en su entorno previsto.

Se podría resumir el proceso de verificación y validación con las siguientes preguntas:

Validación: ¿Se está construyendo el producto correcto de acuerdo con las especificaciones?

Verificación: ¿Se está construyendo el producto correctamente, satisface las expectativas del usuario?

2.5.3 Error, falta, defecto y falla.

Basados en (Abadia, 2010) las pruebas de software se utilizan para descubrir defectos, siendo necesario determinar los siguientes términos (que serán utilizados en este trabajo):

Error: Equivocación o malentendido realizada por una persona (ingenieros, analistas, desarrolladores y testers de software).

Defecto o Falta (fault): Es un error en un ambiente, algoritmo o dato (no contempla sus especificaciones).

Falla (failure): Es cuando se mantiene un defecto, dándose comportamientos anormales (desvió del comportamiento del sistema y no de acuerdo a sus especificaciones) tales como un error, una excepción, un comportamiento erróneo o un fracaso.

Como se observa en la Ilustración 2.12 existen comportamientos anormales, causados por los defectos, dependiendo si estos son detectados o no a continuación se detallan cada uno de ellos:

Error: Cuando se detectó una falla y esta no afecta al normal desenvolvimiento del sistema, pudiéndose continuar.

Excepción: Cuando se detectó una falla y esta se reconoce sin poderla manejar, sin afectar al normal desenvolvimiento del sistema, pudiéndose continuar.

Comportamiento erróneo: Cuando no se detecta una falla y esta no causa una transgresión visible en las especificaciones del sistema.

Fracaso: Cuando no se detecta una falla y esta causa transgresión visible en las especificaciones del sistema.

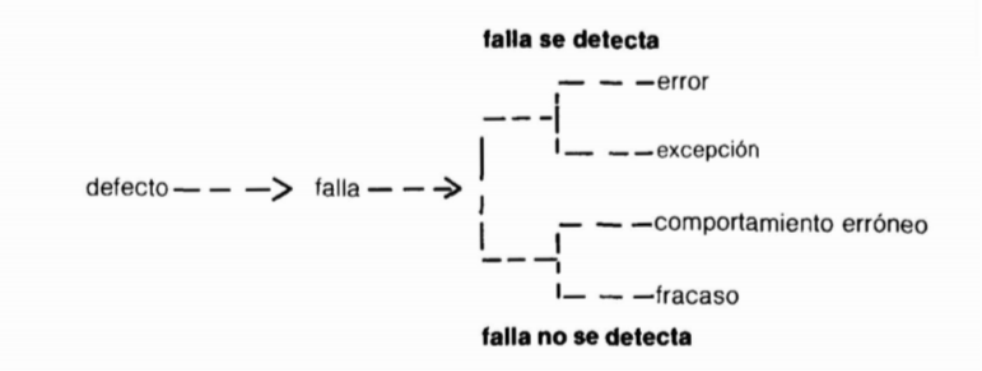


Ilustración 2.12: Estructura de defectos y fallas.
Fuente: Procedimiento para Pruebas de Software. [9]

Cabe destacar que un término que también se puede encontrar es el denominado bug que según (Kaner, Bach, & Pettichord, 2001) no es más que un defecto o una falla (cualquier cosa que pueda estar mal en el software) que hace que este pierda lo valioso para el usuario. Se puede asociar con artefactos de software tales como los requisitos y documentos de diseño. Los defectos que ocurren en estos artefactos también son causados por errores y suelen detectarse en el proceso de revisión.

2.5.4 Requerimientos y su relación con la Prueba.

Según (Beizer, 1990b) los requerimientos son que debe hacer y /o características del software a probar (deben ser consistentes, completos, realizables y verificables), para lo cual se diseñan pruebas que demuestren su veracidad.

[9] Tomado de (Abadia, 2010)

Los requerimientos funcionales determinan las funciones del software a probar y los requerimientos no funcionales son aquellos que tienen que ver con probar los atributos de calidad (seguridad, confiabilidad, portabilidad entre otros).

Basados en (Pytel et al., 2011) la especificación de requerimientos de software es una descripción de cómo se va a comportar el sistema (antes de su desarrollo) tomando en consideración las interacciones con los usuarios, se incluyen requerimientos no funcionales.

Es adecuado que la organización donde se ejecuten las pruebas disponga de un estándar formal (pudiendo ser el IEEE 830-1998) para el levantamiento de requisitos, a fin de que se estos sean comprensibles, consistentes y completos, permitiéndose que estos puedan ser validados (software correcto - software que los usuarios quieren).

Según la (SWEBOK, 2004) una de las tareas más importante de las pruebas es la de planificar la verificación los requerimientos, utilizando por lo general las pruebas de aceptación.

2.5.5 Elementos de la Prueba.

Existen elementos básicos que se deben considerar al momento de ejecutar una prueba de software y estos son según el estándar (IEEE, 1990a) :

Prueba (test): Una actividad en la que un sistema o componente se ejecuta en condiciones especificadas, se observan los resultados o registrados, y se realiza una evaluación de algún aspecto del sistema o componente. Un conjunto de uno o más casos y procedimientos de prueba.

Caso de Prueba (test case): Es un conjunto de entradas, precondiciones de ejecución, resultados esperados y pos condiciones desarrollados como un objetivo en particular o condición de prueba, como ejercitar una ruta de programa en particular o para verificar el cumplimiento de un requisito específico.

Procedimiento de prueba (test procedure): Son Las instrucciones detalladas para la puesta en marcha, ejecución y evaluación de los resultados de una prueba determinada. Es la documentación que especifica una secuencia de acciones para la ejecución de una prueba. Un caso de prueba puede ser usado en más de un procedimiento de prueba

Basados en (Board, 2012) se determinaron los siguientes términos :

Ciclo de prueba (test cycle): Es la ejecución del proceso de pruebas nuevamente en contra de una versión del producto en prueba.

Conjunto de Pruebas (test suite): Es Un conjunto de varios casos de prueba para un componente o sistema bajo prueba, donde la post condición de una prueba se utiliza a menudo como una condición previa para el próximo ciclo.

Datos de prueba (test suite): Son datos que existen (por ejemplo, en una base de datos) antes de que una prueba se ejecuta, y que afecta o se ve afectada por el componente o sistema bajo prueba.

Ejecución de la prueba (test execution): Es el proceso de ejecución de una prueba en el componente o sistema bajo prueba, produciendo un resultado real.

Programa de Pruebas (test suite): Es un programa que especifica la prueba es decir determina el procedimiento de la misma.

Resultado real (actual result): Es el comportamiento producido / que se observa cuando un componente o sistema se prueba.

Resultado esperado (expected result): Es el comportamiento previsto por la especificación, o de otra fuente, del componente o sistema en las condiciones especificadas.

2.6 Pruebas de Software.

2.6.1 Consideraciones respecto a la Prueba.

De acuerdo con (G. J. Myers, 1979) determino mediante un programa simple, que probarlo en su totalidad conllevaría 100 trillones de posibilidades, pudiendo un tester probarlo en un billón de años. De acuerdo con (Myers, 2004) las pruebas de software deben establecer como diseñar y seleccionar los casos de prueba adecuados, en el tiempo requerido, permitiendo reducir costos.

Al momento de la ejecución de una prueba, su razón es determinar cuando el programa fallo, siendo primordial conocer sus datos de entrada y su salida esperada según (Beizer, 1990a), lo que permite decidir si se comportó correctamente es decir

si pasó o fallo, dando como resultado si cumple o no con su especificación pero según (Kaner C., 1999) las especificaciones pueden estar mal hechas.

2.6.2 Requisitos basados en las pruebas.

Hablar de requisitos funcionales es hablar del insumo principal al momento de hacer pruebas (lo que el software debe hacer - su funcionamiento), estos permiten su elaboración y diseño por parte de los testers, siendo un inconveniente que estos no estén disponibles, estén incompletos o mal realizados. Siendo un aspecto importante que los testers pueden indicar la calidad del producto al finalizar la ejecución de la prueba mediante la validación de los requerimientos a través de una planificación, generalmente con el diseño de pruebas de aceptación según la (SWEBOK, 2004)

2.6.3 Actitud frente a las pruebas.

Basado en (Kaner C., 1999) el objetivo de las pruebas es que los testers ejecuten casos de prueba que hagan que el programa falle y no verificar que funcione correctamente, utilizándose posteriormente a estos para asegurar de alguna manera.

2.6.4 Clasificaciones de la Prueba.

No existe una clasificación formal de los diversos tipos de pruebas de software, siendo una clasificación aceptable según el tipo de prueba que se esté haciendo: de acuerdo (Whittaker, 2002) puede ser funcional o estructural y según cómo se realiza (unitaria, de integración o del sistema) siendo la prueba funcional la que se utiliza para este trabajo.

a. Prueba estructural vs. Funcional.

La prueba estructural se la conoce como caja blanca (código fuente del software) su objetivo es determinar los caminos del programa a ser evaluados en las pruebas y la pruebas funcional conocida como caja negra (especificaciones del software) su objetivo es validar si cumple con sus especificaciones (enfoque del usuario) realizando entradas y observando sus salidas, según (Kaner C., 1999) es raro que se considere el código fuente.

Las pruebas funcionales utilizan, la especificación del producto para diseñar los casos de prueba (entrada – salida esperada), existiendo técnicas como partición de equivalencia, análisis del valor límite, grafo causa-efecto y conjetura de errores y de acuerdo con (Myers, 2004) la prueba funcional debe mostrar errores y disconformidades con la especificación y no enfocarse a determinar si el programa cumple con su especificación.

b. Prueba unitaria, de integración y del sistema.

Las pruebas se clasifican en unitarias, de integración o del sistema, pudiendo estas ser enfocadas según el objeto a probar, de acuerdo con (SWEBOK, 2004) este puede cambiar según: por el propósito, el uso, el comportamiento o su estructura.

b.1 Prueba Unitaria.

Fundamentado en (Kit, 1995) se prueba los componentes individuales (subprogramas o procedimientos) de un programa, a fin de determinar disconformidades entre la especificación de su interface y su comportamiento. Cabe destacar que el software más pequeño a probar se lo denomina unidad (cien o menos líneas de código).

b.2 Prueba de Integración.

Basado en (Beizer, 1990a) es probar componentes grandes que resultan de la agregación de otros más pequeños (pasados por las pruebas unitarias), siendo esta correcta o incorrecta. Sus estrategias para software estructurado jerárquicamente (modulo principal llama a sub-módulos y estos a los de nivel inferior) son:

b.2.1 Incremental.

Se prueba mediante pruebas unitarias las partes del software, luego la combinación de las partes (prueba de integración) creado grupos integrados, así hasta tener todo el sistema completo. De acuerdo con (Kaner C., 1999) existen dos formas de realizarse Bottom up don0de los módulos de nivel bajo, se prueban primero, luego los del nivel siguiente según la jerarquía y así repetidamente. Top down donde los módulos de nivel más alto, se prueban primero, luego los de menor nivel que son invocados siguiente según la jerarquía y así repetidamente.

b.3 Prueba de Sistema.

Basado en (SWEBOK, 2004) se prueba el sistema completo, los inconvenientes respecto a lo funcional ya se resolvieron en las pruebas de integración, es recomendada para examinar requerimientos no funcionales

(seguridad, desempeño, confiabilidad, fiabilidad, etc) además se evalúa la infraestructura o entorno. Cabe destacar que cada tipo de prueba (unitaria, integración o sistema) puede enfocarse desde la parte funcional o estructural. De acuerdo con (Myers, 2004) los tipos de pruebas de sistema son:

b.3.1 Prueba de Volumen.

Su objetivo es exponer al sistema a un volumen de datos exagerado con el fin de demostrar que este no puede ser estable como se manifiestas sus especificaciones, se necesitan recursos significativos (hardware y personal) para su ejecución.

b.3.2 Pruebas de Estrés.

Su objetivo es que el programa soporte cargas pesadas (volumen máximo de datos o una acción en un lapso corto), lo más importante aquí es el tiempo de respuesta.

b.3.3 Pruebas de Usabilidad.

Su objetivo es determinar inconvenientes entre la interacción entre el sistema y el usuario sin causar esfuerzo en su utilización.

b.3.4 Pruebas de Seguridad.

Su objetivo es vulnerar la seguridad del sistema, y establecer la manera de asegurar la confidencialidad de la información (evaluar métodos utilizados).

b.3.5 Pruebas de Desempeño.

Su objetivo es la eficiencia del programa enfocándose a sus tiempos de respuesta y rendimiento (características a ser evaluadas) según la conIlustracióncIón y carga de trabajo, aquí el diseño de los casos de prueba permiten medir su desempeño.

b.3.6 Pruebas de Almacenamiento.

Su objetivo es medir la capacidad de los programas en su almacenamiento (memoria principal y secundaria, capacidad de la base de datos), aquí el diseño de los casos de prueba permiten establecer falencias en este aspecto.

b.3.7 Pruebas de Configuración.

Su objetivo es evaluar compatibilidad, conversión y conIlustraciónción según su hardware y software que el sistema requiera, aquí el diseño de los casos de prueba permiten establecer que la compatibilidad y los procedimientos de conversión no funcionen adecuadamente.

b.3.8 Pruebas de Instalación.

Su objetivo es evaluar la instalación del producto, siendo un factor importante para el cliente o usuario, ya que esto puede ser fácil o dificultoso dependiendo del entorno de producción donde se encuentre (plataformas informáticas y configuraciones de hardware).

b.3.9 Pruebas de Confiabilidad.

Su objetivo es mejorar la confiabilidad del programa evaluando la precisión que posee en relación a las prestaciones para cuales fue diseñado (especificaciones originales versus fallos).

b.3.10 Pruebas de Recuperación.

Su objetivo es evaluar el procedimiento que el programa posee para su recuperación en caso de existir fallas de hardware o software, siendo necesario demostrar que este no funciona correctamente.

b.3.11 Pruebas de Mantenimiento.

Su objetivo es determinar si el programa permite ser reajustado (procedimientos, documentación, etc) de acuerdo con el cambio de los requerimientos que el cliente o usuario propone, ya que los escenarios de los negocios cambian continuamente.

2.7 Otros Tipos de Pruebas.

Además de las pruebas antes mencionadas existen otros tipos, como son las denominadas de regresión y de humo las cuales se detallan a continuación:

2.7.1 Pruebas de Regresión.

Su objetivo es verificar que los cambios realizados a los programas, no causen nuevos defectos (alterando su calidad), siendo necesario según (Black, 2002) repetir la prueba o pruebas de una versión anterior y compararlas con la versión actual que entregan los desarrolladores (aparentemente sin defectos), las veces que sean necesarias hasta solucionar los problemas.

De acuerdo con (Whittaker, 2002) un arreglo específico en un programa, puede llegar a:

i. Solucionar solo el defecto que fue encontrado.
ii. Fallar en solucionar el defecto que fue encontrado.
iii. Solucionar el defecto pero dañar otra cosa que antes funcionaba correctamente.

iv. Fallar en solucionar el defecto que fue reportado y dañar otra cosa que antes funcionaba correctamente.

Según (Kaner et al., 2001) existen tres tipos de pruebas de regresión y estas son:

A.1 Regresión de defectos solucionados.

El objetivo es informar sobre una falla que no fue solucionada en una nueva versión del programa.

A.2 Regresión de defectos viejos.

El objetivo es buscar que un cambio en el programa produjo una falla que ya se solucionó volviendo a reaparecer.

A.3 Regresión de efectos secundarios.

El objetivo es que un cambio en el programa produjo que algo que funcionaba ya no funciona.

2.7.2 Prueba de Humo

El objetivo es verificar que cada nueva versión del programa cumple con sus funcionalidades básicas (según sus especificaciones), no es exhaustiva pero ayuda a buscar inconvenientes claves. De acuerdo con (Kaner et al., 2001) si la versión no pasa la prueba de humo, no se realizan a ésta las pruebas planificadas.

2.8 Técnicas de Prueba.

Basados en (SWEBOK, 2004) no existe clasificación homogénea de las técnicas de pruebas dependiendo de varios factores que el ingeniero de software debe considerar como son la experiencia, las especificaciones, la estructura del código, los

fallos (reales o artificiales) por descubrir, el uso de campo, o, por último, la naturaleza de la aplicación. Se determinaron las técnicas de pruebas y cuáles son sus estrategias como se puede observar en la tabla 2.1.

Tabla 2.1: Técnicas de Prueba de Software.

Técnicas de Pruebas	Estrategias
Caja Negra.	Participación de Equivalencia. Análisis del Valor Límite. Tablas de decisión. Máquinas de estado finito. Grafo causa efecto. Prueba de Casos de Uso. Pruebas de Dominio.
Caja Blanca.	Basadas en el Flujo de Control. Basadas en el Flujo de los datos. Mutantes.
Según quién hace la Prueba.	Pruebas de Aceptación. Pruebas Alfa y beta. Pruebas de usuario. Pruebas en pares.
Basadas en la experiencia.	Prueba Ad hoc. Conjetura de errores. Testing exploratorio.

Fuente: Guide to the Software Engineering Body of Knowledge SWEBOK, 2004 version. IEEE Computer Society. 10

A continuación se realiza una descripción de las técnicas de prueba establecidas en la Tabla 2.1

2.8.1 Técnicas de Caja negra.

Son aquellas conocidas como técnicas que prueban la funcionalidad, utilizando como principal insumo, las especificaciones del producto, razón por la

[10] Tomado de http://www.swebok.org

cual estas son sumamente importantes en este trabajo. Comprueban la funcionalidad del software utilizando datos de entrada que produzcan resultados deseados o correctos, no se valida la estructura interna. Las estrategias que se utilizan son las siguientes:

A.1 Partición de Equivalencia.

Pretende seleccionar los casos de prueba que tengan la mayor probabilidad de encontrar errores, ya que no se puede probar todo para lo cual se consideran las siguientes características:

i. Disminuir el número de casos de prueba a desarrollar, tomando en consideración el mayor número de entradas posibles.
ii. Considerar otros casos de prueba con el objetivo de que si se detecta un error en uno de estos, el resto presente el mismo error.

Según (Myers, 2004) para aplicar la partición de equivalencia al diseñar casos de prueba es necesario:

i. Establecer las clases de equivalencia: Mediante el análisis de las entradas que se detallan en las especificaciones, se organizar en grupos los casos de prueba, definiéndose dos clases de equivalencia, las validas (entradas aceptables al programa) y las inválidas (posibles valores erróneos de entradas).

ii. Determinar casos de prueba: Basados en las clases de equivalencia, se establecen los casos de prueba que permitan abarcar mayor cantidad de equivalencias válidas y los casos de prueba que abarquen cada uno de las equivalencias inválidas.

A.2 Análisis del valor límite

Basados en (Myers, 2004) el análisis de valor límite es determinar situaciones límite es decir aquellos valores (entradas y salidas) por arriba y por debajo de las clases de equivalencia.

A.3 Tablas de decisión.

De acuerdo con la (SWEBOK, 2004) las tablas de decisión son representaciones de las relaciones lógicas entre las entradas y salidas de los casos de prueba, es una forma de obtener requerimientos con condiciones lógicas (verdaderas o falsas). Estas se utilizan al tener condiciones del negocio complejas. Cada columna de la tabla identifica una regla del negocio, la cual establece una combinación única de condiciones y acciones a tomar, teniéndose una prueba por cada con sus posibles combinaciones.

A.4 Máquinas de estado finito.

Según la (SWEBOK, 2004) máquinas de estado finito es utilizar las pruebas para determinar estados y transiciones, es decir establecer los cambios que el software sufre según ciertas condiciones ya sean estas actuales o anteriores, permitiendo observar las transformaciones que sucedieron, las causas que las generaron y sus resultados. Se diseñan pruebas que se enfoquen a una secuencia de

estados finitos del software y así ejecutar las diferentes transiciones (validas e inválidas).

A.5 Grafo Causa Efecto

Basados en (Myers, 2004) el grafo causa efecto es una técnica difícil de aplicar al momento de transformar el grafo en una tabla de decisión, no se examinan las posibles combinaciones de las clases de equivalencia. Es adecuada para seleccionar los casos de prueba de manera ordenada. Las especificaciones se las describe en un grafo que relaciona cusas y efectos, los cuales se describen en una tabla de decisión siendo cada columna un caso de prueba.

A.6 Prueba de Casos de Uso.

Fundamentado en (Rumbaugh, Jacobson, & Booch, 1999) el caso de uso es un conjunto de actividades con sus precondiciones y post condiciones, que un sistema ejecuta a través de un actor que es quien interactúa con el sistema, para obtener un resultado deseado. Un caso de uso posee un escenario, un camino principal y varios caminos posibles. Las pruebas de aceptación pueden diseñarse basándose en los casos de uso lo cuales modelan el negocio, siendo una ventaja al momento de detectar defectos, apoyándose en los usuarios o clientes del sistema.

A.7 Prueba de Dominios.

La prueba de dominio determina los posibles valores de las variables (entradas o salidas) para las funciones del sistema para lo cual se utilizan las clases de equivalencia a fin de encontrar mayor número de defectos. Según (Kaner et al.,

2001) la parte importante es el análisis de las variables y luego la ejecución de las pruebas en cada una de las funciones.

2.8.2 Técnicas de Caja Blanca.

Son aquellas técnicas que validan la lógica de los sistemas, a través de pruebas que verifican su estructura interna sin considerar los requisitos. Existe muchas variedades siendo las más conocidas la inspección del código fuente para encontrar errores en la programación como lógica, subutilización y estructura de los datos. En este tipo de técnicas los casos de prueba se fundamentan en la estructura del sistema.

A.1 Basadas en el flujo de control

De acuerdo con la (SWEBOK, 2004) el flujo de control se enfocan en las sentencias o bloques del código fuente de un programa determinando, analizándose las trayectorias del flujo de control desde su entrada a su salida.

A.2 Basadas en el flujo de los datos

Según la (SWEBOK, 2004) el flujo de datos se enfocan en determina el manejo de las variables desde su creación , utilización y su destrucción en un programa.

A.3 Mutantes

Basados en (SWEBOK, 2004) se considera mutante a la versión levemente alterada (cambio sintáctico) del programa sometido a prueba. Cada caso de prueba es ejecutado sobre el mutante y su original, de existir diferencia se lo denomina mutante muerto, generándose pruebas al azar o específicas para matar la cantidad suficiente de mutantes o mutantes sobrevivientes de ser el caso.

2.8.3 Técnicas según quién hace la Prueba

Dependiendo quien realiza la prueba describiéndose a continuación cada una de ellas (pruebas de aceptación, las pruebas alfa y beta, las pruebas de usuario y las pruebas en pares):

A.1 Prueba de Aceptación

Según (Myers, 2004) las pruebas de aceptación permiten determinar si el sistema cumple con los requerimientos iniciales y si este satisface las necesidades del usuario (cliente o usuario final), permiten determinar el cumplimiento de los contratos de desarrollo. Basados en (DAICH et al., 1994) es necesario determinar casos de pruebas que demuestren las funcionalidades principales del sistema para su posterior aprobación.

A.2 Pruebas Alfa y Beta

De acuerdo con la (SWEBOK, 2004) las pruebas alfa y beta permiten probar el sistema con un grupo de usuarios internos denominados alfa y con usuarios externos denominados beta antes poner el producto en producción.

A.3 Prueba de usuario.

Basados en (Kaner et al., 2001) las pruebas de usuario son aquellas que serán realizadas por quienes utilizaran el sistema, pudiendo ejecutarse en la etapa de desarrollo.

A.4 Prueba en pares

Según (Kaner et al., 2001) es trabajar entre dos testers para encontrar fallas en un programa o sistema.

2.8.4 Técnicas basadas en la intuición o en la experiencia.

Estas técnicas de pruebas se enfocan en la experiencia de quien la ejecuta, siendo estas ad hoc, la conjetura de errores y el testing exploratorio.

A.1 Prueba Ad hoc.

De acuerdo con la(SWEBOK, 2004) es una técnica muy utilizada, que depende de habilidad, intuición, y experiencia del ingeniero de software basándose en programas similares, además se manejan para realizar pruebas especiales que no son fáciles de hacer manipulando técnicas más formalizadas.

A.2 Conjetura de errores

Según (Myers, 2004) existen personas que encuentran errores sin utilizar ninguna metodología y basados en sus destrezas o en su intuición, sospechan de

errores, diseñando casos de prueba para comprobarlos, pudiendo estos basarse en aspectos que se omitieron en las especificaciones ya sea por accidente o por ser tan obvias, así como el historial de proyectos pasados.

A.3 Testing Exploratorio

Según la (SWEBOK, 2004) el testing exploratorio es la forma de probar sin poseer un plan de pruebas previo, diseñándose, ejecutándose y modificándose de manera dinámica, basados en la experiencia de los ingenieros de software (comportamiento del producto, familiaridad con la aplicación, la plataforma, el tipo de posibles fallos y fracasos, el riesgo asociado).

Basados en (Bach, 2003) el testing exploratorio es aquel que se basa en controlar el diseño de las pruebas, mientras estas se ejecutan, utilizando la información adquirida para el diseño y optimización de las mismas (minimizando tiempos y costos).

De acuerdo con (Kaner C., 1999) los resultados al ejecutar testing exploratorio, deben ser documentados considerando lo realizado y lo sucedido. Además según (Bach, 2001) estos no son distintos a los conseguidos en pruebas diseñadas previamente, pudiendo utilizarse los dos enfoques.

El testing exploratorio se utiliza cuando es necesaria una retroalimentación rápida del producto en su: funcionalidad, aprendizaje, análisis y asilamiento de un defecto, investigación de un riesgo en particular, evaluación para diseñar pruebas.

Su ventaja es que es medible y de bajo costo pero necesita tester con experiencia y preparación adecuada.

2.9 Planificación de la prueba

Según (Burnstein, 2010) un plan puede definirse, como un documento que proporciona un marco o enfoque para lograr un conjunto de objetivos. En el dominio de software, los planes pueden ser estrictamente orientados a los negocios, por ejemplo, los planes a largo plazo para apoyar el crecimiento económico de una organización, o pueden ser de naturaleza más técnica, por ejemplo, un plan para desarrollar un producto de software específico. Los planes de prueba tienden a ser de carácter técnico, sin embargo, un plan de proyecto de software que puede contener un plan de pruebas y con frecuencia se refiere a los objetivos del negocio.

La planificación de la prueba, es una práctica esencial para cualquier organización que desee desarrollar un proceso de prueba que repetible y manejable. Perseguir los objetivos de madurez integrados en la estructura de TMMI, no es una condición previa necesaria para iniciar un proceso de planificación de prueba, sin embargo, un proceso de prueba de esfuerzo de mejora proporciona un marco adecuado para la adopción de esta práctica esencial. La planificación de pruebas debe comenzar temprano en el ciclo de vida del software, aunque para muchas organizaciones cuyos procesos de prueba son inmaduros, ésta práctica no está todavía en su lugar.

Los hitos son hechos tangibles que se esperan que ocurran en un momento determinado en un tiempo de vida del proyecto. Los administradores los utilizan para determinar el estado del proyecto. El seguimiento de la ocurrencia real de los hitos permite al administrador si el proyecto avanza según lo previsto. Por último los planes de prueba para proyectos de software son documentos muy complejos y detallados. El planificador suele incluir los siguientes objetivos de alto nivel, que son esenciales:

a) Objetivos generales de las pruebas. Como testers ¿por qué estamos probando lo que se logra mediante las pruebas y cuáles son los riesgos asociados con las pruebas de este producto?
b) Qué probar (alcance de las pruebas). ¿Qué elementos, características, procedimientos, funciones, objetos, grupos y subsistemas serán probados?
d) ¿Quién va a probar. ¿Quiénes son el personal responsable de las pruebas?
e) Cómo probar. ¿Qué estrategias, métodos, equipos, herramientas de software y técnicas se van a aplicar? ¿Qué documentos de prueba y entregables deben producirse?
f) Cuándo hacer la prueba. ¿Cuáles son los calendarios de las pruebas? ¿Qué elementos deben estar disponibles?
g) Cuándo dejar de probar. No es económicamente viable o práctico, planear probar, hasta que todos los defectos hayan sido revelados. Este es un objetivo que los probadores nunca pueden estar seguros de que han alcanzado. Debido a los presupuestos, cronogramas y los plazos de los clientes, las condiciones específicas que se indica en el plan de pruebas que permita a los testers y gerentes decidir cuando la prueba se considera completa.

Los planes de prueba se pueden organizar de varias maneras dependiendo de la política de la organización. A menudo existe una jerarquía de planes que incluye varios niveles de calidad y de aseguramiento de planes de prueba. La complejidad de la jerarquía depende: del tipo, tamaño, riesgo propensión, criticidad, misión y seguridad del sistema de software a desarrollar. Todos los planes de calidad y las pruebas también deben coordinarse con el plan de proyectos de software en general. Una jerarquía de plan se muestra en la Ilustración 2.13

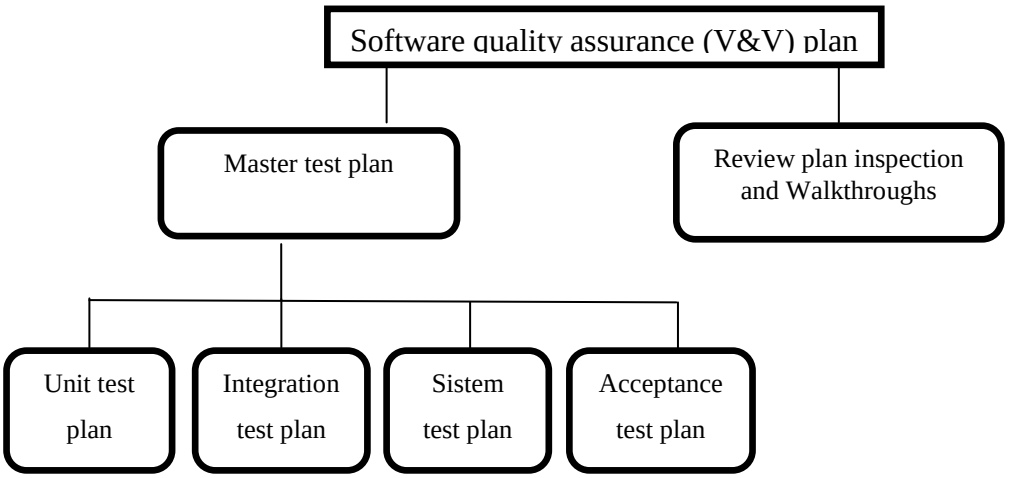


Ilustración 2.13: Jerarquía de Planes de Prueba.
Fuente: Practical Software Testing. [11]

En la parte superior de la jerarquía del plan puede existir un plan de aseguramiento de calidad del software. Este plan ofrece una visión general de todas las actividades de verificación y validación para el proyecto, así como los detalles relativos a los problemas de calidad tales como auditorías, normas, control de configuración y control de proveedores.

[11] Tomado de (Burnstein, 2003)

A continuación vemos que en la jerarquía del plan puede existir, un plan de pruebas maestro, que incluye la descripción de todas las pruebas de ejecución basada en el sistema de software. Un plan maestro para la verificación de las inspecciones opiniones y tutoriales también encajaría en este nivel. El plan de pruebas maestro en sí mismo puede ser un componente del plan general del proyecto o existir como un documento separado.

Según la política de organización, otro nivel de la jerarquía podría contener un plan de pruebas independiente para: la unidad, la integración, sistema y pruebas de aceptación. En algunas organizaciones que son parte del plan de prueba principal, los planes de nivel dan una visión más detallada de las pruebas adecuadas a ese nivel.

De acuerdo con la (IEEE, 1990b), existen descripciones útiles para muchos de estos planes , otros tipos de pruebas y documentos relacionados, tales como los planes de verificación y validación.

Las personas responsables de la elaboración de planes de pruebas dependerán del tipo de plan en desarrollo, generalmente el personal de uno o más grupos cooperan en su desarrollo. Es esencial que las actividades de desarrollo se coordinen para que progrese el proyecto sin problemas.

La jerarquía de la organización y el responsable del desarrollo del plan de pruebas, debe ser especificado en los estándares de la organización y documentos de garantía de calidad de software.

Un plan de pruebas de propósito general se conoce como un "Plan de pruebas." Su contenido se basa en una discusión de sus componentes que se recomiendan en el estándar (IEEE, 1983), donde se citan: ejemplos y plantillas relacionadas que permitan ser una guía a los planificadores.

2.9.1 Componentes del plan de pruebas.

Basados en la (IEEE, 1983) el plan de pruebas posee componentes básicos sean estos para el principal y cada nivel base (unidad, integración, etc) con la cantidad de detalle apropiado.

Según (Burnstein, 2010) se deben tomar en cuenta algunos de los elementos de un plan de pruebas con otros documentos relacionados, como el plan del proyecto, tomándose en cuenta dichas referencias como se observa en la tabla 2.2

Tabla 2.2: Componentes del Plan de Prueba.

1)	Identificador del plan
2)	Introducción
3)	Ítems a ser probados
4)	Características a ser probadas
5)	Enfoque
6)	Pasar criterios de fallas
7)	Criterios de suspensión y reanudación
8)	Pruebas entregables
9)	Tareas de prueba
10)	Entorno de pruebas
11)	Responsabilidades
12)	Dotación y necesidad de entrenamiento de personal
13)	Programación
14)	Riesgos y contingencias
15)	Costos de prueba

Fuente: Practical Software Testing. [12]

[12] Tomado de (Burnstein, 2003)

En la parte superior de la jerarquía del plan puede existir un plan de aseguramiento de calidad del software. Este plan ofrece una visión general de todas las actividades de verificación y validación para el proyecto, así como los detalles relativos a los problemas de calidad tales como: auditorías, normas, control de configuración y control de proveedores. En la jerarquía del plan puede existir un plan de pruebas maestro que incluye la descripción de todas las pruebas de ejecución, basado en el sistema de software. Un plan maestro para la verificación de las inspecciones opiniones/tutoriales también encajaría en este nivel). El plan de pruebas maestro en sí mismo puede ser un componente del plan general del proyecto o existir como un documento separado.

2.10 Conclusiones.

Se analiza la evolución histórica de las pruebas de software llegando a determinar las propuestas que se destacaron en cada una de las etapas correspondientes como se observa en la tabla 2.3. Siendo importante destacar que en 1983 se estableció el estándar 829-1983 y actualmente existe el estándar IEEE 829-2008 donde se determinan un conjunto de documentos de pruebas de software pudiéndose ser utilizados por su formato.

Tabla 2.3 Evaluación histórica de las pruebas de software.

Evolución histórica de las pruebas de Software.		
AÑO	**AUTOR**	**PROPUESTA**
1930	Walter Shewhartv	Trabajó en el mejoramiento de los procesos, utilizando control estadístico de la calidad.
1979	Glenford Myers	La ingeniería del software separe las disciplinas fundamentales del desarrollo del software de la verificación y validación.
1986	W. Edwards Deming	Ciclo Deming que fue utilizado en la mejora continua de la calidad dentro de la empresa (Planear, Hacer, Verificar y actuar).
1988	Dr. Dave Gelperin y el Dr. William C. Hetzel	Realizaron una clasificación basada en los modelos de pruebas más influyentes de la época.
Etapa 1: Hasta 1956	**La orientación de las pruebas a la solución de errores.**	
1950	Alan Turing	Escribe el primer artículo basado en pruebas del software, que hoy en día se podrían denominar "pruebas de corrección".
Etapa 2: 1957-1978	**La orientación de las pruebas a la demostración.**	
1957	Charles Baker	Distinguió el concepto de "depuración" del de "prueba" en su revisión "Mathematical Tables and Other Aids to Computation"del libro "Digital Computer Programming" de Dan MacCracken. En el cual se destacaba que la verificación de los programas perseguía dos objetivos fundamentales: • Asegurar el funcionamiento de la aplicación • Asegurar que la aplicación resolvía el problema para el que había sido planteada
1975	Boyer	"A system to generate test data and symbolically execute programs."
1976	Clarke	," Automated test data generation"
1976	Rama	"On the automated generation of program test data"

Continua ⟶

1976	Howden	Publica la primera aproximación teórica hacia como diseñar métodos sistemáticos que puedan ser utilizados para construir pruebas funcionales .
Etapa 3: 1979-1983	**La orientación de las pruebas a la detección**	
1979	Myers	Describió el modelo orientado a la detección y definió éstas como "El proceso de ejecutar un programa con la intención de encontrar errores". el objetivo es demostrar que si un programa falla, los datos de prueba deberían tener una alta probabilidad de detectarlo.
1981	Miller	"Software Testing and Validation Techniques" se analizó el uso de las pruebas en el desarrollo del software y su empleo en las técnicas de revisión
1982	Bird y Muñoz	Nueva metodología orientada a la automatización de casos de prueba denominada "aleatoria". Publicada en "Automatic generation of random self-checking test cases"
Etapa 4: 1983-1987	**La orientación de las pruebas a la evaluación y prevención**	
1983	"Institute for Computer Sciences and Technology of the National Bureau of Standars"	Publica una línea de investigación (NBSR 1983) especialmente orientada a los sistemas de procesamiento de información federal (FIPS). En la cual se describe una metodología que integra el análisis, la revisión y las actividades de prueba con el fin de ofrecer una evaluación del producto a lo largo del ciclo de vida del software, estableciendo la necesidad de obtener en cada fase del ciclo de vida un conjunto de actividades y de productos asociados
1983	Comité técnico de ingeniería del software del IEEE	Publica el estándar ANSI/IEEE STD 829-1983 [IEEE83]

Continua ⟶

1984	Congreso del gobierno americano	Aprobó la creación de un organismo de investigación para el desarrollo de modelos de mejora para los problemas en el desarrollo de los sistemas de software, y evaluar la capacidad de respuesta y fiabilidad de las compañías que suministran software al Departamento de Defensa (DoD), dando lugar al Instituto de Ingeniería del Software (SEI)
1985	Gelperin y Hetzel	Publican su artículo "The growth of Software Testing" en el que hablan de la trayectoria que ha seguido el proceso de pruebas.
1987	IEEE	Publica el estándar ANSI/IEEE STD 1008-1987 (IEEE88) Un año después de comenzarse a desarrollar el estándar orientado a las pruebas unitarias, en 1985 sus autores introdujeron aquel proceso a lo largo de los distintos niveles de prueba existentes, dando como resultado una metodología conocida como "El proceso de evaluación y de pruebas sistemáticas" (STEP)
1988	Hetzel	Definió un sistema de tareas de pruebas, productos y roles con el fin de dar consistencia y salvar costos a la hora de alcanzar los objetivos propuestos en las pruebas

Fuente: Elaboración Propia.

