UNIVERSIDAD TÉCNICA DEL NORTE

INSTITUTO DE POSGRADO

MAESTRÍA EN INGENIERÍA DE SOFTWARE

"Elaboración de un marco de trabajo para pruebas de software, basado en el estándar ISO/IEC/IEEE 29119 y su impacto en el proceso de evaluación del software"

Trabajo de Grado previo a la obtención del Título de

Magíster en Ingeniería de Software

AUTOR:

Ing. Wilson Aníbal Cárdenas Hernández

DIRECTORA:

MSc. Cathy Pamela Guevara Vega

IBARRA - ECUADOR

2019

Dra. Lucía Yépez V. MSc.
DIRECTORA DEL INSTITUTO DE POSGRADO

ASUNTO: Designación de Asesor/Revisor

Señora Directora:

Yo, Cathy Pamela Guevara Vega tutora el Trabajo de Grado "ELABORACIÓN DE UN MARCO DE TRABAJO PARA PRUEBAS DE SOFTWARE, BASADO EN EL ESTÁNDAR ISO/IEC/IEEE 29119 Y SU IMPACTO EN EL PROCESO DE EVALUACIÓN DEL SOFTWARE", del Ing. Wilson Aníbal Cárdenas Hernández, de la Maestría de INGENIERÍA DE SOFTWARE COHORTE II, solicito de la manera más comedida se sirva designar Asesor/Revisor.

Por la favorable atención prestada al presente, anticipo mi agradecimiento.

Atentamente,

Ing. Cathy Pamela Guevara Vega, MSc.
cguevara@utn.edu.ec

El presente trabajo de titulación fue aprobado por HCD el *7 de 0ctubre de 2017*; *oficio 1637-CDIP*, y sugiero la designación del MSc. Mauricio Rea, como Asesor/Revisor.

Adjunta: Informe de la calificación del Trabajo de Grado, Certificación de inicio y fin de programa académico, Certificado de no adeudar y Record Académico.

Para los fines consiguientes:

Ing. Edwin Marcelo Jurado Ávila, MSc.
COORDINADOR DE LA MAESTRÍA EN INGENIERÍA DE SOFTWARE COHORTE II

Dra. Lucía Yépez V. MSc.
DIRECTORA DEL INSTITUTO DE POSGRADO

ASUNTO: Informe de Revisión de Trabajo de Titulación

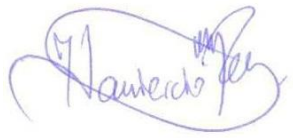
Señora Directora:

Yo, Mauricio Rea, del Trabajo de Grado "ELABORACIÓN DE UN MARCO DE TRABAJO PARA PRUEBAS DE SOFTWARE, BASADO EN EL ESTÁNDAR ISO/IEC/IEEE 29119 Y SU IMPACTO EN EL PROCESO DE EVALUACIÓN DEL SOFTWARE" del maestrante Ing. Wilson Aníbal Cárdenas Hernández, de la Maestría en INGENIERÍA DE SOFTWARE, COHORTE II, remito a usted el informe de la calificación del Trabajo de Grado, así como el detalle de los criterios a corregir en el mismo, que detallo a continuación:

- No tiene correcciones relevantes que realizar.

Lo que informo para los fines pertinentes.

Atentamente,



Ing. Xavier Mauricio Rea Peñafiel, MSc.
mrea@utn.edu.ec
Adjunto: Formato A8 y ejemplar con correcciones

DECLARACIÓN DE RESPONSABILIDAD

Ing. Wilson Aníbal Cárdenas Hernández

DECLARO QUE:

El proyecto de grado denominado **"ELABORACIÓN DE UN MARCO DE TRABAJO PARA PRUEBAS DE SOFTWARE, BASADO EN EL ESTÁNDAR ISO/IEC/IEEE 29119 Y SU IMPACTO EN EL PROCESO DE EVALUACIÓN DEL SOFTWARE"**, y bajo juramento que el contenido e información que se encuentra en el presente trabajo ha sido desarrollado con base a una investigación exhaustiva y de mi autoría, respetando derechos intelectuales de terceros conforme se menciona en la sección bibliográfica de este trabajo.

Ing. Wilson Aníbal Cárdenas Hernández

C.C. 1003090733

AUTOR

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	100309073-3		
APELLIDOS Y NOMBRES:	Ing. Wilson Aníbal Cárdenas Hernández		
DIRECCIÓN:	Atuntaqui, Atahualpa 13-50 y Olmedo		
EMAIL:	wcardenas1283@gmail.com		
TELÉFONO FIJO:	(06) 2951 731	**TELÉFONO MÓVIL:**	0993268628

DATOS DE LA OBRA		
TÍTULO:	ELABORACIÓN DE UN MARCO DE TRABAJO PARA PRUEBAS DE SOFTWARE, BASADO EN EL ESTÁNDAR ISO/IEC/IEEE 29119 Y SU IMPACTO EN EL PROCESO DE EVALUACIÓN DEL SOFTWARE.	
AUTOR:	Ing. Wilson Aníbal Cárdenas Hernández	
FECHA:	02/04/2019	
SOLO PARA TRABAJOS DE GRADO		
PROGRAMA:	☐ Pregrado	■ Posgrado
TÍTULO POR EL QUE OPTA:	Magister en Ingeniería de Software	
ASESORA / DIRECTORA:	MSc. Cathy Pamela Guevara Vega	

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 2 días del mes de abril de 2019

EL AUTOR:

Ing. Wilson Aníbal Cárdenas Hernández
C. C. 1003090733

Dedicatoria

La vida está llena de sueños, metas y objetivos por cumplir,
y es grato saber que cuentas con el apoyo de personas que contribuyen a
alcanzarlos, este trabajo se lo dedico.

A Dios,
por permitirme culminar con éxito esta etapa de mi vida.

A mi madre,
por todo el amor, cariño y fortaleza demostrados
al dedicar su vida a cuidarme y educarme.

A mi esposa,
mi compañera de vida, por ser mi complemento y siempre estar a mí
lado brindándome su amor y apoyo incondicionales.

A mi hijo Matías,
por ser mi inspiración y fortaleza en los momentos más difíciles,
quien, con un abrazo, una palabra o una sonrisa, me da el aliento
necesario para seguir adelante.

Agradecimiento

Mi agradecimiento profundo y sincero, a todos quienes me brindaron su apoyo, para culminar con éxito esta etapa tan importante de mi vida.

Mi gratitud sincera a todos mis maestros, por compartir sus conocimientos y experiencias en las aulas; en especial a mi Directora de Tesis, la MSc. Cathy Guevara, por el compromiso mostrado durante la realización del presente trabajo.

Al personal de la Dirección de Tecnología de la Cooperativa de Ahorro y Crédito "Atuntaqui" Ltda., por facilitar la información necesaria para la investigación; con una mención especial para el Ing. Roberto Peñafiel, por el apoyo incondicional brindado.

Agradecer a mi familia, por estar a mi lado, brindándome su cariño y consejos, animándome a seguir adelante y no permitir que decaiga en alcanzar este objetivo.

A todos y cada uno, mi eterno agradecimiento.

Índice General

Índice de Figuras

Índice de Tablas

UNIVERSIDAD TÉCNICA DEL NORTE

INSTITUTO DE POSGRADO

PROGRAMA DE MAESTRÍA EN INGENIERÍA DE SOFTWARE

wcardenas1283@gmail.com

"ELABORACIÓN DE UN MARCO DE TRABAJO PARA PRUEBAS DE SOFTWARE, BASADO EN EL ESTÁNDAR ISO/IEC/IEEE 29119 Y SU IMPACTO EN EL PROCESO DE EVALUACIÓN DEL SOFTWARE"

Autor: Ing. Wilson Aníbal Cárdenas Hernández

Directora: Ing. Cathy Pamela Guevara Vega, MSc.

Año: 2019

Resumen

Esta investigación presenta el resultado del estudio realizado al Proceso de Gestión de Desarrollo de Software en la Dirección de Tecnología de la COAC Atuntaqui Ltda., implementa un marco de trabajo para la gestión de las pruebas de software, basado en estándares internacionales, y determina el impacto de este, en las fases de pruebas y mantenimiento del software.

El objetivo es corregir los problemas causados por el proceso informal de pruebas de software, que realiza la Dirección de Tecnología de la COAC Atuntaqui Ltda. Se estudian los conceptos, técnicas, procesos, documentación y estándares de pruebas de software, y se elabora un marco de trabajo para pruebas de software, basado en los estándares ISO/IEC/IEEE 29119.

La propuesta se enfoca en el establecimiento de una metodología de pruebas de software, basada en los procesos establecidos en el estándar ISO/IEC/IEEE 29119-2, y la elaboración de plantillas para la documentación de las pruebas de software, basadas en el estándar ISO/IEC/IEEE 29119-3; conformando un marco de trabajo que se convierta en una guía técnica para la evaluación del software desarrollado en la COAC Atuntaqui Ltda.

Se aplica el marco de trabajo para pruebas de software propuesto, en un proyecto de software; se desarrollan las etapas establecidas en la metodología, se genera la documentación correspondiente a cada etapa, y se analizan los resultados obtenidos para determinar las conclusiones y recomendaciones finales de la investigación.

Palabras claves: Pruebas de software, procesos de pruebas, caso de prueba, documentación de las pruebas, técnicas de pruebas, ISO/IEC/IEEE 29119.

UNIVERSIDAD TÉCNICA DEL NORTE

INSTITUTO DE POSGRADO

PROGRAMA DE MAESTRÍA EN INGENIERÍA DE SOFTWARE

wcardenas1283@gmail.com

"ELABORATION OF A FRAMEWORK FOR SOFTWARE TESTING, BASED ON THE ISO/IEC/IEEE 29119 STANDARD AND ITS IMPACT ON THE SOFTWARE EVALUATION PROCESS"

Author: Eng. Wilson Aníbal Cárdenas Hernández

Director: Eng. Cathy Pamela Guevara Vega; MSc.

Year: 2019

Abstract

This research presents the result of the study carried out to the Software Development Management Process in the Technology Directorate of the COAC Atuntaqui Ltda., Implements a framework for the management of the software tests, based on international standards, and determines the impact of this, in the testing and maintenance phases of the software.

The objective is to correct the problems by the informal process of software testing, which is carried out by the Technology Directorate of COAC Atuntaqui Ltda. The concepts, techniques, processes, documentation and software testing standards are studied, and a framework for software testing is developed, based on ISO/IEC/IEEE 29119 standards.

The proposal focuses on the establishment of a software testing methodology, based on the processes established in the ISO / IEC / IEEE 29119-2 standard, and the development of templates for the documentation of software tests, based on the standard ISO / IEC / IEEE 29119-3; forming a framework that becomes a technical guide for the evaluation of the software developed in the COAC Atuntaqui Ltda..

The framework for testing software proposed is applied in a software project; the stages established in the methodology are developed, the documentation corresponding to each stage is generated, and the results obtained are analyzed to determine the conclusions and final recommendations of the investigation.

Keywords: Testing software, testing processes, test cases, test documentation, testing techniques, ISO / IEC / IEEE 29119.

CAPÍTULO 1. EL PROBLEMA

1.1. Antecedentes

La ingeniería de software es una disciplina de la ingeniería que se preocupa por todos los aspectos de la producción de software, en todas sus fases, desde la especificación del sistema hasta la fase de mantenimiento; nace con la necesidad de construir software de calidad, que cumpla con los requisitos de usuario, presupuestos y tiempos establecidos para su desarrollo.

La ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software, que es la aplicación de la ingeniería de software (IEEE[1], 1990).

Los modelos de ciclo de vida del software describen las fases del ciclo del software y el orden en que se ejecutan las fases (INTECO[2], 2009). Los métodos de la ingeniería del software indican "cómo" construir técnicamente el software. Los métodos abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento (Pressman, 2014).

En la actualidad, existe una variedad de modelos de ciclo de vida del software, cada uno con sus características, ventajas y desventajas propias del modelo. Sin embargo, estos modelos concuerdan en las fases principales del desarrollo del software, como son: análisis, diseño, codificación, pruebas y mantenimiento; es decir, todos los modelos incluyen la fase de gestión de pruebas de software, como una fase fundamental del proceso de desarrollo de software, existiendo incluso metodologías, como *Test-Driven Development* [Desarrollo guiado por pruebas], que desarrollan software basándose netamente en la ejecución de pruebas**.**

[1] Institute of Electrical and Electronics Engineers
[2] Instituto Nacional de Tecnologías de la Comunicación

Una vez que se ha generado el código, comienzan las pruebas del programa. Según la IEEE, el proceso de pruebas de calidad de software permite a los desarrolladores brindar productos con altos estándares de calidad y minimiza los riesgos. Las pruebas de software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación (Pressman, 2014).

Algunos métodos de prueba se llevan a cabo a nivel de la ciencia basura, porque a menudo se basan en pequeñas muestras y experimentos mal controlados o documentados (Gallesdic, 2013), se tiende a pensar que el proceso de las pruebas de software se debe realizar en la última etapa del desarrollo y/o no se invierten los tiempos y costos necesarios para una adecuada ejecución de las pruebas. Sin embargo, una correcta gestión de las pruebas de software nos puede evitar el fracaso del proyecto.

1.2. Planteamiento del problema

La Dirección de Tecnología de la COAC[3] Atuntaqui Ltda., hace aproximadamente dieciocho años, conformo el Área de Desarrollo de Software, área encargada de implementar el software requerido por parte de la cooperativa, para brindar un mejor servicio a socios, clientes y trabajadores. Es así como, el Área de Desarrollo, realiza una planificación anual de los proyectos de software a implementar, aportando de esta manera a cumplir los objetivos institucionales.

El proceso de desarrollo de software establecido por la cooperativa no dispone de un modelo para la gestión de pruebas, que permita planificar las pruebas desde el inicio del proyecto, establecer los casos de pruebas a realizar, los responsables de la ejecución de las pruebas y permita generar la documentación inherente a la ejecución de las pruebas, muy necesaria para la fase de mantenimiento.

[3] Acrónimo de Cooperativa de Ahorro y Crédito

Sin un modelo establecido para la gestión de las pruebas, estas se realizan de manera informal, se ejecutan por parte del propio programador y/o usuario experto, quienes no realizan los casos de pruebas suficientes y/o adecuados para la validación del código; a esto se suma, que las personas que ejecutan las pruebas no tienen la cultura de documentar el proceso, con lo cual, la documentación de las pruebas es escasa o simplemente no existe.

Todas estas malas prácticas en el proceso de gestión de las pruebas han generado la inclusión de defectos en el software que se despliega en el ambiente de producción, provocando fallas en los sistemas de información implementados. Es así que, durante el año 2017, el treinta por ciento de los casos de *Service Desk*[4] registrados en la cooperativa, son para corregir defectos encontrados en el software desarrollado, ocasionando varios efectos negativos en el proceso de producción de software, como por ejemplo: el proceso de mantenimiento del software se vuelve complejo, la funcionalidad del software se deteriora tempranamente, la elaboración de nuevos proyectos de software se retrasa por falta de tiempo, se incrementan costos y tiempos no previstos en la estabilización del software, pero sobre todo, se generan molestias a socios y clientes, afectando directamente la imagen institucional de la cooperativa.

Una alternativa a estos inconvenientes es la elaboración de un marco de trabajo para la gestión de pruebas de software, basado en metodologías y estándares internacionales, que permita gestionar las pruebas de software de manera formal, abarque todo el ciclo de vida de desarrollo y genere la documentación correspondiente de las mismas.

1.3. Formulación del problema

¿La implementación de un marco de trabajo para la gestión de pruebas de software, basado en el estándar ISO/IEC/IEEE 29119, permitirá mejorar el proceso de evaluación del software, en la COAC Atuntaqui Ltda.?

[4] Mesa de Ayuda implementada en la COAC Atuntaqui para el Soporte Técnico a los usuarios.

1.4. Justificación de la investigación

En las comunidades de software comercial, las pruebas del software fueron vistas alguna vez como una idea de último momento. Los gerentes de producto novatos e incluso los desarrolladores las consideraban como una disciplina fuera de foco, que casi cualquier persona podía realizar (Schaull, 2011). Incluso, en algunos libros de divulgación sobre el tema, incluyen en sus títulos palabras como arte y oficio, lo que puede haber llevado a algunas personas en la industria, a la conclusión de que las pruebas para la calidad no eran una verdadera disciplina de la Ingeniería de Software.

Por el contrario, la prueba efectiva es una disciplina de esta ingeniería y un componente crítico del ciclo de vida del desarrollo de software, cuyo objetivo es ayudar a mejorar la calidad y fiabilidad del producto, mediante la identificación de defectos, la prevención de errores, y la observación y presentación de reportes (RACCIS[5], 2013).

La fase de pruebas de software tiene como objetivo, verificar el sistema para comprobar si este cumple sus requisitos (Gallesdic, 2013). Dentro de esta fase pueden desarrollarse distintos tipos de pruebas, en función de los objetivos planteados. Algunos tipos de pruebas son: pruebas funcionales, pruebas de usabilidad, pruebas de rendimiento, pruebas de seguridad. Las pruebas funcionales del software verifican que el sistema ofrece a los usuarios la funcionalidad recogida en su especificación (Jorgensen, 2013).

Villarreal manifiesta que: “Se debe fomentar el uso de pruebas de software en los grupos de desarrollo y considerar a las pruebas de software como una forma efectiva de detectar problemas y mejorar la calidad del software” (Villarreal, 2015)

La COAC Atuntaqui es una institución financiera con más de 55 años de trayectoria en el ámbito financiero del país, dedicada a la captación y colocación de recursos económicos; donde la seguridad de la información, infraestructura tecnológica y fiabilidad del software,

[5] Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de software.

cumplen un rol fundamental en el cumplimiento de los objetivos institucionales, cualquier defecto en el software desarrollado, podría representar grandes pérdidas de dinero, afectar el prestigio de la institución e inclusive provocar su cierre.

Por estas razones y de acuerdo con el criterio de Villarreal, resulta primordial establecer un modelo formal para la gestión de pruebas de software, que garantice que el software desarrollado sea: funcional, confiable, eficiente, mantenible y, sobre todo, garantice la seguridad de la información de socios y clientes.

La realización de este proyecto es muy importante, porque con un modelo de gestión de pruebas, se pretende: corregir los defectos del software antes que se encuentre en un ambiente de producción, evitar las fallas de los sistemas informáticos, facilitar la fase de mantenimiento, evitar el deterioro temprano del software, entre otros; contribuyendo de esta manera al cumplimiento de los objetivos institucionales de la cooperativa.

1.5. Objetivos de la investigación

1.5.1. Objetivo General

Elaborar un marco de trabajo para la gestión de pruebas de software y validar su impacto en el proceso de evaluación del software, de la Dirección de Tecnologías de la Información de la COAC Atuntaqui Ltda.

1.5.2. Objetivos Específicos

- Analizar el proceso de evaluación de software del Área de Desarrollo de Software de la COAC Atuntaqui Ltda., mediante la recopilación de información y elaboración de encuestas y entrevistas.
- Desarrollar un marco de trabajo para la gestión de pruebas de software, basado en el estándar ISO/IEC/IEEE 29119.
- Aplicar el marco de trabajo para la gestión de pruebas de software, al proceso de evaluación de software, del Área de Desarrollo de la COAC Atuntaqui Ltda.

- Determinar el impacto obtenido en el proceso de evaluación del software, con la implementación del marco de trabajo para la gestión de pruebas.

1.6. Proposición

La implementación de un marco de trabajo para la gestión de pruebas de software mejorará el proceso de evaluación del software en el Departamento de Tecnologías de la Información de la COAC Atuntaqui Ltda.

CAPÍTULO 2. MARCO REFERENCIAL

2.1. Pruebas de Software

Las pruebas de software son el proceso de análisis de un sistema, o componente de un sistema, para detectar las diferencias entre el comportamiento especificado (requerido) y el observado (existente) (Bruegge & Dutoit, 2002, p. 328).

La prueba es un conjunto de actividades que pueden planearse por adelantado y realizarse de manera sistemática. Por esta razón, durante el proceso de desarrollo de software, debe definirse una plantilla para la prueba del software, un conjunto de pasos que incluye métodos de prueba y técnicas de diseño de casos de prueba (Pressman, 2014, p. 384).

Al final de la ejecución de las pruebas, se contrastan los resultados obtenidos contra los resultados esperados, con el fin de encontrar errores, anomalías o información de atributos no funcionales del programa (Sommerville, 2011, p. 206).

En un análisis rápido, realizado por Roger Pressman, sobre la fase de pruebas de software, se cuestiona: ¿Qué es? – Es la fase donde se prueba el software para descubrir errores que se cometieron de manera inadvertida conforme se diseñó y construyó; ¿Quién lo hace? – El gerente de proyecto, los ingenieros de software y los especialistas en pruebas desarrollan una estrategia para probar el software; ¿Por qué es importante? – Si se realiza sin orden, se desperdicia tiempo, se emplea esfuerzo innecesario y es posible que algunos errores pasen desapercibidos; ¿Cómo me aseguro de que lo hice bien? – Un plan de prueba y procedimientos efectivos conducirán a la construcción ordenada de software y al descubrimiento de errores en cada etapa del proceso de construcción (Pressman, 2014, p. 383).

Según SWEBOK[6], “Es una actividad realizada para evaluar la calidad del producto y mejorarla, identificando defectos y problemas” (SWEBOK, 2014, p. 174).

[6] Software Engineering Body of Knowledge

La *International Software Testing Qualifications Board* (ISTQB) [Junta Internacional de Calificación de Pruebas de Software] define las pruebas de software, como el proceso que consiste en todas las actividades del ciclo de vida, tanto estáticas como dinámicas, relacionadas con la planificación, preparación y evaluación de productos de software y productos relacionados con el trabajo para determinar que cumplen los requisitos especificados, demostrar que son aptos para el propósito y para detectar defectos (ISTQB, 2013).

De acuerdo con Sommerville, el proceso de prueba tiene dos metas distintas:

1. ***Demostrar al desarrollador y al cliente, que el software cumple con los requisitos***. Para el software personalizado, esto significa que en el documento de requisitos debe haber, por lo menos, una prueba por cada requisito. Para los productos de software genérico, esto quiere decir que tiene que haber pruebas para todas las características del sistema, junto con combinaciones de dichas características que se incorporarán en la liberación del producto.
2. ***Encontrar situaciones donde el comportamiento del software sea incorrecto, indeseable o no esté de acuerdo con su especificación.*** Tales situaciones son consecuencia de defectos del software. La prueba de defectos tiene la finalidad de erradicar el comportamiento indeseable del sistema, como caídas del sistema, interacciones indeseadas con otros sistemas, cálculos incorrectos y corrupción de datos.

2.1.1. Conceptos de las pruebas

Las pruebas de software utilizan ciertos términos que, aunque parezcan ser similares, corresponden a conceptos totalmente distintos, por ejemplo: error, falla y defecto; estos conceptos están relacionados entre sí, pero tienen diferentes significados. Para comprender mejor estos conceptos, repasamos las definiciones de Bruegge y Dutoit, quienes dicen que:

- Un ***error*** de programación es la incrustación de sentencias de código incorrectas en la implementación del sistema.

- Un ***defecto*** es un desperfecto de un componente o sistema, que puede causar que el componente o sistema falle al desempeñar las funciones requeridas.
- Una ***falla*** es una manifestación física o funcional de un defecto.

En resumen, un error introduce un defecto en el software, que a su vez causa un fallo al momento de ejecutar las pruebas.

2.1.2. Causas de errores en el software

En un análisis realizado por González, establece que las causas para la presencia de errores en el software, se ubica dentro de un conjunto de posibilidades:

1. ***El usuario ejecuta código no testeado.*** Esto puede deberse a posibles restricciones de tiempo en el desarrollo de software; ocasionando que las pruebas se consideren secundarias, se planteen de forma errónea, no se concluyan y/o no cubran todos los escenarios necesarios para garantizar un mínimo de calidad.
2. ***El orden en el que se ejecutaron las sentencias no es el mismo que el orden que se ha probado.*** Es probable que no se hayan cubierto todas las posibilidades posibles. Sistemas recursivos, inteligencia artificial, interfaces de usuario complejos, etc.
3. ***El usuario ha introducido una combinación de valores de entrada no testeados.*** Por definición es inviable testear todas las posibilidades de entrada de un sistema. Valorando únicamente una entrada numérica nos encontramos ante infinitos valores posibles por lo que asegurar de forma precisa que un sistema funcionará correctamente con todas las posibles entradas es, a efectos prácticos, imposible.
4. ***El entorno operativo del usuario nunca ha sido testeado.*** Es probable que una aplicación concreta desarrollada en un lenguaje concreto y sobre unas librerías concretas no funcione correctamente en todos los sistemas operativos y en todas sus versiones.

2.1.3. Principios de las pruebas

En un análisis sobre el proceso de pruebas de calidad de software, realizado por el MSc. Julián Mera, docente e investigador de la Facultad de Ingeniería de Sistemas de la Universidad Cooperativa de Colombia, resume los siete principios fundamentales de las pruebas de software:

1. ***Las pruebas demuestran la presencia de defectos.*** Las pruebas son unas herramientas que permiten identificar la presencia de defectos; sin embargo, no garantizan que no haya defectos ocultos en el software, y el hecho de que no se identifiquen defectos, no es una evidencia de que el software esté totalmente correcto.
2. ***Las pruebas exhaustivas no existen.*** Probar todo un aplicativo de extremo a extremo con todas las entradas de datos y condiciones es algo imposible; en lugar de tratar de conseguir ese tipo de pruebas, se debe realizar un análisis de los riesgos para establecer prioridades y tomar decisiones adecuadas en la utilización de talento humano y recursos, centralizando los esfuerzos en las pruebas.
3. ***Pruebas tempranas.*** Identificar los defectos en etapas tempranas, cuanto más rápido se identifiquen los defectos, más se ahorrará la empresa en todo tipo de recursos.
4. ***Agrupación de defectos.*** Por lo general, la mayoría de los fallos operativos se concentran en un número reducido de módulos.
5. ***Paradoja del pesticida.*** Si se repite la misma prueba una y otra vez, la misma serie de casos de prueba dejará de encontrar nuevos defectos; es importante que los casos de prueba se revisen periódicamente y escribir nuevos casos de prueba con el objetivo de encontrar más defectos.
6. ***Las pruebas dependen del contexto.*** Las pruebas dependerán del contexto en el cual se ejecuten; por ejemplo, las que sean para un sistema crítico de seguridad, como el

financiero, requieren de un mayor número de pruebas, en comparación con otras aplicaciones de un nivel de complejidad menor o menos críticos.

7. ***Falacia de ausencia de errores.*** La detección y corrección de los defectos no sirven de nada si el sistema no cumple con los requisitos o necesidades del usuario.

Estos principios son una guía para tener en cuenta en todo proceso de pruebas de software; el equipo de trabajo debe procurar que se cumplan y tenerlos presentes en cada actuación y ejecución de herramientas o técnicas, para brindar aseguramiento a la calidad de un producto software.

2.2. Verificación y validación

Las pruebas de software es un elemento de un tema más amplio, que usualmente se conoce como verificación y validación (V&V) del software. Aunque ambas no son lo mismo, se confunden con frecuencia (Sommerville, 2011, p. 206). Barry Boehm, pionero de la ingeniería de software, expresó de manera breve la diferencia entre las dos (Boehm, 1979):

- "Validación: ¿construimos el producto correcto?".
- "Verificación: ¿construimos bien el producto?".

La validación es un conjunto diferente de tareas, que aseguran que el software que se construye sigue los requisitos. La verificación se refiere al conjunto de tareas que garantizan que el software implementa correctamente una función (Pressman, 2014, p. 384).

Según la norma ISO/IEC/IEEE 24765: 2010, se debe tener en cuenta lo siguiente:

> *"Verificación: Proceso de evaluación de un sistema o componente para determinar si un producto de una determinada fase de desarrollo satisface las condiciones impuestas al inicio de la fase".*
>
> *"Validación: Proceso de evaluación de un sistema o componente durante o al final del proceso de desarrollo para determinar cuándo se satisfacen los requisitos especificados".*

2.3. Modelo en V

Existen diferentes modelos de desarrollo de software, como: el modelo en cascada, el modelo general en V o los modelos ágiles de programación. Las pruebas se encuentran dentro de estos modelos de formas diferentes. Por ejemplo, dentro del modelo en cascada las pruebas se ejecutan al final del proyecto, mientras que en el modelo en V las pruebas se realizan en los diferentes niveles de desarrollo.

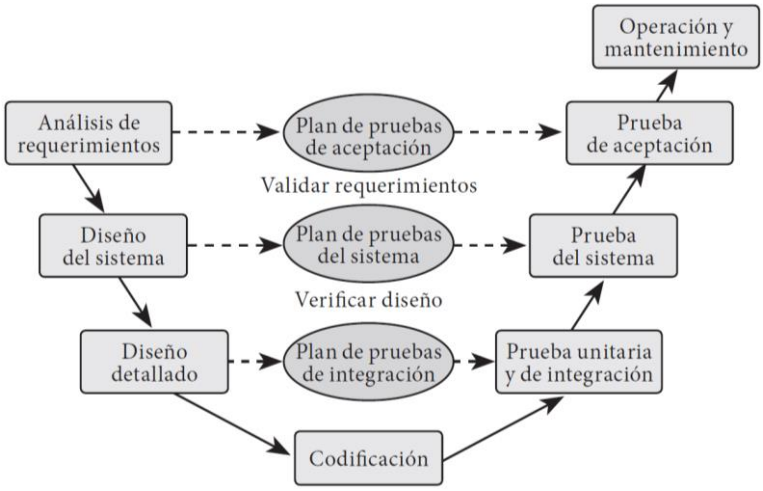


Figura 2.1. Modelo en V de pruebas de software (Lawrence & Atlee, 2009).

En el modelo en V se representan dos secuencias de fases, la primera corresponde a las fases de desarrollo del proyecto y la segunda a las fases de pruebas del proyecto. Las fases de un mismo nivel se pueden realizar en paralelo. Para cada nivel de desarrollo se define un nivel de prueba. Dentro de cada nivel de prueba, el probador debe asegurarse de que los resultados cumplan con la verificación y validación del software.

En un proceso de pruebas formal, suelen confundirse con mucha facilidad, los niveles de pruebas con los tipos de prueba, y a pesar de que se encuentren íntimamente relacionadas, tienen connotaciones diferentes en el proceso. Para entender un poco más, vamos a partir del hecho de que las pruebas pueden ejecutarse en cualquier punto del proceso de desarrollo de software, y es aquí donde los niveles de prueba nos permiten entender con claridad los diferentes puntos o etapas en donde pueden ejecutarse ciertos tipos de prueba (Zapata, 2013).

2.3.1. Pruebas de unidad

Las pruebas de unidad verifican el funcionamiento aislado de partes del software que se pueden probar independientemente. Dependiendo del contexto, estas podrían ser subprogramas individuales o un componente más grande formado por unidades muy relacionadas. Normalmente, las pruebas de unidad se realizan con acceso al código fuente y con el soporte de herramientas de depuración, pudiendo implicar a los programadores que escribieron el código (SWEBOK, 2014).

Las pruebas de unidad tienen por objeto localizar defectos y comprobar el funcionamiento de módulos software, programas, objetos, clases, etc., que puedan probarse por separado; es decir, se pueden realizar de manera independiente al resto del sistema en función del contexto (ISTQB, 2013).

La prueba de unidad enfoca los esfuerzos de verificación en la unidad más pequeña del diseño de software: el componente o módulo de software. Este tipo de pruebas puede realizarse en paralelo para múltiples componentes y por lo general se consideran como adjuntas al paso de codificación (Pressman, 2014).

De acuerdo con Bruegge & Dutoit, existen tres motivaciones para la realización de pruebas unitarias:

1. Reducen la complejidad de las actividades, permitiendo enfocarse en unidades más pequeñas del sistema.
2. Facilita la detección y corrección de defectos.
3. Permite probar cada uno de los componentes de forma independiente.

Sommerville, manifiesta que cuando se prueba un objeto, se tiene que diseñar las pruebas necesarias para cubrir todas las características del objeto (Sommerville, 2011).

- Probar las operaciones asociadas con el objeto;
- Establecer y verificar el valor de todos los atributos relacionados con el objeto;

- Poner el objeto en todos los estados posibles, es decir, simular todos los eventos que causen un cambio de estado del objeto.

2.3.2. Pruebas de integración

Las pruebas de integración son una técnica sistemática para construir la arquitectura del software, mientras se llevan a cabo pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar los componentes probados de manera individual y construir una estructura de programa que se haya dictado por diseño (Pressman, 2014).

Las pruebas de integración detectan defectos que no se han descubierto durante las pruebas unitarias, enfocándose en pequeños grupos de componentes. Dos o más componentes se integran y prueban, y después de que las pruebas ya no detectan ningún nuevo defecto se añaden componentes adicionales (Bruegge, 2002).

Bruegge, Dutoit & Pressman definen cuatro tipos de estrategias para la implementación de las pruebas de integración:

La ***prueba de integración de gran explosión (big bang)*** sucede cuando todos los componentes se combinan y se prueban como un todo. La corrección de errores se dificulta, pues el aislamiento de las causas se complica por la extensión de todo el programa. Una vez corregidos estos errores, otros aparecen y el proceso se repite en un bucle aparentemente interminable.

La ***prueba de integración descendente*** es un enfoque incremental a la construcción de la arquitectura de software. Primero se prueban los componentes de la capa superior y luego se integran los componentes de la siguiente capa hacia abajo. Cuando se han probado juntos todos los componentes de la nueva capa se selecciona la siguiente capa. Los *stubs* de prueba se usan para simular a los componentes de las capas inferiores que todavía no se han integrado.

La ***prueba de integración ascendente*** comienza la construcción y las pruebas con los componentes de los niveles inferiores de la estructura del programa, luego integra los

componentes de la siguiente capa superior. Esto se repite hasta que se combinan todos los componentes de todas las capas. Puesto que los componentes se integran de abajo hacia arriba, la funcionalidad que proporcionan los componentes subordinados en determinado nivel siempre está disponible y se elimina la necesidad de *stubs*.