Luego de analizar la información investigada sobre los modelos más importantes de procesos de pruebas de software, se realizó una comparación como se muestra en la Tabla 2.4, determinándose criterios que permitan compararlos con TMMI, estos fueron publicación, licenciamiento, definido, niveles, factorías, características importantes, riesgos.

Tabla 2.4 Criterios de comparación de modelos más importantes de pruebas de Software.

Criterio	Definición
Publicación.	Año de publicación.
Licenciamiento	Autorización que el autor o autores de los modelos , conceden para su utilización (Privado – Abierto)
Definido.	Especificación del modelo es completa.
Niveles	Se puede establecer el nivel de madurez.
Factorías	Su implementación es sencilla para una factoría de software.
Riesgos	Los modelos consideran las pruebas basadas en los riesgos.

Fuente: Elaboración Propia.

Basándonos en el análisis realizado anteriormente sobre los modelos más importantes de pruebas de software como se observa en la tabla 2.5 se determinan las siguientes conclusiones:

i. Ninguno de los modelos poseen un proceso adecuado y flexible de verificación y validación que permita utilizarse por todo tipo de empresas de software sean estas pequeñas, medianas o grandes.

ii. TMMI y TMM son modelos para mejorar los procesos de pruebas de software, poseen 5 niveles de madurez, cumpliéndose los procesos y actividades a cumplir dependiendo el nivel. Siendo su debilidad la validación y verificación dependiendo esto, de la empresa para su realización. Además para pequeñas empresas no es aconsejable por su complejidad pudiéndose utilizar si se recortan los procesos a cumplir. Otra desventaja es que solo se poseen directrices de su utilización, teniendo que las empresas definir los

aspectos de su realización, se puede utilizar esto como base para la implementación de los procesos o para evaluar procesos ya existentes.

iii. TPI es un modelo de mejora del proceso de pruebas, mediante acciones actividades y tareas, pero no es un mecanismo que permita estructurar o formar uno desde cero.

iv. TMAP es un modelo orientado a la gestión de las pruebas a través de actividades y tareas, pero no es un modelo de procesos de pruebas de software, no es aconsejable para pequeñas o medianas empresas, debido a su complejidad y extensión.

v. Test PAI es un modelo de pruebas similar a TMMI y TMM permite mejorar los procesos de pruebas ya existentes, no es un mecanismo para promover uno, pero es menos complicado y extenso por lo que se podría aplicar a una empresa pequeña.

vi. Todos los modelos de mejoramiento del proceso de pruebas toman en consideración la gestión de riesgos.

La solución propuesta en esta tesis es determinar un modelo que permita establecer un proceso de pruebas formal cumpliendo con los aspectos básicos del nivel 2 de TMMI y recortando ciertos parámetros a fin de que este sirva para una empresa pequeña o mediana y no sea algo muy complejo o extenso, ajustándose a sus necesidades.

Tabla 2.5 Resultados de Comparación.

Criterio/Modelo	TMM	TMMI	TPI	Test PAI	TMap
Publicación	1996	2008	1998	2008	1995
Licenciamiento	Propietario	Propietario	Propietario	Propietario	Propietario
Definido	Si	Si	Si	Si	Si
Niveles	Si	Si	Si	Si	Si
Factorías	No	No	No	Si	No
Riesgos	Si	Si	Si	Si	Si

Fuente: Elaboración Propia.

CAPÍTULO 3

MODELO FORMAL DE PRUEBAS FUNCIONALES PROPUESTO

3. INTRODUCCIÓN.

En este capítulo se establece el modelo formal de pruebas funcionales propuesto en ésta tesis a fin de que la empresa Siredcom alcance el nivel 2 de TMMI. Se inicia con el diagnóstico del sistema de gestión de calidad que posee la empresa, aplicando el estándar de calidad ISO-9001-2000 tomando en consideración la responsabilidad de la dirección, la gestión de los recursos, la realización del producto, la medición, el análisis y la mejora. Posteriormente se realiza un diagnóstico del nivel de TMMI que la empresa posee a fin de conocer el nivel en el que se encuentra, para luego presentar una propuesta de un modelo formal de pruebas que permita alcanzar el nivel 2 de TMMI, en la empresa SIREDCOM.

3.1 Diagnóstico del Sistema de Calidad de la Empresa Siredcom.

3.1.1 Estándar de calidad de Software a nivel de proceso.

Se utilizó la lista de chequeo, basada en la ISO-9001-2000, que especifica los requisitos de un Sistema de gestión de calidad, la cual permite evaluar y demostrar su capacidad, a fin de cumplir con los requerimientos de los clientes y su satisfacción.

Tabla 3.1: Lista de Chequeo, con base en la ISO-9001-2000 referente al Sistema de Gestión de Calidad

Aptdo.	REQUISITO	*SI*	*NO*	
	CAPÍTULO 4 SISTEMA DE GESTIÓN DE LA CALIDAD			
4.1	**La organización:**			
	a) Identifica los procesos necesarios para el sistema de gestión de la calidad y su aplicación a través de la organización		X	
	b) Determina la secuencia e interacción de estos procesos		X	
	c) Determina los métodos y criterios requeridos para asegurar: el funcionamiento efectivo y el control de los procesos		X	
	d) Asegura la disponibilidad de recursos e información necesarios para apoyar el funcionamiento y el seguimiento de los procesos.		X	
	e) Mide, realiza el seguimiento y analiza estos procesos		X	
	f) Implanta las acciones necesarias para alcanzar los resultados previstos y la mejora continua de estos procesos.		X	
	Si la organización tiene contratado externamente cualquier proceso que afecte a la conformidad del producto con los requisitos:			
	• ¿Se asegura el control sobre tales procesos?		X	
	• ¿El control de dichos procesos contratados externamente está identificado en el sistema de gestión de la calidad?		X	
4.2.1	La documentación del sistema de gestión de la calidad incluye:			
	a) declaraciones documentadas de una política de la calidad y de objetivos de la calidad,		X	
4.2.2	¿El manual de la calidad incluye:			
	• El campo de aplicación del sistema de gestión de la calidad, incluyendo detalles de, y justificación para, cualquier exclusión?		X	
	• Una descripción de la interacción entre los procesos incluidos en el sistema de gestión de la calidad?		X	
4.2.3	El procedimiento documentado para el control de documentos contempla entre otras, las disposiciones necesarias:			
	• Para asegurar que las versiones pertinentes de los documentos aplicables se encuentran disponibles en los puntos de uso,	x		
	• Para asegurar que los documentos permanecen legibles y fácilmente identificables,	x		
	• Para asegurar que se identifican los documentos de origen externo y que se controla su distribución.	x		
4.2.4	Los registros de calidad permanecen legibles, fácilmente identificables y recuperables		X	

Fuente: Lista de Chequeo. [13]

[13] Tomado de (Ing. Ernesto Sandoval S, 2010)

Tabla 3.2: Lista de Chequeo, con base en la ISO-9001-2000 referente a la Responsabilidad de la Dirección.

Aptdo.	REQUISITO	*SI*	*NO*	
	CAPÍTULO 5 RESPONSABILIDAD DE LA DIRECCIÓN			
5.1	La alta dirección proporciona evidencia de su compromiso para el desarrollo e implantación del sistema de gestión de la calidad y para la mejora continua de su eficacia por medio de:			
	comunicar a la organización la importancia de satisfacer tanto los requisitos del cliente como los reglamentarios;		X	
	Establecer la política de la calidad;		X	
	Asegurar que se establecen los objetivos de la calidad;		X	
	Llevar a cabo las revisiones por la dirección, y	x		
	Asegurar la disponibilidad de recursos.		X	
5.2	La alta dirección asegura que los requisitos del cliente se determinan y cumplen con el propósito de incrementar la satisfacción del cliente?		X	
5.3	La alta dirección asegura que la política de la calidad:			
	Es adecuada al propósito de la organización;		X	
	Incluye el compromiso de satisfacer los requisitos y de mejorar continuamente fa eficacia del sistema de gestión de la calidad;		X	
	Proporciona un marco de referencia para establecer y revisar los objetivos de la calidad;		X	
	Se comunica y entiende dentro de la organización;		X	
	Se revisa para conseguir que se mantenga adecuada continuamente.		X	
5.4.1	¿Los objetivos de la calidad:			
	- Incluyen aquellos necesarios para satisfacer los requisitos del producto?		X	
	- son establecidos para todas las funciones y niveles relevantes dentro de la organización?		X	
	- Son medibles y coherentes con la política de calidad ?		X	
5.4.2	La alta dirección asegura que:			
	la planificación del sistema de gestión de la calidad se lleva a cabo con el fin de cumplir los requisitos dados en el apartado 4.1, así como los objetivos de la calidad,		X	
	Se mantiene la integridad del sistema de gestión de la calidad cuando se planean e implementan cambios en el sistema de gestión de la calidad.		X	
5.5.1	La alta dirección asegura la definición y comunicación de las responsabilidades, autoridades y su interrelación dentro de la organización.	X		
5.5.2	¿El representante de la dirección:			
	Asegura que se establecen, implantan y mantienen los procesos necesarios para el sistema de gestión de la calidad;		X	
	Informa a la alta dirección del funcionamiento del sistema de gestión de la calidad, incluyendo las necesidades para la mejora;		X	

Continua ⟶

	Asegura que se promueve la toma de conciencia de los requisitos de los clientes en todos los niveles de la organización.		X	
5.5.3	La alta dirección asegura que se establecen los procesos apropiados de comunicación dentro de la organización y que la comunicación se efectúa considerando la eficacia del sistema de gestión de la calidad.		X	
5.6.1	La alta dirección revisa el sistema de gestión de la calidad de la organización a intervalos planificados		X	
5.6.2	La información para la revisión por la dirección incluye información sobre			
	Resultados de auditorías;	X		
	Retroalimentación de los clientes;		X	
	Funcionamiento de los procesos y conformidad del producto;	X		
	Situación de las acciones correctivas y preventivas;	X		
	Seguimiento de las acciones derivadas de las revisiones anteriores de la dirección;	X		
	Cambios planeados que podrían afectar al sistema de gestión de la calidad;		X	
	Recomendaciones para la mejora.	x		

Fuente: Lista de Chequeo. [14]

Tabla 3.3: Lista de Chequeo, con base en la ISO-9001-2000 referente a la Gestión de los Recursos.

Aptdo.	**REQUISITO**	***SI***	***NO***	
	CAPÍTULO 6 **GESTIÓN DE LOS RECURSOS**			
6.2.1	¿Se han definido requisitos de educación (estudios), formación, habilidades prácticas y experiencia para asegurar la competencia del personal que realiza actividades que afectan a la calidad del producto?		X	
6.2.2	¿Se evalúa la efectividad de las acciones adoptadas para dotar de la competencia necesaria al personal de la organización que realiza actividades que afectan a la calidad del producto? ¿Se dispone de algún sistema para concienciar a los empleados de la relevancia e importancia de sus actividades y como contribuyen a la consecución de los objetivos de la calidad?		X	
6.3	La organización determina, proporciona y mantiene las infraestructuras necesarias para lograr la conformidad con los requisitos del producto. La infraestructura incluye, por ejemplo:			
	Edificios, espacio de trabajo e infraestructuras asociadas;		X	
	Equipos para los procesos, tanto hardware como software,		X	
	Servicios de apoyo tales como transporte y comunicación.	x		

Fuente: Lista de Chequeo. [15]

[14] Tomado de (Ing. Ernesto Sandoval S, 2010)

[15] Tomado de (Ing. Ernesto Sandoval S, 2010)

Tabla 3.4: Lista de Chequeo, con base en la ISO-9001-2000 referente a la Realización del Producto.

Aptdo.	**REQUISITO**	*SI*	*NO*	*Observaciones*
	CAPÍTULO 7 REALIZACIÓN DEL PRODUCTO			
7.1	La organización ha planificado y desarrollado los procesos necesarios para la realización del producto. La planificación de la realización del producto es consistente con los requisitos de otros procesos del sistema de gestión de la calidad. En la planificación de la realización del producto, la organización ha determinado, lo siguiente:			
	a) Los objetivos de la calidad y los requisitos para el producto			Parcialmente 50% Existen requisitos del producto mas no un sistema de gestión de calidad establecido formalmente en la organización, por lo cual no existen dichos objetivos.
	b) La necesidad de establecer procesos y documentación, y proporcionar recursos específicos para el producto;			Parcialmente 60% ya que existen procesos que han ido cambiando con el tiempo pero que no han sido documentados desde varios años atrás.
	c) Actividades requeridas de verificación, validación; seguimiento, inspección y ensayos específicos para él producto así como los criterios para la aceptación del mismo;	x		
	d) Los registros que sean necesarios para proporcionar evidencia de que los procesos de realización y el producto resultante cumplen los requisitos. El resultado de la planificación se presenta en forma adecuada para el método de operar de la organización.	x		
7.2.1	¿La determinación de los requisitos relacionados con el producto incluye:			
	• Los requisitos no especificados por el cliente pero necesarios para la utilización prevista o especificada del producto?		x	
	• Los requisitos reglamentarios?		x	

Continua ⟶

7.2.2	Se mantienen registros de los resultados de la revisión de los requisitos relacionados con el producto y de las acciones originadas por la misma.	x		
7.2.3	¿Se han determinado e implantado disposiciones efectivas para la comunicación con los clientes, relativos a:			
	• Información del producto?		x	
	• tratamiento de preguntas, gestión de contratos y pedidos, incluyendo las modificaciones?	x		
	• Realimentación del cliente, incluyendo reclamaciones?			Parcialmente un 30% ya que los usuarios internos del sistema de información, no poseen una cultura organizacional adecuada que permita establecer una retroalimentación inmediata y a tiempo de reclamos que permitan prevenir inconvenientes a futuro, con aquellos procesos que utilizan diariamente en el diferentes módulos que a su cargo tienen para realizar su trabajo
7.3.1	Durante la planificación del diseño y desarrollo la organización determina:			
	• Las etapas de diseño y desarrollo;	x		
	• La revisión, verificación y validación, apropiadas para cada etapa del diseño y desarrollo, y	x		
	• Las responsabilidades y autoridades para el diseño y el desarrollo.	x		
7.3.4	Se mantienen registros de los resultados de las revisiones y las subsiguientes acciones necesarias	x		
7.3.5	Se mantienen registros de los resultados de la verificación y las subsiguientes acciones necesarias	x		
7.3.6	Se mantienen registros de los resultados de la validación y las subsiguientes acciones necesarias	x		
7.3.7	La revisión de los cambios del diseño y desarrollo incluye la evaluación del efecto de los cambios en las partes constitutivas y en el producto entregado. Se mantienen registros de los resultados de la revisión de los cambios y las subsiguientes acciones necesarias.		x	

Continua ⟶

7.4.1	¿Se han definido los criterios para la selección y evaluación periódica de los proveedores?		x	
	¿Se registran los resultados de la evaluación y las subsiguientes acciones de seguimiento de los proveedores?		x	
7.5.2	¿Se han identificado los procesos que requieren ser validados? ¿Se han establecido disposiciones para la validación que incluyan:			
	a) Criterios definidos para la revisión y aprobación de los procesos,		x	
	b) Aprobación de equipos y cualificación del personal,		x	
	c) Utilización de métodos y procedimientos específicos,		x	
	d) Requisitos aplicables a los registros, y		x	
	e) Revalidación		x	

Fuente: Lista de Chequeo. [16]

Tabla 3.5: Lista de Chequeo, con base en la ISO-9001-2000 referente a la Medición, Análisis y Mejora.

Aptdo.	**REQUISITO**	***SI***	***NO***	***Observaciones***
CAPÍTULO 8 MEDICIÓN, ANÁLISIS Y MEJORA				
8.1	La organización planifica e implanta los procesos de seguimiento, medición, análisis y mejora necesarios:			
	para demostrar la conformidad del producto,	x		
	Para asegurar la conformidad del sistema de gestión de fa calidad,		x	
	Para mejorar continuamente la eficacia del sistema de gestión de la calidad. Se incluye la determinación de los métodos aplicables; incluyendo técnicas estadísticas, y la extensión de su utilización.		x	
8.2.1	La organización realiza el seguimiento de la información relativa a la percepción del cliente sobre el grado en que la organización ha satisfecho sus requisitos. Se han determinado los métodos para obtener y utilizar dicha información.		x	
8.2.2	Se planifica el programa de auditorías tomando en consideración el estado y la importancia de los procesos y áreas a auditar, así como los resultados de auditorías previas. Se definen los criterios de auditoría, el alcance de la misma, su frecuencia y metodología.	x		

[16] Tomado de (Ing. Ernesto Sandoval S, 2010)

Continua ⟶

	La selección de los auditores y la realización de las auditorias aseguran la objetividad e imparcialidad del proceso de auditoría. Las actividades de seguimiento incluyen la verificación de las acciones tomadas y el informe de los resultados de la verificación.			
8.2.3	¿Se aplican métodos apropiados para la medida y seguimiento de los procesos de realización necesarios para satisfacer los requisitos del cliente? ¿Estos métodos confirman la continua capacidad de cada proceso parasatisfacer su finalidad prevista?		x	
8.3	Cuando se detecta un producto no conforme después de la entrega o cuando se ha comenzado su utilización, la organización adopta las acciones apropiadas respecto de las consecuencias, o efectos potenciales, de la no conformidad.		x	
8.4	La organización determina, recopila y analiza los datos apropiados para demostrar la adecuación y la eficacia del sistema de gestión de la calidad y para evaluar dónde pueden realizarse mejoras continuas del sistema de gestión de la calidad. Se incluyen los datos generados por las actividades de medición y seguimiento y. por cualquier otra fuente relevante. El análisis de estos datos proporciona información sobre: • la satisfacción del cliente; • la conformidad con los requisitos del producto; • las características y tendencias de los procesos y productos incluyendo las oportunidades para llevar a cabo acciones preventivas, y • los proveedores		x	
8.5.1	La organización mejora continuamente la eficacia del sistema de gestión de la calidad por medio de la utilización de la política de la calidad, objetivos de la calidad, resultados de las auditorias, análisis de datos, acciones correctivas y preventivas y la revisión por la dirección.			Parcialmente Cumplido 50% La organización mejora la calidad de sus productos, por medio de auditorías informáticas internas y externas que se realizan periódicamente, lo cual permite acciones correctivas y preventivas, según autorización de la gerencia. Pero no existe un sistema de gestión de la calidad formal con sus correspondientes políticas y objetivos.

Fuente: Lista de Chequeo.

3.1.2 Análisis de Resultados del Diagnóstico.

Después de haber utilizado la lista de chequeo para realizar un diagnóstico sobre la calidad existente, de la empresa SIREDCOM, se establecieron los siguientes resultados posterior a la verificación de requisitos enfocados a los procesos pertenecientes a la norma UNE-EN ISO 9001:2000 (Sistema de Gestión de la Calidad, Responsabilidad de la Dirección, Gestión de los Recursos, Realización del Producto, y Medición, Análisis y Mejora), como se muestra a continuación:

Tabla 3.6: Resultados del diagnóstico de Sistema de Calidad de la Empresa Siredcom.

PROCESO SEGÚN UNE-EN ISO 9001:2000	CONFORMIDADES	INCONFORMIDADES	PARCIALMENTE	PORCENTAJE DE CONFORMIDAD E INCONFORMIDAD A LA LISTA DE CHEQUEO.
SISTEMA DE GESTIÓN DE LA CALIDAD	**3**	**13**		18,75% Conformidades. 81,25% Inconformidades.
RESPONSABILIDA DE LA DIRECCIÓN	**9**	**23**		28,13% Conformidades. 71,87% Inconformidades
GESTIÓN DE LOS RECURSOS	**1**	**5**		16,66% Conformidades 83,33% Inconformidades
REALIZACION DEL PRODUCTO	**10**	**11**	**1,4**	41,66% Conformidades 24 45,83% Inconformidades. 5,83%Parcialmente Cumplidas. 6.66% Incumplimiento.
MEDICIÓN, ANÁLISIS Y MEJORA	**2**	**6**	**0,5**	22,22% Conformidades. 66.66% Inconformidades. 5,55% Parcialmente Cumplidas. 5,55% Incumplimiento

Fuente: Elaboración Propia.

Los gráficos que se colocan a continuación permitirán realizar un análisis más adecuado de los procesos establecidos en norma UNE-EN ISO 9001:2000, lo que describirá de mejor manera la realidad del departamento de tecnología de la empresa Siredcom.

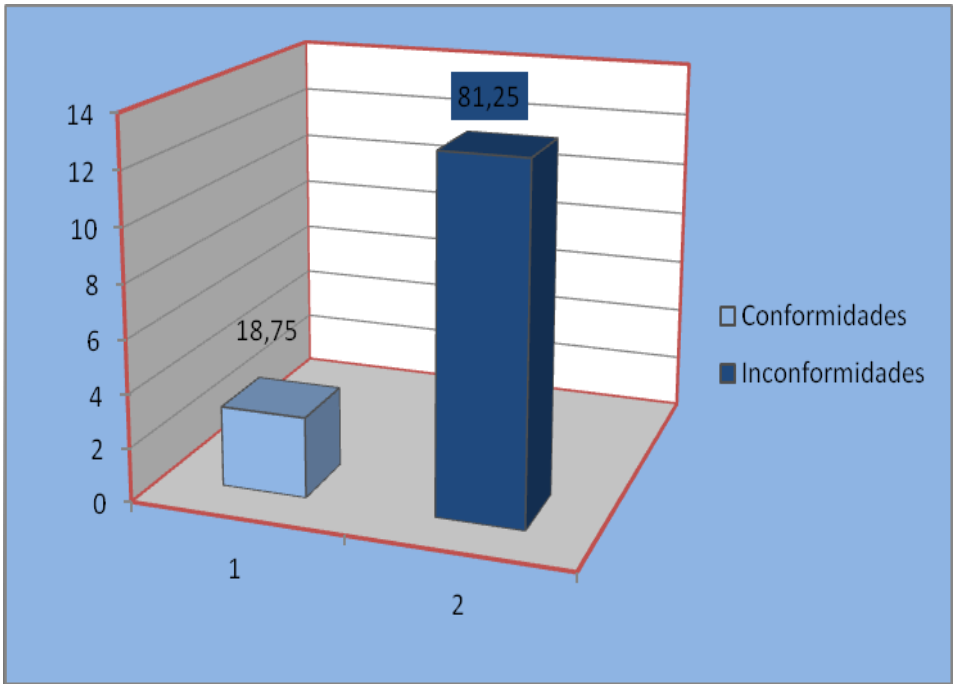


Ilustración 3.1: Proceso - Sistema de Gestión de la Calidad según ISO 9001:2000
Fuente: Elaboración Propia**.**

La Ilustración 3.1 muestra que las inconformidades de los requisitos del proceso del sistema de gestión de calidad están incrementadas en un 62,5% a las conformidades.

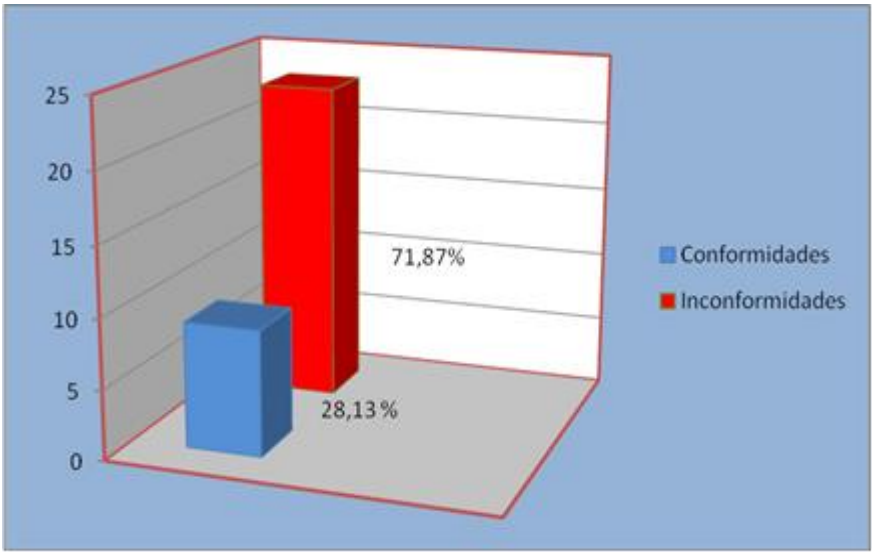


Ilustración 3.2: Proceso – Responsabilidad de la Dirección según ISO 9001:2000

Fuente: Elaboración Propia.

La Ilustración 3.2 muestra que las inconformidades de requisitos correspondientes al proceso de responsabilidad de la dirección están incrementadas en un 43,74% a las conformidades.

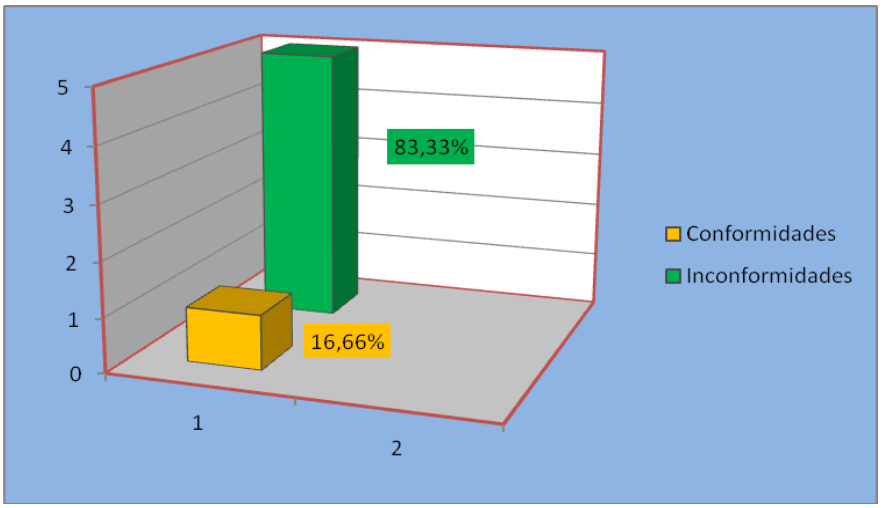


Ilustración 3.3: Proceso – Gestión de Recursos según ISO 9001:2000

Fuente: Elaboración Propia.

La Ilustración 3.3 muestra que las inconformidades de los requisitos correspondientes al proceso de gestión de los recursos están incrementadas en un 66,67 % a las conformidades.

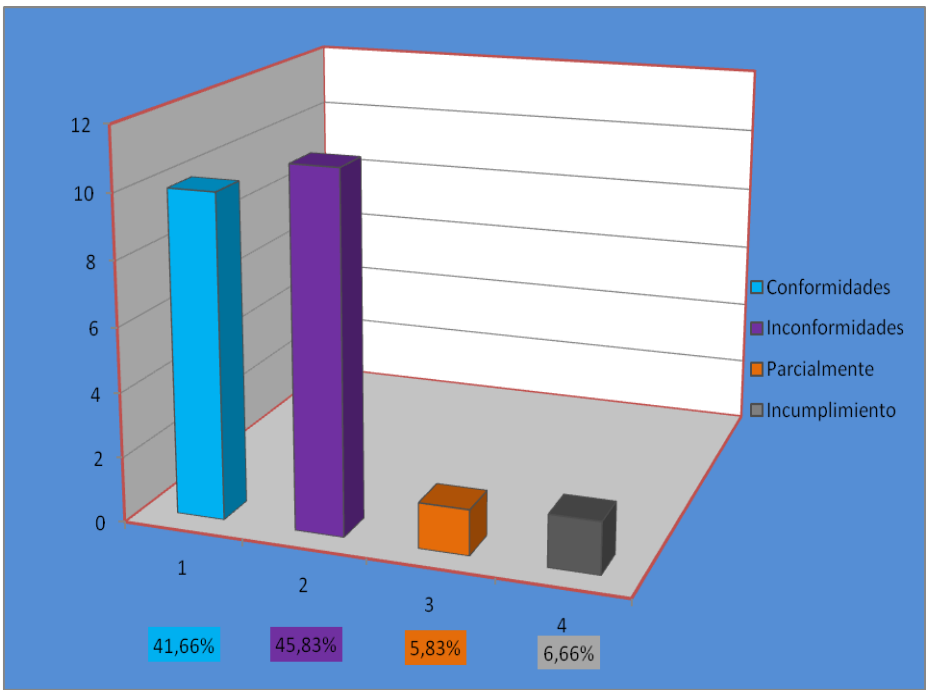


Ilustración 3.4: Proceso – Realización del Producto según ISO 9001:2000

Fuente: Elaboración Propia.

La Ilustración 3.4 muestra que las inconformidades de los requisitos correspondientes al proceso de realización del producto están incrementadas en un 4,17 % a las conformidades. Existiendo Tareas parcialmente cumplidas en 5,83% y tareas incumplidas en un 6,66%, las cuales siendo cumplidas aumentarían las conformidades a un 54,15% balanceando el proceso.

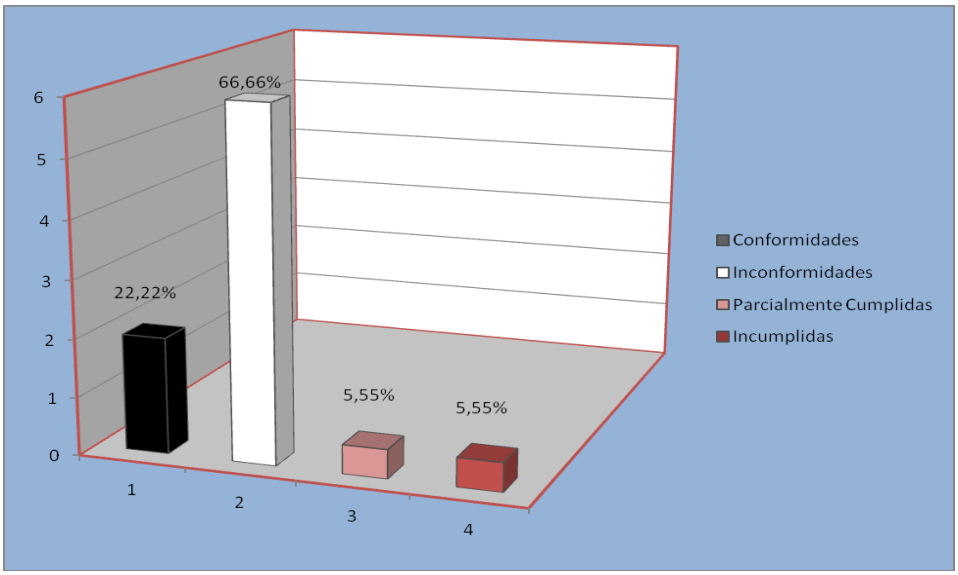


Ilustración 3.5: Proceso – Medición Análisis y Mejora según ISO 9001:2000

Fuente: Elaboración Propia**.**

Las inconformidades de los requisitos correspondientes al proceso de medición análisis y mejora, están incrementadas en un 44,44 % a las conformidades. Existiendo Tareas parcialmente cumplidas en 5,55% y tareas incumplidas en un 5,55%, las cuales siendo cumplidas aumentarían las conformidades a un 33,32% balanceando el proceso.

Al analizar los resultados establecemos que existe mayor número de inconformidades que conformidades en cada proceso pertenecientes a la norma UNE-EN ISO 9001:2000. Se demuestra que la parte que se encuentra en un mejor nivel de conformidades versus inconformidades es el proceso de realización del producto, así como la existencia de requisitos parcialmente cumplidos en dicho proceso.

El sistema de gestión de calidad tiene una cantidad mayor considerable de inconformidades versus conformidades. La responsabilidad de la gerencia se ve muy deteriorada. Se tiene descuidado los la gestión de recursos en la organización. La medición, análisis y mejora poseen tareas parciamente incumplidas que pueden ser una fortaleza para tratar de balancear las inconformidades existentes.

3.2 Diagnóstico del Nivel de TMMI.

Se procede a hacer una evaluación a la empresa Siredcom para determinar el nivel en que se encuentra en referencia al modelo TMMI, iniciando por llenar la información de la organización evaluada como se mira en la tabla 3.7

Tabla 3.7: Información de Evaluación de TMMI.

Información de Evaluación (NIVEL-TMMI)	
Detalles Organización	
Nombre Organización/Departamento	SIREDCOM
Departamentos implicados	GERENCIA – DESARROLLO
Responsable Evaluación	ING. MILTON ESCOBAR
Rol	EVALUADOR

Fuente: Elaboración Propia**.**

Dentro del nivel 2 de TMMI constan los siguientes procesos: Políticas y estrategias, Planificación, Técnicas y Métodos, Entorno; si bien es cierto que con estos procesos se termina el nivel dos, no es menos cierto que el seguimiento y

control de las pruebas es una área que no está definida, pero debe ser tomada en consideración a fin de aprender y mejorar el proceso de prueba.

Para poder llevar a cabo la lista de verificación del nivel de TMMI en el que se encuentra la empresa Siredcom es necesario responder tomando en cuenta la siguiente ponderación:

i. 0= Práctica no implementada, no existe documentación apropiada de proceso.

ii. 1= Práctica aplicada parcialmente en proyectos o existe documentación de proceso pero no es aplicada o sólo parcialmente.

iii. 2= Proceso documentado que casi cubre la práctica o proceso no documentado pero se aplica de forma global en los proyectos.

iv. 3= Práctica totalmente documentada y aplicada de forma global.

A continuación en la tabla 3.8 se presenta el cuestionario correspondiente a la evaluación que se realiza a la empresa Siredcom para determinar el nivel en que se encuentra dentro del estándar de pruebas TMMI.

Tabla 3.8: Procesos que constituyen el nivel 2 de TMMI.