La ***prueba de integración combinada*** aplica las estrategias descendente y ascendente, tratando de usar lo mejor de ambas. Durante este tipo de pruebas, el sistema se descompone en subsistemas de tres capas: la capa de destino de la prueba, una capa superior y una capa inferior; usando la capa destino como foco de atención, ahora pueden realizarse en paralelo las pruebas descendentes y ascendentes de la capa destino.

2.3.3. Pruebas del sistema

Las pruebas unitarias y de integración se enfocan en encontrar defectos en componentes individuales y en las interfaces entre los componentes. Una vez que se han integrado todos los componentes, las pruebas del sistema aseguran que el sistema completo se apegue a los requisitos funcionales y no funcionales del sistema (Bruegge, 2002).

Las pruebas de sistema demuestran que los componentes son compatibles, que interactúan correctamente y que transfieren los datos correctos en el momento adecuado, a través de sus interfaces (Sommerville, 2011).

Para la mayoría de los sistemas es difícil saber cuántas pruebas de sistemas son esenciales y cuándo hay que dejar de hacer pruebas. Las pruebas exhaustivas, donde se pone a prueba cada secuencia posible de ejecución del programa, son imposibles. Por lo tanto, las pruebas deben basarse en un subconjunto de probables casos de prueba. De manera ideal, para elegir este subconjunto, las compañías de software cuentan con políticas, las cuales pueden basarse en políticas de prueba generales, como una política de que todos los enunciados del programa se ejecuten al menos una vez (Sommerville, 2011).

2.3.4. Pruebas de aceptación

A la hora de realizar estas pruebas, el producto está listo para implantarse en el entorno del cliente. El usuario debe ser el que realice las pruebas, ayudado por personas del equipo de pruebas, siendo deseable, que sea el mismo usuario quien aporte los casos de prueba.

Estas pruebas se caracterizan por:

- Participación del usuario, quien debe ejecutar los casos de prueba ayudado por los miembros del equipo de pruebas.
- Corresponden a la fase final del proceso de desarrollo de software.
- Están enfocadas a probar los requisitos de usuario; o mejor dicho, a demostrar que no se cumplen los requisitos, los criterios de aceptación o el contrato. Si no se consigue demostrar esto, el cliente deberá aceptar el producto.

Es muy recomendable que las pruebas de aceptación se realicen en el entorno donde se va a usar el sistema, incluido el personal. En caso de un producto de interés general, se realizan pruebas con varios usuarios que reportarán sus valoraciones sobre el producto. Para la generación de casos de prueba de aceptación, se utilizan técnicas de caja negra.

2.4. Técnicas de pruebas de software

Uno de los objetivos de las pruebas es detectar tantas fallas como sea posible. Se han desarrollado muchas técnicas para hacer esto. Estas técnicas intentan "romper" un programa siendo lo más sistemáticas posible, identificando las entradas que producirán comportamientos representativos del programa; por ejemplo, al considerar subclases del dominio de entrada, escenarios, estados y flujos de datos. A veces estas técnicas se clasifican como caja blanca, si las pruebas se basan en información sobre cómo se ha diseñado o codificado el software; o como caja negra si los casos de prueba se basan únicamente en las entradas y salidas del software (SWEBOK, 2014).

2.4.1. Revisiones

Al igual que las pruebas de software, el proceso de verificación y validación implicaría inspecciones y revisiones del software. Estas últimas analizan y comprueban los requisitos del sistema, los modelos de diseño, el código fuente del programa e incluso las pruebas propuestas del sistema. Estas son las llamadas técnicas “estáticas”, donde no es necesario ejecutar el software para verificarlo (Sommerville, 2011).

Hay tres ventajas en la inspección del software sobre las pruebas.

1. Durante las pruebas, los errores pueden ocultar otras fallas. Puesto que la inspección es un proceso estático, no hay que preocuparse por las interacciones entre errores, en consecuencia, una sola sesión de inspección descubriría muchos errores en un sistema.
2. Las versiones incompletas de un sistema se pueden inspeccionar sin costos adicionales. Si un programa está incompleto, entonces es necesario desarrollar equipos de prueba especializados para poner a prueba las partes disponibles, esto genera costos para el desarrollo del sistema.
3. Además de buscar defectos de programa, una inspección puede considerar también atributos más amplios de calidad de un programa, como el cumplimiento con estándares, la portabilidad y la mantenibilidad.

Sin embargo, las inspecciones no sustituyen las pruebas de software, ya que no son eficaces para descubrir defectos que surjan por interacciones inesperadas entre diferentes partes de un programa, problemas de temporización o dificultades con el rendimiento del sistema.

2.4.2. Pruebas de caja blanca

Las pruebas de caja blanca también suelen ser llamadas estructurales o de cobertura lógica. Este método se centra en cómo diseñar los casos de prueba, atendiendo al comportamiento interno y la estructura del programa. Se examina así la lógica interna del programa sin considerar los aspectos de rendimiento.

El objetivo de la técnica es diseñar casos de prueba para que se ejecuten, al menos una vez, todas las sentencias del programa y todas las condiciones tanto en su vertiente verdadera como falsa.

Aunque las pruebas de caja blanca son aplicables a varios niveles (unidad, integración y sistema), habitualmente se aplican a las unidades de software. Su cometido es comprobar los flujos de ejecución dentro de cada unidad, pero también pueden testear los flujos entre unidades durante la integración, e incluso entre subsistemas durante las pruebas de sistema.

Puede ser impracticable realizar una prueba exhaustiva de todos los caminos de un programa. Por ello se han definido distintos criterios de cobertura lógica, que permiten decidir qué sentencias o caminos se deben examinar con los casos de prueba. Estos criterios son:

- ***Cobertura de Sentencias:*** Se escriben casos de prueba suficientes para que cada sentencia en el programa se ejecute, al menos una vez.
- ***Cobertura de Decisión:*** Se escriben casos de prueba suficientes para que cada decisión en el programa se ejecute una vez con resultado verdadero y otra con el falso.
- ***Cobertura de Condiciones:*** Se escriben casos de prueba suficientes para que cada condición en una decisión tenga una vez resultado verdadero y otra falso.
- ***Cobertura Decisión/Condición:*** Se escriben casos de prueba suficientes para que cada condición en una decisión tome todas las posibles salidas, al menos una vez, y cada decisión tome todas las posibles salidas, al menos una vez.
- ***Cobertura de Condición Múltiple:*** Se escriben casos de prueba suficientes para que todas las combinaciones posibles de resultados de cada condición se invoquen al menos una vez.
- ***Cobertura de Caminos:*** Se escriben casos de prueba suficientes para que se ejecuten todos los caminos de un programa. Entendiendo camino como una secuencia de sentencias encadenadas desde la entrada del programa hasta su salida.

1. Cobertura de Caminos

La aplicación de este criterio de cobertura asegura que los casos de prueba diseñados permitan que todas las sentencias del programa sean ejecutadas al menos una vez y que las condiciones sean probadas tanto para su valor verdadero como falso.

Una de las técnicas empleadas para aplicar este criterio de cobertura es la Prueba del Camino Básico. Esta técnica se basa en obtener una medida de la complejidad del diseño procedimental de un programa (o de la lógica del programa). Esta medida es la complejidad ciclomática de McCabe, y representa un límite superior para el número de casos de prueba que se deben realizar para asegurar que se ejecuta cada camino del programa.

Los pasos para aplicar esta técnica son:

- Representar el programa en un grafo de flujo
- Calcular la complejidad ciclomática
- Determinar el conjunto básico de caminos independientes
- Derivar los casos de prueba que fuerzan la ejecución de cada camino.

A continuación, se detallan cada uno de estos pasos.

El ***grafo de flujo*** se utiliza para representar flujo de control lógico de un programa. Para ello se utilizan los tres elementos siguientes:

- ***Nodos:*** representan cero, una o varias sentencias en secuencia. Cada nodo comprende como máximo una sentencia de decisión (bifurcación). Cuando en una condición aparecen uno o más operadores lógicos (AND, OR, XOR, ...) se crea un nodo distinto por cada una de las condiciones simples. Cada nodo generado de esta forma se denomina *nodo predicado*.
- ***Aristas:*** líneas que unen dos nodos.
- ***Regiones:*** áreas delimitadas por aristas y nodos. Cuando se contabilizan las regiones de un programa debe incluirse el área externa como una región más.

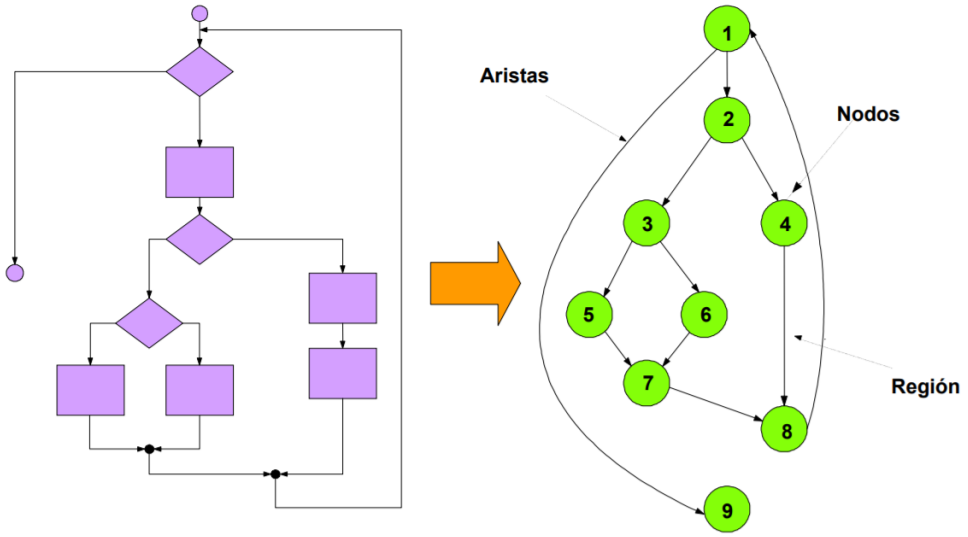


Figura 2.2. Grafo de flujo correspondiente a un diagrama de módulos (Sánchez, 2015).

La ***complejidad ciclomática*** es una métrica del software que proporciona una medida cuantitativa de la complejidad lógica de un programa. En el contexto del método de prueba del camino básico, el valor de la complejidad ciclomática define el número de caminos independientes de dicho programa, y por lo tanto, el número de casos de prueba a realizar.