			Resp.	Resultados	
	Política y estrategia de Pruebas				
SG 1	**Establecer una política de Testing**				
	SP 1.1	Definir objetivos del Testing	0		
	SP 1.2	Definir Política de Testing	0		
	SP 1.3	Distribuir la Política a los stakeholders.	0		
				No Satisface	
SG 2	**Establecer una estrategia de Testing**				
	SP 2.1	Realizar una evaluación de riesgos de producto genérica	0		
	SP 2.2	Definir una estrategia de Testing	0		
	SP 2.3	Distribuir la estrategia a los stakeholders.	0		
				No Satisface	
SG 3	**Establecer indicadores de rendimiento del testing**				
	SP 3.1	Definir los indicadores de rendimiento del Testing	0		
	SP 3.2	Implementar los indicadores	0		
				No Satisface	
GG 2	**Institucionalizar el Proceso Gestionado**				
	GP 2.1	Establecer una Política organizacional	0		
	GP 2.2	Planificar el proceso	0		
	GP 2.3	Proporcionar recursos	0		
	GP 2.4	Asignar responsabilidades	0		
	GP 2.5	Formar al personal	0		
	GP 2.6	Gestionar conIlustraciónciones	0		
	GP 2.7	Identificar e involucrar a los stakeholders (partes interesadas relevantes)	0		
	GP 2.8	monitorizar y controlar el proceso	0		
	GP 2.9	Evaluar objetivamente la adherencia	0		
	GP 2.10	Revisar el estado con el nivel directivo	0		
				No Satisface	
GG 3	**Institucionalizar el Proceso Definido**				
	GP 3.1	Establecer un proceso definido	0		
	GP 3.2	Recoger información de mejora	0		
				No Satisface	
			Total	**0,0**	**No Satisface**

Planificación de las Pruebas			Resp.	Resultados	
SG 1	**Evaluar los riesgos del producto**				
	SP 1.1	Definir las fuentes y categorías de los riesgos de producto	0		
	SP 1.2	Identificar los riesgos	0		
	SP 1.3	Analizar los riesgos	0		
				No Satisface	
SG 2	**Establecer el enfoque del Testing**				
	SP 2.1	Identificar productos y características a testear	1		
	SP 2.2	Definir el enfoque del Testing	0		
	SP 2.3	Definir los criterios de entrada	1		
	SP 2.4	Definir los criterios de salida	1		
	SP 2.5	Definir los criterios de suspensión y reanudación	0		
				No Satisface	
SG 3	**Establecer estimaciones del Testing**				
	SP 3.1	Establecer una estructura de descomposición del trabajo (WBS) de alto nivel	0		
	SP 3.2	Definir el ciclo de vida del Testing	0		
	SP 3.3	Determinar las estimaciones de esfuerzo y coste del testing	0		
				No Satisface	
SG 4	**Desarrollar un plan de Testing**				
	SP 4.1	Establecer el calendario del testing	0		
	SP 4.2	Planificar el personal del testing	0		
	SP 4.3	Planificar la involucración de los stakeholders	0		
	SP 4.4	Identificar los riesgos de Testing del proyecto	0		
	SP 4.5	Establecer el Plan de Testing	1		
				No Satisface	
SG 5	**Obtener el compromiso con el plan de Testing**				
	SP 5.1	Revisión del test plan.	0		
	SP 5.2	Reconciliar los niveles de trabajo y de recursos	0		
	SP 5.3	Obtener el compromiso con el plan de Testing	0		
				No Satisface	
GG 2	**Institucionalizar el Proceso Gestionado**				
	GP 2.1	Establecer una Política organizacional	0		
	GP 2.2	Planificar el proceso	1		
	GP 2.3	Proporcionar recursos	1		
	GP 2.4	Asignar responsabilidades	1		

Continua ⟶

	GP 2.5	Formar al personal	0		
	GP 2.6	Gestionar conIlustraciónciones	0		
	GP 2.7	Identificar e involucrar a los stakeholders (partes interesadas relevantes)	1		
	GP 2.8	Monitorizar y controlar el proceso	0		
	GP 2.9	Evaluar objetivamente la adherencia	0		
	GP 2.10	Revisar el estado con el nivel directivo	0		
				No Satisface	
GG 3	**Institucionalizar el Proceso Definido**				
	GP 3.1	Establecer un proceso definido	0		
	GP 3.2	Recoger información de mejora	0		
				No Satisface	
			Total	0,1	**No Satisface**

PA 2.3	**Seguimiento y Control del Testing**			**Resp.**	**Resultados**	
	SG 1	**Monitorizar el proceso de Testing frente al plan**				
		SP 1.1	Monitorizar los parámetros de planificación del Testing	0		
		SP 1.2	Monitorizar los recursos de entorno proporcionados y usados	0		
		SP 1.3	Monitorizar los compromisos	0		
		SP 1.4	Monitorizar los riesgos	0		
		SP 1.5	Monitorizar la involucración de los stakeholders	0		
		SP 1.6	Llevar a cabo revisiones de progreso	0		
		SP 1.7	llevar a cabo revisiones de hitos	0		
					No Satisface	
	SG 2	**Monitorizar la calidad del producto frente al plan y las expectativas**				
		SP 2.1	Verificar los criterios de entrada	0		
		SP 2.2	Monitorizar las indecencias	0		
		SP 2.3	Monitorizar los riesgos	0		
		SP 2.4	Monitorizar los criterios de salida	0		

Continua ⟶

		SP 2.5	Monitorizar los criterios de suspensión y reanudación	0		
		SP 2.6	Llevar a cabo revisiones de calidad del producto	0		
		SP 2.7	Llevar a cabo revisiones de hitos de calidad	0		
					No Satisface	
	SG 3	**Gestionar las acciones correctivas hasta su cierre**				
		SP 3.1	Analizar los problemas	1		
		SP 3.2	Tomar acciones correctivas	1		
					No Satisface	
	GG 2	**Institucionalizar el Proceso Gestionado**				
		GP 2.1	Establecer una Política organizacional	0		
		GP 2.2	Planificar el proceso	0		
		GP 2.3	Proporcionar recursos	0		
		GP 2.4	Asignar responsabilidades	0		
		GP 2.5	Formar al personal	0		
		GP 2.6	Gestionar configuraciones	0		
		GP 2.7	Identificar e involucrar a los stakeholders (partes interesadas relevantes)	0		
		GP 2.8	monitorizar y controlar el proceso	0		
		GP 2.9	Evaluar objetivamente la adherencia	0		
		GP 2.10	Revisar el estado con el nivel directivo	0		
					No Satisface	
	GG 3	**Institucionalizar el Proceso Definido**				
		GP 3.1	Establecer un proceso definido	0		
		GP 3.2	Recoger información de mejora	1		
					No Satisface	
				Total	0,3	**No Satisface**

Continua ⟶

PA 2.2	**Entorno del Testing**			**Resp.**	**Resultados**	
	SG 1	**Desarrollar las necesidades del entorno de Testing**				
		SP 1.1	Obtener las necesidades de entorno de testing	0		
		SP 1.2	Desarrollar los requisitos de entorno de testing	0		
		SP 1.3	Analizar los requisitos de entorno de testing	0		
					No Satisface	
	SG 2	**Preparar el entorno de Testing**				
		SP 2.1	Implementar el entorno de testing	0		
		SP 2.2	Crear datos de testing genéricos	1		
		SP 2.3	Especificar procedimiento de entrada al entorno de testing	0		
		SP 2.4	Ejecutar test de entrada de entorno	0		
					No Satisface	
	SG 3	**Gestionar el entorno de Testing**				
		SP 3.1	Gestionar el sistema	0		
		SP 3.2	Gestionar los datos de testing	0		
		SP 3.3	Coordinar la disponibilidad y uso de los entornos de testing	1		
		SP 3.4	Reportar y gestionar las incidencias de entorno	0		
					No Satisface	
	GG 2	**Institucionalizar el Proceso Gestionado**				
		GP 2.1	Establecer una Política organizacional	0		
		GP 2.2	Planificar el proceso	0		
		GP 2.3	Proporcionar recursos	0		
		GP 2.4	Asignar responsabilidades	0		
		GP 2.5	Formar al personal	0		
		GP 2.6	Gestionar configuraciones	0		
		GP 2.7	Identificar e involucrar a los stakeholders (partes interesadas relevantes)	1		
		GP 2.8	monitorizar y controlar el proceso	0		
		GP 2.9	Evaluar objetivamente la adherencia	0		

Continua ⟶

		GP 2.10	Revisar el estado con el nivel directivo	0		
					No Satisface	
	GG 3	**Institucionalizar el Proceso Definido**				
		GP 3.1	Establecer un proceso definido	0		
		GP 3.2	Recoger información de mejora	0		
					No Satisface	
				Total :	0,1	**No Satisface**
				NIVEL 2:	0,1	**No Satisface**

Fuente: Elaboración Propia**.**

.

3.3 Conclusiones.

3.3.1 Conclusiones (9001:2000)

Después de haber realizado el análisis de los resultados y de las ilustraciones establecidas en base a la lista de chequeo, se establecen las siguientes conclusiones:

- En todos los procesos involucrados en la norma UNE-EN ISO 9001:2000, el número de inconformidades es mayor a las conformidades.

En el Sistema de Gestión de Calidad se determinó que:

- No está establecido de una manera formal, un sistema de gestión de calidad en la organización, que mediante objetivos claros y una planificación adecuada, aseguren el mejoramiento continuo, y control adecuado de sus procesos.
- Falta de políticas de calidad adecuadas que coadyuven, al cumplimiento de los objetivos organizacionales y de las del departamento de tecnología de

información, que son la base fundamental del normal funcionamiento de sus actividades.

En la Responsabilidad de la Dirección se determinó que:

- Falta de compromiso por parte de la dirección de la empresa Siredcom para establecer un sistema de gestión de calidad tanto a nivel gerencial como departamental como es el caso del departamento de tecnología.
- Falta de compromiso y apoyo de la dirección de la empresa Siredcom con las necesidades del departamento de tecnología, con el propósito de que los clientes internos y externos, incrementen su satisfacción.
- La alta gerencia no establece los procesos apropiados de comunicación dentro de la organización según la directriz de un sistema de gestión de calidad.

En la Gestión de los Recursos se determinó:

- Falta de una planificación para la capacitación del personal que trabaja en el departamento de tecnología de información de la empresa Siredcom.
- Falta de evaluación continua de los conocimientos técnicos que posee el personal que trabaja en el departamento de tecnología de información de la empresa Siredcom.
- Falta de motivación en los empleados del departamento de tecnología a fin de concienciarlos sobre la relevancia e importancia de sus actividades a fin de que contribuyan a la consecución de los objetivos de la calidad, en la empresa Siredcom.
- Falta de infraestructura adecuada (espacio adecuado, equipos, falta de suficiente memoria, falta de herramientas de desarrollo) para el personal que

trabaja en el departamento de tecnología de la empresa Siredcom, lo cual no brinda un ambiente de trabajo adecuado, para realizar sus labores, de una manera más adecuada y eficaz.

En la Realización del Producto se determinó que en el departamento de tecnología de la información de la empresa Siredcom:

- Existen requisitos propios del producto de software desarrollados, más no un sistema de gestión de calidad establecido formalmente en la organización, por lo cual no se cumplen los objetivos.
- Existen procesos que han ido cambiando con el tiempo, que han sido automatizados en su mayoría, pero que no han sido documentados desde varios años atrás.
- Falta de políticas que controlen los requisitos no especificados por el cliente pero necesarios para la utilización prevista o especificada del producto de software desarrollado en dicho departamento, bajo reglamentos institucionales.
- Falta de políticas sobre procedimientos adecuados para establecer la retroalimentación del cliente, con los desarrolladores de los productos de software incluyendo reclamos, ya que los usuarios de los sistemas de información, no poseen una cultura organizacional adecuada que permita establecer una retroalimentación inmediata y a tiempo que permitan prevenir inconvenientes a futuro, con aquellos procesos que se utilizan diariamente en diferentes módulos para realizar su trabajo.

- Falta de políticas que permitan mantener un historial sobre los resultados de la revisión de los cambios entre los desarrolladores y los usuarios correspondientes.
- No existen políticas que definan los criterios para la selección y evaluación periódica de los proveedores de hardware y software requerido por analistas, diseñadores y programadores. Así como el manejo de un historial sobre los resultados de la evaluación y las subsiguientes acciones de seguimiento de los proveedores.
- No Existe un proceso de comunicación por parte del personal del departamento de tecnología de información de la de la empresa Siredcom, con los usuarios correspondientes de los módulos que se encuentran en producción, a fin de capacitar continuamente sobre cambios en los sistemas y la creación de nuevos reportes.
- Falta establecer políticas que disponga las pautas necesarias al momento de validar los productos de software allí desarrollados.

En la Medición, Análisis y Mejora se determinó que:

No existe un sistema de gestión de calidad que permita a la organización y al departamento de tecnología de información planificar e implantar procesos de seguimiento, medición, análisis y mejora continua.

- La organización al no poseer un sistema de gestión de calidad que mejore los productos de software desarrollados en la empresa Siredcom, solamente se apoya en auditorías informáticas internas y externas que se realizan

periódicamente, lo cual permite acciones correctivas y preventivas, según autorización de la gerencia.

- No se realiza seguimiento de la información relativa a la percepción del cliente interno sobre el grado en que dicho departamento ha satisfecho sus requisitos.
- No existe un procedimiento escrito que defina los pasos y acciones a seguir, cuando se detecta que un producto que está ya en producción posee defectos a fin de evitar las consecuencias, o efectos potenciales, de la no conformidad.

3.3.2 Conclusiones (TMMI)

En los procesos involucrados en el nivel de TMMI llegamos a las siguientes conclusiones, basándonos en los resultados obtenidos en la tabla 3.8.

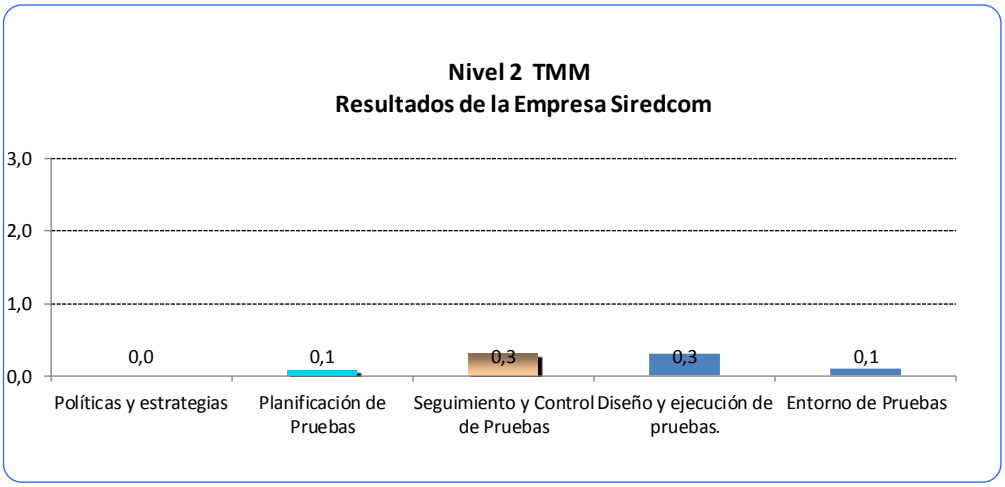


Ilustración 3.6: Resultados de la Evaluación del Nivel de TMMI de la Empresa Siredcom.

Elaborado por: Milton Escobar.

Después de haber ejecutado la lista de verificación para determinar en que nivel de TMMI se encuentra la empresa Siredcom se llega a concluir que está en un nivel 1, ya que no se encuentra definido un proceso que permita distinguirlo del debugging, no se determina de forma adecuada de la especificación del software a ser probado. Las pruebas se las realiza en el camino es decir cuando se ha completado la codificación, las versiones de los productos software son puestas en producción sin un aseguramiento de la calidad además se denota que el personal de la empresas no posee la competencia adecuada.

3.4 Modelo Formal de Pruebas Funcionales Propuesto

En la presente investigación se presenta una propuesta para la empresa SIREDCOM, basada en el proceso de pruebas, la cual constituirá una guía para las pruebas funcionales, a partir de las especificaciones del producto e independiente del ciclo de vida de desarrollo, tomando en consideración el riesgo del producto y la prioridad con la cual se realizarán las pruebas.

El modelo formal establecido como propuesta en esta investigación está definido en cuatro partes, a fin de que la empresa Siredcom alcance el nivel 2 de TMMI. Además se define un equipo de pruebas con sus respectivos roles, los cuales permiten establecer responsabilidades al momento de realizar la gestión de las actividades de pruebas y sus correspondientes entregables. Este capítulo consta de lo siguiente partes:

a) Caracterización del Modelo de Pruebas Funcionales Propuesto.

b) Estructura del Modelo de Pruebas Funcionales Propuesto.

3.4.1 Caracterización del Modelo de Pruebas Funcionales Propuesto.

En esta parte se establecen las principales características del modelo. Cabe destacar que el modelo de pruebas propuesto, es independiente del proceso de desarrollo de software. El modelo propuesto tiene como objetivo permitir a la empresa Siredcom alcanzar el nivel 2 de TMMI para lo cual se prueba la funcionalidad del software a partir de sus especificaciones, basándose en el análisis de riego del mismo, a fin de determinar las prioridades con las cuales serán evaluadas, se planifica utilizando métodos y técnicas básicas de pruebas funcionales.

A.1 Ser Independiente de la metodología de desarrollo

El Modelo de Pruebas propuesto se lo aplica de manera paralela a proyectos de desarrollo de software, cuando el software está por ser terminado o requiere mantenimiento, es decir en cualquier fase que se encuentre, sin importar que metodología es usada para el desarrollo de software que se esté probando, es el proceso que la empresa Siredcom debe aplicar a fin de alcanzar el nivel 2 de TMMI.

A.2 Validar las especificaciones funcionales del software.

El propósito de la validación es conocer si el software probado cumple o no con sus especificaciones, para lo cual se necesita los requerimientos y versión

ejecutable del producto, lo que permitirá establecer los casos de prueba, de existir especificaciones, se requiere que cliente y desarrolladores se reúnan.

A.3 Evaluar Riesgos del software a probar.

El modelo propuesto se enfoca en analizar el riesgo del software a probar, a fin de determinar cómo se van especificar y ejecutar los casos de pruebas, ya que probar todo es imposible, razón por lo cual se debe propender a realizar pruebas más efectivas que minimicen el costo, pero que a la vez aseguren confiabilidad, seguridad y satisfagan los requerimientos de los clientes. Casi siempre el factor tiempo al hacer pruebas es muy corto o es escaso por lo que se deben definir prioridades, siendo una buen alternativa el saber que probar, según los riegos analizados.

Basado en (Kit, 1995) un riesgo es la probabilidad que se produzca un evento no deseado y que este traiga consecuencias negativas, siendo una práctica recomendable conversar con los desarrolladores sobre que funcionalidades son las más usadas y cuáles son las que se deben tener mayor cuidado durante la ejecución de las pruebas, en otras palabras aquellas que en caso de falla causarían graves inconvenientes. Lo que permitirá minimizar los riesgos y prevenir errores posteriores.

De acuerdo con (Bach, 1999) para analizar los riesgos se deben tomar en consideración las siguientes directrices:

i. Elaborar una lista de riesgos;
ii. Realizar pruebas para cada riesgo y
iii. Mitigar riesgos.

El riesgo está directamente relacionado a la probabilidad de ocurrencia de que un problema se produzca y cause un impacto, razón por la cual entre más alto es el impacto, mayor será el riesgo asociado al problema. Existen dos maneras de analizar el riesgo, el primero desde adentro hacia fuera y el segundo desde afuera hacia adentro. De acuerdo con (Bach, 1999) la manera de adentro hacia afuera analiza que riesgos se asocian a una funcionalidad y la manera de afuera hacia adentro analiza que funcionalidades se asocian a una clase de riesgo, basándose en una lista de riesgos posibles y su correspondencia con los detalles del software. El modelo de pruebas propuesto utiliza el análisis de riesgos de adentro hacia afuera, siendo este detallado a continuación:

A.3.1 Análisis de riesgo de adentro hacia fuera.

Se realiza un estudio detallado sobre el software y así se determinan los riesgos relacionados a este. Para lo cual se determinan preguntas para cada parte del software:

- **Vulnerabilidades**: ¿qué debilidades o fallas pueden existir en este software?
- **Amenazas**: ¿qué entradas o escenarios pueden aprovechar una vulnerabilidad y hacer que el software falle?
- **Afectados**: ¿A quién o qué le causa un impacto, si una amenaza se materializa y cuál es su impacto?

En este análisis se necesita que el desarrollador a través de reuniones de trabajo, basándose en su entendimiento técnico, determine los riesgos del software y a su vez diseñe las pruebas que permitan atenuar ese riesgo, previniendo la

efectividad de las amenazas. Esto permite que quien realiza la prueba comprenda la funcionalidad y logre establecer una lista de riesgos y estrategias relacionadas con lo que se está probando.

El analizar riesgos requiere habilidades de comunicación y actitud positiva, debido a que se basa en un método heurístico, y esto podría conllevar el afectar a las funcionalidades; el modelo de pruebas propuesto utiliza pruebas basadas en las especificaciones, y testing exploratorio. Cuando se encuentra un defecto crítico para el negocio al hacer testing exploratorio y no fue hallado en la prueba que se planificó en un ciclo determinado, se debe revisar el análisis de riesgo.

Cabe destacar que el modelo de pruebas propuesto además de hacer análisis de riesgo del software a probar, también hace un análisis de riesgo del proyecto, siendo estos los que ponen en peligro la consecución de sus objetivos. Algunos ejemplos de riesgos en un proyecto de pruebas son: personal sin una adecuada capacitación, falta de una comunicación adecuada y falta de trabajo de equipo por parte de los desarrolladores, generación inadecuada de información y datos, confidencialidad de datos, desconocimiento del dominio del negocio, no encontrar errores significativos e importantes, entre otros.

A.4 Tener Ciclos de Prueba.

Según (Black, 2002) un ciclo de prueba permite ejecutar las pruebas que se hayan planificado para el producto, sean estas una o varias, sin importar el proceso de desarrollo que se utilice. Cada ciclo está relacionado con una versión del software

a probar, razón por lo cual cada nuevo ciclo que se genera crea una nueva versión. La gestión de las actividades de pruebas se las realiza para una versión específica del software a probar, sobre la cual se hacen las pruebas y se reportan las novedades encontradas.

Cabe destacar que las pruebas que se realizan sobre una versión están planificadas con antelación y deben ejecutarse, solamente si las prioridades cambian sería alterada su realización. Mientras se realizan las pruebas para una versión, paralelamente a ello se están haciendo mejoras a esa versión por parte de los desarrolladores.

El momento que se detecta un incidente, generalmente los desarrolladores saben el porqué de la falla y lo arreglan de inmediato, lo cual genera una versión nueva del software. Si no se necesita seguir probando la versión anterior, no se debería aceptar esta nueva versión por parte del equipo de pruebas ya que no fue planificada. Debido al error corregido, se pueden haber adaptado nuevas funcionalidades, teniéndose que ejecutar nuevamente todas las pruebas, sin tener la posibilidad de conocer la calidad total de la versión que se está probando.

Algo difícil de definir es el número de ciclos de pruebas necesarios, ya que no todas las versiones que realizan los desarrolladores son probadas por el equipo de pruebas, pudiendo existir más de dos versiones del software desarrolladas por los desarrolladores entre dos ciclos de prueba.

3.4.2 Estructura del Modelo de Pruebas Funcionales Propuesto.

La propuesta planteada, para ser independiente se fundamenta en los ciclos de prueba. En la Ilustración 3.7 se llega a un nivel de abstracción uno

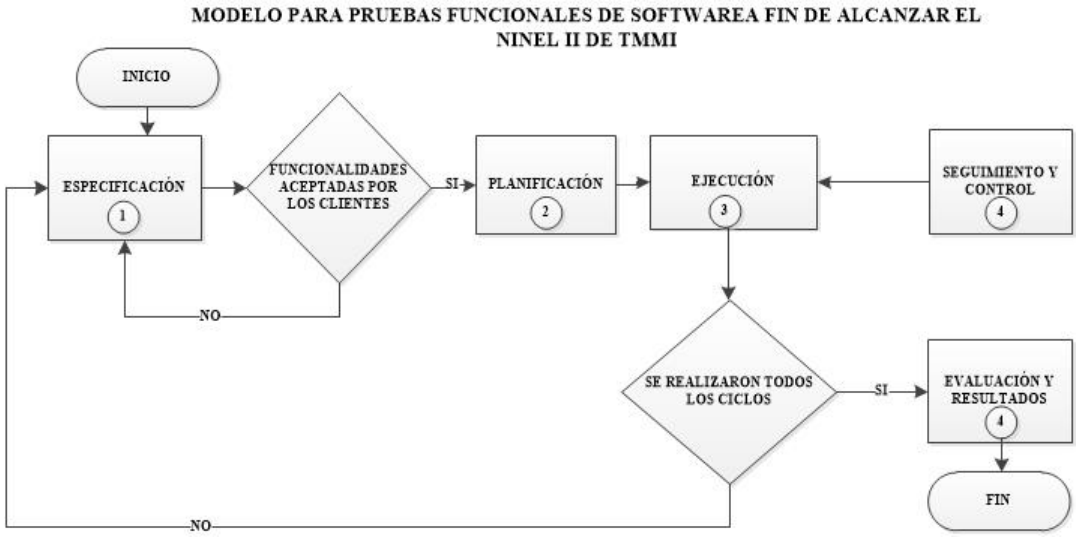


Ilustración 3.7: Modelo propuesto para Pruebas Funcionales de Software Nivel de Abstracción 1
Elaborado por: Milton Escobar.

En la Ilustración 3.8 se muestra el modelo de pruebas planteado en ésta tesis, el cual posee cuatro partes: Especificación, Planificación, Ciclo y Resultado. En la Ilustración 3.8 se llega al nivel dos, es decir a los componentes internos del modelo.

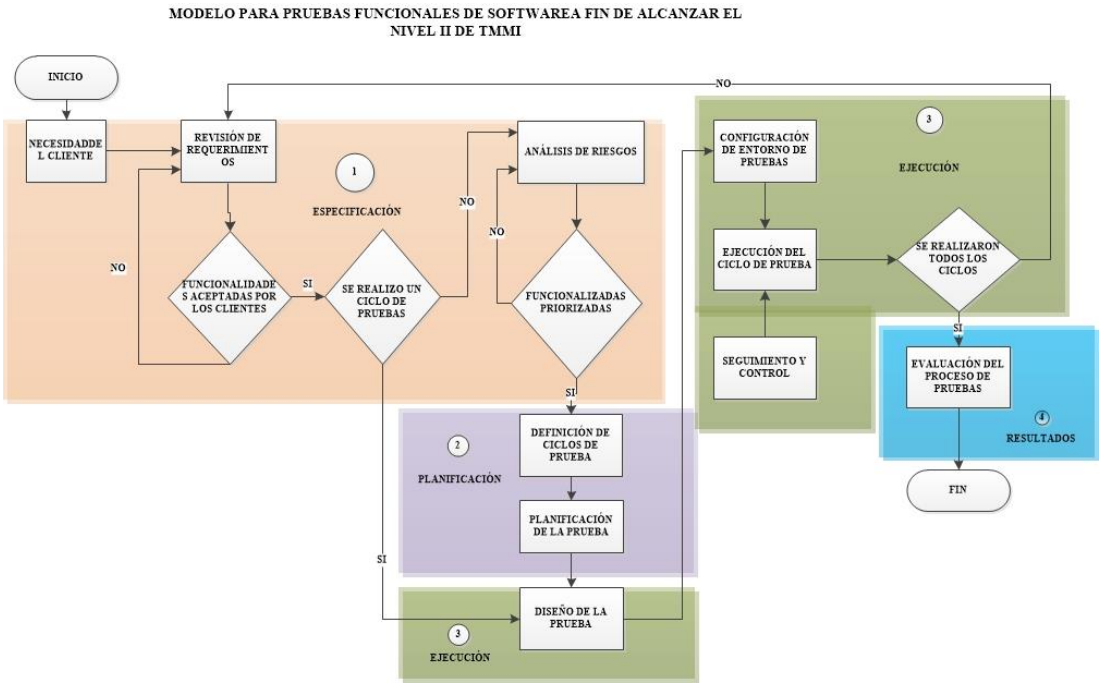


Ilustración 3.8: Modelo propuesto para Pruebas Funcionales de Software Nivel de Abstracción 2

Fuente: Elaboración Propia

Los objetivos de cada una de las partes definidas en el modelo de pruebas propuesto son:

i. **Especificación**: El objetivo es analizar las funcionalidades más importantes del software a probar, con el fin de determinar el alcance de las pruebas y así establecer un cronograma de los ciclos correspondientes. Posteriormente si el cliente está de acuerdo, se procede con la planificación, caso contrario se vuelve a realizar la especificación donde se analizan los inconvenientes encontrados por parte del cliente, lo cual se documenta y guarda para tener una referencia para futuros proyectos.

ii. **Planificación:** El objetivo es planificar y diseñar las pruebas, ya aceptadas por el usuario. Se establecen los ciclos, las funcionalidades y el análisis de riesgo, en esta parte se genera un plan de pruebas.

iii. **Ejecución:** El objetivo es determinar y realizar las pruebas de una versión del software, relacionándolas con el ciclo correspondiente.

iv. **Evaluación y resultados:** El objetivo de esta parte es determinar el informe final sobre las pruebas realizadas, la satisfacción del usuario con respecto al software probado y al proceso utilizado, a fin de realizar una mejora continua. Cabe destacar que esta información es almacenada a manera de histórico para pruebas futuras.

i. Especificación de la prueba.

En esta etapa se estudia la factibilidad de realizar las pruebas. En la Tabla 3.9 se muestran las actividades involucradas y los documentos generados en cada una.

Tabla 3.9 – Especificación de la Prueba.

ACTIVIDADES	Documentos Generados
A1 - Estimación de las tareas	ET - Estimación de Tareas
A2 – Medición de Esfuerzo	ME - Matriz de Esfuerzo
EP1- Necesidad del cliente	AR- Acta de Reunión AA- Acuerdo alcanzado
EP2 - Revisión de requerimientos	LP - Listado de Prueba
EP3 - Análisis de riesgo	LP - Listado de Prueba Priorizado

Fuente: Elaboración Propia

Las actividades A1- Estimación de las Tareas y A2 – Medición de Esfuerzo, son actividades realizadas en todas las etapas del proceso, al comienzo de cada etapa se debe estimar el tiempo que lleva realizar cada una de las actividades de la etapa y luego que las actividades son realizadas, se debe reportar el tiempo consumido en las mismas. Estas actividades generales no se muestran en los diagramas de actividad de las etapas.

En la Ilustración 3.9 se muestra el diagrama de flujo en esta fase. La actividad Ep1- Necesidad del cliente, tiene como objetivo llegar a un acuerdo con el Cliente respecto al alcance del proyecto de prueba. En la actividad Ep2 - Revisión de Requerimientos, se estudian las fuentes existentes para conocer si es posible derivar las pruebas a partir de ellas y se listan los grupos de funcionalidades del producto, los cuales son analizados y priorizados junto al cliente y desarrolladores. En la actividad Ep3 - Análisis de Riesgo. Con las funcionalidades priorizadas se retoma la actividad Ep1 - Necesidad del cliente, para definir el alcance y la agenda del proyecto de prueba. A partir del alcance y la agenda definidos el equipo de prueba realiza la

cotización del proyecto de prueba. A partir del alcance y la agenda definidos se genera el listado de pruebas priorizado.

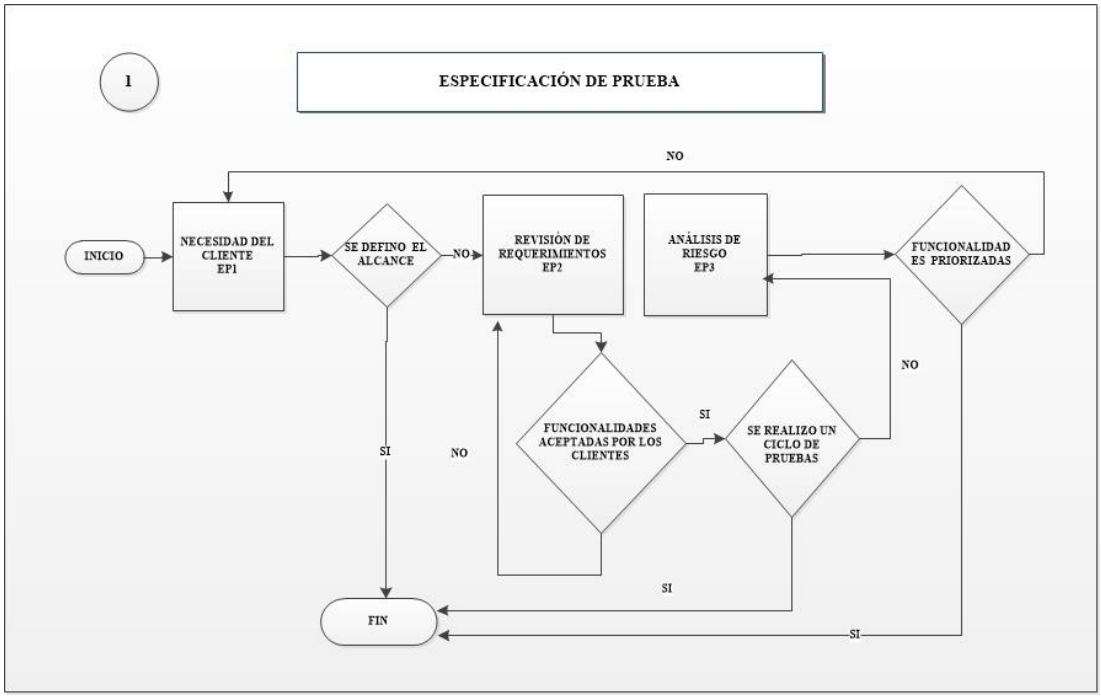


Ilustración: 3.9 Actividades de Especificación de Prueba.

Fuente: Elaboración Propia

A partir del diagrama de flujo explicado en la Ilustración 3.9 se describen las actividades definidas para la fase Especificación de la prueba:

a. Ep1. Necesidad del Cliente.

- Objetivo.

Determinar cuál es el trabajo que el equipo de pruebas realizará, tomando en consideración el alcance que las pruebas tendrán sobre el producto software a probar y su planificación en relación a los ciclos a pruebas a realizarse, tomando en

consideración como factor fundamental lo que el cliente necesita del producto software.

- Descripción.

La necesidad del cliente es una actividad que se realiza repetidas veces durante la fase Especificación de la Prueba a fin de determinar el alcance de las mismas, para lo cual se consideran las funcionalidades que se van a probar y cuáles no. Es necesario realizar reuniones con el Líder de Pruebas, el Cliente y los Desarrolladores. Dependiendo del producto software a probar, este se lo podrá dividir en partes o subsistemas, lo que facilitará determinar el alcance, ya que no todos los componentes tendrán la misma importancia, pudiendo darse casos en que no se probarán.

Es necesario separar las funcionalidades del producto software a probar, hasta que sea factible determinar su alcance. Posteriormente en la actividad Ep2. Revisión de Requerimientos, se analizan las funcionalidades a ser probadas, a fin de obtener el LP- Listado de Prueba y luego en Ep3-Análisis de riesgo, priorizar dichas funcionalidades. Al poseer el listado de prueba priorizado, se vuelve nuevamente a la actividad Ep1-Necesidad del cliente y con ayuda del PD-Plan de Desarrollo del producto software a probar, que es donde se establecerán las versiones del producto, pudiéndose definir el alcance de las pruebas. Lo que se verá reflejado en la actividad AA-Acuerdo Alcanzado, reflejándose el alcance, el número de ciclos de prueba y un bosquejo preliminar o básico de las pruebas a realizar.

Basándose en el LP-Listado de Pruebas realizado, se realiza una revisión junto al cliente a fin de confirmar las funcionalidades del producto de software a probar. Además se revisan los atributos de calidad que se analizaran al ejecutar las pruebas, según lo establecido en la fase EP1 Necesidad del Cliente, donde se determinaron atributos y prioridades.

Luego se establecen que productos se generan de las actividades de las pruebas realizadas al producto software, los cuales serán dados al cliente, cabe destacar que sebe establecer la manera en que se comunicará el cliente con el equipo de pruebas de una manera periódica, pudiéndose optar por varias posibilidades como es: reuniones, informes, reportes, etc.

Cabe destacar que en la Especificación de la Prueba se debe tomar en consideración lo siguiente:

i. Establecer donde se realizarán las pruebas del producto software a probar, en las instalaciones de la empresa Siredcom o en las instalaciones del cliente o de una manera mixta.
ii. Dependiendo de la complejidad y del tamaño del producto software a probar, se realizan las reuniones necesarias pudiendo establecerse como mínimo 2 reuniones para realizar esta actividad, las mismas que se reflejan en la AR-Acta de Reunión, la cual se entrega al cliente para ser validada posteriormente.

iii. Para poder planificar las pruebas del producto software a ser probado es necesario realizar un acuerdo con el cliente, sobre el tiempo que cada ciclo durará versus las pruebas de regresión a realizarse.

iv. El modelo propuesto relaciona las funcionalidades del producto software a ser probado con los riesgos asociados a dichas funcionalidades, lo que permite estimar aproximadamente la cantidad de ciclos de pruebas requeridos y su planificación.

b. EP2. Revisión de requerimientos.

- Objetivo.

Establecer si las especificaciones del producto de software a ser probado, permiten establecer las pruebas requeridas. Además las especificaciones del producto software a ser probado, a fin de establecer si se pueden generar casos de pruebas a partir de los requerimientos correspondientes. De existir inconvenientes en estos se debe establecer con el cliente la forma de mejorarlos, completarlos en el caso de estar incompletos o generarlos.

- Descripción.

Para realizar la actividad Revisión de requerimientos, es necesario validar los requerimientos del producto software a ser probado, mediante la ejecución de pruebas funcionales para determinar si al realizarlas, se obtiene la salida deseada.

Se analiza la información que se tiene sobre los requerimientos del producto software a ser probado y a partir de esto se determina si se puede diseñar las pruebas correspondientes, de existir inconvenientes, se debe realizar un trabajo conjunto con

el cliente y desarrolladores, a fin de mejorar la información de los requerimientos existentes, obteniéndose como producto de esta actividad el LP-Listado de Prueba.