Existen varias formas de calcular la complejidad ciclomática de un programa a partir de un grafo de flujo:

1. El número de regiones del grafo coincide con la complejidad ciclomática, V(G).
2. La complejidad ciclomática, V(G), de un grafo de flujo G se define como

   V(G) = Aristas – Nodos + 2
3. La complejidad ciclomática, V(G), de un grafo de flujo G se define como

   V(G) = Nodos Predicado + 1

La Figura 3 representa, por ejemplo, las cuatro regiones del grafo de flujo, obteniéndose así la complejidad ciclomática de cuatro. Análogamente se puede calcular el número de aristas y nodos predicados para confirmar la complejidad ciclomática. Así:

V(G) = Número de regiones = 4

V(G) = Aristas – Nodos + 2 = 11-9 + 2 = 4

V(G) = Nodos Predicado + 1 = 3 +1 = 4

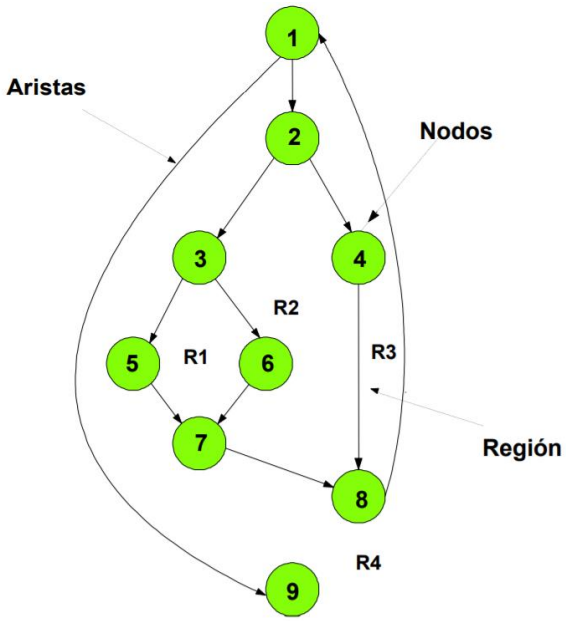


Figura 2.3. Número de regiones del grafo de flujo (Sánchez, 2015).

Esta complejidad ciclomática determina el número de casos de prueba que deben ejecutarse para garantizar que todas las sentencias de un programa se han ejecutado al menos una vez, y que cada condición se habrá ejecutado en sus vertientes verdadera y falsa.

Un ***camino independiente*** es cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una condición, respecto a los caminos existentes. En términos del diagrama de flujo, un camino independiente está constituido por lo menos por una arista que no haya sido recorrida con anterioridad a la definición del camino. En la identificación de los distintos caminos de un programa para probar, se debe tener en cuenta que cada nuevo camino debe tener el mínimo número de sentencias nuevas o condiciones nuevas respecto a los que ya existen. De esta manera se intenta que el proceso de depuración sea más sencillo.

El conjunto de caminos independientes de un grafo no es único. No obstante, a continuación, se muestran algunas heurísticas para identificar dichos caminos:

1. Elegir un camino principal que represente una función válida, que no sea un tratamiento de error. Debe intentar elegirse el camino que atraviese el máximo número de decisiones en el grafo.

2. Identificar el segundo camino mediante la localización de la primera decisión en el camino de la línea básica, alternando su resultado mientras se mantiene el máximo número de decisiones originales del camino inicial.
3. Identificar un tercer camino, colocando la primera decisión en su valor original, a la vez que se altera la segunda decisión del camino básico, mientras se intenta mantener el resto de las decisiones originales.
4. Continuar el proceso hasta haber conseguido tratar todas las decisiones, intentando mantener como en su origen el resto de ellas.

Este método permite obtener V(G) caminos independientes cubriendo el criterio de cobertura de decisión y sentencia.

Así, por ejemplo, para el grafo de la Figura 2.3, los cuatro posibles caminos independientes generados serían:

Camino 1: 1-10

Camino 2: 1-2-4-8-1-9

Camino 3: 1-2-3-5-7-8-1-9

Camino 4: 1-2-5-6-7-8-1-9

Estos cuatro caminos constituyen el camino básico para el grafo de flujo correspondiente.

El último paso es construir los casos de prueba que fuerzan la ejecución de cada camino. Una forma de representar el conjunto de casos de prueba se muestra en la siguiente tabla.

Tabla 2.1. *Generación de casos de prueba para cada camino.*

Número de camino	Caso de prueba	Resultado esperado
1-10	Caso Prueba 1	Resultado 1
1-2-4-8-1-9	Caso Prueba 2	Resultado 2
1-2-3-5-7-8-1-9	Caso Prueba 3	Resultado 3
1-2-5-6-7-8-1-9	Caso Prueba 4	Resultado 4

Fuente: (Sánchez, 2015)

2.4.3. Pruebas de caja negra

Las pruebas de caja negra, también denominadas por el ISTQB como técnicas basadas en especificación, son una forma de derivar y seleccionar condiciones, datos y casos de prueba a partir de la documentación de requisitos del sistema. Las pruebas de caja negra tienen en cuenta únicamente las entradas y salidas del sistema o componente que se va a probar, es decir, ignoran el mecanismo interno del software. Consideran el comportamiento del software desde el punto de vista de un observador externo.

Aplicar pruebas de caja negra sobre un sistema:

- Posibilita de forma rápida y sencilla validar un sistema.
- Es muy útil cuando un sistema contiene grandes cantidades de código.
- Informa sobre si un componente o sistema no se comporta según su especificación, es decir, si los resultados para unos datos concretos no son los esperados.
- No es necesario conocer el código, por lo tanto, pueden ser realizadas por los usuarios del sistema y no sólo por los programadores.

Entre sus principales deficiencias se encuentran:

- El escaso control sobre la cobertura y la imposibilidad práctica de probar todas las entradas de un sistema.
- Es necesario seleccionar un subconjunto finito de entradas que validen el producto, esta selección no siempre es sencilla, por lo que el nivel de dificultad en el diseño de este tipo de pruebas puede aumentar considerablemente.

Al igual que ocurre con las técnicas de caja blanca, para confeccionar los casos de prueba de caja negra existen distintos criterios. A continuación, se describen las principales técnicas o métodos de pruebas de caja negra, tomando en cuenta las definiciones del Syllabus del ISTQB.

1. **Partición de equivalencias**

Consiste en clasificar los datos de entradas del sistema en grupos que presentan un comportamiento similar, por lo cual serán procesados de la misma forma. Se pueden definir particiones tanto para datos válidos, como no válidos (que deben ser rechazados por el sistema).

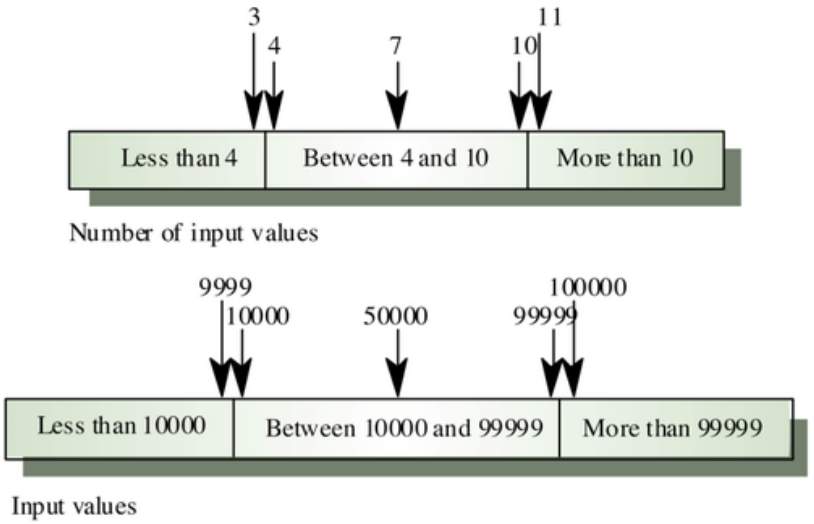


Figura 2.4. Ejemplo de partición de equivalencias. (Web Labs, 2013).

Las particiones también pueden definirse en función de las salidas de datos, valores internos, valores relacionados antes o después de ciertos eventos, y también para los valores que reciben las interfaces.

A partir de allí se definen pruebas para cubrir todos o parte de las particiones de datos válidos y datos inválidos. Es aplicable a entradas de datos realizadas por personas o vía interfaces con otros sistemas.

2. **Análisis de valores borde**

Parte del principio que el comportamiento al borde de una partición de datos tiene mayores probabilidades de presentar errores (bugs). Los valores máximos y mínimos de una partición son sus valores bordes. Se aplican tanto para datos válidos como inválidos.

Al incluirlas en el diseño de casos de prueba, se define una prueba por cada valor borde. La capacidad de identificar defectos de esta técnica es alta, se pueden revisar las especificaciones funcionales para identificar datos interesantes.

3. Tablas de decisión

Las tablas de decisión son una herramienta útil para documentar reglas de negocio de alta complejidad que el sistema debe cumplir. Se crean a partir del análisis de la especificación funcional y la identificación de las reglas de negocio. Las condiciones de entrada y acciones se expresan a menudo en términos de verdadero o falso.

La tabla de decisión contiene las condiciones desencadenantes, que son la combinación de valores de verdadero o falso para cada entrada de datos, así como la acción que resulta de cada combinación. Cada columna de la tabla corresponde con una regla de negocio que representa la combinación de condiciones y las acciones que resultan.

Tabla 2.2. *Ejemplo de tabla de decisión.*

Condiciones	**TC1**	**TC2**	**TC3**	**TC4**
Solicitud login	0	1	1	1
Validar usuario	x	0	1	1
Validar password	x	x	0	1
Acciones				
Recuperar contraseña	0	1	1	0
Activar casilla usuario	0	1	1	0
Activar casilla password	0	0	1	0
Ingresar en área privilegiada	0	0	0	1

Fuente: (Sánchez, 2015)

4. Transición entre estados

Un sistema puede presentar diferentes comportamientos según su estado actual o eventos previos. Este aspecto del sistema se puede representar en un diagrama de transición entre estados.

El diagrama de estados permite al Tester visualizar los estados, transiciones, entradas de datos o eventos que las desencadenan y las acciones que pueden resultar.

Una tabla de estados muestra las relaciones entre los estados y las entradas de datos. Puede ayudar a identificar posibles transacciones inválidas.

5. Pruebas de casos de uso

Los casos de uso describen las interacciones entre actores (usuarios o sistemas) que producen un resultado que agrega algún valor. A partir de estos se pueden derivar casos de prueba. Tienen precondiciones que deben cumplirse para que estos funcionen de forma exitosa.

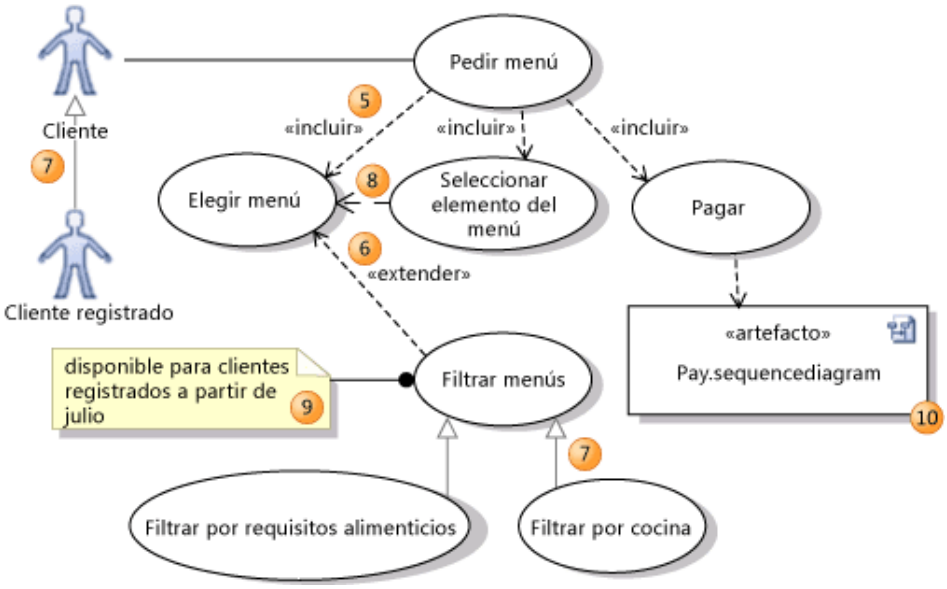


Figura 2.5. Ejemplo de pruebas de casos de uso.

Fuente: (Microsoft Developer Network, 2014).

Los casos de uso terminan con post condiciones, que son resultados observables y el estado del sistema después de la ejecución. Son útiles para definir las pruebas de aceptación, en las que participa el usuario o cliente.

6. Pruebas de historias de usuario

En metodologías ágiles como Scrum, los requisitos de usuario son preparados en forma de historias de usuario. La historia de usuario describe una funcionalidad (o parte de ella), que puede ser desarrollada y probada en una sola iteración, describe la funcionalidad a implementar, requisitos no funcionales y los criterios de aceptación.

La cobertura mínima de pruebas para una historia de usuario está compuesta por los criterios de aceptación. Por ende, los casos de prueba se derivan de estos criterios de aceptación.

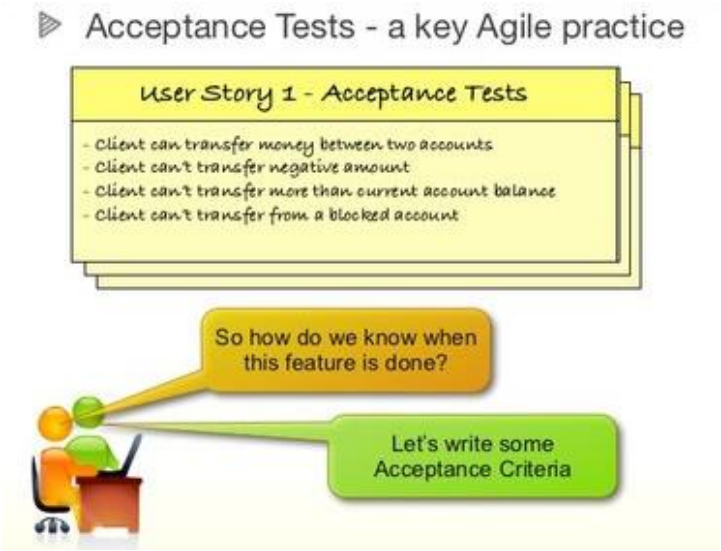


Figura 2.6. Ejemplo de pruebas de historias de usuario.

Fuente: (Wakaleo Consulting, 2013)

2.5. ISO/IEC/IEEE 29119: Software Testing

Bedini, define las pruebas de software como: "Proceso planificado, basado en estándares previamente establecidos…", por lo tanto, resulta indispensable estudiar un estándar internacional, especializado en el proceso de pruebas de software.

La ISO/IEC/IEEE 29119 es un conjunto de estándares acordados internacionalmente para las pruebas de software, que se pueden utilizar dentro de cualquier ciclo de vida u organización de desarrollo de software. Son los únicos estándares reconocidos y acordados internacionalmente para las pruebas de software, proporcionan un enfoque de alta calidad para las pruebas y se pueden comunicar en todo el mundo (Veenendaal, 2016).

El objetivo de la norma ISO/IEC/IEEE 29119 es definir un solo estándar de pruebas, que cubra todo el ciclo de vida del software, incluyendo los aspectos relacionados con la organización, gestión, diseño y ejecución de las pruebas; y unifique estándares anteriores, como: el BS 7925-2 para Pruebas de Software de Componentes y el IEEE 829 para Documentación de las Pruebas (Reid, 2017).

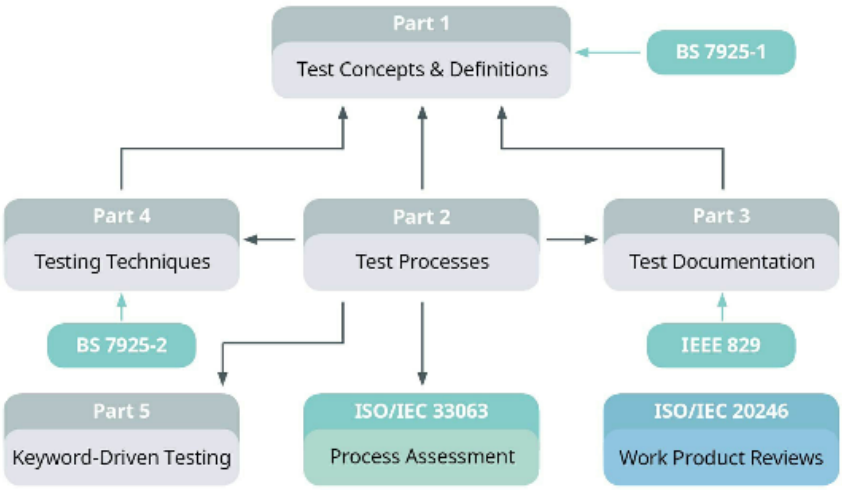


Figura 2.7. Estructura del estándar ISO/IEC/IEEE 29119 (ISO/IEC/IEEE 29119:1, 2013)

El estándar se estructura de cinco partes:

- ISO/IEC/IEEE 29119-1: Conceptos y definiciones
- ISO/IEC/IEEE 29119-2: Procesos de prueba

- ISO/IEC/IEEE 29119-3: Documentación de las pruebas
- ISO/IEC/IEEE 29119-4: Técnicas de pruebas
- ISO/IEC/IEEE 29119-5: Pruebas guiadas por palabra clave

El modelo utilizado como base para el nuevo conjunto de estándares se puede ver en la Figura 2.7, con los procesos de prueba en el centro. La documentación de las pruebas se produce al ejecutar los procesos de prueba; por lo tanto, la documentación de las pruebas describe los resultados de los procesos de prueba. El requisito de utilizar técnicas para diseñar los casos de prueba lo especifican los procesos de prueba en la Parte 2, mientras que las diferentes técnicas de diseño de las pruebas se definen por separado en la Parte 4. Los conceptos generales y la terminología común utilizada por las otras partes se definen en la Parte 1.

Poco después de comenzar el trabajo en las primeras cuatro partes, en la evaluación de los procesos de prueba, la ISO/IEC 33063 fue creada por una propuesta separada y esta fue seguida por el desarrollo de la Parte 5, pruebas guiadas por palabra clave. Las tres primeras partes se publicaron en 2013, la Parte 4 en 2015 y la Parte 5 se publicó en 2016. Posteriormente, WG26 desarrolló una norma separada sobre revisiones (ISO/IEC 20246) para complementar las pruebas dinámicas cubiertas por las otras normas, que fue publicado en febrero de 2017 (GIIS, 2017).

Los estándares ISO/IEC/IEEE 29119 están destinados a soportar las pruebas en una amplia variedad de dominios de aplicación, para varios niveles de criticidad y en cualquier ciclo de vida; por lo tanto, los estándares son genéricos y se pueden aplicar a:

- Todos los dominios industriales.
- Sistemas críticos para la seguridad y no críticos para la seguridad.
- Pruebas exploratorias y pruebas guiadas.
- Cualquier modelo de ciclo de vida, que incluye tradicionales y ágiles.
- Pruebas automatizadas.

2.5.1. ISO/IEC/IEEE 29119 - 1: Conceptos y definiciones

La Parte 1 presenta el conjunto de estándares e incluye las definiciones comunes de todas las partes, así como también describe los conceptos elementales en los que se basan las pruebas. Esta parte explica que cubren las otras partes y describe cómo pueden usarse los estándares para diferentes modelos de ciclo de vida.

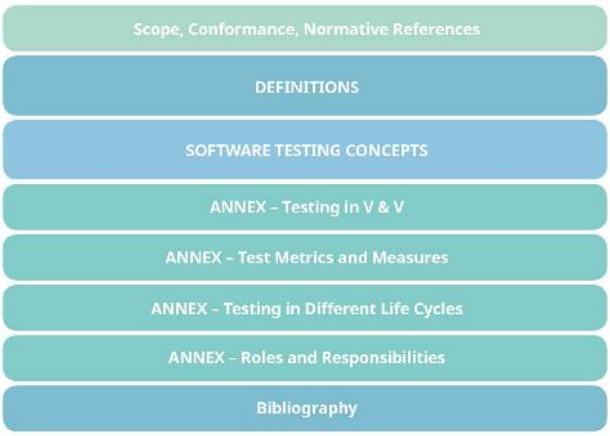


Figura 2.8. ISO/IEC/IEEE 29119 - Parte1: Conceptos y definiciones.

Fuente: (ISO/IEC/IEEE 29119-1, 2013)

El objetivo de esta parte es dar una visión general de los estándares y los conceptos generales de las pruebas de software; así como, proporcionar un vocabulario de términos sobre pruebas de software que cubra todo el ciclo de vida del software.

2.5.2. ISO/IEC/IEEE 29119 - 2: Procesos de prueba

La Parte 2 define los procesos de prueba utilizando un modelo de tres capas, como se muestra en la Figura 2.9.

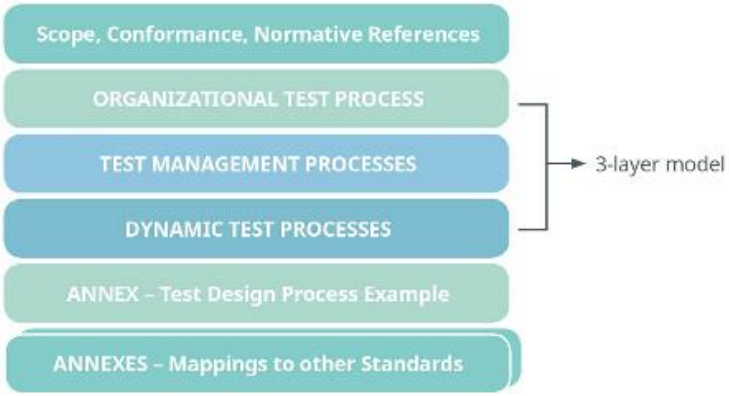


Figura 2.9. ISO/IEC/IEEE 29119 - Parte 2: Procesos de prueba (Reid, 2017).

La capa superior corresponde al proceso de prueba organizacional que se utiliza para generar y mantener las políticas y estrategias organizacionales para las pruebas. La capa intermedia incluye los procesos de gestión de pruebas para la planificación, supervisión, control y finalización de las pruebas. La capa inferior corresponde a los procesos de prueba dinámicos, por lo que el modelo general no incluye ningún proceso de prueba estático, como el análisis estático y las revisiones. El grupo de trabajo de pruebas de software ISO, WG26, ha desarrollado un estándar separado, ISO/IEC 20246: Revisiones de productos de trabajo, que define un proceso genérico para todas las formas de revisión.

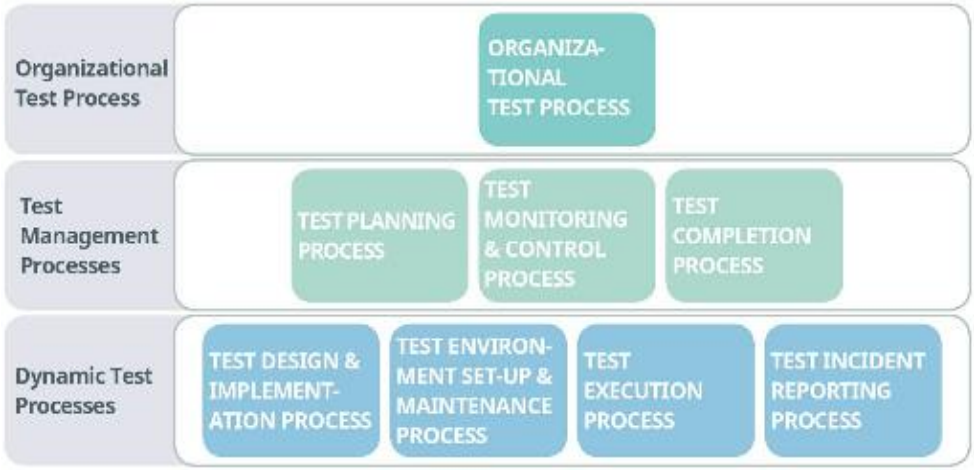


Figura 2.10. Procesos de pruebas definidos en el estándar ISO/IEC/IEEE 29119.