Partiendo de la actividad Ep1. Necesidad del Cliente, donde se analizaron las especificaciones y funcionalidades del producto software a ser probado. Cabe destacar que dichas especificaciones pueden ser conseguidas de: Especificación de Requerimientos, Manuales de usuario y de sistema, entrevistas con los usuarios y/o clientes, reportes de fallos y/o defectos, etc.

Se analizan las funcionalidades que serán probadas en el primer ciclo de pruebas, con sus respectivas especificaciones (siendo estas las de mayor prioridad según el análisis de riesgo efectuado, ya que es imposible probar todo por el costo y tiempo que esto causa). Se planifica aquellos requerimientos que aún no se han probado, para probarlos en los siguientes ciclos.

De existir inconveniente en los requerimientos, estos deben ser optimizados y si no existen, elaborados, para así poder realizar las pruebas correspondientes.

c. Análisis de Riesgo.

- Objetivo.

Determinar una lista de prueba que priorice las funcionalidades más trascendentales del producto software a ser probado según su importancia y riesgo asociado, así como también aquellas funcionalidades que no serán comprobadas.

- Descripción.

Partiendo del documento generado LP-Listado de Prueba (funcionalidades establecidas en la fase anterior), se estudian y se discuten con el cliente y desarrolladores los riesgos que están asociados a las funcionalidades del producto software a ser probado. Cabe destacar la importancia del rol que realizan los clientes en esta actividad ya que ellos, conocen las funcionalidades más importantes y riesgosas en su negocio y lo que representan en caso de no funcionar adecuadamente. Otro rol importante es el de los desarrolladores ya que ellos aportan con determinar que parte del software a ser probado tiene mayor probabilidad de fallos. Obteniéndose después de este análisis la funcionalidad del producto a ser probado con sus respectivas priorizaciones en base a los riesgos analizados.

En otras palabras se establece las funcionalidades más importantes y así se especifica un alcance adecuado que permita realizar un cronograma de las pruebas a realizar. Cabe destacar que algunas funcionalidades a las cuales se les determino prioridad alta en la fase de Necesidad del Cliente, al ser dividido en algunas se convierten en prioridad alta, prioridad media y sin prioridad de realizarse pruebas sobre ellas.

ii. Planificación.

- Objetivo.

El objetivo de esta etapa es planificar el proyecto de prueba, una vez que el cliente aprobó. En la Ilustración 4 se muestran las actividades involucradas y los documentos generados.

Tabla 3.10 – Planificación.

Actividades	Documentos generados
A1 - Estimación de las tareas	ET - Estimación de Tareas
A2 – Medición de Esfuerzo	ME - Matriz de Esfuerzo
PL1 - Exploración del Producto	LP - Listado de Pruebas
PL2 - Definición de los Ciclos de Prueba	LCP - Listado de Ciclos de Prueba
PL3 - Definición del Testware	TW – Testware
PL4 - Planificación de las Pruebas	PP - Plan de Pruebas
PL5 - Definición del Proceso de Incidentes	PI - Proceso de Incidentes

Fuente: Elaboración Propia

En la Ilustración 3.10 se muestran las actividades de la parte de la planificación.

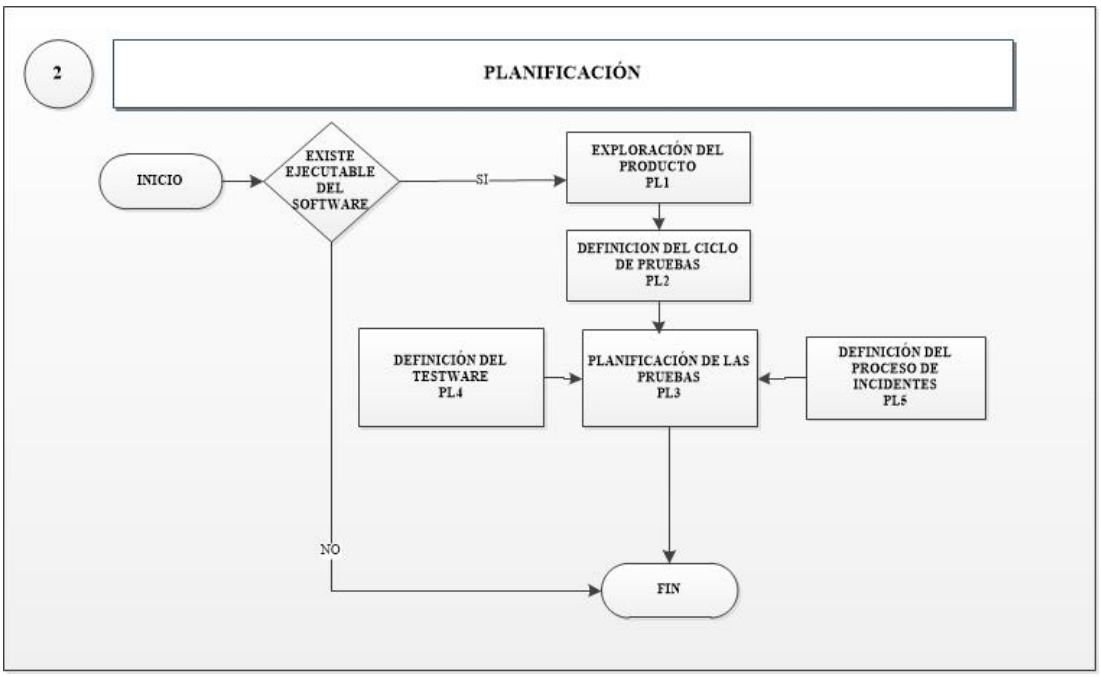


Ilustración: 3.10 Actividades de la Planificación de Prueba.

Fuente: Elaboración Propia

Luego de la especificación en donde se han definido todas las funcionalidades a probar y la prioridad con que será probada cada una, junto con el cliente, en la actividad EP2 -Revisión de Requerimientos, se estudian las funcionalidades del

producto y se realiza un listado de pruebas. En EL3 - Análisis de Riesgo, se conoce la importancia de incluir o no una funcionalidad en el alcance de las pruebas y la prioridad con que deben ser probadas en caso de ser incluida. A partir de esta información, se realiza PL2 - Definición de los Ciclos de Prueba, donde se listan los ciclos de prueba según las versiones que serán generadas por el equipo de desarrollo y se definen las funcionalidades que serán probadas en cada ciclo de prueba.

En el caso de que ya exista una versión ejecutable del producto al comenzar el proyecto de prueba, se puede realizar la actividad PL1 - Exploración del Producto, donde se explora el producto con el fin de entender rápidamente los requerimientos y ayudar en la actividad Ep2 - Revisión de Requerimientos. En paralelo con esto, se desarrollan dos actividades: PL4- Definición del Testware (el testware es el producto resultante de la prueba), donde se define cuáles serán los elementos que lo conformarán para el proyecto y cómo se organizarán, PL8 - Definición del Proceso de Incidentes, donde se define junto al cliente y los desarrolladores la herramienta a seguir para el reporte de incidentes y las etapas involucradas en dicho proceso.

El plan de pruebas del proyecto es realizado en la actividad PL3 - Planificación de las Pruebas. Dicho plan resume toda la información del proyecto de prueba y las decisiones tomadas durante la etapa de Planificación.

d. PL1 – Exploración de Producto.

- Objetivo.

Explorar el producto software a ser probado para determinar sus funcionalidades.

- Descripción.

Explorar el producto según Kaner et al., (2001) es experimentar sin establecer fronteras, navegando y explorando el producto software a ser probado, llegando más allá de la forma si no llegando al fondo de su funcionalidad.

Al momento de evaluar aquellos productos que fueron realizados por personas desconocidas, en esta actividad se enfoca a conocer y saber cómo funciona el producto software a probar, siempre y cuando se posea una versión ejecutable del mismo. Cabe destacar que una forma de analizarlo es que el evaluador se habitúe a su uso, mientras que el diseñador, coteje las especificaciones del mismo.

e. PL2 - Definición de los Ciclos de Prueba

- Objetivo.

Establecer la planificación de los ciclos de pruebas, con sus correspondientes funcionalidades a probar en cada uno de los ciclos.

- Descripción.

Partiendo del LP-Listado de Pruebas priorizado el cual se obtuvo de la actividad PL2- Revisión de Requerimientos y PL3- Análisis de riesgo, se realiza un reajuste a la planificación de los ciclos de pruebas que se realizó en la fase anterior, especificando de cada ciclo de prueba su inicio y su fin en relación con la versión del producto de software a probar. Es necesario mediar el tiempo con el equipo de desarrollo a fin de que realizar rectificaciones y establecer las nuevas versiones a probar del producto software a probar.

Basándose en el análisis del riesgo del producto a probar se establecen las pruebas que se realizaran en cada ciclo. Se debe tomar en consideración escoger un conjunto de pruebas para cada ciclo. Según (Black, 2002) existen técnicas ya establecidas:

i. Técnica de Priorización: Se ejecutan las pruebas en el orden de prioridad dado en el análisis de riesgo al comenzar las pruebas.
ii. Técnica de Priorización Dinámica: A medida que los ciclos se desarrollan pueden cambiar las prioridades, se ejecutan las pruebas en cada ciclo según el orden de prioridad definido en el ciclo. El riesgo con este enfoque es que podría ocurrir que no se ejecutan todas las pruebas a lo largo de los ciclos, lo que podría llevar a un cambio de agenda al final.
iii. Tren: Si todas las pruebas tienen la misma importancia se pueden ejecutar las pruebas una tras otra a través de los ciclos.

Siempre que se arregle una falla, es necesario realizar de nuevo las pruebas de regresión correspondientes, ya que no se puede saber con exactitud cuántos errores o fallos existen. Se debe considerar en la planificación las pruebas de regresión en cada ciclo.

f. PL2 - Definición del Testware

- Objetivo.

Determinar que herramientas y como estas se organizaran para realizar las pruebas del producto de software a ser probado.

- Descripción.

Determinar cómo se organizará la forma del testware y el acceso al mismo a fin de poder ser utilizado de manera continua, razón por la cual los casos de pruebas deben ser ordenados y enlistados. Se pueden agrupar los casos de prueba que necesiten un mismo entorno para su realización o agrupar aquellas pruebas que necesitan mantenimiento a fin de optimizar recursos y tiempo De acuerdo con (Kit, 1995) es necesario que al ejecutar un caso de prueba especificar los datos, procedimientos y registros de pruebas relacionados (datos, reportes, etc).

g. PL3 - Planificación de las pruebas

- Objetivo.

Construir el plan de pruebas, utilizando: Que, cuando, donde, quien y como se realizarán las pruebas y sus correspondientes actividades.

- Descripción.

El plan de pruebas establece una relación entre el equipo de pruebas y el cliente, según (Black, 2002) este posee:

i. Alcance
ii. Cronograma
iii. Ciclos y condiciones a realizarse
iv. Requerimientos del entorno de pruebas (software y hardware)
v. Estrategias de prueba
vi. Técnicas para diseño de pruebas
vii. Roles y responsabilidades
viii. Procedimientos de incidencias
ix. Análisis de riesgos
x. Clasificación de defectos
xi. Reportes al Cliente

Para administrar las pruebas es necesario planear de acuerdo al listado de pruebas donde está la funcionalidad del producto software a probar, de acuerdo a cada uno de los ciclos, lo que permite realizar la estimación del tiempo de

realización de las actividades de pruebas, pudiéndose basarse en experiencias de pruebas de productos de software anteriores. Se debe hacer notar que el plan de pruebas será hecho por el Líder de Pruebas y por el Diseñador de Pruebas, revisado y validado por el cliente.

h. PL4 - Definición del Proceso de Incidentes.

- Objetivo.

Establecer el proceso por el que pasa un incidente, es decir desde que sucede hasta su arreglo, tomando en consideración las responsabilidades correspondientes.

- Descripción.

Un incidente generalmente se manifiesta porque una falla fue encontrada a realizar las pruebas del producto software, siendo necesario determinar un proceso que determine la forma de que el equipo de pruebas administre los incidentes generados en cada ciclo de prueba para los cual se debe:

a. Establecer un instrumento de seguimiento de los incidentes generados.
b. Determinar sobre alguien la responsabilidad de recibir los incidentes y realizar las pruebas de regresión después de que las fallas sean arregladas.
c. De acuerdo con (Kaner C., 1999) se puede establecer que estados pueden tener los incidentes:
    i. El incidente es reportado por un tester.

ii. El Líder de Pruebas asigna una prioridad al incidente reportado y se asigna un desarrollador para su arreglo, este puede rechazar por no poder reproducir el error o determinar que no existe un incidente.

iii. Si el incidente se arregló por parte del desarrollador, el estado es reparado

iv. El tester ejecuta una prueba sobre la versión que tuvo el incidente si no existe novedad el estado es resuelto, de no ser así a estado pendiente.

iii. Ejecución de las Pruebas.

- Objetivo.

El objetivo de esta fase es realizar las pruebas de una versión determinada del software. En éste ciclo se puede ejecutar una, alguna o todas aquellas planificadas para el software a probar. Esta fase se divide a su vez en sub-fases como se muestra en la Ilustración 3.12. Las sub-fases son: Acuerdo con él Cliente, Configuración del Entorno, Diseño de las Pruebas y Ejecución de las Pruebas.

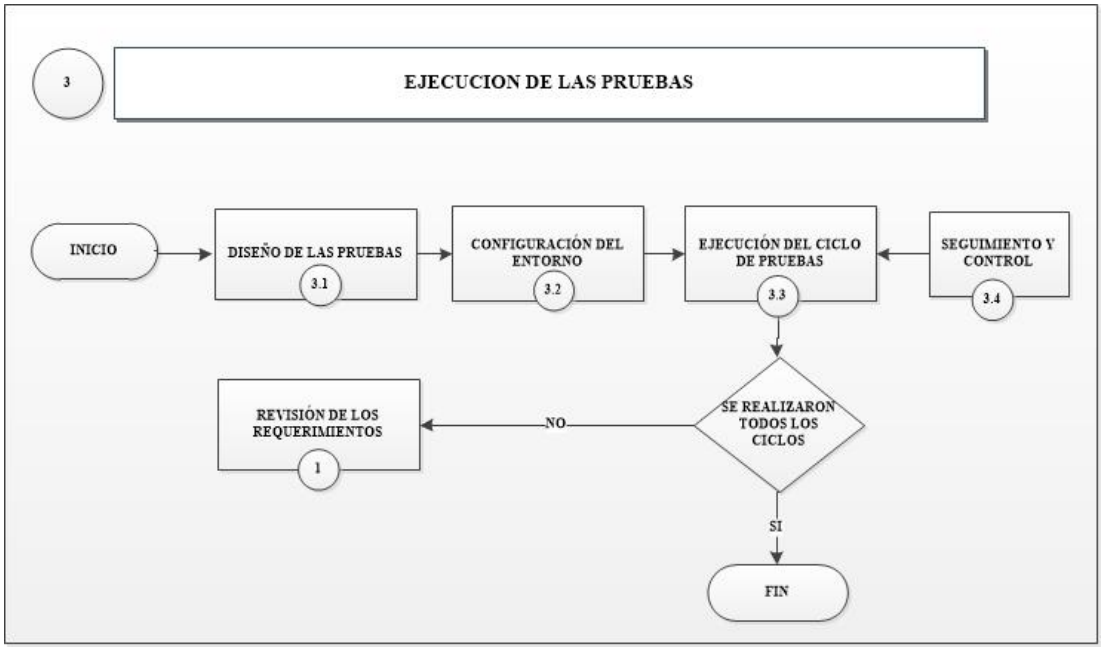


Ilustración 3.11 Ejecución de la Prueba.

Fuente: Elaboración Propia.

Los objetivos de cada sub fase se describen a continuación:

a. **Diseño de las Pruebas:** Consiste en el diseño de los casos de prueba a partir de la especificación del producto. En caso de ser necesario, en esta sub-fase se mejoran o generan las especificaciones junto con el Cliente.
b. **Configuración del Entorno:** Su objetivo de esta sub-fase es configurar el ambiente de pruebas, separando los ambientes de desarrollo y prueba, instalar las herramientas necesarias y el software a probar en la versión correspondiente a cada ciclo de prueba.
c. **Ejecución de las Pruebas:** El objetivo de esta etapa es contrastar el comportamiento esperado del software con su comportamiento real, analizar las diferencias y reportar los resultados.
d. **Seguimiento y Control:** Su objetivo es dar seguimiento y controlar el ciclo de prueba. La planificación realizada al principio del proyecto es

revisada al comenzar cada ciclo de prueba, se planifican en detalle las tareas para el ciclo y se ajusta la planificación según las estimaciones y posteriores mediciones realizadas en los ciclos anteriores. Luego que se realizaron las otras tres sub-etapas del ciclo, se vuelve a esta sub-etapa, donde se evalúa el ciclo y se decide si se finaliza o si es necesario generar y ejecutar nuevas pruebas para el ciclo.

En cada ciclo ejecutado, se procede con la sub-fase: Seguimiento del Ciclo, donde se administra y planifica el mencionado ciclo, una vez definidas las tareas que serán realizadas, se comienza la sub-fase Diseño de las Pruebas, apenas se cuente con la versión ejecutable del producto a probar. Posteriormente comienza la sub-fase de Configuración del Entorno y luego se realiza la Ejecución de las Pruebas.

Si no se cuenta con la versión ejecutable del producto, el Diseño de las Pruebas puede realizarse de todos modos, ya que se realiza a partir de la especificación del producto. En cambio la Configuración del Entorno y la Ejecución de las Pruebas requieren de la versión ejecutable del producto. Al terminar de ejecutar las pruebas, se vuelve a la sub-fase Seguimiento del Ciclo, donde se evalúan los resultados de la ejecución de las pruebas y se decide si es necesario derivar nuevos casos de prueba, en caso contrario se termina el ciclo de prueba.

Los ciclos de prueba se solapan en el tiempo, mientras se ejecutan las pruebas, se puede al mismo tiempo estar diseñando las pruebas siguientes. Se revisan solamente las especificaciones de las funcionalidades que serán probadas.

A continuación se describen cada una de las sub-etapas para la Ejecución de un Ciclo de Prueba.

a. Diseño de las pruebas.

El objetivo de esta sub-fase es diseñar las pruebas que luego se las ejecutarán sobre el software a probar. En la Tabla 3.11 se muestran las actividades involucradas en esta fase y los documentos generados en cada una.

Tabla 3.11 – Diseño de las Pruebas.

Actividades	Documentos generados
PL2 – Revisión de requerimientos.	LP- Listado de Pruebas
E2 – Configuración.	RI - Reporte de Inconvenientes.
DP1 – Diseño de los Casos de Prueba.	CP – Casos de Prueba MT – Matriz de Trazabilidad
DP2 – Validación de los Casos de Prueba.	RR – Resumen de Reunión
DP3 – Asignación de los Casos de Prueba.	PE – Plan de Ejecución

Fuente: Elaboración Propia

En la Ilustración 3.12 se muestra el diagrama de flujo en la sub-fase de Diseño de las Pruebas. Se comienza con la actividad P2- Revisión de Requerimientos, donde se revisan solo las especificaciones de las funcionalidades que serán probadas en este ciclo, en caso de que no se cuente con las especificaciones o que estén incompletas, se trabaja junto con el cliente para generarlas o mejorarlas. A partir de los requerimientos para las pruebas se diseñan los casos de prueba en la actividad DP1-Diseño de los Casos de Prueba, luego la actividad DP2-Validación de los Casos de Prueba, tiene como objetivo que el cliente valide los casos de prueba generados para asegurarse de que son de valor para el

negocio. Por último en la actividad DP3-Asignación de los Casos de Prueba, quienes diseñan los casos de prueba, se los asignan a los testers encargados de ejecutarlos en la siguiente sub-fase: Ejecución de las Pruebas.

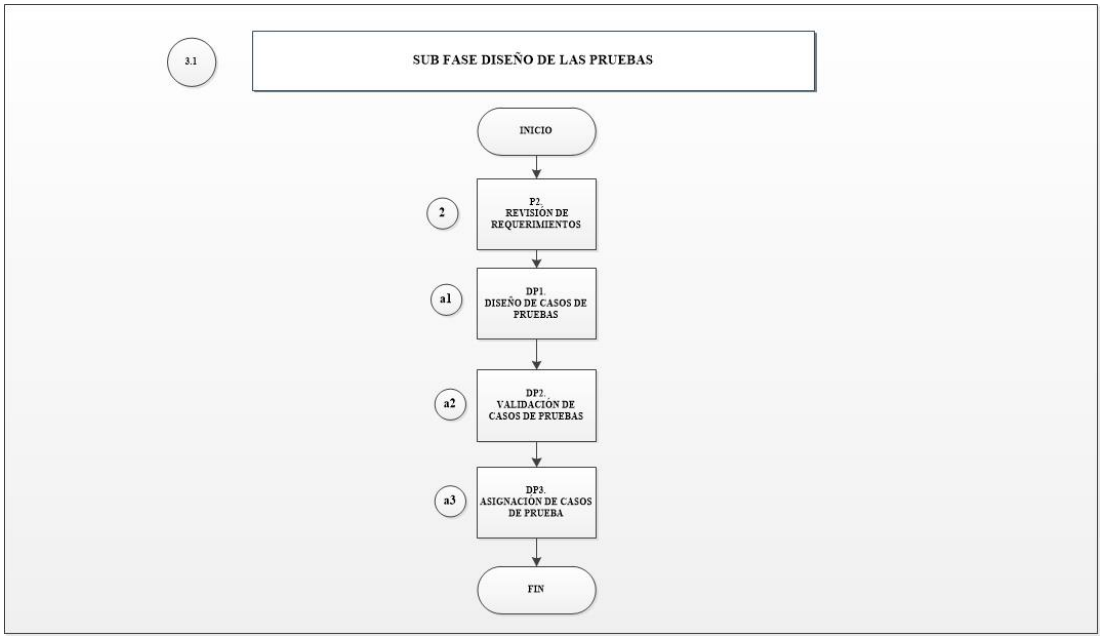


Ilustración: 3.12 Actividades de la sub-etapa Diseño de las Pruebas

Fuente: Elaboración Propia

a.1 DP1 - Diseño de los Casos de Prueba

- Objetivo.

Determinar los casos de prueba basándose en las especificaciones del producto software a probar.

- Descripción.

Basándose en las funcionalidades planeadas en cada ciclo del producto software a probar, se establecen qué casos de prueba se van a generar mediante

técnicas de caja negra. Además se diseñan las pruebas y se establecen los datos de pruebas correspondientes. Basado en (Kit, 1995) es necesario crear una matriz, para probar la funcionalidad del producto software a probar, relacionando los ítems de prueba y los casos de prueba.

En el diseño de los casos de prueba se puede utilizar técnicas de caja negra como son: partición de equivalencia, valor límite, conjetura de errores, grafo causa efecto, partición en categorías, máquinas de estado finitas y escenarios, las cuales fueron resumidas en el capítulo 2.

Según (Kit, 1995) los casos de prueba se relacionan con el LP-Listado de Pruebas para cada ciclo correspondiente. Para lo cual es necesario:

- Identificar que se debe probar
- Determinar las prioridades de lo que se va a probar según un análisis riesgo correspondiente.
- Realizar el diseño de las pruebas de alto nivel (grupos relacionados)
- Realizar los casos de pruebas individuales (según el diseño de alto nivel)
- Construir una matriz que permita relacionar las funcionalidades a probar y los casos de prueba correspondientes.

Fundamentado en Pol M (Pol M., 2002) se debe establecer el orden de ejecución de las pruebas según los ciclos y funcionalidad correspondiente, tomando como fundamento el ciclo de vida de los datos: crearlo, consultarlo, modificarlo y borrarlo.

Debido a que los requerimientos del producto software ser probado cambian frecuente o de manera tardía. Según (Kaner et al., 2001) es de suma importancia tomar en cuenta las siguientes buenas prácticas:

- Desarrollar pruebas según se necesiten.
- No realizar documentos de pruebas muy grandes (alto costo de mantenimiento).
- No basar las pruebas en detalles de interfaz de usuario (pueden cambiar)
- Diseñar pruebas automatizadas (maximizan mantenibilidad y portabilidad).
- Construir un conjunto de pruebas genéricas (producto a producto) permite optimizar el tiempo y con esto la ejecución de la planificación.

a.2 DP2 – Validación de los Casos de Prueba

➢ Objetivo.

Determinar si los casos de prueba construidos por el equipo de pruebas, generan un valor al negocio, siendo necesaria la validación por parte del cliente del producto software a probar.

➢ Descripción.

Se establece una validación por parte del cliente de los casos de prueba, tomando en consideración los datos y el valor que estos generan al negocio, según la prioridad establecida en LP-Listado de pruebas.

a.3 DP3 – Asignación de los casos de prueba

- Objetivo

Determinar a cada uno de los testers que caso de prueba está a su cargo, estableciendo el plan de ejecución del ciclo de pruebas del software a ser probado.

- Descripción

El diseñador determina que testers deben ejecutar los casos de prueba, según la planificación establecida para el ciclo de pruebas, esta asignación se establece en PE-Plan de Ejecución, de acuerdo (Kaner et al., 2001) una buena práctica es rotar a los testers en los distintos ciclos existentes, así se permite que estos no pierdan interés y no exista especialización, lo que permite prever si alguno deja su empleo y se va. Otra manera de afrontarlo es organizar a los testers en pares.

b. Configuración del Entorno.

El objetivo de esta sub-fase es configuración del ambiente de pruebas, separando los ambientes de desarrollo y prueba e instalar las herramientas necesarias y el software a probar en la versión correspondiente a cada ciclo de prueba. En la Tabla 3.12 se muestran las actividades involucradas en esta etapa y los artefactos generados en cada una.

Tabla 3.12 – Entorno de Pruebas.

Actividades	Documentos generados
E1 - Instalación de Herramientas	RI- Reporte de Inconvenientes.
E2 – Configuración	RI - Reporte de Problemas

Fuente: Elaboración Propia

En la Ilustración 3.13 se muestra el diagrama de flujo de la sub-fase Entorno de Pruebas. En la actividad E1-Instalación de Herramientas, se instalan las herramientas necesarias para realizar las pruebas del software, las cuales fueron definidas en la etapa de Planificación en la actividad P6-Definición del Testware. En el momento en que se cuenta con la versión ejecutable del producto a probar en el ciclo, se instala y se configuración de los datos relacionados en la actividad E2-ConIlustracióncion. Los problemas que puedan surgir durante estas actividades, se reportan en RI – Reporte de Inconvenientes.

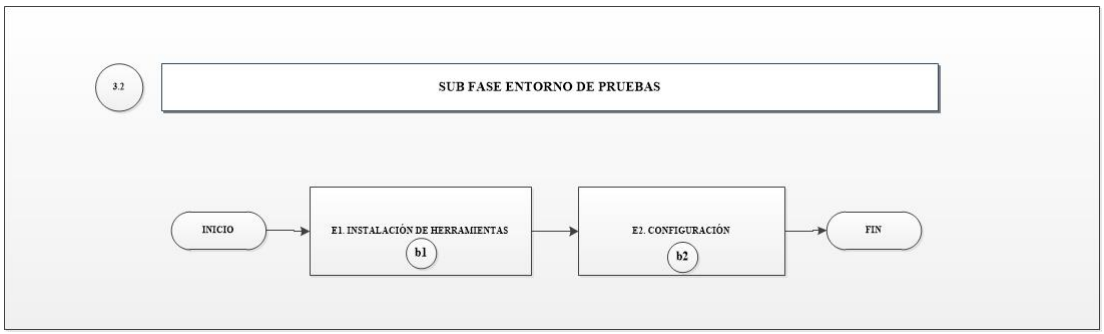


Ilustración: 3.13 - Actividades de la sub-fase Entorno de Pruebas.

Fuente: Elaboración Propia

b.1 Instalación de Herramientas

- Objetivo

Realizar la instalación de las herramientas que se necesitan para la ejecución de las pruebas, que se consideraron en el Testware.

- Descripción

Se realiza la instalación para realizar las pruebas (servidores, estaciones de trabajo), están son aquellas que se definieron en el testware (sistemas operativos, bases de datos, herramientas de gestión de la configuración, herramientas para la realización de reportes, etc.) de tener inconvenientes estos son documentados en RI-Reporte de Incidentes.

c.2 Configuración.

- Objetivo

Configuración e instalar el producto software a ser probado en el ciclo correspondiente.

- Descripción

Para cada ciclo de pruebas se debe configurar e instalar en el ambiente de pruebas la versión correspondiente en la cual se va a ejecutar la prueba funcional, considerando los datos necesarios. De existir inconvenientes se lo registrará en RP-Reporte de Problemas.

c. Ejecución de las Pruebas.

En ésta esta sub-fase se analiza el comportamiento que se desea del software en forma real, se analizan las diferencias y se reportan los resultados. En la Tabla 3.13 se muestran las actividades involucradas en esta etapa y los documentos generados.

Tabla 3.13– Ejecución de las Pruebas.

Actividades	Documentos generados
EP1 - Pruebas de Humo.	RE - Reporte de Ejecución.
EP2 - Ejecución de las Pruebas.	RE - Reporte de Ejecución.
EP3 - Testing Exploratorio.	RE - Reporte de Ejecución.
EP4 - Reporte de Incidentes.	RI - Reporte de Incidentes
EP5 - Validación de los Incidentes.	RI - Reporte de Incidentes
EP6 – Verificación de las Correcciones.	RE - Reporte de Ejecución.

Fuente: Elaboración Propia.

En la Ilustración 3.14 se muestra el diagrama de flujo de la sub-etapa Ejecución de las Pruebas, comienza con la actividad EP1- Pruebas de Humo, donde se valida que las funcionalidades básicas de la versión a probar están presentes, si esto sucede se continúa con la actividad EP2 – Ejecución de las Pruebas y EP3- Testing Exploratorio, donde se ejecutan las pruebas del software. Para los incidentes encontrados en ciclos anteriores, que fueron corregidos para la versión que se está probando, se ejecutan las pruebas de regresión en la actividad EP6- Verificación de las Correcciones. Los incidentes encontrados en estas actividades se reportan en EP4- Reporte de Incidentes. Los incidentes reportados son validados por el cliente y los desarrolladores en la actividad EP5 – Validación de los Incidentes.

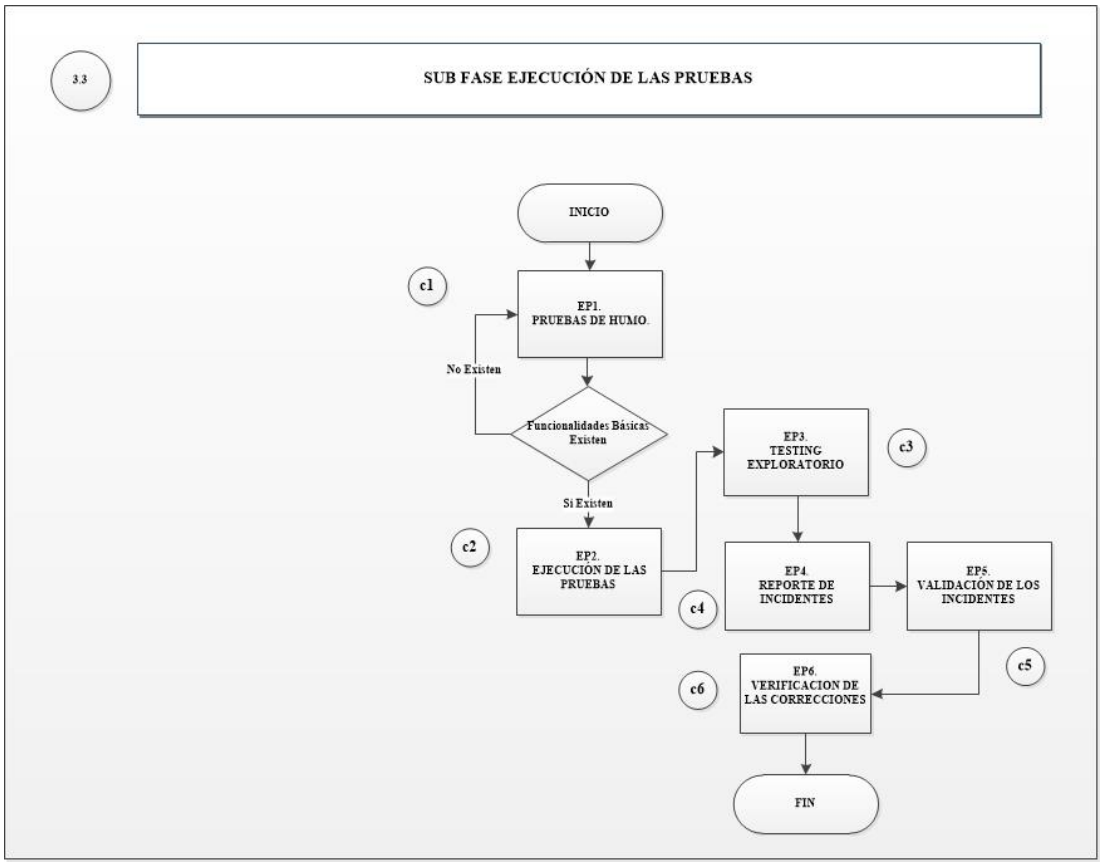


Ilustración: 3.14 – Sub Fase Ejecución de las Pruebas.

Fuente: Elaboración Propia.

c.1 EP1 - Pruebas de Humo

- Objetivo

Validar que las funcionalidades principales del producto software a ser probado existen y funcionan.

- Descripción

Basado en (Kaner et al., 2001) las pruebas de humo permiten validar las funcionalidades básicas del producto software a ser probado en cada nueva versión según lo especificado. Este tipo de pruebas examinan en búsqueda de desequilibrios

o elementos primordiales faltantes o defectos, que imposibilitan ejecutar las pruebas según su planificación para cada ciclo. Si no pasa la prueba de humo, dicha versión no realizara la ejecución de su prueba.

De acuerdo con (Kaner et al., 2001) una versión puede ser denegada si:

- Si la importancia de la versión es porque posee una funcionalidad crítica y esta no se encuentra o falla
- Si la importancia de la versión es que agrega una funcionalidad crítica y se descubre que no está presente en la versión o que falla inmediatamente
- Si la versión no presenta las funcionalidades principales es decir no funciona según las especificaciones, pudiendo ser porque esta fue hecha con los requerimientos incorrectos.
- Si se está por recibir una nueva versión en poco tiempo, depende del costo de su instalación, lo que permita definir si es mejor continuar probando la versión anterior hasta que llegue la nueva

c.2 EP2 - Ejecución de las pruebas

- Objetivo

Realizar los casos de prueba planeados en el ciclo de pruebas y analizar su resultado.

- Descripción

Según (Kit, 1995) se debe escoger los casos de prueba, realizarlos, registrar sus resultados y analizar si existen incidentes de ser así determinar si es por errores en el producto software o errores en las pruebas. Si se encontraron incidentes se

registran en EP4- Reporte de Incidentes. Al ejecutarse un caso de prueba y su resultado no es el adecuado el tester debe:

- Nuevamente realizar el caso de prueba, por lo menos una vez, para saber si fue un incidente.
- Aislar el incidente mediante la realización del caso de prueba en otros escenarios y con otros datos.

c.3 EP3 - Testing Exploratorio

➢ Objetivo

Minimizar la posibilidad de dejar de lado una funcionalidad imprescindible para el negocio que pudo ser obviada al realizar el análisis de riesgo del producto software a probar. Al encontrar un defecto crítico para el negocio al utilizar el testing exploratorio y no haberse encontrado por medio de una prueba planificada en el ciclo es necesario reanalizar el análisis de riesgo realizado anteriormente.

➢ Descripción

De acuerdo con (Burnstein, 2003) los testers realizan el diseño, el desarrollo y la ejecución de las pruebas mientras se ejecuta el producto mediante el testing exploratorio a fin de enfocar desde una perspectiva diferente al producto software a probar, siendo un método barato y rápido. Si al realizar el testing exploratorio se encuentran fallas, que no fueron encontradas por aquellas pruebas planificadas y que son sensibles para el producto software a probar o para el cliente. Siendo necesario revisar el análisis de riesgo que se posee.

c4. EP4 - Reporte de incidentes

- Objetivo

Documentar los incidentes encontrados en la realización de las pruebas.

- Descripción

En la actividad P8- Definición del Proceso de Incidentes, se determinó el proceso que en concordancia con el cliente se requiere para reportar los incidentes. Al reportar un incidente se debe determinar con precisión si es un defecto, mejora o una observación. Cabe destacar que se debe relacionar una incidencia con su correspondiente caso de prueba que lo genero a fin de poderse volver a reproducir, también debe determinarse el impacto del mismo. Los reportes deben ser precisos, simples y claros.

Según (Kaner et al., 2001) las siguientes recomendaciones se deben considerar al realizar un reporte de incidentes :

- Describir los pasos para reproducirlo de una manera sencilla
- Para llegar al incidente realizarlo pasó a paso.
- Enumerar cada uno de los pasos.
- No saltarse ningún paso.
- Determinar pasos cortos para llegar a la falla.
- Utilizar sentencias simples y corta
- Explicar que se esperaba y que es lo que sucedió.