Fuente: (ISO/IEC/IEEE 29119-2, 2013)

La Figura 2.10 muestra el conjunto completo de los ocho procesos de prueba definidos en el estándar para pruebas de software.

Proceso de Prueba Organizacional, cuyo objetivo es crear y mantener las especificaciones de la organización para las pruebas (políticas, estrategias, procesos, procedimientos y otra normativa de la organización para las pruebas); se compone de un único proceso, el Proceso de Prueba Organizacional.

Procesos de Gestión de Pruebas, cuyo objetivo es planificar, programar, estimar, realizar el seguimiento, comunicar, controlar y cerrar las actividades de las pruebas; incluye los Procesos de Planificación, Seguimiento y Control, y Finalización de las Pruebas.

Procesos de Pruebas Dinámicas, cuyo objetivo es diseñar e implementar las pruebas, establecer y mantener el entorno de prueba, ejecutar las pruebas y en caso de incidentes, comunicarlos; incluye los procesos de Diseño e Implementación de las Pruebas, Establecimiento y Mantenimiento del Entorno de Pruebas, Ejecución de las Pruebas y Reporte de Incidentes de Prueba (Lerche-Jensen, 2015).

Cada proceso se compone de una o más actividades y cada una de estas actividades comprende una o más tareas, como se muestra en la Figura 2.11. Las tareas especifican las acciones requeridas o proporcionan las recomendaciones para realizar los procesos con un nivel de detalle más bajo.

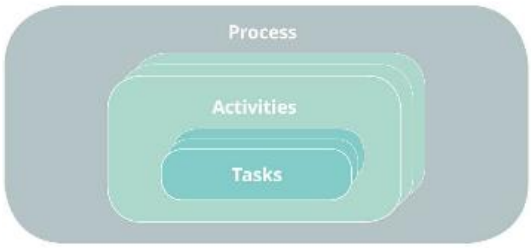


Figura 2.11. Estructura de un proceso de pruebas (Reid, 2017).

2.5.3. ISO/IEC/IEEE 29119 - 3: Documentación de las Pruebas

La Parte 3 proporciona plantillas con descripciones de los contenidos para cada uno de los principales tipos de documentos de las pruebas:

- Documentación organizacional de pruebas: Políticas organizacionales de prueba y estrategia organizacional de pruebas.
- Documentación de la gestión de pruebas: Plan de pruebas, informe de estado de las pruebas e informe de finalización de las pruebas.
- Documentación de pruebas dinámicas: Especificación de las pruebas, requisitos del entorno de prueba e informe de preparación, requisitos de los datos de prueba e informe de preparación, resultados de la prueba, registro de ejecución de las pruebas e informe de incidentes.

La estructura de la Parte 3 del estándar, se muestra en la Figura 2.12.

Figura 2.12. ISO/IEC/IEEE 29119 - Parte 3: Documentación de las pruebas (Reid, 2017).

Existe un fuerte vínculo entre los estándares de la Parte 2 (Procesos) y Parte 3 (Documentación). Los resultados de los procesos definidos en la Parte 2 corresponden a la documentación definida en la Parte 3. Los nombres de los documentos utilizados en la Parte 3 son coherentes con los productos descritos en la Parte 2, pero no es necesario utilizar la misma estructura de documento o la misma denominación (es decir, puede combinar o dividir documentos descritos en el estándar y usar diferentes nombres), siempre que el contenido requerido esté documentado en el mismo nivel.

2.5.4. ISO/IEC/IEEE 29119 - 4: Técnicas de Prueba

Se requiere que los usuarios que siguen la Parte 2, elaboren planes de prueba que especifiquen qué técnicas de diseño de caso de prueba se usarán y qué criterios de finalización de prueba se deben alcanzar. En la Parte 4 se define una amplia gama de técnicas de prueba y medidas de cobertura correspondientes. Su estructura se muestra en la Figura 2.13.

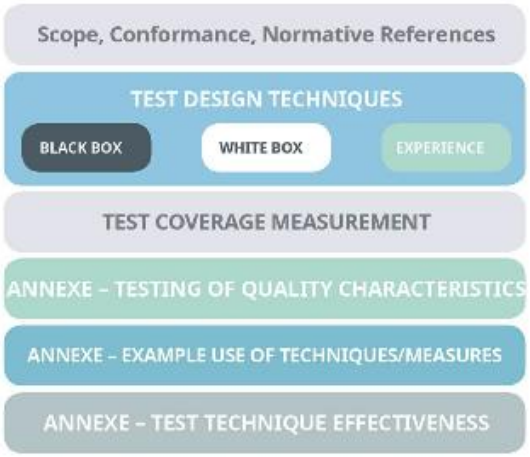


Figura 2.13. ISO/IEC/IEEE 29119 - Parte 4: Técnicas de Pruebas (Reid, 2017).

En la Parte 4 se brinda orientación para implementar las siguientes técnicas de diseño de casos de prueba:

- ***Técnicas de prueba basadas en especificación:*** partición de equivalencia, análisis de valor límite, prueba de transición de estado, prueba de tabla de decisión, prueba de escenario, gráficas de causa y efecto, método de árbol de clasificación, técnicas de prueba combinatorias, pruebas de sintaxis, pruebas aleatorias.
- ***Técnicas de prueba basadas en la estructura:*** pruebas de declaración, pruebas de ramificación, pruebas de decisión, condición de rama, prueba de combinación de condición de rama, prueba de cobertura de decisión de condición modificada (MCDC) y prueba de flujo de datos.
- ***Técnicas de prueba basadas en la experiencia:*** adivinar errores.

2.5.5. ISO/IEC/IEEE 29119 - 5: Pruebas dirigidas por palabras clave

La Parte 5, que se muestra en la Figura 2.14, proporciona una introducción y un enfoque de referencia para implementar pruebas basadas en palabras clave.

Figura 2.14. ISO/IEC/IEEE 29119 - Parte 5; Pruebas basadas en palabras clave. (Reid, 2017).

Define los requisitos en los marcos para las pruebas basadas en palabras clave y los requisitos mínimos para las herramientas de apoyo, que son necesarios para utilizar por completo el enfoque de las pruebas basadas en palabras clave. Las interfaces definidas y un formato común de intercambio de datos aseguran que los usuarios de la norma puedan compartir artefactos de prueba, como casos de prueba, datos de prueba y resultados de pruebas.

2.6. Marco Legal

Sobre el particular, la Ley Orgánica de Defensa del Consumidor, publicada en el Registro Oficial No. 116 el 10 de julio del 2000, y modificada el 16 de enero del 2015, establece entre otros aspectos:

Artículo 4.- Derechos del consumidor. Son derechos fundamentales del consumidor, a más de los establecidos en la Constitución Política de la República, tratados o convenios internacionales, legislación interna, principios generales del derecho y costumbre mercantil, los siguientes:

…

ii. Derecho a que proveedores públicos y privados oferten bienes y servicios competitivos, de óptima calidad, y a elegirlos con libertad;

…

Artículo 27.- Servicios Profesionales. - Es deber del proveedor de servicios profesionales, atender a sus clientes con calidad y sometimiento estricto a la ética profesional, la ley de su profesión y otras conexas.

CAPÍTULO 3. MARCO METODOLÓGICO

3.1. Descripción del área de estudio

La presente investigación se realizará en las oficinas de la Dirección de Tecnologías de la Información, de la Cooperativa de Ahorro y Crédito Atuntaqui Ltda., ubicadas en la ciudad de Atuntaqui, provincia de Imbabura.

La Dirección de Tecnologías de la Información está conformada por tres áreas: Seguridad de la Información, Operaciones y Soporte TI, y Desarrollo de Software.

Figura 3.1. Organigrama de la Dirección de Tecnología de la COAC Atuntaqui Ltda.

Fuente: (Cooperativa de Ahorro y Crédito Atuntaqui Ltda., 2018)

El Área de Desarrollo de Software se encuentra integrada por un Administrador de Desarrollo y tres Analistas Programadores, su responsabilidad es implementar los proyectos de software que la institución requiera.

3.2. Diseño y tipo de investigación

La investigación utiliza un enfoque ***cualitativo*** que permita observar el proceso de gestión de pruebas en la Dirección de Tecnologías de la Información, validarlo bajo el criterio del personal técnico involucrado y la documentación existente del proceso.

Además, se usa un enfoque ***cuantitativo*** que permita la recolección de datos en la fase de mantenimiento del software, con la elaboración de encuestas al personal de soporte e informes de la herramienta de soporte *"Service Desk"* del Departamento de Tecnología, para contrastarlos posteriormente con los resultados obtenidos de la aplicación del modelo de gestión de pruebas propuesto.

3.2.1. Modalidad de la investigación

Para la elaboración del proyecto se utiliza:

La ***investigación bibliográfica*** en fuentes de consulta como: libros, artículos científicos, publicaciones, revistas, estándares y toda la bibliografía necesaria para comprender el estado del arte, conocimientos científicos, filosóficos y legales que fundamentan la investigación.

Una ***investigación de campo*** para comprender el proceso de gestión de pruebas de software que emplea el Área de Desarrollo de la COAC Atuntaqui Ltda.

Además, para comprender de mejor manera la fase de pruebas, se realizará una ***investigación documental*** a los informes previos de las pruebas del software realizadas.

3.2.2. Tipos de investigación

Durante el desarrollo se utilizaron algunos tipos de investigación.

Se realiza una ***investigación exploratoria*** al entorno de producción de software de la Dirección de Tecnología de la Cooperativa Atuntaqui, para conocer el proceso de gestión de pruebas, la problemática del proceso y sus efectos en la fase de mantenimiento.

Se realiza una ***investigación descriptiva*** para examinar las características del proceso de gestión de pruebas de software, determinar sus actores, métodos, técnicas y herramientas, y establecer la relación entre el proceso de gestión de pruebas y la fase de mantenimiento.

Al final, a través de una ***investigación explicativa***, se intenta comprobar la relación entre la fase de pruebas y la fase de mantenimiento, con la detección de defectos durante el proceso de pruebas del software.

3.3. Procedimiento de investigación

3.3.1. Métodos

Se aplica el ***método analítico*** para estudiar las fases del proceso de desarrollo, políticas establecidas, informes previos y toda la documentación existente que nos permita comprender el proceso de desarrollo de software en la COAC Atuntaqui Ltda.

Se utiliza el ***método inductivo*** basado en el razonamiento y experiencia para obtener conclusiones generales que partan de hechos particulares, como las actividades y prácticas empleadas en el proceso de gestión de pruebas de software.

Se usa el ***método deductivo*** para realizar un estudio general del proceso de desarrollo de software, para posteriormente realizar un análisis de sus fases, procesos y subprocesos.

3.3.2. Población y muestra

La investigación se realiza con la participación de Director de Tecnología, personal del Área de Desarrollo y el personal de Soporte Técnico de la Dirección de Tecnología de la COAC Atuntaqui Ltda.