- Si las consecuencias son graves, especificar el porqué de su argumento

c.5 EP5 - Validación de los incidentes

➢ Objetivo

El cliente valida los incidentes encontrados.

➢ Descripción

El equipo de pruebas reporta los incidentes encontrados los cuales deben se validados por el cliente, además este puede cambiar el estado del incidente (prioridad, estado y categoría), según la actividad P8- Definición del Proceso de Incidentes.

c.6 EP6 -Verificación de las Correcciones

➢ Objetivo

Mediante la ejecución de las pruebas de regresión se validan las correcciones que los desarrolladores hicieron de los incidentes encontrados.

➢ Descripción.

Al terminar la ejecución de las pruebas en un ciclo determinado, aquellos incidentes reportados se corregirían por los desarrolladores, pudiendo realizarse esta actividad en el segundo ciclo de pruebas, puesto que se revisa que dicho incidente se ha corregido en la nueva versión y así poderlo cerrar. Posteriormente se realizan las pruebas de regresión que permitan establecer que los cambios efectuados resultaron

correctos y no generaron comportamientos no requeridos o generaron nuevos errores, para lo que se requiere seleccionar casos de prueba para re ejecutarlos. De encontrar novedades estas se reportan en la actividad EP4-Reporte de incidentes.

d. Seguimiento y control.

El objetivo de esta sub-fase es realizar el seguimiento y control del proyecto de prueba en cada ciclo de prueba, es muy importante llegar a un acuerdo con el cliente. En la Tabla 3.12 se muestran que actividades se realizan en esta etapa y los documentos generados.

Tabla 3.14 – Seguimiento y Control.

Actividades	Documentos generados
AC1 - Planificación del Ciclo de Pruebas.	PPC - Plan de Pruebas del Ciclo
AC2 - Gestión de la Configuración.	RGC - Reporte de la Gestión de la Configuración.
AC3 - Seguimiento y Control del Ciclo de Pruebas	RACP - Reporte de Avance del Ciclo de Pruebas.
AC4 - Evaluación del Ciclo de Pruebas	RECP - Reporte de Evaluación del Ciclo de Pruebas.
A1 - Estimación de las tareas	ET - Estimación de Tareas
A2 - Medición de Esfuerzo	ME - Matriz de Esfuerzo

Fuente: Elaboración Propia

En la Ilustración 3.15 se muestra el diagrama con las actividades de la sub-fase Seguimiento y Control.

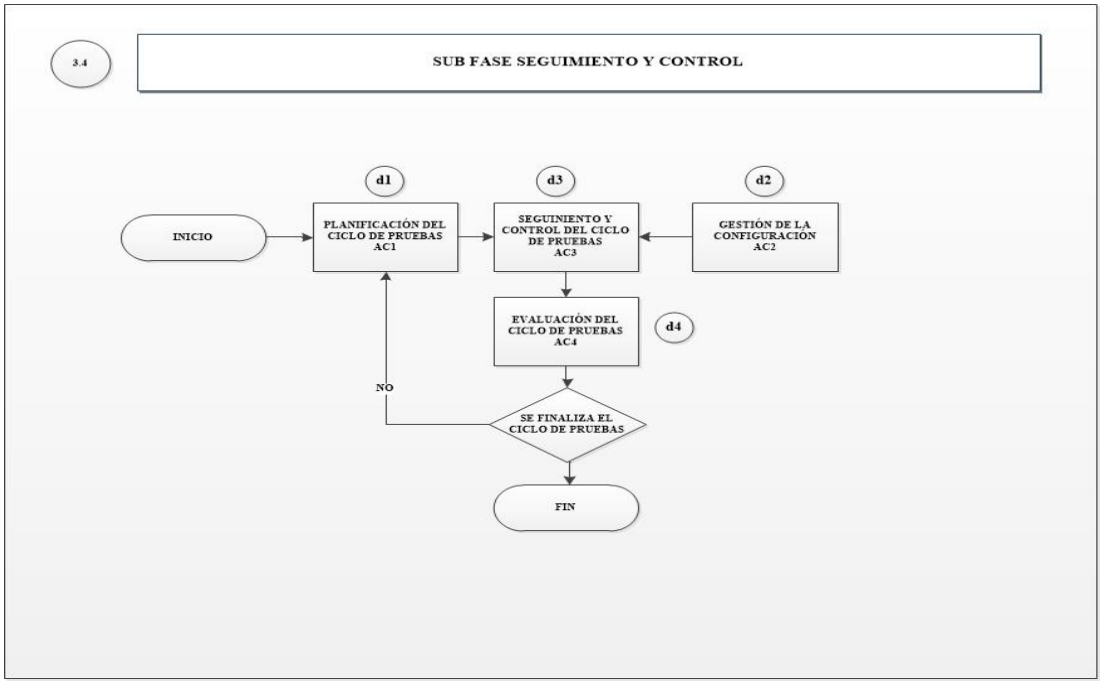


Ilustración: 3.15 Seguimiento y Control

Fuente: Elaboración Propia.

La planificación del proyecto de prueba realizada en la fase de Planificación, es revisada al comenzar cada ciclo en la actividad AC1-Planificación del Ciclo de Pruebas. El detalle de la planificación se hace a partir de las estimaciones realizadas en la actividad A1-Estimación de la Tareas. En la actividad PL3- Análisis de Riesgo del se definen las funcionalidades a probar en el ciclo. Durante todo el ciclo de prueba, se administra la configuración de los elementos que componen el testware en AC2 - Gestión de la Configuración y se realiza AC3 - Seguimiento y Control del Ciclo de Pruebas. Al finalizar el ciclo de pruebas se evalúa en la actividad AC4-Evaluación del Ciclo de Pruebas si es necesario derivar nuevos casos de prueba o si se termina el ciclo.

d.1 AC1 - Planificación del ciclo de pruebas.

- Objetivo.

Construir el plan de pruebas del ciclo de pruebas, estableciendo las correspondientes responsabilidades a fin de ejecutarlo.

- Descripción.

Aquí se realiza el plan de pruebas con la descripción de que se va a hacer en cada ciclo de pruebas, para lo cual se utiliza la actividad A1- Estimación de tareas, que se ven reflejadas en el documento ET-Estimación de Tareas, se utiliza la actividad RACP- Reporte de Avance del Ciclo de Pruebas, donde se encuentra lo sucedido en ciclos anteriores, se utiliza la actividad PL3- Análisis de riesgo, referente al riesgo asociado al producto de software a probar y mediante el LP-listado de pruebas se establecen las funcionalidades para cada ciclo del producto de software a probar y el PP-Plan de Pruebas donde está la planificación de las pruebas. Cabe destacar que es necesario considerar las pruebas de regresión que se realizarse en los ciclos correspondientes, ya que aquellos incidentes que se pudieron encontrar en ciclos anteriores son corregidos para los ciclos de la versión actual.

d2. AC2 - Gestión de la Configuración.

- Objetivo.

Administrar la configuración de los artefactos que forman el testware.

- Descripción.

Basado en (Kit, 1995) la gestión de la configuración permite que los artefactos generados durante el desarrollo de un producto de software, mantengan su integridad, para lo cual se controlan los cambios realizados. Al realizar las pruebas en un producto de software la Gestión de la Configuración permite administrar las pruebas, relacionando la versión de la prueba con él la versión del producto software a probar, es decir con su correspondiente configuración del software y hardware, que permita reproducir un incidente y así establecer con seguridad que se está probando la versión adecuada. Esta actividad se utiliza durante todo el ciclo de prueba, estableciéndose un reporte a su finalización lo que permite determinar los elementos que conforman el testware para cada ciclo.

d.3 AC3 - Seguimiento y Control del Ciclo de Pruebas.

- Objetivo

Determinar si las actividades del ciclo de pruebas se están realizando según lo planificado.

- Descripción

Esta actividad se ejecuta durante todo el ciclo de pruebas, permite controlar que los tiempos de cada tarea se realicen según lo planificado. Aquí se establecen la manera de mitigar los riegos y establecer contingencias que sean necesarias para probar el producto software. Una manera adecuada de realizar un seguimiento y control adecuado , según (Kit, 1995) es a través de determinar el número de casos de pruebas según lo siguientes enfoques:

- Casos de Prueba Planificados: Casos de Prueba que se planifican realizar.
- Casos de Prueba Disponibles: Casos de Prueba planificados que están disponibles para ser ejecutados.
- Casos de Prueba Ejecutados. Casos de Prueba Disponibles que han sido ejecutados.
- Casos de Prueba Exitosos: Casos de Prueba Ejecutados que en las ejecuciones más recientes no han detectado incidentes.

Realizar un gráfico del número de los distintos casos de prueba de manera habitual permite establecer las tendencias de las pruebas. Para llevar un control adecuado del ciclo de pruebas es necesario tomar las acciones correctivas establecidas en el seguimiento. Algunas consideraciones a tomarse en cuenta según la ISTQ (International Software Testing Qualifications Board) en el control de pruebas de producto software a probar son:

- Cuando un riesgo se manifiesta es necesario volver a priorizar las pruebas.
- Cambiar la planificación de las pruebas.
- Volver a priorizar las pruebas si ocurre un error.

En las pruebas de software pueden ocurrir inconvenientes los cuales son documentados en RI-Reporte de inconvenientes donde se pose un listado de los inconvenientes encontrados durante el ciclo de pruebas. Posteriormente se establece el RACP-Reporte de Avance del Ciclo de Pruebas, donde se establece la evolución de las actividades de prueba y la calidad de la versión del producto de pruebas probado, tomando en consideración: incidentes, requerimientos, configuraciones, etc.

d.4 AC4 - Evaluación de las Pruebas

- Objetivo

Determinar si es necesario realizar nuevas pruebas, de acuerdo a las pruebas realizadas en ciclos anteriores.

- Descripción

Antes de ejecutar esta actividad deben haberse realizado las siguientes tres sub-fases del ciclo de pruebas: Diseño de las Pruebas, Configuración del Entorno y Ejecución de las Pruebas.

La evaluación del ciclo de pruebas según (Kit, 1995) debe considerar lo siguiente:

- Evaluación de las funcionalidades: Evaluar los casos de prueba según las funcionalidades.
- Evaluación de los incidentes: Evaluar la calidad del producto según la ejecución de las pruebas realizadas.
- Evaluación de la efectividad: Evaluar las pruebas según el criterio de completitud y decidir cuándo parar o aumentar más pruebas y seguir, se basa en los dos anteriores.

Se deben responder las siguientes preguntas:

¿Continuar o parar las pruebas?

¿Qué pruebas adicionales son necesarias si se decide continuar?

iV. Evaluación y Resultados de la Prueba.

En esta fase se conocerá el nivel de satisfacción del cliente, se realizará un informe final, se evaluará el proceso de prueba para mejorarlo y guardar los elementos del proyecto de prueba para su uso en proyectos posteriores. En la Tabla 3.15 se muestran las actividades que se realizan en esta fase y los documentos generados.

Tabla 3.15 – Evaluación del Proyecto.

Actividades	Documentos generados
RP1- Determinar la Satisfacción del Cliente	ISC - Informe de Satisfacción del Cliente
RP2 - Reporte de Finalización del Proyecto	RFP - Reporte Final del Proyecto.
RP3 - Archivar el Proyecto	AP - Archivo del Proyecto
RP4 - Mejoramiento del Proceso de Prueba	MPP - Mejora del Proceso de Prueba
A1 - Estimación de las tareas	ET - Estimación de Tareas
A2 - Medición de Esfuerzo	ME - Matriz de Esfuerzo

.

Fuente: Elaboración Propia.

En la Ilustración 3.16 se muestra el diagrama de flujo para la fase Resultado de prueba. La actividad RP1- Determinar la Satisfacción del Cliente tiene como objetivo conocer el grado de satisfacción del cliente con el proyecto de prueba, a partir de este dato en la actividad RP4 - Mejoramiento del Proceso de Prueba, el equipo de prueba mejora el proceso para próximos proyectos, en la actividad RP2- se realiza el informe final del proyecto de prueba y en RP3- Archivar el Proyecto se almacenan los elementos del proyecto para ser consultados en próximos proyectos.

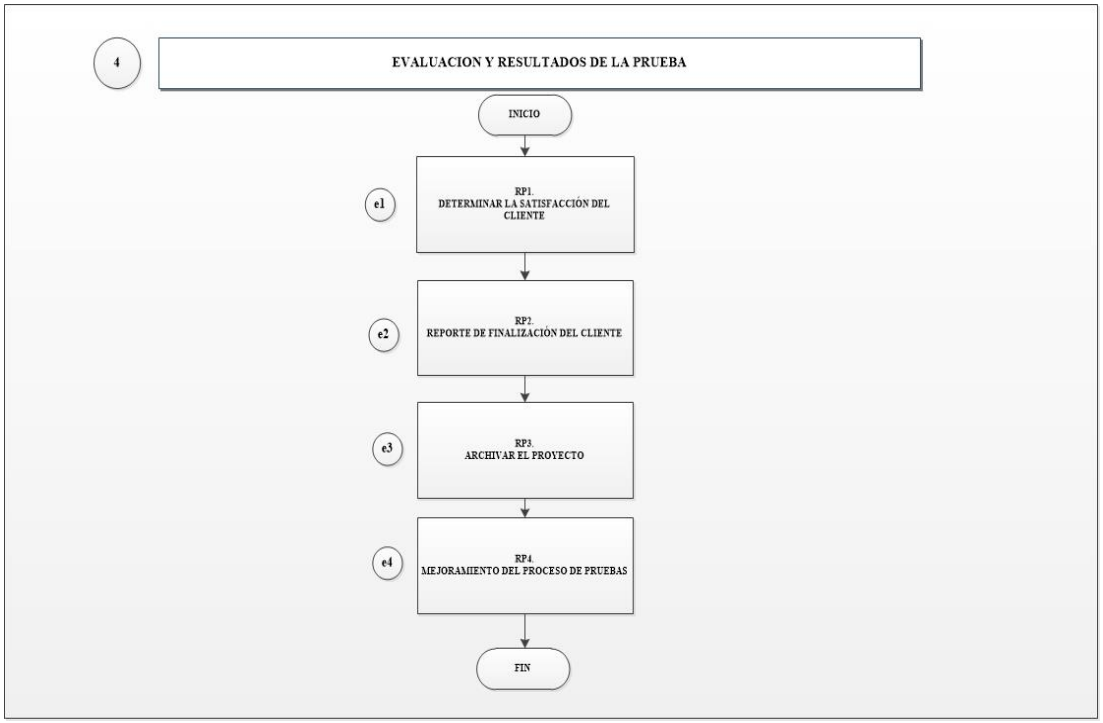


Ilustración: 3.16 – Actividades de la fase de Resultado de la Prueba.

Fuente: Elaboración Propia.

e1. RP1. Determinar la satisfacción del Cliente

- Objetivo.

Determinar la satisfacción del cliente en lo referente a las pruebas realizadas al producto software probado y su resultado.

- Descripción.

Cuando se terminan las pruebas se debe determinar el nivel de satisfacción del cliente con respecto a los resultados obtenidos del producto software probado, además se debe analizar que clase de comunicación existió y la perspectiva de las pruebas realizadas. Esto permitirán mejorar el proceso de pruebas, para esto se puede aplicar un cuestionario o una entrevista.

e2. RP2 – Reporte de Finalización de Proyecto.

- Objetivo

Generar un reporte final de las pruebas realizadas.

- Descripción

Al finalizar las pruebas de software, se genera un reporte final de las pruebas realizadas, tomando en consideración el esfuerzo total, por ciclo, por fase y por actividad así como las razones del porque existieron inconvenientes en seguir lo planificado. Cabe destacar que este reporte permite realizar estimaciones para pruebas futuras.

e3. RP3 – Archivar el Proyecto.

- Objetivo

Archivar el proyecto de pruebas para de esta manera utilizarlo en futuros proyectos.

- Descripción

Se organizan las partes que forman parte del proyecto de pruebas a fin de utilizarlos en futuras pruebas, es importante aclarar que el reporte final es parte del testware.

Las actividades A1-Estimacion de tareas y A2- Medición de Esfuerzo, son actividades que permiten establecer cómo va la planificación de las pruebas y la

realidad, permitiendo de esta manera re planificar los ciclos de pruebas o hacer una nueva planificación del ciclo o de todas las pruebas. Cabe destacar que se puede determinar cuáles son los casos de pruebas con el mayor esfuerzo.

e4. RP4 – Mejoramiento del Proceso de Prueba.

- Objetivo.

Establecer que se debe cambiar y reajustar en el proceso de pruebas para su mejoramiento.

- Descripción.

De acuerdo con (Burnstein I., 1996) para mejorar el Proceso de Pruebas es necesario tomar en consideración modelos de mejora ya preestablecidos como es el caso de Testing Maturity Model Integration (TMMI) siendo la base primordial el poseer un equipo de trabajo que se enfoque exclusivamente a la mejora continua. Según PSP (The Personal Software Process) es importante que luego de generar las pruebas correspondientes a un producto de software, el equipo de pruebas en conjunto con el equipo de mejoras del proceso de pruebas debe analizar y recomendar cambios para pruebas futuras, estableciéndose en ciertas ocasiones tareas específicas según el dominio del producto software a probar, siendo importante mantener memorias de las experiencias adquiridas. Las memorias de pruebas pasadas ayudan a ser una base de consulta para cualquier cambio en el proceso y las posibles causas de inconvenientes. Basado en PSP (The Personal Software Process) una alternativa importante es enfocar la mejora del proceso hacia la rentabilidad, es decir requerir un menor esfuerzo al realizar los cambios además es necesario luego de

realizar cualquier cambio, realizar un seguimiento para establecer si los cambios realizados han sido eficaces.

f1. A1 - Estimación de Tareas

- Objetivo

El equipo de pruebas debe estimar el tiempo que utilizara para realizar sus actividades en cada ciclo de pruebas a ejecutar.

- Descripción

Esta actividad se realiza tomando en consideración S1-Planificacion del Ciclo, organizando el tiempo requerido para cada miembro del equipo de pruebas. Según PSP (The Personal Software Process) la estimación es algo que con el tiempo se mejora, basándose en experiencias pasadas y archivos históricos, razón por lo cual la estimación puede equivocarse y se debe realizarse preferiblemente por partes.

f2. A2 Medición de Esfuerzo

- Objetivo.

Documentar el esfuerzo que se requiere para cada actividad que se realiza en las pruebas.

- Descripción

Determinar el esfuerzo medido en tiempo real es decir horas – persona de los integrantes del equipo de pruebas al realizar cada una de las actividades de prueba.

3.5 Roles.

El modelo de pruebas propuesto define roles y responsabilidades para la empresa Siredcom. Los roles establecidos son: Líder de Pruebas, Diseñador de Pruebas, Tester, Cliente y desarrollador.

El equipo de prueba está conformado por:

- Líder de Pruebas
- Diseñador de Pruebas
- Los Testers.

El líder es quien administra el proyecto de prueba. El diseñador es quien establece el diseño de los casos de prueba a partir de las especificaciones del producto y el Tester es quien ejecuta los casos de prueba y reporta los resultados encontrados.

3.6 Relación con el ciclo de vida de Desarrollo.

Aquí se establece la relación entre el modelo de pruebas propuesto y las etapas del ciclo de vida de desarrollo. Siendo independiente del ciclo de vida de desarrollo. Cuando el proyecto de pruebas empieza, cuando el proyecto de desarrollo en una fase más adelantada o cuando el producto de software ya está instalado en producción y se está realizando mantenimiento.

En la Ilustración 3.17 se establece el ciclo de vida de desarrollo a modo de iteraciones, realizándose en cada una las siguientes fases: Requerimientos (R),

Diseño (D), Implementación (I) y Prueba (P), aquí se enfocan a pruebas unitarias y de integración, generándose en cada iteración una nueva versión del producto de software. Además se muestra el modelo propuesto, relacionado con el ciclo de vida de desarrollo, siendo la fase Especificación de la Prueba (EP) la que se realiza antes de desarrollar el producto software, en la fase Planificación de la Prueba (PP) se lo realiza de manera paralela con el análisis de requerimientos, para determinar un primer análisis del riesgo del producto software, permitiendo definir los ciclos de prueba. Ya planificada la prueba, los ciclos de prueba siguen al ciclo de desarrollo de esta manera:

- La sub-fase Diseño de las Pruebas (DP) se estructura desde el análisis de requerimientos realizado por el equipo de desarrollo. Esta sub-fase se la puede realizar de manera paralela con las fases de Diseño, Implementación y Prueba del ciclo de vida de desarrollo. Al liberar la versión 1, se ejecuta la sub etapa Entorno de Pruebas (EP), instalándose la versión en el ambiente de prueba, para posteriormente realizar la fase de Ejecución de Pruebas (EJP). Mientras se realiza el ciclo de pruebas se realiza la sub-fase Seguimiento del Ciclo (SC).

- Es de mayor utilidad para el equipo de desarrollo la ejecución de las pruebas de la versión anterior, antes de liberar una versión posterior, a fin de que los incidentes encontrados en las pruebas no se desperdicien y permitan realizarse las correcciones necesarias.

- Como se observa en la Ilustración 15, los ciclos de prueba se solapan en el tiempo, ya que mientras se están ejecutando las pruebas de una versión se puede realizar a la vez el diseño de las pruebas del ciclo siguiente.

Al terminar el proyecto de prueba se realiza la etapa Resultado de Prueba (RP).

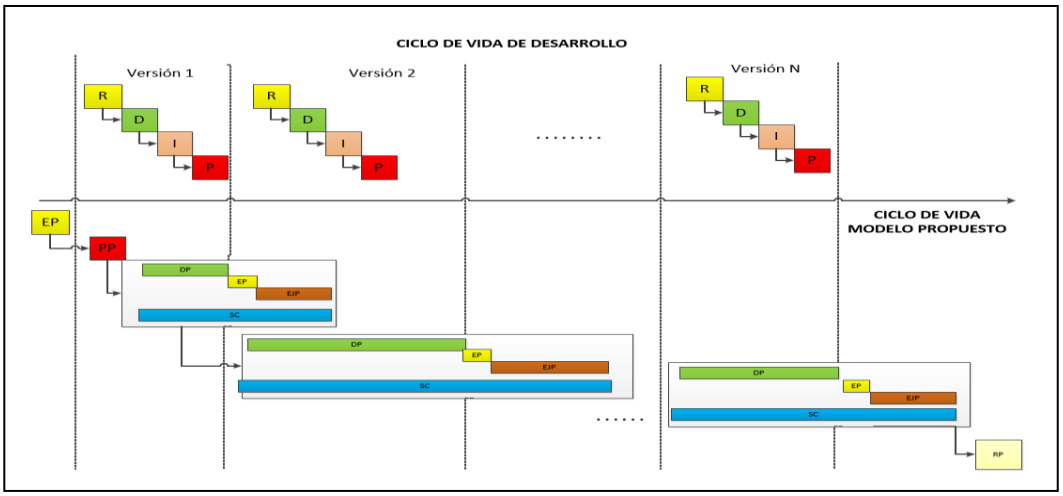


Ilustración 3.17 – El modelo propuesto en relación al ciclo de vida de desarrollo

Fuente: Elaboración Propia.

El modelo de pruebas propuesto es enfocado de una manera genérica, razón por lo cual se puede adaptar a las características propias de un producto software, a continuación se detallan ciertas pautas que sucederán, cuando el modelo propuesto de pruebas acompaña al proyecto de desarrollo, siendo importante la planificación:

- El equipo de pruebas puede sugerir mejoras a la especificación de requisitos, basándose en los documentos de requerimientos, lo cual se verá reflejado en los casos de prueba.

- Generalmente el equipo de desarrollo genera muchas versiones al inicio, razón por lo cual se debe establecer cuales se probaran y cuales no se tomaran en consideración.
- En las primeras iteraciones del producto software por lo general lo vuelven muy inestable razón por la cual los incidentes ya han sido solucionados por el equipo de desarrollo.
- Generalmente mucho de los casos de pruebas generados en las primeras iteraciones se descartan, debido al constante cambio que sufre el producto software.
- Se deben realizar entrevistas con los clientes, a fin de definir los datos de pruebas a ser considerados lo que permitirá ser los adecuados, ya que es difícil al inicio.
- Ya que al inicio el producto de software cambia contantemente, en los ciclos de prueba iniciales se encontraran muchos incidentes, por lo que se requerirá realizar varias pruebas de regresión.
- La planificación de las pruebas en relación a los ciclos de prueba, variara considerablemente ya que el plan de desarrollo también variara.

El modelo propuesto para pruebas puede ser aplicado cuando el mismo ya se encuentra en una fase avanzada en su desarrollo o se encuentra en mantenimiento, teniendo que el equipo de pruebas aprender del producto software a ser probado. Se describen algunas pautas que se deben considerar cuando al planificar las pruebas en cada ciclo:

- Si no existen especificaciones adecuadas o incompletas, se puede mejorar basándose en los requerimientos de la última versión del producto software que se disponga.
- Es de suma utilidad disponer del manual de usuario del producto software, al momento en que se le está dando mantenimiento, lo que permitirá entender que se está probando y de ser necesario mejorar sus requerimientos.
- Si el producto software se encuentra en mantenimiento, se puede contar con datos reales para generar los datos de prueba.
- Según (Kaner et al., 2001) al realizar un cambio en el producto software es necesario que en el ciclo de pruebas se incluya las establecen funcionalidades que se modificaron, para lo cual se debe consultar a las sugerencias del cliente y de los desarrolladores lo que permitirá analizar los posibles impactos y así determinar que pruebas de regresión se deben ejecutar.
- Se respetan las fechas de comienzo de los ciclos de prueba, debido a que las versiones del producto de software son más estables, cumpliéndose la planificación de desarrollo.

3.7 Documentación Generada del Modelo de Pruebas funcional propuesto.

En la tabla 3.16 se detallan los documentos generados en el modelo de pruebas propuesto, donde se detalla el roles (responsabilidades), así como la entrada y salida de las diferentes fases propuestas.

Tabla 3.16 – Documentos Generados y Sus Relaciones.

Documento Generado	Responsable	Entrada	Salida
CP - Ciclos de Prueba	Líder de Pruebas.	PL7. Planificación de las Pruebas. PL5 - Definición de los Ciclos de Prueba	PL1. Acuerdo con el cliente. PL5 - Definición de los Ciclos de Prueba
MPP - Mejoramiento del Proceso de Pruebas.	Líder de Pruebas.		RP4. Mejoramiento del Proceso de Prueba
CP- Casos de Prueba	Diseñador de Pruebas.	AC3 - Seguimiento y Control del Ciclo de Pruebas. AC4 - Evaluación del Ciclo de Pruebas. DP2. Validación de los Casos de Prueba DP3. Asignación de los Casos de Prueba EP2 - Ejecución de las Pruebas. EP4 - Reporte de Incidentes.	DP1. Diseño de los Casos de Prueba
CPR – Casos de Prueba de Regresión	Diseñador de Pruebas.	EP6 – Verificación de las Correcciones.	
Versión Ejecutable del Producto (EJ)	Tester.	EP1 - Pruebas de Humo. EP2 - Ejecución de las Pruebas. EP3 - Testing Exploratorio.	C2- Configuración
Estimación de Tareas (ET)	Líder de Pruebas.	AC1 - Planificación del Ciclo de Pruebas.	A1 - Estimación de las tareas
Especificación de Requerimientos (ERS)	Cliente	Ep2 - Revisión de requerimientos PL2-Revisión de requerimientos.	
LP - Listado de Prueba	Diseñador de Pruebas	Ep1- Necesidad del cliente Ep2- Revisión de requerimientos Ep3- Análisis de riesgo PL1- Acuerdo con el Cliente PL2 - Revisión de requerimientos PL3 - Análisis de riesgo PL4 – Exploración del Producto PL5 - Definición de los Ciclos de Prueba PL7 - Planificación de las Pruebas AC1 - Planificación del Ciclo de Pruebas. DP1 – Diseño de los Casos de Prueba	Ep1- Necesidad del cliente Ep2- Revisión de requerimientos Ep3- Análisis de riesgo PL1- Acuerdo con el Cliente PL2 - Revisión de requerimientos PL3 - Análisis de riesgo PL4 – Exploración del

Continua ⟶

		PL6 - Definición del Testware	Producto
ISC – Informe de Satisfacción del Cliente	Líder de Prueba	RP4 – Mejoramiento del Proceso de Prueba	RP1 – Determinar la satisfacción del Cliente
MT – Matriz de Trazabilidad	Diseñador de pruebas	AC3 - Seguimiento y Control del Ciclo de Pruebas.	DP1. Diseño de los Casos de Prueba
PD-Plan de Desarrollo del producto software a probar.		Ep1- Necesidad del cliente PL1 - Acuerdo con el Cliente PL6 - Definición del Testware PL5 - Definición de los Ciclos de Prueba	
PE-Planilla de Esfuerzo	Líder de Prueba	AC3 - Seguimiento y Control del Ciclo de Pruebas A2 - Medición de Esfuerzo	A2 - Medición de Esfuerzo
PEC - Plan de Ejecución del Ciclo	Diseñador de Pruebas		DP3 – Asignación de los Casos de Prueba
PI - Proceso de Incidentes	Líder de prueba	PL7 - Planificación de las Pruebas	PL8 - Definición del Proceso de Incidentes
PP – Plan de Pruebas		AC1 - Planificación del Ciclo de Pruebas. A1 - Estimación de las tareas. AC4 - Evaluación del Ciclo de Pruebas RP3 - Archivar el Proyecto	PL7 - Planificación de las Pruebas
PPC - Plan de Pruebas del Ciclo	Líder de Prueba	AC3 - Seguimiento y Control del Ciclo de Pruebas.	AC1 - Planificación del Ciclo de Pruebas.
AA - Acuerdo Alcanzado			Ep1- Necesidad del cliente
PT - Proceso de Prueba	Líder de Pruebas	RP4 – Mejoramiento del Proceso de Prueba	
RACP - Reporte de Avance del Ciclo de Pruebas.		AC1 - Planificación del Ciclo de Pruebas. A1 - Estimación de las tareas. RP4 – Mejoramiento del Proceso de Prueba RP3 - Archivar el Proyecto	AC3 - Seguimiento y Control del Ciclo de Pruebas
RGC - Reporte de la Gestión de la Configuración.	Tester	RP3 - Archivar el Proyecto	AC2 - Gestión de la Configuración
RECP - Reporte de Evaluación del Ciclo de Pruebas	Líder de Prueba	RP4 – Mejoramiento del Proceso de Prueba RP3 - Archivar el Proyecto	AC4 - Evaluación del Ciclo de Pruebas
RFP - Reporte Final del Proyecto.	Líder de Prueba		RP3 - Archivar el Proyecto
RI – Reporte de Incidentes	Tester	AC4 - Evaluación del Ciclo de Pruebas EP5 - Validación de los Incidentes. EP6-Verificación de las Correcciones.	EP4 - Reporte de Incidentes. EP5- Validación de los Incidentes

Continua ⟶

		AC3 - Seguimiento y Control del Ciclo de Pruebas.	
RP- Reporte de Problemas	Tester	AC4 - Evaluación del Ciclo de Pruebas	AC3 - Seguimiento y Control del Ciclo de Pruebas. E1 - Instalación de Herramientas E2 – Configuración.
RE - Reporte de Ejecución.	Tester	AC3 - Seguimiento y Control del Ciclo de Pruebas. AC4 - Evaluación del Ciclo de Pruebas EP4 - Reporte de Incidentes. RP4 – Mejoramiento del Proceso de Prueba	EP1 - Pruebas de Humo. EP2 - Ejecución de las Pruebas. EP3 - Testing Exploratorio EP6-Verificación de las Correcciones.
RR – Resumen de Reunión	Líder de la Prueba		Ep1- Necesidad del cliente. PL1 - Acuerdo con el Cliente. DP2 – Validación de los Casos de Prueba.
TW - Testware	Diseñador de Pruebas.	E1 - Instalación de Herramientas RP3 - Archivar el Proyecto	PL6 - Definición del Testware RP3 - Archivar el Proyecto

Fuente: Elaboración Propia.

A continuación se describe cada uno de los documentos generados por el modelo propuesto.

3.7.1 Ciclos de Prueba (CP).

El Documento donde se definen los ciclos de prueba que se ejecutaran, tomando en consideración fechas de inicio y de finalización, versiones de desarrollo, lista de funcionalidades de cada ciclo, la prioridad correspondiente. Este documento se relaciona con el PP- Plan de Pruebas.

3.7.2 Mejoramiento del Proceso de Prueba (MPP).

Posee que mejoras y arreglos que se deben hacer la Proceso de Pruebas, argumentando por qué y quien lo argumenta.

3.7.3 Casos de Prueba (CP).

Según (Black, 2002) cada caso de prueba consiste de tres valores: acciones, datos y resultados esperados.

Los datos necesarios para identificar cada caso de prueba incluyen:

- Identificador: Identificador único para el caso de prueba
- Conjunto de prueba: Identificador de los conjuntos de prueba donde el caso de prueba es usado.
- Nombre.
- Funcionalidad: Funcionalidad del IP-Inventario de Prueba que se prueba con este caso de prueba.
- Prioridad: Prioridad asignada al caso de prueba.
- HW: Lista de Hardware requerido para ejecutar el caso de prueba
- SW: Lista de Software requerido para ejecutar el caso de prueba Configuración (setup): Lista de pasos necesarios para comenzar las pruebas.
- Retroceder acciones (cleanup): Lista de pasos necesarios para retroceder al estado previo a la prueba.

- Condiciones: Lista de condiciones que deben cumplirse al ejecutar las pruebas.
- Resultado esperado: Especificación del resultado esperado de ejecutar las pruebas.
- Los siguientes datos se completan una vez que el caso de prueba se ejecutó:
- Resultado real: Especificación de lo que realmente ocurrió: Pasó/Falló/No se pudo ejecutar
- Duración: Tiempo reloj en ejecutar el caso de prueba.
- Esfuerzo: Horas-Persona requeridas para conIlustraciónr y ejecutar el caso de prueba.
- Fecha: Fecha en la que se realizó la prueba
- Identificación del Defecto: En caso de que la prueba falle se debe identificar el incidente reportado para su seguimiento
- Versión: Identificador de la versión probada
- Tester: Nombre del tester que ejecutó el caso de prueba

3.7.4 Casos de Prueba de Regresión (CPR).

Son casos de prueba que se ejecutan al iniciar cada ciclo, que permite establecer si un incidente se corrigió y se lo puede cerrar. Se conservan los datos que identifican al caso de prueba y se modifican solamente los datos de la ejecución, donde se determina que resultados existieron al ejecutar el caso de prueba de la nueva versión del software probado.

3.7.5 Versión Ejecutable del Producto (EJ).

Es la versión ejecutable del producto software a probar para ciclo de pruebas, debe poseer información sobre que se hizo a la versión para arreglarla y optimizarla.

3.7.6 Estimación de Tareas (ET)

Se establece según la planificación de la pruebas para cada uno de los ciclos, la estimación del esfuerzo de las personas que intervienen, estas estimaciones son preferibles realizarlas semanalmente.

3.7.7 Especificación de Requerimientos (ERS).

Es de donde se obtiene la información correspondiente a las especificaciones del producto a probar, pudiendo ser los ERS, manuales, entrevistas, prototipos entre otros.

3.7.8 Informe de Satisfacción del Cliente (ISC)

Aquí se establece la satisfacción del cliente en relación a los resultados obtenidos sobre las pruebas del producto de software probado y como fue el desempeño del equipo de pruebas.

3.7.9 Listado de Pruebas (LP)

El listado de pruebas es un documento donde se especifica las funcionalidades del producto de software a ser probado, donde se debe tener en consideración lo siguiente:

- Identificador: Referencia única de la especificación de requerimientos, además se escribe, es donde se lleva la descripción detallada de la funcionalidad a probar.
- Nombre: Nombre de la funcionalidad a probar.
- Complejidad: Indica lo complejo que es para los desarrolladores implementar dicha funcionalidad. Los valores posibles son: Alta, Media y Baja.
- Criticidad: Indica cuán crítica para el negocio es dicha funcionalidad. Los valores posibles son: Alta, Media y Baja.
- Prioridad: Indica la prioridad que tiene esa funcionalidad para las pruebas. Los valores posibles son: Alta, Media y Baja.

3.7.10 Matriz de Trazabilidad (MT).

Para cada elemento del LP – Listado de Prueba, se construye una matrìz donde se relaciona la funcionalidad y los casos de prueba del producto software que permitan probar dicha funcionalidad. Esto permite saber qué casos de prueba cubren qué requerimientos de prueba de acuerdo con (Kit, 1995). Esta matriz permite establecer una trazabilidad de funcionalidades, pudiéndose establecer el impacto que causa el cambio de los requerimientos.

3.7.11 Plan de Desarrollo del Producto (PD).

En este plan se establece la planificación de las versiones que el equipo de desarrollo generará con sus correspondientes funcionalidades, siendo esto un

documento que permite al equipo de pruebas planificar las pruebas correspondientes a cada versión del producto software a probar.

3.7.12 Planilla de Esfuerzo (PE).

Es aquí donde se establece el esfuerzo (horas/persona) que se proyecta para realizar las actividades de prueba del producto software a ser probado, tomando en consideración:

- Actividad: Identificador de la actividad que se reporta
- Fecha: Día en que se realizó la actividad
- Esfuerzo: Minutos que se realizó la actividad
- Observaciones: Descripción breve de la actividad realizada

Existen actividades en las que se puede requerir mayor información. Para las siguientes actividades, interesa contar con el tiempo requerido como se detalla en la Tabla 3.17

Tabla 3.17 – Información adicional para estimación del esfuerzo.

Actividad de Prueba	Información Adicional
PL2-Revisión de requerimientos.	Tiempo que se requiere para cada requerimiento
DP1 – Diseño de los Casos de Prueba	Tiempo que se requiere para cada caso de prueba
EP2 - Ejecución de las Pruebas.	Tiempo que se requiere para: - Ejecutar el caso de prueba por primera vez - Ejecutar el caso de prueba para determinar si existe verdaderamente una incidencia. - Examinar otros casos de prueba a fin de determinar otros escenarios donde se manifiestan los incidentes.

EP3 - Testing Exploratorio	Tiempo que se requiere para hacer testing exploratorio
EP4 - Reporte de Incidentes.	Tiempo que se consume por cada incidente reportado
EP6-Verificación de las Correcciones.	Tiempo que se requiere para cada caso de regresión

Fuente: Elaboración Propia.

3.7.13 Plan de Ejecución del Ciclo (PEC)

Se planifica la ejecución de los casos de los casos de prueba en cada ciclo además se establece un orden y testers encargados de realizarlos.

3.7.14 Proceso de Incidentes (PI)

Aquí se determinan los estados en que un incidente puede llegar a alcanzar, desde que el equipo de pruebas lo detecta hasta que el equipo de desarrollo lo arregla. Cabe destacar que el proceso de incidentes es requerido por el plan de pruebas.

El proceso de incidentes toma en consideración lo siguiente:

- Estados del proceso de incidentes
- Personas involucradas en el proceso de incidentes
- Condiciones para pasar de un estado a otro del proceso de incidentes

3.7.15 Plan de Pruebas (PP).

En el Plan de Pruebas se define quién, cuándo, dónde y cómo se realizarán las actividades en el proyecto de prueba. Los puntos principales del Plan de Pruebas según (Black, 2002) son :

1. Introducción
2. Agenda
3. Condiciones
4. Requerimientos del Entorno
5. Desarrollo de las pruebas
6. Ejecución de las pruebas
7. Personas y roles
8. Requerimientos de Capacitación
9. Proceso de Seguimiento de Incidentes
10. Clasificación de los defectos
11. Administración de las Versiones
12. Ciclos
13. Entregables

Se detallan de una manera general cada uno:

1. Introducción: Determina que se va a probar establece objetivos, políticas y metodologías que se utilizan. Contiene:

    - Alcance: Se establece hasta donde se llevara a cabo las pruebas y que no se probara.
    - Lugar: Donde físicamente se realizaran las pruebas.
    - Definiciones: Se determina un glosario de términos referentes a pruebas de software
    - Referencias: Listar los documentos referenciados por este plan junto con su versión.

2. Planeación: Se determina la planeación de las pruebas, requiriéndose las fechas de cada una de las actividades de pruebas a realizarse es decir su inicio y su finalización.

3. Condiciones: Determina que se debe cumplir y hacer en cada una de las fases de las pruebas. Siendo necesario:

   - Condiciones de entrada: Se cumplen cuando inicia un ciclo de pruebas, como por ejemplo:

     - El sistema de seguimiento de incidentes fue definido.
     - Todos los componentes están bajo administración de la configuración.
     - El equipo de desarrollo configuró el entorno.
     - El equipo de pruebas tiene los permisos adecuados de acceso al sistema.
     - El equipo de desarrollo ha completado todas las funcionalidades (y reparado todos los defectos) establecidas para esta versión.
     - El equipo de desarrollo ha realizado las pruebas unitarias de todos los componentes de esta versión.
     - El equipo de desarrollo provee el software al equipo de pruebas 3 días calendario antes de comenzar las pruebas.
     - El equipo de prueba realiza pruebas de humo durante 3 días y reporta los resultados en una reunión junto con el gerente del proyecto. En esa reunión deben resolverse los plazos en caso de no poder seguir las pruebas.

   - Condiciones para continuar: Se cumplen las condiciones establecidas para continuar con las pruebas, como por ejemplo:

- Todo el software liberado al equipo de pruebas está acompañado de notas de la versión.
- Se hacen reuniones semanales de revisión de incidentes para gestionar los incidentes abiertos y los tiempos de cierre.

- Condiciones de salida: Se cumplen ciertas condiciones para terminar las pruebas de un ciclo, como por ejemplo:
  - El equipo de pruebas ha ejecutado todas las pruebas planificadas.
  - El equipo de pruebas ha verificado que todos los incidentes en el sistema de seguimiento están cerrados o rechazados, verificado por medio de pruebas de regresión.
  - Se tiene una reunión entre el Gerente del Proyecto y el equipo de prueba para acordar que se ha terminado el ciclo.

4. Requerimientos del Entorno: Establece el recurso técnico (hardware – software) que se necesitan para poder ejecutar las pruebas.

   - Hardware Base: Presenta los requerimientos de Hardware necesarios para probar el sistema.
   - Software Base: Presenta los requerimientos de Software necesarios para probar el sistema.
   - Herramientas: Presenta las herramientas necesarias para probar el sistema.

5. Desarrollo de las pruebas: Establece como el equipo de pruebas construye el Testware, así como los datos que se necesitan para generar las pruebas.

6. Ejecución de las pruebas: Establece como se administraran las pruebas, las versiones correspondientes y los datos que se requieren para su ejecución.
7. Personas y Roles: Establece la información de quienes representan los roles en la ejecución de las actividades de pruebas, con sus correspondientes responsabilidades.
8. Requerimientos de Capacitación: Establece que capacitación necesita el equipo de pruebas para poder hacer las pruebas.
9. Proceso de Seguimiento de Incidentes: Establece como se informa de un incidente y su posterior seguimiento.
10. Clasificación de los Incidentes: Establece como se clasifican los incidentes hallados.
11. Administración de las Versiones: Establece quien se encargara de instalar y configurar las versiones a ser probadas en el ambiente de pruebas.
12. Ciclos: Se sigue lo establecido en CP-Ciclos de pruebas.
13. Entregables: Establece que documentos (entregables) se darán al cliente durante la ejecución de las pruebas.

3.7.16 Plan Pruebas del Ciclo (PPC).

Este plan posee la planificación que se va a seguir durante un ciclo de pruebas y las actividades a realizar, incluye:

- Planificación para cada actividad de pruebas a realizarse en cada ciclo, tomando en consideración las correspondientes responsabilidades de cada miembro del equipo de pruebas y su correspondiente tiempo a ocupar.

- Funcionalidades que se probarán en cada ciclo tomando en consideración su orden.
- Incidentes reparados en la versión.
- Cambios realizados en la versión.
- Pruebas de regresión que se ejecutarán en el ciclo.
- Desviaciones respecto a lo planificado en PP-Plan de Pruebas.
- Riesgos del proyecto de prueba y mecanismos para su mitigación y contingencia.

3.7.17 Acuerdo Alcanzado (AA).

El Acuerdo Alcanzado es el resultado de la fase EP1 – Necesidad del Cliente, donde se establece el alcance de las pruebas, la planificación preliminar de los ciclos de prueba.

3.7.18 Reporte de Avance del Ciclo de Pruebas (RACP)

Es establecer que fue lo que se realizó en cada ciclo de prueba. Posee lo siguiente:

- Las mediciones realizadas durante el ciclo
- Gráficas mostrando la información resumida
- La planificación real del ciclo y la desviación respecto a la planificación estimada.
- Los obstáculos ocurridos durante el ciclo de prueba.
- Acciones tomadas al realizar el control del ciclo.

Para poder reportar el avance del ciclo, se debe haber medido los siguientes elementos:

- Funcionalidades: Mediciones relacionadas con las funcionalidades probadas durante el ciclo.
- Casos de prueba: Mediciones relacionadas con la planificación, diseño y ejecución de los casos de prueba.
- Incidentes: Mediciones relacionadas con los incidentes encontrados durante el ciclo.
- Esfuerzo: Tiempo invertido en realizar cada actividad del Proceso.
- Obstáculos: Problemas que ocurrieron durante las pruebas

En la Tabla 3.18 se listan las métricas que se requieren tomar en consideración en el modelo propuesto para la empresa SiredCom, según (Kit, 1995) y (Kaner, 2001).

Tabla 3.18 – Métricas sobre el ciclo de pruebas.

Componentes del Ciclo de Prueba	**Mediciones (Métricas)**	**Fórmula**
Funcionalidades del producto software a probar en cada ciclo	Funcionalidades probadas	Número de Casos de Prueba / Número total de funcionalidades a ser probadas.
	Tiempo utilizado para probar la funcionalidad del producto software a probar.	Número total de tiempo en ejecutar las pruebas / Total de Funcionalidades probadas
	Defectos encontrados al probar la funcionalidad del producto software a probar.	Número total de incidentes / Número de funcionalidades probadas
	Funcionalidades cambiantes	Número de funcionalidades que han cambiado a partir de la última versión probada.
	Porcentaje de pruebas que se han cambiado por haberse cambiado la funcionalidad del producto software a probar.	Número de pruebas modificadas por cambios en las funcionalidades / Número total de pruebas
Casos de Prueba que se ejecutan para probar el producto software en cada ciclo	Casos de pruebas exitosos	Número de pruebas ejecutadas que llegaron a pasar.
	Casos de pruebas fallidos	Número de pruebas ejecutadas que llegaron a fallaron.
	Casos de pruebas ejecutados	Es la suma de casos de pruebas exitosos más casos de pruebas ejecutados.

Continua ⟶

	Casos de pruebas no ejecutados	Número de casos de prueba Planificados para el ciclo que no fueron ejecutados
	Casos de pruebas planificados	Número de casos de prueba Planificados para el ciclo que fueron ejecutados
	Caso de pruebas disponibles	Número de casos de prueba que se diseñaron para el ciclo
	Caso de pruebas disponibles ejecutados (3)	Número de casos de prueba disponibles en el ciclo que fueron ejecutados.
	Pruebas de regresión ejecutadas(4)	Número de casos de prueba ejecutados en el ciclo, que se ejecutaron en ciclos anteriores.
	Tiempo medio por caso de prueba	Tiempo total en ejecutar los casos de prueba / Número total de casos de prueba ejecutados
	Incidentes por caso de prueba	Número total de incidentes reportados en el ciclo / Número de casos de prueba ejecutados
	Eficiencia de las pruebas.	Número de pruebas ejecutadas / Número total de incidentes encontrados.
Testing Exploratorio	Sesiones de Testing exploratorio	Número de sesiones de Testing exploratorio que se realizaron en el ciclo
	Esfuerzo de Testing Exploratorio	Tiempo que se requiere para el testing exploratorio en el ciclo.
	Incidentes encontrados en el Testing Exploratorio	Número de incidentes encontrados durante el testing exploratorio en el ciclo
	Funcionalidades Exploradas	Número de Funcionalidades exploradas en el ciclo, durante el testing exploratorio
Incidentes	Incidentes por estado	Número de incidentes en el ciclo según su estado(nuevo, resuelto, pendiente, reabierto, etc)
	Incidentes por categoría	Número de incidentes en el ciclo según su categoría
	Incidentes por tipo	Número de incidentes en el ciclo según su tipo.
	Incidentes por severidad	Número de incidentes en el ciclo según su severidad.
	Incidentes que no se pudieron reproducir	Número de incidentes que no se pudieron reproducir por el equipo de desarrollo en el ciclo.
	Falsos Incidentes	Número de incidentes que el equipo de pruebas en el ciclo reporto, que el cliente no considero un incidente.
	Tasa de reparación de incidentes.	Número de Incidentes reparados en el ciclo / Número de incidentes encontrados en el ciclo anterior.

Continua ⟶

	Tiempo Medio en resolver incidentes	Número total de tiempo en resolver los incidentes desde que se reportaron / Número de incidentes resueltos en el ciclo
	Calidad de la reparación de incidentes	Número de incidentes reabiertos en el ciclo/Número de incidentes reparados para el ciclo
	Tiempo medio en reportar incidentes	Total de tiempo en ejecutar casos de prueba / Cantidad de incidentes reportados en el ciclo
Esfuerzo	Esfuerzo Planificado (5)	Sumar el tiempo planificado para todas las actividades de las pruebas
	Esfuerzo realizado en el ciclo	Sumar el tiempo invertido en todas las actividades del ciclo
	Esfuerzo realizado hasta el momento (6)	Sumar el tiempo realizado desde que se inician las pruebas hasta el último ciclo
	Esfuerzo por realizar(7)	Sumar el tiempo el tiempo que se planifica para cada actividad hasta q las pruebas terminen
	Esfuerzo total (8)	Sumar (6) y (7)
	Desviación en el esfuerzo planificado	Restar (8) – (5)
	Esfuerzo por persona	Contar el esfuerzo por persona del equipo de prueba.
	Esfuerzo en reportar incidentes	Sumar el tiempo en reportar los incidentes encontrados en el ciclo
	Esfuerzo por actividad	Para cada actividad del ciclo, sumar el tiempo en realizarla por cada integrante.
	Esfuerzo por etapa	Para cada fase del ciclo, sumar el tiempo en realizarla por cada integrante
Problemas	Problemas reportados	Número de problemas que fueron establecidos en el ciclo.
	Esfuerzo en problemas	Contar el tiempo invertido por el tiempo de pruebas en resolver los problemas
	Casos de prueba bloqueados	Número de pruebas bloqueadas por incidentes.
	Pruebas de Humo que fallaron	Número de fallas al pasar las pruebas de humo en el ciclo.
	Esfuerzo perdido en el ambiente de prueba	Sumar el esfuerzo invertido en problemas con el ambiente de pruebas.
	Incidentes causados por el ambiente de pruebas	Sumar la cantidad de incidentes relacionados al ambiente de pruebas.

Fuente: Measurement Issues and Software Testing. Y Software Testing In The Real World : Improving The Process [17]

[17] Tomado de (Kit, 1995) y (Kaner, 2001)

Existen métricas externas que miden la calidad del producto de software a ser probado durante las pruebas, aquellas están definidas en la norma ISO 9126.

3.7.19 Reporte de la Gestión de la Configuración (RGC).

En este reporte se encuentran los elementos de configuración de las pruebas con su correspondiente línea base y las últimas versiones.

3.7.20 Reporte de Evaluación del Ciclo de Pruebas (RECP)

En este reporte se establece un informe que determina como se realizaron las pruebas en el ciclo, hace un resumen de la información contenida en el RACP - Reporte de Avance del Ciclo de Pruebas, el cual es entregado al cliente

3.7.21 Reporte Final del Proyecto (RFP)

Este reporte hace un resumen de todo el proyecto de pruebas, aquí se complementa la información del RACP - Reporte de Avance del Ciclo de Pruebas y posee lo siguiente:

i. Las mediciones para todo el proyecto de prueba: Se realizan las mediciones de la Tabla 3.18 para todo el proyecto, también se realizan las mediciones de la Tabla 3.19
ii. Gráficas mostrando la información más importante.
iii. La planificación real del proyecto y la desviación respecto a lo planificado.
iv. Los obstáculos ocurridos durante el proyecto de prueba.
v. Visión global de la calidad del producto.

Tabla 3.19 – Métricas finales del proyecto.

Medición	Fórmula
Ciclos Planificados	Número de Ciclos Planificados para las pruebas del software a probar.
Ciclos Realizados	Numero de ciclos de pruebas realizados para probar el producto software.
Desviación de lo planeado y lo sucedido	Esfuerzo planificado para las pruebas – Esfuerzo real de las pruebas.
Cambios en el plan de pruebas	Numero de cambios hechos en el plan durante todas las pruebas
Porcentaje de documentación y entregables dados al cliente	Cantidad de documentos y entregables dados al cliente / Total de documentos y entregables planificados dar al cliente.
Incidentes no resueltos al finalizar las pruebas	Número de incidentes sin resolver al terminar las pruebas
Incidentes en los ciclos	Número de incidentes encontrados en los ciclos

Fuente: Measurement Issues and Software Testing. Y Software Testing In The Real World : Improving The Process [18]

3.7.22 Reporte de Incidentes (RI)

Se documenta cualquier incidente que se haya presentado al realizar las pruebas, los cuales requieren un análisis posterior. Para documentar un incidente es necesario:

i. Identificador: Identificador único.
ii. Caso de prueba: Identificador del caso de prueba donde se encontró el incidente.
iii. Resumen: Descripción breve del incidente.
iv. Pasos: Pasos necesarios para la reproducción del incidente.
v. Versión: Versión del producto que se está probando.
vi. Prioridad: Importancia del incidente encontrado, esta puede ser:

[18] Tomado de (Kit, 1995) y (Kaner, 2001)

- Crítica: La ejecución del sistema es interrumpida, no se puede seguir con la prueba
- Alta: La aplicación sigue funcionado pero el problema tiene un alto impacto
- Media: El problema tiene impacto medio en la aplicación
- Bajo: El problema tiene poco impacto en la aplicación

vii. Categoría: Categoría a la que corresponde el incidente.

viii. Tester: Nombre del tester que reporta el incidente.

ix. Estado: Estado del incidente según el PI – Proceso de Incidentes definido.

3.7.23 Reporte de Ejecución (RE)

Este reporte posee la información necesaria para realizar cada CP-Caso de Prueba que se realizan en cada ciclo.

3.7.24 Reporte de Problemas (RP)

Los problemas encontrados son riesgos que se materializaron cuando se estaban ejecutando las pruebas. En este reporte se determinan las causas de que las pruebas no sean eficientes. Según (Kaner, 2001) se establece por cada problema el esfuerzo realizado, para lo cual se debe tomar en consideración:

i. Cantidad de pruebas bloqueadas por defectos.

ii. Tentativas falladas de pasar las pruebas de humo.

iii. Cambios en las especificaciones durante la prueba.

iv. Pruebas que cambiaron por requerimientos modificados.

v. Tiempo perdido en cosas del entorno (por ejemplo dificultades para obtener el equipo de prueba o configurar el sistema).

3.7.25 Resumen de Reunión (RR).

Aquí se establece un acta de las reuniones realizadas, donde se establecen participantes, fecha y lugar de realización, temas abordados, acuerdos, sugerencias, actividades y responsables de realizarlas según lo acordado en la reunión correspondiente, para lo cual todos los participantes deben estar de acuerdo.

3.7.26 Testware (TW)

Según (Black, 2002) el resultado de todas las actividades de pruebas realizadas al producto software a ser probado, son los siguientes:

i. Herramientas: Las herramientas necesarias para realizar las pruebas pueden comprender Sistemas Operativos, herramientas de automatización de las pruebas, manejadores de Bases de Datos, herramientas para la Gestión de las Pruebas y herramientas para la realización de reportes.
ii. Inventario: Reúne las funcionalidades a ser probadas.
iii. Casos de Prueba
iv. Biblioteca de casos de prueba: Para cada caso de prueba incluye: la configuración, las condiciones que deben cumplirse para poder ejecutar el caso de prueba y cómo volver al estado inicial.
v. Conjunto de Pruebas: Organizan los casos de prueba. Incluyen la configuración del conjunto, los casos de prueba que los componen y cómo volver al estado inicial. Un mismo caso de prueba puede ejecutar en varias suites.
vi. Resultados (Logs): Los resultados pueden ser creados automáticamente por las herramientas o manualmente por los testers.

vii. Reportes: Se realizan a partir de los casos de prueba y los resultados de ejecutarlos.

viii. Arquitectura de las pruebas: Documento que muestra la estructura del Testware y las herramientas que aplican en el ambiente de pruebas. Muestra también la estructura del sistema a probar.

3.8 ROLES.

La empresa Siredcom requiere que su equipo de pruebas posea sus roles y correspondientes responsabilidades, estableciéndose lo siguientes roles:

i. Líder del Proyecto de Prueba
ii. Diseñador de Pruebas
iii. Tester
iv. Cliente
v. Desarrollador.

El equipo de prueba está constituido por el Líder de Proyecto de Prueba, los Diseñadores de Pruebas y los Testers. Además existe un usuario o cliente que es a quien se le realizaran las pruebas del producto software a probar pudiéndose establecer los siguientes escenarios:

I. Usuario o Cliente a quien se le desarrolla el producto software.
II. Usuario o Cliente que va a comprar el producto de software

A continuación se detallan competencias y responsabilidades para cada rol.

3.8.1 Líder de Proyecto de Prueba.

Los requisitos y competencias para el Líder del Proyecto de Prueba son:

i. Capacidad de Liderazgo.

ii. Sólidos conocimientos en administración de proyectos.

iii. Excelente relación interpersonal.

iv. Fuerte experiencia de trabajo en equipo.

v. Experto en técnicas de prueba.

vi. Excelente comprensión de las fases del proceso de prueba.

vii. Sólidos conocimientos en programación, tecnologías de bases de datos, sistemas operativos.

viii. Fuerte experiencia en Ingeniería de Requerimientos.

En la Tabla 3.20 se describen las responsabilidades del Líder de Prueba, dentro del modelo de pruebas propuesto.

Tabla 3.20 – Responsabilidades del líder de pruebas.

Actividad	Responsabilidades
Ep1- Necesidad del cliente	Definir el alcance, los criterios de aceptación para el proyecto de prueba y los criterios de finalización del mismo del proyecto de prueba junto con el Cliente y Desarrolladores.
Ep2 - Revisión de requerimientos	Estudiar los principales requerimientos del producto junto con el Cliente y el Diseñador para realizar la propuesta de servicio.
Ep3 - Análisis de riesgo	Identificar junto con el Cliente y los Desarrolladores las funcionalidades más riesgosas y las que tengan mayor probabilidad de tener defectos.
PL1 - Acuerdo con el Cliente	Negociar con el Cliente y los Desarrolladores el alcance definitivo y la agenda de los ciclos de prueba
PL2 - Revisión de requerimientos	Estudiar las especificaciones para ver si es posible generar las pruebas a partir de ellas.

Continua ⟶

PL3 - Análisis de riesgo	Identificar junto con el Cliente y los Desarrolladores la prioridad de las funcionalidades incluidas en el alcance.
PL5 - Definición de los Ciclos de Prueba	Definir junto con el Cliente y los Desarrolladores la agenda de los ciclos de prueba y las funciones que serán probadas en cada ciclo.
PL7 - Planificación de las Pruebas	Planificar el proyecto, definiendo la estrategia de Prueba y las técnicas a usar en el Proyecto Planificar el armado del ambiente de pruebas junto con el Cliente y los especialistas técnicos necesarios.
PL8 - Definición del Proceso de Incidentes	Definir junto con el Cliente y Desarrolladores la forma en que se reportan los incidentes encontrados y quienes serán los responsables por parte del cliente de validar y corregir.
AC1 - Planificación del Ciclo de Pruebas	Planificar en detalle el ciclo de prueba y las actividades a realizar por el equipo de pruebas.
AC3 - Seguimiento y Control del Ciclo de Pruebas	Liderar técnicamente el proyecto. Supervisar el equipo de prueba. Administrar costos y procesos. Implementar el proceso de prueba Seguir el progreso del proyecto de prueba y reportar el porcentaje de definición y ejecución de casos de prueba, así como la cantidad de incidentes detectados por caso de prueba y totales Interactuar con el Cliente en todas las etapas técnicas de la ejecución del proyecto Asegurar la calidad de todos los productos del proyecto de prueba Validar el armado del ambiente de prueba y su preservación.
AC4 - Evaluación del Ciclo de Pruebas	Evaluar las pruebas del ciclo y definir cuando es necesario generar nuevos casos de prueba.
RP1- Determinar la Satisfacción del Cliente	Evaluar la satisfacción del Cliente con el proyecto de prueba y los resultados de las pruebas.
RP2 - Reporte de Finalización del Proyecto	Elaborar el informe final del proyecto de prueba.
RP4 - Mejoramiento del Proceso de Prueba	Sugerir mejoras en el proceso de prueba.
A1 - Estimación de las tareas	Estimar el esfuerzo que insumirá cada actividad y pedir al resto del equipo que estime sus tareas.
A2 - Medición de Esfuerzo	Reportar el esfuerzo que insume cada actividad y controlar que el resto del equipo reporte su esfuerzo

Fuente: Measurement Issues and Software Testing. Y Software Testing In The Real World : Improving The Process [19]

[19] Tomado de (Kit, 1995) y (Kaner, 2001)

3.8.2 Diseñador de Pruebas

Los requisitos y competencias para el Diseñador de Pruebas son:

i. Experto en técnicas de prueba.

ii. Sólidos conocimientos en diseño de casos de prueba.

iii. Sólidos conocimientos en Ingeniería de Requerimientos.

iv. Excelente comprensión de las fases del proceso de prueba.

v. Sólidos conocimientos en programación, tecnologías de bases de datos y sistemas operativos.

vi. Excelente experiencia en Pruebas de Software.

vii. Buen manejo de la expresión escrita.

viii. Excelente relación interpersonal.

ix. Experiencia de trabajo en equipo.

En la Tabla 3.21 se describen las responsabilidades del Diseñador de Pruebas.

Tabla 3.21 – Responsabilidades del líder de pruebas.

Actividad	Responsabilidad
Ep2 - Revisión de requerimientos	Estudiar los principales requerimientos del producto junto con el Cliente y el Diseñador para realizar la propuesta de servicio.
Ep3 - Análisis de riesgo	Identificar junto con el Cliente y los Desarrolladores las funcionalidades más riesgosas y las que tengan mayor probabilidad de tener defectos
PL2 - Revisión de requerimientos	Definir la estrategia para generar los datos de prueba junto con la contraparte técnica del Cliente. Verificar la calidad de los requerimientos para realizar las pruebas Detectar los problemas de especificación de requerimientos y elaborar propuestas para su mejora
PL3 - Análisis de riesgo	Definir junto con el Cliente las prioridades de las funcionalidades en cada ciclo de prueba
PL4 - Exploración del Producto	Estudiar el producto para conocerlo rápidamente.

Continua ⟶

PL6 - Definición del Testware	Definir los artefactos que serán generados durante el proyecto de prueba y cómo serán organizados
PL7 - Planificación de las Pruebas	Ayudar al Líder de Proyecto a planificar el proyecto de prueba
PL8 - Definición del Proceso de Incidentes	Definir junto con el Cliente y Desarrolladores la forma en que se reportan los incidentes encontrados y quienes serán los responsables por parte del cliente de validar y corregir
AC1 - Planificación del Ciclo de Pruebas.	Asistir al Líder de Proyecto en la planificación del ciclo.
AC2 - Gestión de la Configuración	Administrar los elementos definidos en el Testware durante todo el proyecto junto con el tester
AC3 - Seguimiento y Control del Ciclo de Pruebas	Coordinar el equipo de testers y asistirlos en los momentos necesarios Coordinar reuniones técnicas con la Contraparte técnica del cliente o el equipo de desarrollo. Generar reportes de avance
AC4 - Evaluación del Ciclo de Pruebas	Analizar los resultados de las pruebas realizadas Elaborar los reportes de prueba Seguir la ejecución de las pruebas
DP1 – Diseño de los Casos de Prueba	Definir criterios para realizar las pruebas Diseñar los casos de prueba Generar y mantener la matriz de trazabilidad entre requerimientos y pruebas
DP2 – Validación de los Casos de Prueba	Validar con el Cliente y/o los Desarrolladores que los casos de prueba sean de valor para el negocio
DP3 – Asignación de los Casos de Prueba	Definir junto con el equipo de testers, quién ejecutará las pruebas definidas
EP4 - Reporte de Incidentes.	Revisar los incidentes reportados por los testers
RP4 - Mejoramiento del Proceso de Prueba	Sugerir mejoras en el proceso de pruebas
A1 - Estimación de las tareas	Estimar el esfuerzo que insumirá cada actividad
A2 - Medición de Esfuerzo	Reportar el esfuerzo que insume cada actividad

Fuente: Measurement Issues and Software Testing. Y Software Testing In The Real World: Improving The Process [20]

3.8.3Tester.

Los requisitos y competencias para el Tester son:

i. Buen conocimiento en desarrollo de casos de prueba

ii. Buen conocimiento en probar software y en herramientas automáticas

iii. Sólidos conocimientos en base de datos y en programación.

iv. Buenos conocimientos en técnicas de prueba.

[20] Tomado de (Kit, 1995) y (Kaner, 2001)

v. Comprensión de las fases del proceso de prueba

En la Tabla 3.22 se describen las responsabilidades del Tester.

Tabla 3.22 – Responsabilidades del tester.

Actividad	Responsabilidades
PL4 - Exploración del Producto	Explorar el Producto para aprender sobre él
PL6 - Definición del Testware	Definir junto con el Diseñador los artefactos que serán generados durante el proyecto de prueba
E1 - Instalación de Herramientas	Instalar las herramientas definidas en el Testware
E2 – Configuración	Instalar y configuración la versión a probar del producto
AC1 - Planificación del Ciclo de Pruebas.	Planificar las pruebas en conjunto con los Diseñadores
AC2 - Gestión de la Configuración	Administrar los elementos definidos en el Testware durante todo el proyecto.
AC3 - Seguimiento y Control del Ciclo de Pruebas	Seguir la ejecución de las pruebas
AC4 - Evaluación del Ciclo de Pruebas	Analizar los resultados de las pruebas realizadas Elaborar los reportes de prueba
DP3 – Asignación de los Casos de Prueba	Definir junto con el equipo de testers y diseñadores, quién ejecutará las pruebas definidas
EP1 - Pruebas de Humo.	Ejecutar las pruebas de humo Registrar los resultados reales de la ejecución de las pruebas
EP2 - Ejecución de las Pruebas.	Ejecutar las pruebas Registrar los resultados reales de la ejecución de las pruebas
EP3 - Testing Exploratorio.	Realizar testing exploratorio Registrar los resultados del testing exploratorio
EP4 - Reporte de Incidentes.	Detectar y registrar los incidentes en el sistema de seguimiento de incidentes
EP5 - Validación de los Incidentes.	Asegurarse de que el Cliente valide los incidentes reportados
EP6 – Verificación de las Correcciones.	Validar que el incidente fue solucionado en la nueva versión del producto y que puede ser cerrado.
RP3 - Archivar el Proyecto	Dejar los elementos que componen el testware organizados de forma de poder ser accedidos fácilmente en otros proyectos
RP4 - Mejoramiento del Proceso de Prueba	Sugerir mejoras en el proceso de prueba.
A1 - Estimación de las tareas	Estimar el esfuerzo que insumirá cada actividad
A2 - Medición de Esfuerzo	Reportar el esfuerzo que insume cada actividad

Fuente: Measurement Issues and Software Testing. Y Software Testing In The Real World : Improving The Process [21]

[21] Tomado de (Kit, 1995) y (Kaner, 2001)

3.8.4 Cliente.

El cliente es quien contrata el proyecto de prueba independiente, por lo que no tiene requisitos y competencias definidas. En la Tabla 3.23 se describen las responsabilidades del Cliente.

Tabla 3.23 – Responsabilidades del cliente.

Actividad	Responsabilidades
Ep1- Necesidad del cliente	Definir el alcance, los criterios de aceptación para el proyecto de prueba y los criterios de finalización del mismo del proyecto de prueba junto con el Líder de Proyecto y Desarrolladores
Ep2 - Revisión de requerimientos	Brindar acceso a las fuentes de requerimientos existentes (documentos de requerimientos, manuales, documentación técnica, usuarios del sistema, etc.)
Ep3 - Análisis de riesgo	Identificar junto con el Líder de Proyecto las funcionalidades más riesgosas y las que tengan mayor probabilidad de tener defectos
PL1 - Acuerdo con el Cliente	Negociar con el Líder de Proyecto y los Desarrolladores el alcance definitivo y la agenda de los ciclos de prueba Definir los interlocutores del Cliente con el equipo de prueba
PL2 - Revisión de requerimientos	Brindar acceso a las fuentes de requerimientos existentes (documentos de requerimientos, manuales, documentación técnica, usuarios del sistema, etc.)
PL3 - Análisis de riesgo	Identificar junto con el Líder de Proyecto y los Desarrolladores la prioridad de cada funcionalidad que compone el alcance
PL5 - Definición de los Ciclos de Prueba	Definir junto con el Líder de Proyecto y los Desarrolladores la agenda de los ciclos de prueba y las funciones que serán probadas en cada ciclo.
PL7 - Planificación de las Pruebas	Planificar el proyecto, definiendo la estrategia de Prueba y las técnicas a usar en el Proyecto Planificar el armado del ambiente de pruebas junto con el Líder de Proyecto y los Desarrolladores necesarios.
PL8 - Definición del Proceso de Incidentes	Acordar con el Líder de Proyecto el procedimiento a seguir en caso de aparición de defectos que impidan continuar con las pruebas Identificar quiénes serán los responsables por parte del cliente de validar y corregir los incidentes encontrados en las pruebas
DP2 – Validación de los Casos de Prueba	Validar que los casos de prueba sean de valor para el negocio.
EP5 - Validación de los Incidentes.	Validar los incidentes reportados en el sistema de seguimiento.
RP1- Determinar la Satisfacción del Cliente	Realizar la encuesta de satisfacción
RP2 - Reporte de Finalización del Proyecto	Validar el informe final del proyecto de prueba

Fuente: Measurement Issues and Software Testing. Y Software Testing In The Real World: Improving The Process 22

[22] Tomado de (Kit, 1995) y (Kaner, 2001)

3.8.5 Desarrollador o Contraparte técnica.

Al igual que el cliente, el desarrollador es un rol externo al equipo de pruebas, por lo que no tiene requisitos y competencias. En la Tabla 3.24 se describen las responsabilidades del Desarrollador.

Tabla 3.24 – Responsabilidades del desarrollador.

Actividad	Responsabilidades
Ep1- Necesidad del cliente	Definir el alcance, los criterios de aceptación para el proyecto de prueba y los criterios de finalización del mismo del proyecto de prueba junto con el Líder de Proyecto y el Cliente
Ep3 - Análisis de riesgo	Identificar junto con el Líder de Proyecto y el Cliente las funcionalidades más riesgosas y las que tengan mayor probabilidad de tener defectos
PL1 - Acuerdo con el Cliente	Negociar con el Líder de Proyecto y el Cliente el alcance definitivo y la agenda de los ciclos de prueba Definir los interlocutores con el equipo de prueba.
PL3 - Análisis de riesgo	Identificar las áreas del producto que tienen más probabilidad de contener defectos
PL5 - Definición de los Ciclos de Prueba	Definir los ciclos de prueba acordes con las fechas del plan de desarrollo
PL8 - Definición del Proceso de Incidentes	Identificar quiénes serán los responsables por parte del cliente de validar y corregir los incidentes encontrados en las pruebas
DP2 – Validación de los Casos de Prueba	Validar los casos de prueba y sus datos.
EP6 – Verificación de las Correcciones.	Realizar el seguimiento y validación de los incidentes encontrados Responsabilizarse por la corrección de incidentes

Fuente: Measurement Issues and Software Testing. Y Software Testing In The Real World : Improving The Process [23]

[23] Tomado de (Kit, 1995) y (Kaner, 2001)

CAPÍTULO 4

APLICACIÓN DEL MODELO.

4. INTRODUCCIÓN.

El modelo propuesto, para probar un producto software y obtener el nivel 2 de TMMI fue aplicado en uno de los productos que se encuentra en producción en la empresa. Para el diagnóstico se aplicó el estándar de calidad ISO-9001-2000 tomando en consideración: la responsabilidad de la dirección, el sistema de gestión de la calidad, la gestión de los recursos, la realización del producto, medición análisis y mejora. Para continuar con el diagnóstico del nivel de TMMI en sus respectivos procesos: Políticas y estrategias de prueba, Planificación de las Pruebas, Técnicas y Métodos de Pruebas, Entorno del Testing.

A continuación se procedió a aplicar el modelo formal propuesto, el mismo que es independiente del proceso de desarrollo de software y se validó las especificaciones funcionales del mismo, se evaluó los riesgos a probar y se realizaron los ciclos de prueba.

Se definió en el modelo de pruebas la Especificación, lo que permitió analizar las funcionalidades más importantes del software a probar, determinándose el alcance de las pruebas y así un cronograma de los ciclos correspondientes.

Luego que el cliente estuvo de acuerdo, se procedió con la planificación, luego de que se volvió a realizar la especificación donde se analizaron los inconvenientes encontrados por parte del cliente, el objetivo de planificar y diseñar las pruebas, ya aceptadas por el usuario lo que permitió continuar con la ejecución, es decir determinar y realizar las pruebas de una versión del software, relacionándolas con el ciclo correspondiente, para pasar a la evaluación y determinar el informe final sobre las pruebas realizadas, la satisfacción del usuario con respecto al software probado y al proceso utilizado fue excelente lo que permitirá realizar una mejora continua, esta información será guardada y será utilizada en pruebas futuras.

4.1 Producto a probar de la empresa Siredcom.

El producto a ser probado es el sistema modular de administración educativa el cual se encuentra en producción varios años, y que se le da mantenimiento según las necesidades de la institución donde se encuentra instalado. Al realizar las pruebas se disponía de manual de usuario el cual no está actualizado, el cliente es personal que labora en la institución pero la mayoría ya no está ya que ha existido un recorte de personal. La empresa Siredcom está interesada en establecer un proceso que le permita llegar a cumplir con el nivel 2 de TMMI.

Ya que la confidencialidad es un derecho, no se establece el nombre de la empresa, ni la aplicación, ni el cliente. Ya que la información para la empresa es muy sensible e importante para ser divulgada, tampoco se permite exponer los casos de pruebas y sus funcionalidades correspondientes, dejándose solamente presentar los datos generados al aplicar el modelo.

4.2 Aplicación del Modelo Propuesto.

Se realizan las pruebas utilizando el modelo propuesto:

- Especificación de la prueba.
- Planificación.
- Ejecución de la Prueba (Ciclo de Prueba).
- Resultado de la Prueba.

4.2.1 Especificación de la prueba.

Esta fase se establece la factibilidad de realización de las pruebas. A continuación se describe lo realizado.

La actividad Ep1 – Necesidad del cliente. Esta se realizó en cuatro reuniones con el cliente a fin de determinar las funcionalidades más importantes que el cliente busca en el producto software a ser probado, llegándose a un acuerdo después de la validación del cliente.

En la actividad Ep2 – Revisión de Requerimientos. A partir de los manuales de usuario y de las versiones que se poseen del producto, detectándose que la documentación no estaba completa, teniendo que mejorar los requerimientos existentes. Posterior a esto se determinó un listado de las funcionalidades más importantes, las cuales aparecen en el LP- Listado de Prueba, donde se encontraron 40 funcionalidades, las cuales fueron validadas por el cliente.

Las prioridades que se dieron a las 40 funcionalidades que están en LP-Listado de Pruebas, se las realizo en la actividad Ep3 – Análisis de Riesgo, determinándose con el cliente el alcance de las pruebas a 20 funcionalidades de las 40 especificadas al inicio como se observa en la tabla 4.1. En el primer cronograma estableció que las pruebas durarían dos meses, realizándose una versión en cada mes según la planificación. Realizándose las pruebas en la empresa.

Tabla 4.1 – Alcance de la Especificación de la Prueba.

Alcance			
Prioridad	**Complejidad**	**Criticidad**	**Número de Funcionalidades**
Alta	**Alta**	**Alta**	**5**
Alta	**Alta**	**Alta**	**5**
Alta	**Media**	**Media**	**5**
Media	**Media**	**Media**	**5**
		Total	**20**

Fuente: Elaboración Propia.

En la actividad Ep3 – Análisis del Riesgo se conversó con el cliente y se determinó lo importante del análisis del riesgo del producto a probar y su implicación con el alcance de las pruebas, razón por la cual la versión que se tenía del LP-Listado de Pruebas cambio, siendo necesario establecer prioridad, criticidad e importancia.

Ya poseyendo esta información se realizó PL1- Acuerdo con el Cliente donde se establece el cronograma de pruebas y el alcance. Después que el cliente valido la propuesta se siguen con la siguiente fase: La planificación.

Esfuerzo de la fase: Especificación de la Prueba. En la tabla 4.2 se muestra el esfuerzo que se invirtió en la especificación de la prueba, por parte de los integrantes del equipo de pruebas.

Tabla 4.2 – Esfuerzo de la Especificación de la Prueba.

Especificación de la Prueba			
Horas	**Líder**	**Diseñador**	**Total**
Ep1- Necesidad del cliente	**7**	**3,5**	**10,5**
Ep2 - Revisión de requerimientos	**3,5**	**12,5**	**16**
Ep3 - Análisis de riesgo	**4**	**4**	**8**
Total	**14,5**	**20**	**34,5**

Fuente: Elaboración Propia.

4.2.2 Planificación.

Se realizaron tres reuniones con el cliente en la fase de planificación, realizándose:

- PL1 – Acuerdo con el Cliente
- PL3 – Análisis de riesgo
- PL5 – Definición de los Ciclos de Prueba
- PL8 – Definición del Proceso de Incidentes.

Para poder realizar la planificación se necesitó que el cliente mantuviese una comunicación continua con el equipo de pruebas, llegándose a acordar que se tendrían reuniones dos veces a la semana, además de utilizar el correo electrónico para acortar tiempos. Se necesitó que el cliente asignara a quienes tomarían

decisiones trascendentales, decisiones importantes para la realización de las pruebas (alcance, validación de casos de prueba y resultados obtenidos). El equipo de prueba instalaran las versiones correspondientes a ser probadas, el cliente resolverán los inconvenientes causados al momento de las configuraciones. Los reportes referentes a la ejecución de los casos de prueba y los incidentes correspondientes se entregarán al finalizar la semana.

Cabe destacar que los datos de prueba son reales y consistentes por estar trabajando en el servidor de producción de la institución, siendo esto responsabilidad del cliente.

En la actividad PL2 - Revisión de requerimientos se revisó la documentación existente sobre el producto a probar, existiendo especificaciones incompletas, teniéndose que estos ser completados no en su totalidad razón por la cual algunos serán probados en el primer ciclo y otros en el siguiente ciclo.

La actividad PL4 - Exploración del Producto, sirvió de apoyo ya que se disponía solamente del ejecutable del producto a probar, permitiéndose entenderlo de mejor manera, mejorándose sus requerimientos, lo cual reduce tiempos y costos.

En la actividad PL5 - Definición de los Ciclos de Prueba, se determina el inicio y fin de cada ciclo que se utilizara para probar con sus respectivas funcionalidades a probar.

En la Tabla 4.3 se especifica el número de funcionalidades a ser probadas en cada ciclo, tomando en consideración prioridad, complejidad y criticidad.

Tabla 4.3- Alcance definido para la fase de Planificación de los Ciclos de Pruebas.

Planificación de los Ciclos de Pruebas					
Prioridad	**Complejidad**	**Criticidad**	**Numero Funcionalidades**	**Ciclo 1**	**Ciclo 2**
Alta	**Alta**	**Alta**	**5**	**0**	**5**
Alta	**Alta**	**Alta**	**5**	**4**	**1**
Alta	**Media**	**Media**	**5**	**4**	**1**
Media	**Media**	**Media**	**5**	**4**	**1**
		Total	**20**	**12**	**7**

Fuente: Elaboración Propia.

En la actividad PL8 – Definición del Proceso de Incidentes, se lleva documento en exel el cual está compartido para todo el equipo de pruebas, llevándose el registro correspondiente de los incidentes suscitados, lo que permite realizar reportes y seguimiento de los mismos. En la Ilustración 1 se definen los incidentes que se pueden presentar.

En la Ilustración 48 se muestra el proceso de incidentes definido para el proyecto de prueba.

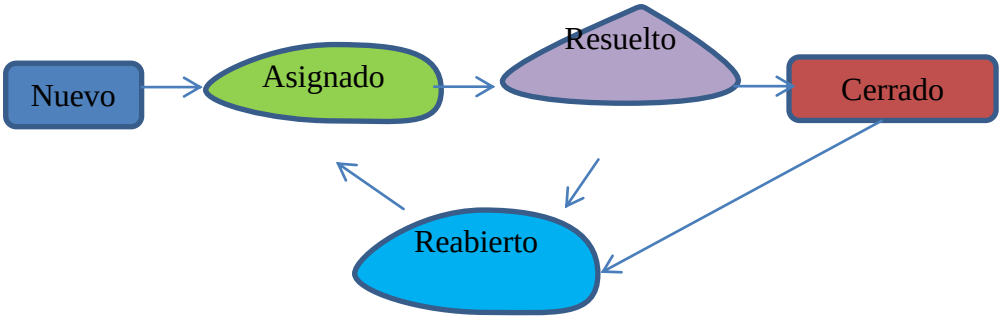


Ilustración 4.1– Estados de un Incidente.

Fuente: Elaboración Propia.

Se procede a determinar que es cada estado:

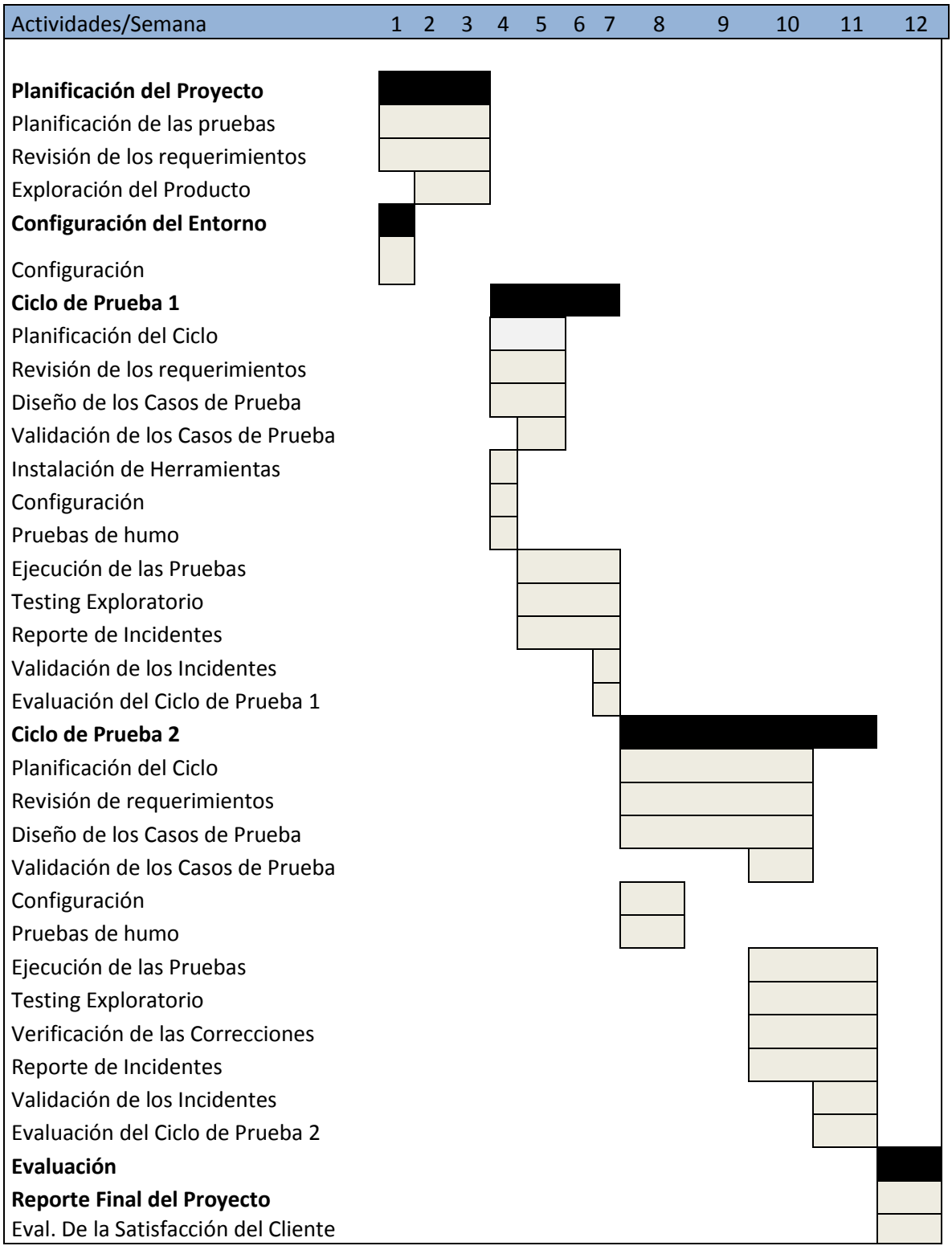
- Nuevo: Incidente ingresado debe ser atendido.
- Asignado: Incidente procesándose está a cargo de un desarrollador para su arreglo, aún no está resuelto.
- Reabierto: El incidente se resolvió, pero su solución fue incorrecta.
- Resuelto: El incidente fue arreglado por el equipo de desarrollo, estableciéndose una versión del producto. Al ejecutarse las pruebas de regresión y de no existir inconvenientes su estado es cerrado caso contrario es reabierto.
- Cerrado: Incidente arreglado y finalizado.

El cliente debía pasar los incidentes al estado Asignado o Resuelto. Los estados Nuevo, Reabierto y Cerrado fueron usados por el equipo de prueba. A fin de determinar grupos de incidentes, estos se los clasifico en:

- Implementación incorrecta
- Mejora
- Interfaz de usuario
- Manejo de errores
- Validación faltante o incorrecta
- Funcionalidad no disponible
- Instalación o ambiente
- Salida anormal
- Seguridad
- Usabilidad
- Observación
- No clasificado

En la actividad P7 – Planificación de las Pruebas se estableció un plan de pruebas tomando como base la actividad PL5 - Definición de los Ciclos de Prueba, PL6 - Definición del Testware y PL8 - Definición del Proceso de Incidentes, estableciéndose la planificación de las pruebas como se muestra en la tabla 4.4 , la etapa de planificación durara tres semanas, el ciclo de pruebas 1, durara cuatro semanas, el ciclo de pruebas 2, cuatro semanas y la Evaluación del proyecto de prueba se realizaría en la última semana.

Tabla 4.4 - Planificación de las Pruebas.



Fuente: Elaboración Propia.

Cabe destacar que el líder de pruebas es quien hace el plan de pruebas el cual es validado por el cliente, siendo importante para tener un medio de relacionarnos con el cliente y las pruebas a realizarse.

a. Esfuerzo de la fase: Planificación de la Prueba.

En la Tabla 4.5 se establece cual fue el esfuerzo que necesito el equipo de pruebas en la fase de Planificación de la prueba. La actividad PL1 - Acuerdo con el Cliente se toma en consideración el esfuerzo de PL5 - Definición de los Ciclos de Prueba y PL8 - Definición del Proceso de Incidentes por haberse trabajado con el cliente. En la actividad P7-Planificación de las pruebas se consideran las actividades P6 – Definición del Testware. La actividad E2 –Configuración, esta actividad se la utilizo en la etapa de Planificación.

Tabla 4.5 – Esfuerzo de la Planificación de las Prueba.

Etapa: Planificacion					
	Horas por Actividad	Lider	Disenador	Tester	Total
P	Acuerdo con el Cliente	5	5	0	10
P	Revision de Requerimientos	11	16,5	0	27,5
P	Planificacion de las Pruebas	6	3	0	9
P	Exploracion del Producto	0	6	0	6
C	Configuracion	0	0	9	9
Total		**22**	**30,5**	**9**	**61,5**

Fuente: Elaboración Propia.

b. Ciclos de Prueba.

Para probar el producto, se necesitaron dos ciclos a continuación se establece lo que se realizó en cada ciclo.

b.1 Ciclo de Prueba 1

Se detalla las actividades realizadas en él ciclo 1 según el modelo propuesto. El ciclo 1 se inició al realizar la actividad PL3- Análisis del riesgo, determinaron prioridades y alcances de las pruebas para este ciclo. Se diseñaron casos de pruebas para probar funcionalidades, los cuales determinaron el alcance del ciclo, lo cual se conversó con el cliente, ya que en la actividad AC1 - Planificación del Ciclo de Pruebas, se estableció que era necesario aumentar una semana más de lo planeado ya que se aumentarían funcionalidades, de 12 a 20 estando planificadas al inicio 13, aumentándose 7. Acordándose acortar el ciclo de pruebas 2.

En la Tabla 4.6 se establece las funcionalidades planificadas y las reales que se realizaron en el ciclo 1.

Tabla 4.6 – Alcance del Ciclo 1 (Planificado Versus Real).

Alcance para Ciclo 1				
Prioridad	Complejidad	Criticidad	Número Plan. Ciclo 1	Número Real Ciclo 1
Alta	Alta	Alta	0	1
Alta	Alta	Alta	4	4
Alta	Media	Media	4	4
Media	Media	Media	4	4
Media	Baja	Media	0	4
Baja	Baja	Media	0	3
		Total	12	20

Fuente: Elaboración Propia.

La actividad AC3 - Seguimiento y Control del Ciclo de Pruebas se hizo mediante reuniones cada dos días del equipo de pruebas. Realizando además las siguientes actividades: A2 - Medición de Esfuerzo donde se reporta el esfuerzo del ciclo y AC2 - Gestión de la Configuración donde se administró los elementos creados en las pruebas según el Testware.

c. Configuración del Entorno

Según lo planeado para el primer ciclo se tenía una nueva versión del producto, pero al no generarse a tiempo se usó la versión instalada utilizándose la exploración en la etapa de Planificación siendo este el motivo de no realizarse la actividad E2 – Configuración en este ciclo. En la actividad E1 - Instalación de Herramientas se determinó como herramienta alternativa inicial Excel para gestionar las pruebas (Casos de Prueba) y reporte de seguimiento de incidentes.

d. Diseño de las pruebas.

Aquí se diseñan las pruebas mediante la actividad DP1 – Diseño de los Casos de Prueba y al mismo tiempo la actividad PL2 – Revisión de requerimientos. Los casos de prueba se generaron utilizando las técnicas de caja negra de partición de equivalencia y valor límite. Se hizo una matriz que permita establecer la relación que existe entre funcionalidades y los casos de prueba generados, para lo cual el cliente valida los casos de prueba. En la tabla 4.7 mira para cada una de las funcionalidades a probarse en el ciclo 1 sus correspondientes casos de prueba y el número de incidentes generados al ser ejecutados.

Tabla 4.7 – Casos de Prueba e Incidentes Encontrados por Funcionalidad.

Casos Definidos por Funcionalidad-Ciclo de Prueba 1					
Prioridad	Complejidad	Criticidad	Funcionalidad	Numero Casos	Numeros Incidentes
Alta	Baja	Media	Funcionalidad 1	4	0
Alta	Media	Alta	Funcionalidad 2	3	5
Alta	Media	Alta	Funcionalidad 3	7	4
Alta	Baja	Alta	Funcionalidad 4	5	4
Alta	Media	Alta	Funcionalidad 5	6	5
Alta	Media	Alta	Funcionalidad 6	2	4
Alta	Media	Alta	Funcionalidad 7	13	8
Alta	Media	Alta	Funcionalidad 8	20	11
Alta	Baja	Alta	Funcionalidad 9	10	5
Alta	Media	Alta	Funcionalidad 10	10	5
Baja	Baja	Media	Funcionalidad 11	6	2
Media	Media	Media	Funcionalidad 12	2	0
Baja	Baja	Media	Funcionalidad 13	7	0
Media	Baja	Media	Funcionalidad 14	7	0
Baja	Baja	Media	Funcionalidad 15	9	
Baja	Baja	Media	Funcionalidad 16	8	0
Baja	Baja	Media	Funcionalidad 17	2	0
Baja	Baja	Media	Funcionalidad 18	4	1
Baja	Baja	Media	Funcionalidad 19	5	3
Baja	Baja	Media	Funcionalidad 20	4	2
			Total	134	59

Fuente: Elaboración Propia.

4.2.3 Ejecución de la Prueba.

En el ciclo 1 la versión que se está probando es la que se tenía instalada en la fase de Planificación, razón por lo cual no se requirió ejecutar la actividad EP1 - Pruebas de Humo ya que las funcionalidades básicas no presentaban ningún inconveniente. Se realizaron las actividades EP2 - Ejecución de las Pruebas y EP3 - Testing Exploratorio, registrando los incidentes encontrados en el programa de excel y posteriormente validados por el cliente. En la Tabla 4.8 se detallan los incidentes según su tipo.

Tabla 4.8 – Incidentes por Tipo (Ciclo 1).

Incidentes por Tipo	Ciclo 1
Funcional- Implementación Incorrecta	20
Interfaz de Usuario	10
Mensajes erróneos o falta de mensaje	30
Validación faltante o incorrecta	2
Total	62

Fuente: Elaboración Propia.

En la Tabla 4.9 se especifican los incidentes según su criticidad.

Tabla 4.9 – Incidentes según Criticidad (Ciclo 1).

Incidentes según criticidad	Ciclo 1
Alta	6
Media	26
Baja	30
Total	62

Fuente: Elaboración Propia.

Al finalizar la sub fase EP2 - Ejecución de las Pruebas, se realizó AC4 - Evaluación del Ciclo de Pruebas, donde se evalúan las actividades del ciclo, en la Tabla 4.10 se muestran las diferencias entre el ciclo planificado y lo real, siendo estas:

- La actividad AC1 - Planificación del Ciclo de Pruebas se ejecutó en una semana y no en dos, como estaba planificado.

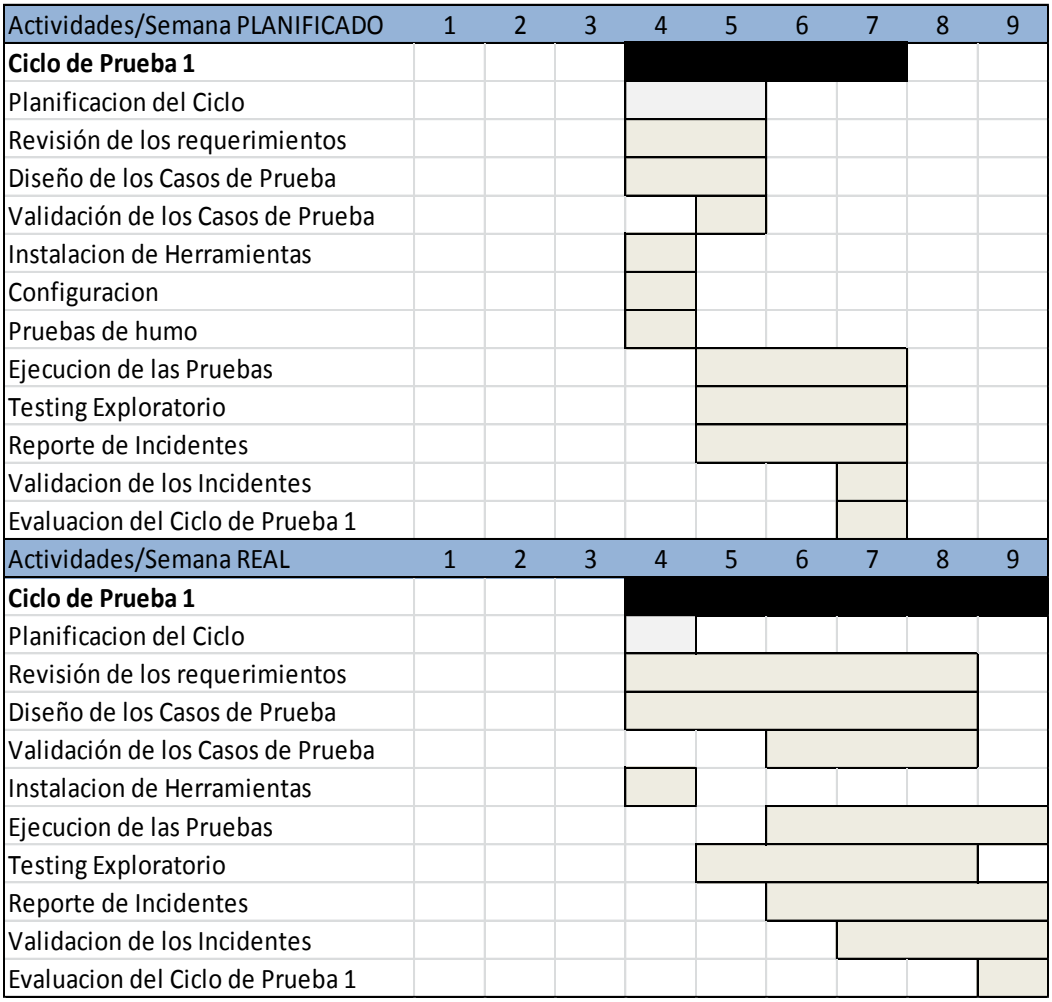
- Las actividades E2 – Configuración y EP1 - Pruebas de Humo no se necesitaron hacer ya que se probó una versión del producto que ya estaba instalada en la fase de planificación

- Las actividades PL2 - Revisión de requerimientos, DP1 – Diseño de los Casos de Prueba y DP2 – Validación de los Casos de Prueba se realizaron dos semanas después de lo planeado en el plan de pruebas. La actividad AC1 - Planificación del Ciclo de Pruebas, duro una semana más.

- Las actividades EP2 - Ejecución de las Pruebas., EP3 - Testing Exploratorio yEP4 - Reporte de Incidentes, se realizaron en una semana más de lo planificado.

Se hizo una reunión con el cliente para explicarle sobre los resultados obtenidos en las pruebas del ciclo número 1 a fin de proceder a cerrar el ciclo y continuar con el siguiente. En la Tabla 4.11 se muestra el resultado de la ejecución del Ciclo de Prueba 1.

Tabla 4.10 – Cronograma Planificado – Cronograma Real (Ciclo 1).



Fuente: Elaboración Propia.

Tabla 4.11 – Resumen del Ciclo No 1.

Resumen Ciclo de Prueba No 1	
Cantidad de Funcionalidad	20
Cantidad de casos de pruebas	134
Casos por Funcionalidad	5,2
Cantidad de Incidentes	59
Incidente por funcionalidad	2,95
Incidentes por caso ejecutado	0,44

Fuente: Elaboración Propia.

En la Tabla 4. 12 se especifica el esfuerzo que tuvo que hacer el equipo de Pruebas.

Tabla 4.12 – Esfuerzo del Ciclo de Prueba 1.

Actividades Realizadas.	Líder	Diseñador	Tester	Total de Horas
Planificación del Ciclo de Pruebas.	4	4		8
Análisis de riesgo	4	4		8
Gestión de la Configuración	0	5		5
Seguimiento y Control del Ciclo de Pruebas	5	5		10
Evaluación del Ciclo de Pruebas	20	6		26
Instalación de Herramientas			2,5	2,5
Revisión de requerimientos		12,5		12,5
Diseño de los Casos de Prueba		60,5		60,5
Validación de los Casos de Prueba	6	6		12
Ejecución de las Pruebas.			49,5	49,5
Testing Exploratorio.			16	16
Reporte de Incidentes			16	16
Validación de los Incidentes.		7	7	14
TOTAL	39	110	91	240

Fuente: Elaboración Propia.

a. Ciclo de Prueba 2

Se detalla las actividades realizadas en él ciclo 2 según el modelo propuesto.

Cuando se finalizó el Ciclo de Prueba 2, debido a que la empresa Siredcom se atrasó en la planificación es decir en el desarrollo que estaba atrasado, no se poseía la última versión del producto software que se estaba probando, estableciéndose una estimación de que en una semana se poseería la nueva versión en ocho días, por lo que el ciclo 2 comenzara en una semana más. Lamentablemente el desarrollo no terminó en una semana y se decidió realizar el ciclo de pruebas 2, haciéndose hasta poseer la versión, el diseño de los casos de pruebas del ciclo. La nueva versión estuvo en dos semanas, ejecutándose las pruebas de inmediato. Ya que en ciclo de prueba 1 se establecieron nuevas funcionalidades, por lo que se recortó el alcance del ciclo de pruebas 2.

En la tabla 4.13 se puede observar lo planificado para el ciclo 2 y el alcance real, según las funcionalidades a ser probadas. Al ser la nueva versión una versión mejorada de la anterior, el usuario quería probar funcionalidades de la anterior versión en la nueva, para estar seguro de que no se hayan visto afectadas.

Tabla 4.13 – Ciclo 2 Planificado Versus Real.

Funcionalidades Ciclo 2				
Prioridad	**Complejidad**	**Criticidad**	**Ciclo 2 Planificado**	**Ciclo 2 Real**
Alta	Alta	Alta	0	1
Alta	Alta	Alta	6	3
Media	Media	Alta	0	1
Media	Baja	Alta	6	3
Baja	Baja	Media	0	0
		Total	12	8

Fuente: Elaboración Propia.

No se planifico el realizar pruebas de regresión, ya que en ese momento no se sabía que incidentes se encontrarían ni cuales fueron corregidos en la nueva versión.

Se decidió que se volverían a realizar cuarenta casos de pruebas de regresión. En la Tabla 4.14 se observan los incidentes que ocurrieron al realizar las pruebas de regresión.

b. Configuración del Entorno.

Se realizó la instalación de la nueva versión en el servidor de la institución donde se encuentra instalada la aplicación, se tomó en consideración el documento donde el cliente determino los cambios que se efectuaron y las funcionalidades nuevas que están hechas en la nueva versión.

c. Diseño de Pruebas.

Aquellos requerimientos nuevos a ser probados en ciclo 2 fueron analizados, diseñándose las pruebas con anterioridad a disponer de la última versión del producto en ejecución. Se utilizó técnicas de caja negra, para realizar los casos de prueba. Se hizo una matriz que permita relacionar las funcionalidades versus los casos de prueba ejecutados. Los casos de prueba fueron aprobados por el usuario en una reunión previa

En la tabla 4.14 se determinan las funcionalidades que se establecieron en el ciclo de prueba 2, el número de casos de prueba realizados y el número de incidentes que se generaron.

Tabla 4.14 – Caso de Prueba e Incidentes según funcionalidad.

Casos de Prueba según funcionalidad Ciclo 2					
Prioridad	**Complejidad**	**Criticidad**	**Funcionalidad**	**Numer de Casos**	**Incidentes**
Alta	Alta	Media	Funcionalidad 21	4	0
Alta	Alta	Media	Funcionalidad 22	15	10
Alta	Media	Media	Funcionalidad 23	5	1
Alta	Media	Baja	Funcionalidad 24	6	3
Alta	Baja	Media	Funcionalidad 25	7	5
Alta	Alta	Alta	Funcionalidad 26	7	0
Alta	Media	Media	Funcionalidad 27	3	0
Alta	Alta	Alta	Funcionalidad 28	4	2
			Total	51	21

Elaborado por: Milton Escobar.

d. Ejecución de las Pruebas.

Se utilizó la actividad EP1 - Pruebas de Humo. El producto software probado poseía funcionalidades básicas, realizándose los casos de prueba en las actividades EP2 - Ejecución de las Pruebas y EP3 - Testing Exploratorio. También al inicio del

ciclo se realizaron pruebas de regresión. En la tabla 4.15 se ve los resultados obtenidos. En la tabla 4.16 muestra los incidentes que se generaron según su tipo. En la tabla 4.17 muestra los incidentes que se generaron según la importancia de impacto.

Tabla 4.15– Caso de Prueba de Regresión y sus Incidentes.

Casos de Prueba de Regresión Ciclo 2					
Complejidad	**Criticidad**	**Prioridad**	**Funcionalidad**	**Numer de Casos**	**Incidentes**
Alta	Media	Alta	Funcionalidad 3	2	0
Alta	Media	Alta	Funcionalidad 7	4	1
Alta	Media	Alta	Funcionalidad 9	4	2
Alta	Media	Alta	Funcionalidad 11	5	3
Alta	Media	Media	Funcionalidad 14	4	4
Alta	Baja	Media	Funcionalidad 17	12	5
Media	Baja	Media	Funcionalidad 18	4	0
Media	Baja	Media	Funcionalidad 19	8	5
			Total	43	20

Fuente: Elaboración Propia.

Tabla 4.16– Tipos de Incidentes en el Ciclo 2.

Incidentes Ciclo de Prueba 2		
Tipo de Incidente	**Ciclo 2**	**Regresión**
Implementación Incorrecta	6	3
Interfaz	6	10
Mensajes Incorrectos	2	0
Mensajes Faltantes	2	0
Inexistencia de Validación	2	0
Incorrecta Validación	2	0
Total	20	13

Fuente: Elaboración Propia.

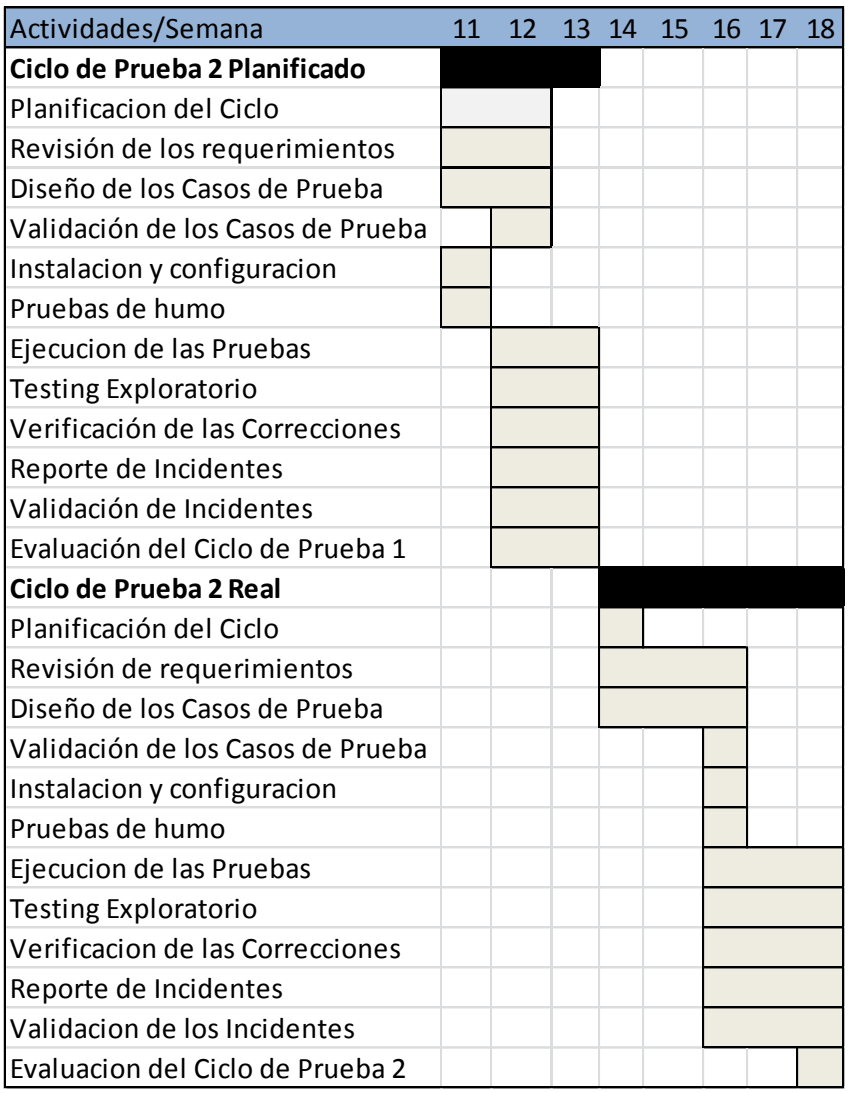
Tabla 4.17 – Criticidad en Incidentes en el Ciclo 2.

Incidentes Ciclo de Prueba 2		
Criticidad en Incidente	**Ciclo 2**	**Regresión**
Alta	4	3
Media	5	0
Baja	20	12
Total	**29**	**15**

Fuente: Elaboración Propia.

Luego de realizar las pruebas es necesario hacer una evaluación donde se utiliza la actividad AC4 - Evaluación del Ciclo de Pruebas. En la tabla 4.18 se establece las diferencias entre lo planificado del ciclo de prueba 2 y lo real. En la tabla 4.19 se establece el resultado de las pruebas en el Ciclo2, se tomó en consideración las pruebas de regresión realizadas.

Tabla 4.18 – Cronograma Planificado Versus Real para el Ciclo 2.



Fuente: Elaboración Propia.

Tabla 4.19 – Resumen de las Pruebas en el Ciclo2.

Resumen Ciclo de Prueba No 2	
Cantidad de Funcionalidad	16
Cantidad de casos de pruebas	94
Casos por Funcionalidad	5,8
Cantidad de Incidentes	41
Incidente por funcionalidad	2,56
Incidentes por caso ejecutado	0,44

Fuente: Elaboración Propia.

En la Tabla 4.20 se determina el esfuerzo que necesito el equipo de pruebas en el ciclo 2.

Tabla 4.20 – Esfuerzo en el Ciclo2.

Actividades Realizadas.	Líder	Diseñador	Tester	Total de Horas
Planificación del Ciclo de Pruebas.	3	3		6
Análisis de riesgo	3	3		6
Seguimiento y Control del Ciclo de Pruebas	5	5		10
Evaluación del Ciclo de Pruebas	22	12		34
Instalación de Herramientas			6	6
Revisión de requerimientos	9,5	17		26,5
Diseño de los Casos de Prueba	10	28,5		38,5
Validación de los Casos de Prueba	3	3		6
Pruebas de Humo			4	
Ejecución de las Pruebas.			30	30
Testing Exploratorio.			8	8
Reporte de Incidentes			10	10
Validación de los Incidentes.			15	15
TOTAL	55,5	71,5	73	200

Fuente: Elaboración Propia.

4.2.4 Resultado y Evaluación del Proyecto.

La actividad Reporte de Incidentes, se inició al mismo momento en que se estaba realizando el Ciclo de Pruebas 2. En la tabla 4.21 se ve una comparación entre la planificación del proyecto y la realidad de la realización de los Ciclos de Prueba. El cliente estuvo de acuerdo con lo realizado en el ciclo de pruebas 2 y con lo realizado en el proyecto de pruebas. Se evaluó en una reunión la satisfacción del Cliente, en la tabla 4.22 se observa una parte de la encuesta realizada. Se estableció como parámetros: 1 Mal y 5 Excelente.

Tabla 4.21 – Planificación de las Pruebas Versus la Planificación Real de las Pruebas.

FASE -SEMANA	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Planificación de las Pruebas																		
Estudio Preliminar																		
Planificacion del Proyecto																		
Ciclo Prueba 1																		
Ciclo Prueba 2																		
Evaluación																		
Planificación Real de las Pruebas																		
Estudio Preliminar																		
Planificacion del Proyecto																		
Ciclo Prueba 1																		
Ciclo Prueba 2																		
Evaluación																		

Fuente: Elaboración Propia.

Tabla 4.22– Encuesta de Satisfacción del Cliente.

Encuesta de Satisfacción del Cliente					
Cree usted que fueron suficientes los ciclos de pruebas realizados?	1	2	3	4	5
Se sintió satisfecho con las reuniones realizadas con el equipo de pruebas ?	1	2	3	4	5
Se sintió satisfecho con su participacion en las pruebas realizadas?	1	2	3	4	5
Se siente satisfecho con el producto software obtenido ?	1	2	3	4	5
Cree que la priorizacion de funcionalidades del producto probado es adecuado	1	2	3	4	5

Fuente: Elaboración Propia.

A continuación se miran gráficos sobre las pruebas realizadas, en la Ilustración 4.2 y Ilustración 4.3 se miran las funcionalidades que se probaron, los casos de pruebas ejecutados y los incidentes que se mostraron en cada ciclo de pruebas, así como las pruebas de regresión realizadas en las pruebas de software.

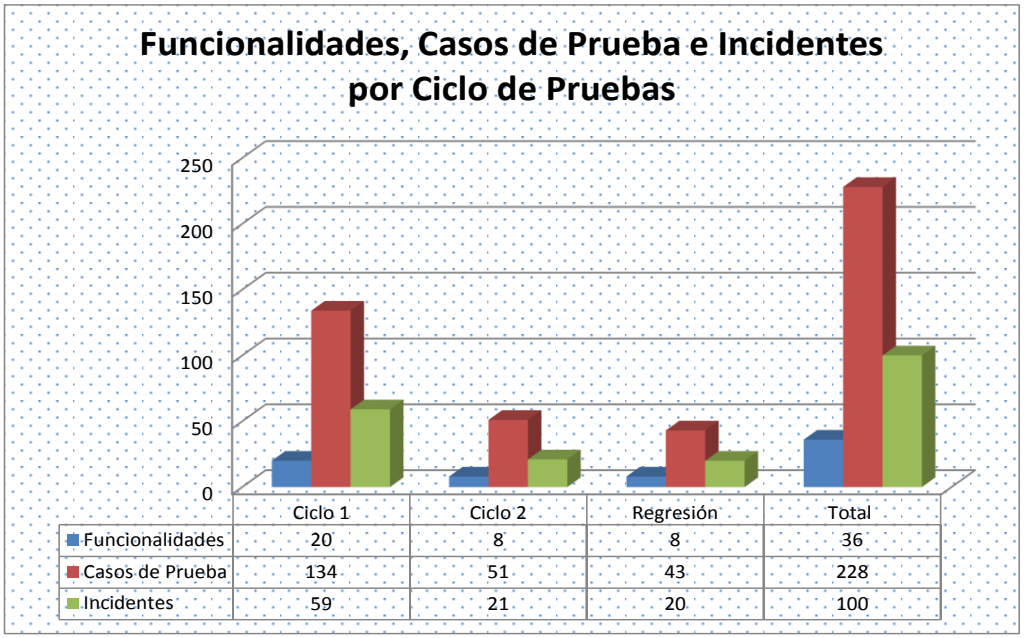


Ilustración 4.2– Funcionalidades, Casos de Prueba e Incidentes de las Pruebas Realizadas en Cada Ciclo.

Fuente: Elaboración Propia.

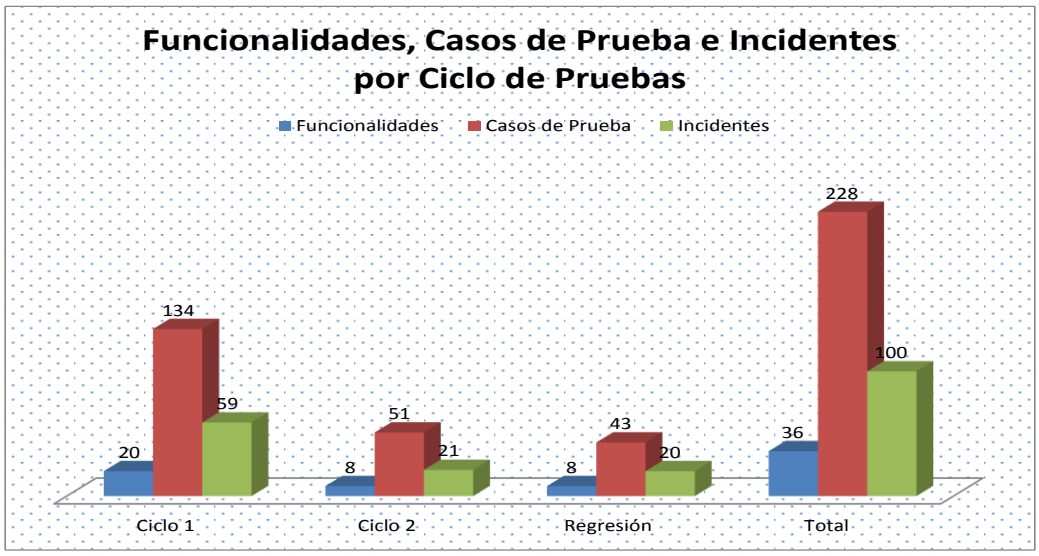


Ilustración 4.3– Funcionalidades, Casos de Prueba e Incidentes de las Pruebas Realizadas en Cada Ciclo.

Fuente: Elaboración Propia.

En la Ilustración 4.4 e Ilustración 4.5 se establecen los incidentes encontrados según su tipo.

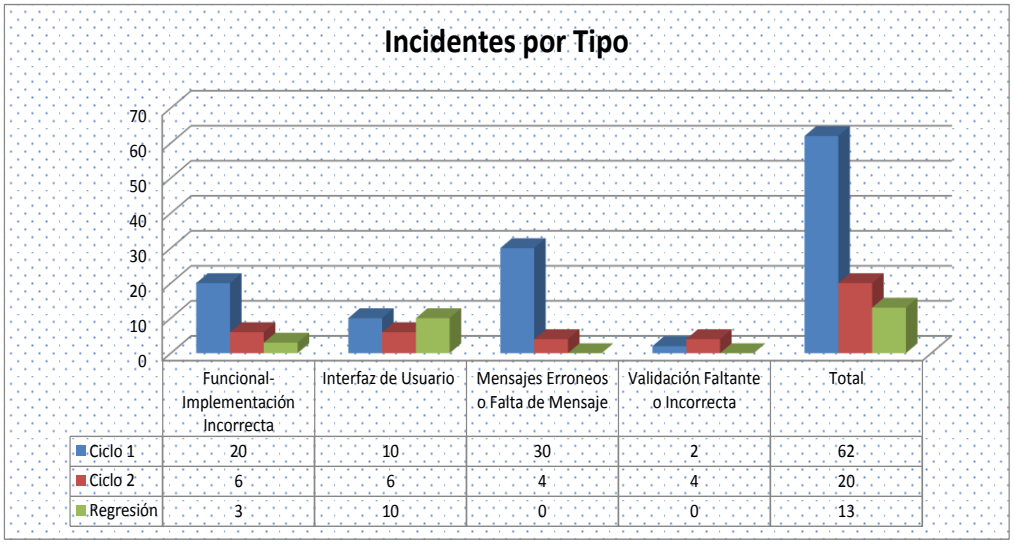


Ilustración 4.4 – Incidentes por Tipo según su Ciclo.

Fuente: Elaboración Propia.

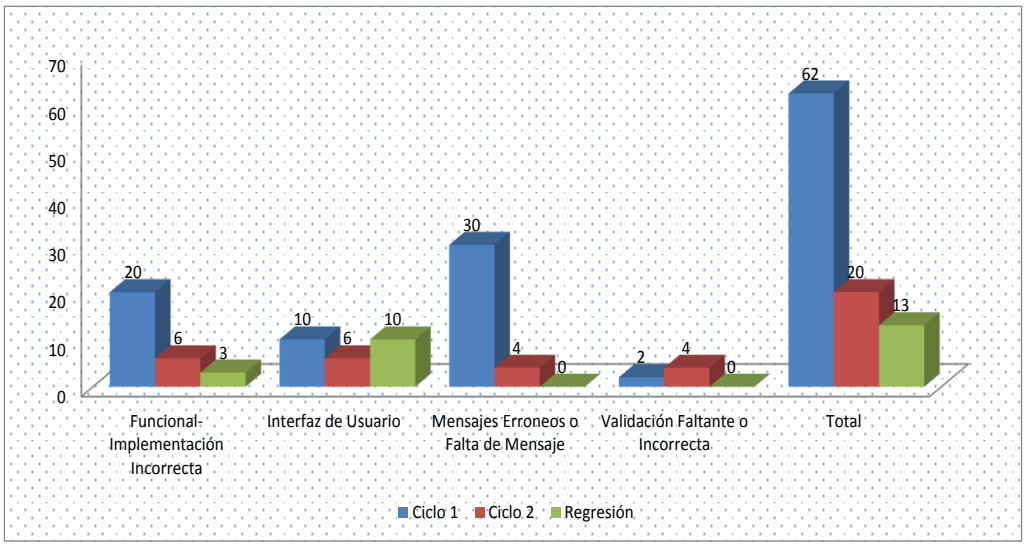


Ilustración 4.5– Incidentes por Tipo según su Ciclo.

Fuente: Elaboración Propia.

En la Ilustración 4.6 y Ilustración 4.7 se determina la criticidad de los incidentes encontrados durante la ejecución de las pruebas, se observan que son aquellos de importancia baja reduciéndose totalmente al realizar la segunda versión del producto.

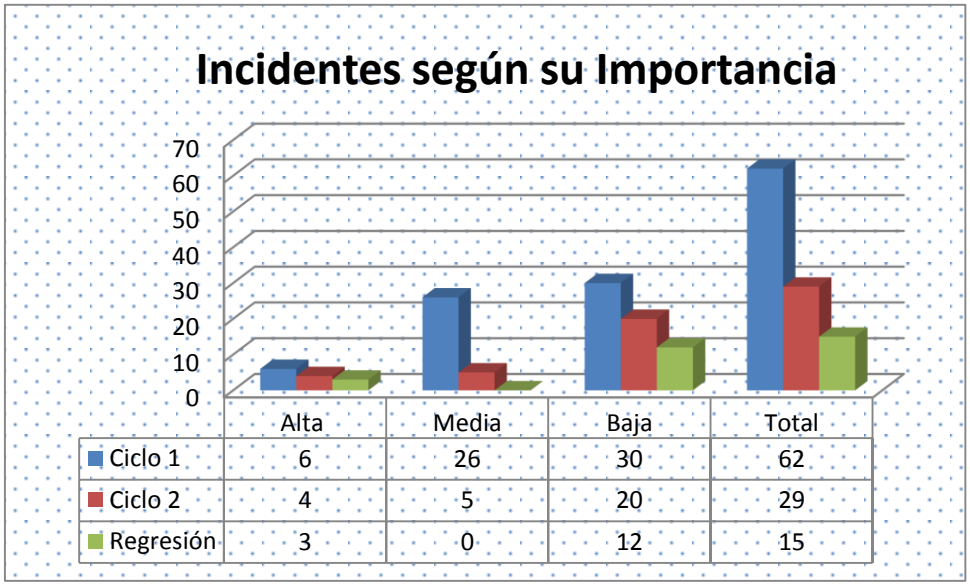


Ilustración 4.6 – Incidentes según Importancia.
Fuente: Elaboración Propia.

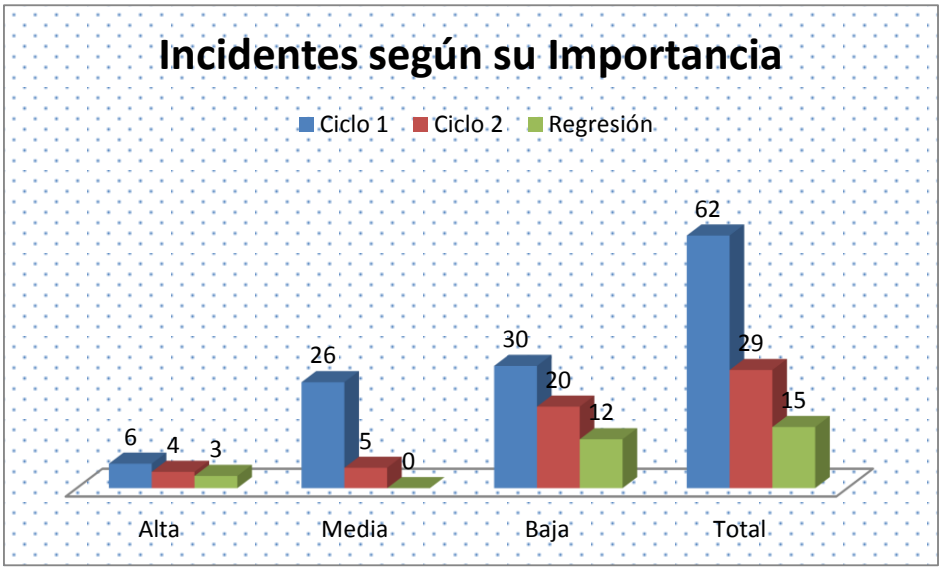
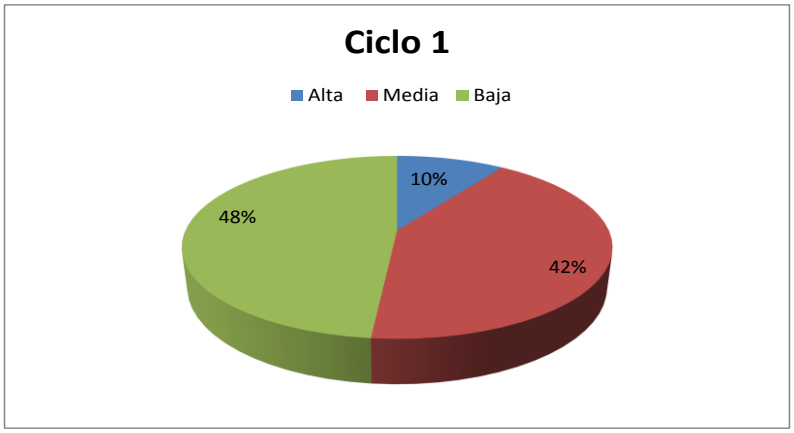
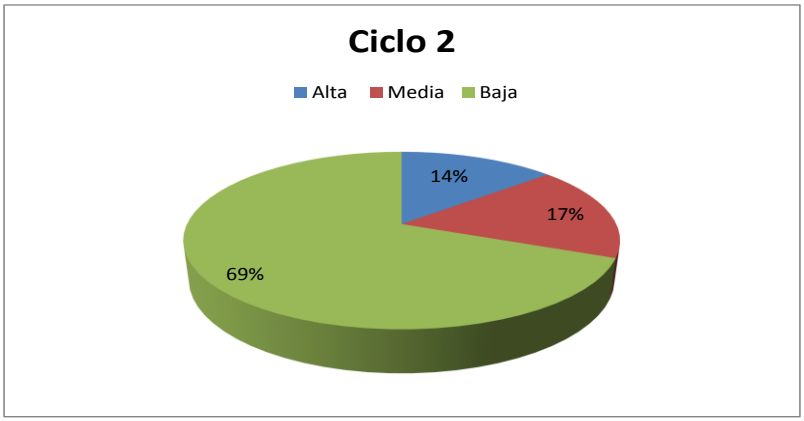


Ilustración 4.7 – Incidentes según Importancia.

Fuente: Elaboración Propia.

En la Ilustración 4.8 se observa el porcentaje de incidentes según su ciclo de pruebas, observándose un incremento de importancia alta en el segundo ciclo ya que el producto software probado tuvo alteraciones importantes en su funcionalidad.





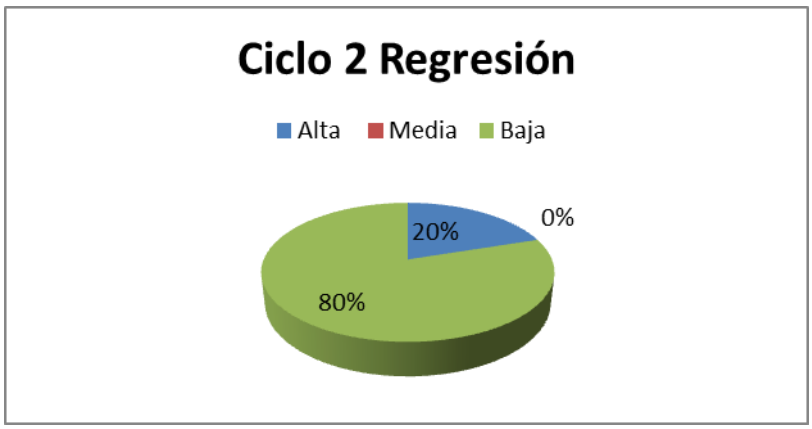


Ilustración 4.8 – Porcentaje de Incidentes según la Importancia (Alta, Media, Baja).

Fuente: Elaboración Propia.

En la Ilustración 4.9 se muestran indicadores en cada uno de los ciclos de prueba:

Casos de Prueba por Funcionalidad: Total de Casos de Prueba / Total de Funcionalidades. Determina cantidad de casos de prueba generados en promedio por funcionalidad.

Incidentes por funcionalidad: Total de Incidentes / Total de Funcionalidades. Determina el promedio de incidentes que tiene cada funcionalidad.

Incidentes por Caso de Prueba: Se calcula como el Total de Incidentes / Total de Casos de Prueba. Determina el promedio de incidentes que encontró cada caso de prueba ejecutado.

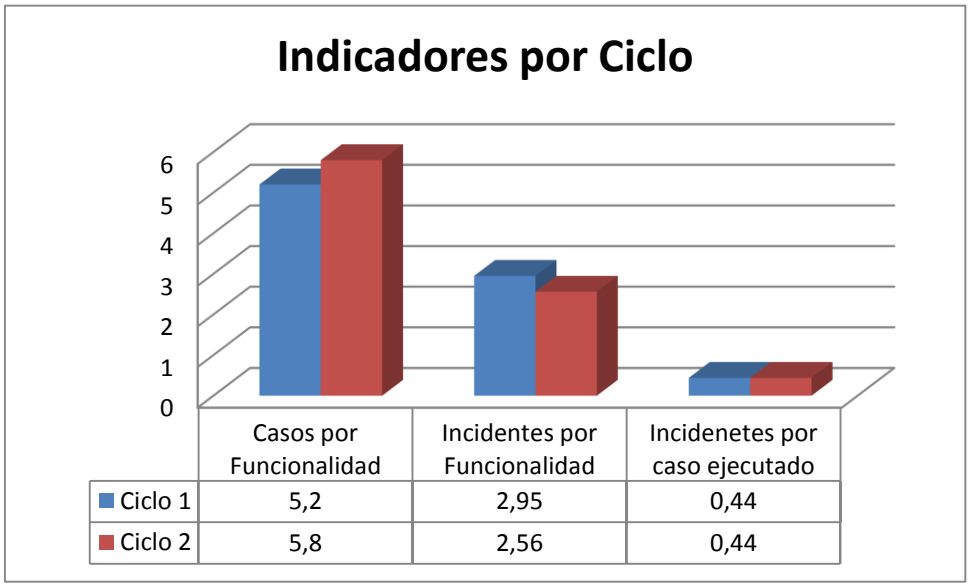


Ilustración 4.9 – Indicadores por Ciclo de Pruebas.

Fuente: Elaboración Propia.

4.3 CONCLUSIONES.

Para el diagnóstico se aplicó el estándar de calidad ISO-9001-2000 tomando en consideración: la responsabilidad de la dirección, el sistema de gestión de la calidad, la gestión de los recursos, la realización del producto, medición análisis y mejora.

Se realizó el diagnóstico del nivel de TMMI a la Gerencia y área de desarrollo en lo concerniente a las Políticas y estrategias de prueba, Planificación, Técnicas y Métodos, Entorno del testing, así como el seguimiento y control de las pruebas.

Se procedió a aplicar el modelo formal propuesto, el mismo que es independiente del proceso de desarrollo de software y se debieron validar las especificaciones funcionales del software, evaluar los riesgos del software a probar y realizar ciclos de prueba.

Las actividades de la especificación, permitieron analizar la necesidad del cliente en cuatro reuniones a fin de determinar las funcionalidades más importantes del software a probar, una vez validado por el cliente se revisó los requerimientos a partir de los manuales de usuario y de las versiones que se poseen del producto, detectándose que la documentación no estaba completa. Se determinó un listado de las funcionalidades más importantes, 40 funcionalidades, las cuales fueron validadas por el cliente, quedando en 20; determinando el alcance de las pruebas y estableciendo un cronograma que duraría dos meses.

Luego se procedió con la planificación para lo cual el equipo de pruebas mantuvo tres reuniones con el cliente, los reportes referentes a la ejecución de los casos de prueba y los incidentes correspondientes se entregaron al finalizar la semana, se revisó la documentación existente sobre el producto a probar, existiendo especificaciones incompletas, teniéndose que estos ser completados no en su totalidad, razón por la cual algunos fueron probados en el primer ciclo y otros en el siguiente ciclo. El ejecutable del producto a probar permitió entender de mejor manera el producto, lo cual redujo tiempos y costos en mejorar los requerimientos.

De las 20 funcionalidades a ser probadas se las realizó en dos ciclos la etapa de planificación duró tres semanas, el ciclo de pruebas 1, duró cuatro semanas, el ciclo de pruebas 2, cuatro semanas y la Evaluación del proyecto de prueba se realizó en la última semana.

El ciclo 1 se inició al realizar el Análisis del riesgo, se diseñaron casos de pruebas para probar funcionalidades, los cuales determinaron el alcance del ciclo, se estableció que era necesario aumentar una semana más de lo planeado ya que se aumentaron funcionalidades, de 12 a 20 estando planificadas al inicio 13, aumentándose 7, acordándose acortar el ciclo de pruebas 2.

El seguimiento y Control del Ciclo de Pruebas se hizo mediante reuniones cada dos días del equipo de pruebas, realizándose la Medición de Esfuerzo donde se reportó el esfuerzo del ciclo, la Gestión de la Configuración y se administró los elementos creados en las pruebas según el Testware.

Según lo planeado para el primer ciclo, se tenía una nueva versión del producto, pero al no generarse a tiempo se usó la versión instalada utilizándose la exploración en la etapa de Planificación, se determinó como herramienta alternativa inicial Excel para gestionar las pruebas y el reporte de seguimiento de incidentes.

Los casos de prueba se generaron utilizando las técnicas de caja negra de partición de equivalencia y valor límite. Se hizo una matriz que permita establecer la relación que existe entre funcionalidades y los casos de prueba generados.

En el ciclo 1 la versión que se está probando es la que se tenía instalada en la fase de Planificación razón por lo cual no se requirió ejecutar Pruebas de Humo ya que las funcionalidades básicas no presentaban ningún inconveniente. Se realizaron la Ejecución de las Pruebas de Testing Exploratorio, registrando los incidentes encontrados en el programa de excel y posteriormente fueron validados por el cliente.

Se hizo una reunión con el cliente para explicarle sobre los resultados obtenidos en las pruebas del ciclo número 1 a fin de proceder a cerrar el ciclo y continuar con el siguiente.

La nueva versión estuvo en dos semanas, ejecutándose las pruebas de inmediato ya que en el ciclo de prueba 1 se establecieron nuevas funcionalidades, por lo que se recortó el alcance del ciclo de pruebas 2.

Se puede observar lo planificado para el ciclo 2 y el alcance real, según las funcionalidades a ser probadas. Al ser la nueva versión una versión mejorada de la anterior, el usuario quería probar funcionalidades de la anterior versión en la nueva, para estar seguro de que no se hayan visto afectadas.

Aquellos requerimientos nuevos a ser probados en ciclo 2 fueron analizados, Se utilizó técnicas de caja negra, para realizar los casos de prueba para lo cual se hizo una matriz que permita relacionar las funcionalidades versus los casos de prueba ejecutados.

Se utilizaron Pruebas de Humo, el producto software probado poseía funcionalidades básicas, realizándose los casos de prueba la Ejecución de las Pruebas y Testing Exploratorio. También al inicio del ciclo se realizaron pruebas de regresión. Luego de realizar las pruebas se procedió a la evaluación, donde se utilizó la Evaluación del Ciclo de Pruebas.

La actividad Reporte de Incidentes, se inició al mismo momento en que se estaba realizando el Ciclo de Pruebas 2, se puede ver una comparación entre la planificación del proyecto y la realidad de la realización de los Ciclos de Prueba. El cliente estuvo de acuerdo con lo realizado en el ciclo de pruebas 2 y con lo realizado en el proyecto de pruebas. Se evaluó en una reunión la satisfacción del Cliente.

Bibliografía:

Abadia, M. E. V. (2010). Procedimientos para prueba de software. *Publicaciones Icesi*(18).

Ampuero, M. A., & López Trujillo, Y. (2010). Creando un profesional con disciplina en el proceso de desarrollo de software. *Ingeniería Industrial, 27*(1), 4 pág.

Bach, J. (1999). Heuristic risk-based testing. *Software Testing and Quality Engineering Magazine*, *11*, 99.

Bach, J. (2001). What is Exploratory Testing? And How it differs from Scripted Testing. *StickyMinds*, *Enero*.

Bach, J. (2003). Exploratory testing explained. *Online: http://www. satisfice. com/articles/et-article. pdf*.

Bahill, A. T., Bharathan, K., & Curlee, R. F. (1995). How the testing techniques for a decision support system changed over nine years. *Systems, Man and Cybernetics, IEEE Transactions on*, *25*(12), 1533-1542.

Bajarlía, M. V. S., Eterovic, J., & Ierache, J. (2008). *Elaboración de Especificación de Requerimientos de Seguridad en el desarrollo de Sistemas de Información basado en la Modelización de Conocimientos.* Paper presented at the X Workshop de Investigadores en Ciencias de la Computación.

Beizer. (1990a). *Software testing techniques* ((2nd ed) ed.).

Beizer. (1990b). *"Software testing techniques (2nd ed.)"*.

Beizer, B. Software testing techniques. 1990. *New York, ISBN: 0-442-20672-0*.

Bird, D. L., & Munoz, C. U. (1983). Automatic generation of random self-checking test cases. *IBM systems journal, 22*(3), 229-245.

Black. (2002). *Managing the Testing Process*.

Board, I. S. T. Q. (2005). Certified Tester Foundation Level Syllabus.

Burnstein, I. (2003). *Practical software testing: a process-oriented approach*: Springer.

Burnstein, I. (2010). *Practical software testing a process-oriented approach*. Verlag New York inc.

Castillo Chamorro, J. M. (2007). *Metodología para la construcción de modelos multiagente en entornos de planificación.* Universidad Rey Juan Carlos.

Crespo, Ê. G., Palacios, R. C., & Seco, A. A. (2005). Desarrollo de competencias mediante la creación de una herramienta de soporte al PSP.

Cristiá, M. (2009). Introducción al Testing de Software. *Universidad Nacional de Rosario*.

DAICH, G., PRICE, G., RAGLAND, B., & DAWOOD, M. (1994). Test and reengineering tool evaluation project: Software test technologies report.

de Abadia, M. E. V. (2010). Procedimientos para prueba de software. *Publicaciones Icesi*(18).

Deming, W. E. (1986). *Out of the Crisis*.

Deutsch, M. S. (1981). *Software verification and validation: Realistic project approaches*: Prentice Hall PTR.

Escalona, M. J., & Koch, N. (2002). Ingeniería de Requisitos en Aplicaciones para la Web–Un estudio comparativo. *Universidad de Sevilla*.
Esteban, A. S. (2012). *Marco metodológico para la mejora de las actividades de verificación y validación de productos software.* Universidad Carlos III de Madrid.
Gelperin, D., & Hetzel, B. (1988). The growth of software testing. *Communications of the ACM*, *31*(6), 687-695.
Glenford, J. M., & Sandler, C. (1979). The art of software testing. *John Wiley and Sons, ISBN*, *10*, 0471043281.
Hetzel, B. (1991). Software Testing: Some Troubling Issues and Opportunities. *BCS Special Interest Group in Software Testing*, 707-726.
Hetzel, W. C., & Hetzel, B. (1988). *The complete guide to software testing*: QED Information Sciences Wellesley, MA.
Howden, W. E. (1976). Reliability of the path analysis testing strategy. *Software Engineering, IEEE Transactions on*(3), 208-215.
Howden, W. E. (1978). Theoretical and empirical studies of program testing. *Software Engineering, IEEE Transactions on*(4), 293-298.
Howden, W. E. (1982). Life-cycle software validation. *Computer*, *15*(2), 71-78.
HOWDEN.W.E., M. E. a. (1981). Software Testing and Validation Techniques. *IEEE Computer Society*.
Hu, H., Jiang, C.-H., Cai, K.-Y., Wong, W. E., & Mathur, A. P. (2013). Enhancing software reliability estimates using modified adaptive testing. *Information and Software Technology*, *55*(2), 288-300. doi: http://dx.doi.org/10.1016/j.infsof.2012.08.012
Humphrey, W. S. (2001). Introducción al Proceso Software Personal (PSP). *Edición Addison Wesley*.
Humphrey, W. S. (Madrid, 2001.). Introducción al proceso de software personal,.
Humphrey, W. S., Snyder, T. R., & Willis, R. R. (1991). Software process improvement at Hughes Aircraft. *Software, IEEE, 8*(4), 11-23.
IEEE. (1983). Standard for Software Test Documentation (IEEE,Std 829-1983).
IEEE. (1990a). Standard Glossary of Software Engineering Terminology Institute of Electrical and Electronics Engineers.
IEEE. (1990b). Standart Glossary of Software Enginieering Terminology (Std610.12-1990). Copyright 1990 by IEEE. All rights reserved.
IEEE (IEEE Std. 610, -. (1990). Standard Glossary of Software Engineering Terminology (IEEE Std. 610,12-1990).
Ince, D. C. (1987). The automatic generation of test data. *The Computer Journal*, *30*(1), 63-69.
Jung, H.-W., Kim, S.-G., & Chung, C.-S. (2004). Measuring software product quality: A survey of ISO/IEC 9126. *IEEE software, 21*(5), 88-92.
Kaner, C. (2001). Measurement Issues and Software Testing.
Kaner, C., Bach, J., & Pettichord, B. (2001). Lessons Learned in Software Testing, 2001. *John Wiley & Sons. completion of, and return to use case*, *247*, 189-190.
Kaner, C., Bach, J., & Pettichord, B. (2008). *Lessons learned in software testing*: John Wiley & Sons.
Kaner C., F. J., Nguyen H. (1999). *Testing Computer Software*, (2nd Edition ed.).
Kit. (1995). *Software Testing In The Real World : Improving The Process*.

Korel, B. (1990). Automated software test data generation. *Software Engineering, IEEE Transactions on*, *16*(8), 870-879.
León Perdomo, Y., Enrique Góngora Rodríguez, A., & Febles Estrada, A. (2013). Aplicando métricas de calidad a proyectos y procesos durante las pruebas exploratorias. *Revista Cubana de Ciencias Informáticas*, *7*(2), 193-205.
López, O., Laguna, M. Á., & Marqués, J. M. (2000). Normalización de Assets de Requisitos en el Contexto de la Reutilización Sistemática del Software. *sistema*, *13*(15), 9.
McCracken, D. D. (1957). *Digital computer programming*: John Wiley & Sons.
Mendoza, L. E., Pérez, M. A., & Grimán, A. C. (2005). Prototipo de modelo sistémico de calidad (MOSCA) del software. *Computación y Sistemas*, *8*(3), 196-217.
Mesquida, A. L., Mas, A., & Amengual, E. (2009). La madurez de los servicios TI. *Innovación, Calidad e Ingeniería del Software*, *5*(2), 77.
Myers. (2004). *The art of software testing* (2nd edition ed.).
Myers, G. J. (1979). The Art of software Testing John Wiley & sons. *New York*.
Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing*: John Wiley & Sons.
Padayachee, I., Kotze, P., & van Der Merwe, A. (2010). ISO 9126 external systems quality characteristics, sub-characteristics and domain specific criteria for evaluating e-Learning systems. *The Southern African Computer Lecturers' Association, University of Pretoria, South Africa*.
Perona Ossaba, L. Á., & Velásquez Isaza, J. E. (2013). Gestión de conocimiento: la solución para disminuir el reproceso en las pruebas de software.
The Personal Software Process. Technical Report CMU/SEI-2000-TR-022. (2000).
Pomeroy-Huff M., M. J., Cannon R., Sebern S. . (2005). The Personal Software Process (PSP) Body of Knowledge.
Puello, O. (2013). Model verification and validation based on CMMI.
Pytel, P., Uhalde, C., Ramón, H. D., Castello, H., Tomasello, M., Pollo Cattaneo, M. F., . . . García Martínez, R. (2011). *Ingeniería de requisitos basada en técnicas de ingeniería del conocimiento.* Paper presented at the XIII Workshop de Investigadores en Ciencias de la Computación.
Rumbaugh, J., Jacobson, I., & Booch, G. (1999). The unified software development process: Addison Wesley.
Shewhart, W. A. (1931). *Economic control of quality of manufactured product* (Vol. 509): ASQ Quality Press.
Shorten, F. (1985). NBS (National Bureau of Standards) reactor: summary of activities July 1983 through June 1984. Technical note: National Bureau of Standards, Washington, DC (USA). Reactor Radiation Div.
Sommerville, I. (2005). *Ingeniería del software 7/e*: Pearson Educación.
Sommerville., I. (1995). *Software engineering (5th ed.)*. USA, Redwood City: Addison Wesley Longman Publishing Co.
Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 433-460.
Wallmüller, E. (2007). Qualität & Informatik, from www.itq.ch/pdf/tpi/2007_TPI_Intro_eV10.pdf
Whittaker. (2002). *"How to break Software, a practical Guide"*.
Wilford Rivera, I., & López Trujillo, Y. (2010). USO EFECTIVO DE LOS RECURSOS HUMANOS IMPLICADOS EN EL PROCESO DE DESARROLLO DE SOFTWARE. *Ingeniería Industrial*, *27*(2-3), 4 pág.

Linkografía:

Board, I. S. T. Q. (2012), from http://www.istqb.org/downloads/finish/20/101.html

Burnstein I., S. T., Carlson C. (1996). Aminer, from http://pdf.aminer.org/000/361/558/european_and_american_software_process_maturity_models_and_assessments_tutorial.pdf

Hu, H., Jiang, C.-H., Cai, K.-Y., Wong, W. E., & Mathur, A. P. (2013). Enhancing software reliability estimates using modified adaptive testing. Information and Software Technology, 55(2), 288-300. doi: http://dx.doi.org/10.1016/j.infsof.2012.08.012

Informática, A. T. d. (2008), from http://www.ati.es/IMG/pdf/SanzVol4Num4.pdf

Ing. Ernesto Sandoval S, G. R. D. (2010). Lista de chequeo, from http://ebookbrowsee.net/lista-de-chequeo-iso-9001-pdf-d603376220

Lemos, O. A. L., Ferrari, F. C., Eler, M. M., Maldonado, J. C., & Masiero, P. C. (2013). Evaluation studies of software testing research in Brazil and in the world: A survey of two premier software engineering conferences. Journal of Systems and Software, 86(4), 951-969. doi: http://dx.doi.org/10.1016/j.jss.2012.11.040

Sevilla, U. d. (2014), from http://www.ccoo.us.es/uploads/descargas/documentacion/NormaInternacionalISO9001.pdf

Sogeti. (2010). Sogeti, from www.es.sogeti.com/Global/Fichas%20comerciales/Ficha_TMap.pdf

SWEBOK, v. I. C. S. (2004). Guide to the Software Engineering Body of Knowledge from http://www.swebok.org