Tabla 3.1. *Población de la investigación.*

Población	Frecuencia	Porcentaje
Director de Tecnología	1	12.5 %
Administrador de Desarrollo	1	12.5 %
Analistas Programadores	3	37.5 %
Soporte Técnico	3	37.5 %
Total	**8**	**100%**

Fuente: Autor

Como la población del proyecto de investigación no sobrepasa las cien personas, se trabaja con la totalidad de esta, sin que sea necesario sacar muestras representativas.

3.3.3. Estrategias técnicas

Para la investigación se utilizan las siguientes técnicas:

Se aplican ***encuestas*** al personal técnico del Área de Desarrollo y el Área de Soporte Técnico, para medir las opiniones del proceso de gestión de pruebas que actualmente usa la cooperativa y medir los efectos causados en la fase de mantenimiento.

Al Administrador de Desarrollo, quien es el encargado de planificar y gestionar los proyectos de software, se le realiza una ***entrevista***; con el fin de, comprender la forma como se realiza la gestión de los proyectos, específicamente, en la fase de pruebas.

Se realizan visitas de ***observación*** para analizar las actividades, tareas, técnicas y prácticas utilizadas en el proceso de gestión de pruebas de software.

3.3.4. Instrumentos

Para la recolección de la información se utilizarán los siguientes instrumentos:

- Cuestionario de la encuesta para el Área de Desarrollo (Anexo 1).
- Cuestionario de la encuesta para Soporte Técnico (Anexo 2).
- Cuestionario base de la entrevista al Administrador de Desarrollo (Anexo 3).
- Computador portátil, teléfono celular.
- Microsoft Office.
- Fotocopias, papel Bonn, bolígrafos.

3.3.5. Operacionalización de variables

Tabla 3.2. *Operacionalización de la variable independiente.*

Conceptualización	Dimensión	Indicadores	Items Básicos	Técnicas e Instrumentos
Gestión de Pruebas de Software **Las pruebas de software son el proceso de análisis de un sistema, o componente de un sistema, para detectar las diferencias entre el comportamiento especificado (requerido) y el observado (existente).**	Metodología	Aplicación de metodología para el proceso de gestión de pruebas (Si/No). ¿Cuál? Priorización de las funcionalidades (Muy Bueno, Bueno, Regular) Selección de los casos de prueba (Muy Bueno, Bueno, Regular)	¿El Área de Desarrollo emplea alguna metodología para la gestión de pruebas de software? ¿La priorización de las funcionalidades es? ¿La selección de los casos de prueba es?	Entrevista Encuesta
	Procesos	Etapas definidas para la gestión de las pruebas de software (Si/No). ¿Cuáles? El proceso de pruebas (Muy Bueno, Bueno, Regular).	¿Las pruebas a ser ejecutadas se planifican con anticipación, estableciendo datos de entrada y resultados esperados? ¿Quiénes integran en equipo de pruebas? ¿El proceso de pruebas de software es?	Entrevista, Encuesta
	Técnicas	Técnicas establecidas para la ejecución de pruebas, dependiendo de su alcance. (Si/No). ¿Cuáles?	¿Se utiliza técnicas para la ejecución de las pruebas?, ¿Cuáles?	Entrevista
	Documentación	Documentación resultante del proceso de pruebas (Si/No). ¿Cuál? La documentación de las pruebas (Muy Bueno, Bueno, Regular).	¿Se documentan los resultados obtenidos en las pruebas para establecer su efectividad? ¿Se documentan los defectos encontrados en las pruebas? ¿La documentación de las pruebas es?	Entrevista, Encuesta, Revisión documental

Fuente: Autor

Tabla 3.3. *Operacionalización de la variable dependiente.*

Conceptualización	Dimensión	Indicadores	Items Básicos	Técnicas e Instrumentos
Mantenimiento del Software **Es el proceso del desarrollo del software que efectúa cambios al software para corregir errores, defectos o bugs encontrados durante su uso, así como, mejora y optimiza el software desplegado para adicionar nuevas funcionalidades.**	Mantenimiento para corrección de errores	Incidentes ingresados para corregir fallas.	¿Cuántos incidentes de soporte fueron ingresados para corregir fallas de software?	Service Desk
		Afectaciones a la base de datos.	¿Cuántas afectaciones directas a la base de datos se realizaron?	
	Mantenimiento para mejora de funcionalidades	Incidentes ingresados para mejorar alguna funcionalidad.	¿Cuántos incidentes de soporte fueron ingresados para mejorar alguna funcionalidad?	Service Desk
		Reportes de información	¿Cuántos incidentes de soporte fueron ingresados para la obtención de información?	
	Tiempo de respuesta	Tiempo empleado en solventar fallas de software	¿Tiempo promedio empleado en la solución de incidentes?	Service Desk
		Tiempo empleado en la implementación de nuevas funcionalidades	¿Tiempo promedio empleado en implementar nuevas funcionalidades?	
			¿Tiempo disponible del personal técnico para nuevos proyectos?	Entrevista

Fuente: Autor

3.3.6. Resultados de la encuesta realizada al Área de Desarrollo

La encuesta se realiza al personal técnico del Área de Desarrollo para medir su percepción sobre el proceso de pruebas software actual; se solicitó que califiquen cada uno de los ítems en una escala de: Muy Bueno, Bueno, Regular.

Los resultados obtenidos en la encuesta se resumen en la siguiente figura.

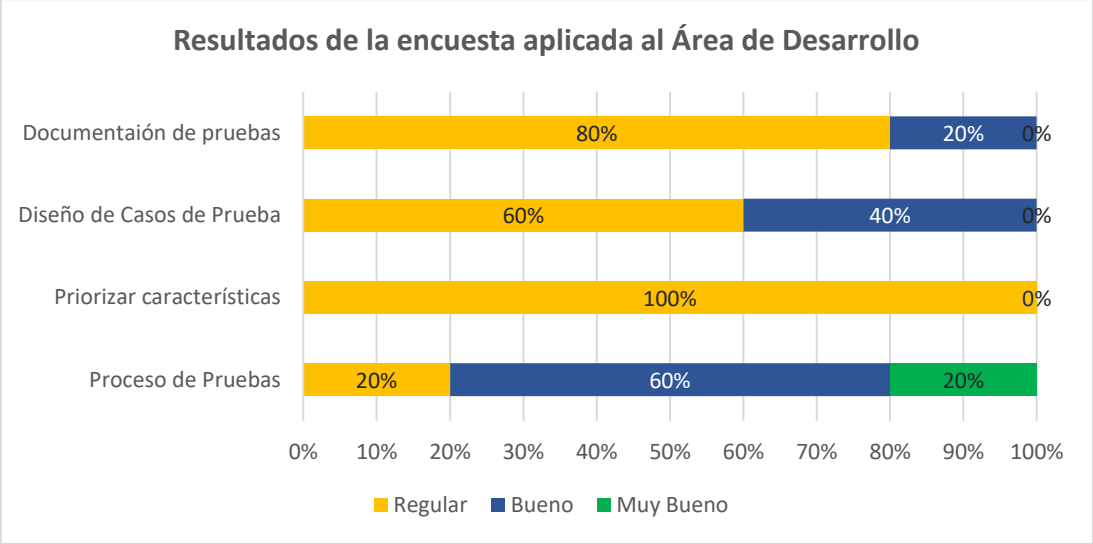


Figura 3.2. Resultados de la encuesta aplicada al personal de Desarrollo de la COAC Atuntaqui Ltda.

Fuente: Autor

Se puede observar que el 60% de los encuestados califican al ***proceso de pruebas de software actual*** como *Bueno*, el 20% lo califica como *Muy Bueno* y un 20% como *Regular*.

Todos los encuestados califican al ***proceso de priorización de las características a probar*** como *Regular*.

El 60% de los encuestados, piensa que el proceso de ***diseño de casos de prueba*** es *Regular*, mientras que un 40% cree que es *Bueno*.

Por último, la mayoría de los encuestados, un 80%, califican a la ***documentación*** de las pruebas de software como *Regular* y un 20% la califican como *Buena*.

La percepción final del proceso de pruebas de software, por parte del personal de desarrollo, es 65% *Regular*, 30% *Bueno* y tan solo un 5% como *Muy Bueno*.

3.3.7. Resultados de la encuesta realizada al personal de Soporte Técnico

La encuesta al personal técnico del Área de Soporte se realiza para medir la percepción del software implementado por el Área de Desarrollo, una vez que este se encuentra en ambiente de producción.

Los resultados de la encuesta se resumen en la siguiente figura:

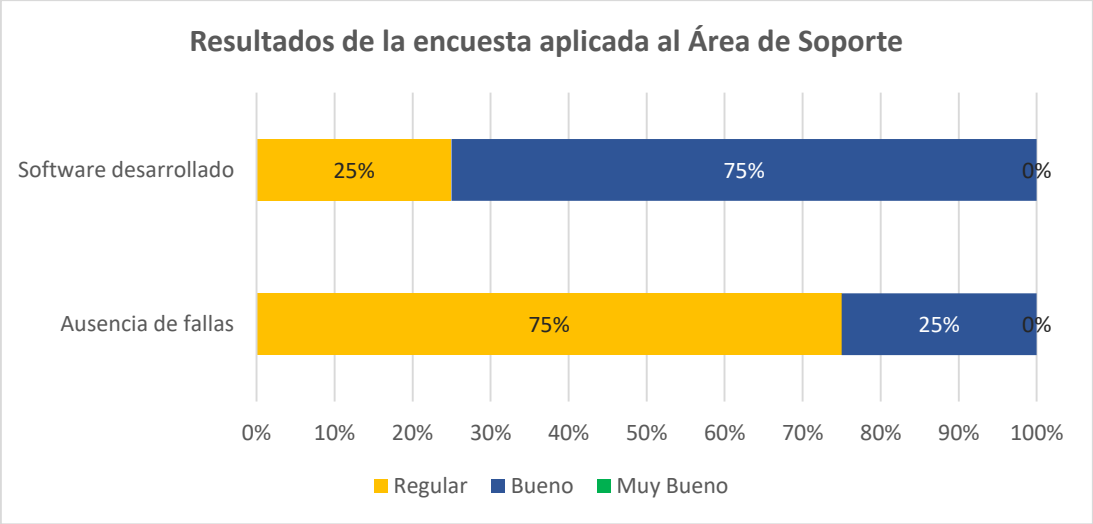


Figura 3.3. Resultados de la encuesta aplicada al personal de Soporte de la COAC Atuntaqui Ltda.

Fuente: Autor

Como se puede apreciar en la figura, el 75% de los encuestados, creen que el ***software desarrollado*** en la institución es *Bueno*, mientras que un 25% cree que es *Regular*. Mientras, que el 75% del personal de soporte encuestado, califica como *Regular* a la ***ausencia de fallas*** en el software desarrollado, mientras que un 25% la califica como *Buena*.

Ninguno de los aspectos evaluados por el personal de soporte, lo califica como *Muy Bueno*, porque manifiesta que existen algunos inconvenientes en el software que se deben corregir.

3.3.8. Entrevista realizada al Administrador de Desarrollo

La entrevista se realizó al Administrador de Desarrollo, Ing. Javier Collaguazo, persona responsable del proceso de Gestión de Desarrollo de Software en la COAC Atuntaqui Ltda. El objetivo de la entrevista es conocer el proceso de desarrollo establecido en la Dirección de Tecnología, específicamente a las pruebas de software.

Las preguntas y respuestas obtenidas se detallan a continuación:

1. **¿La institución cuenta con un proceso de desarrollo de software establecido?**

   Actualmente, la institución cuenta con un proceso para la Gestión de Desarrollo de Software, que se encuentra aprobado por el Consejo de Administración.

2. **¿Cuáles son las fases del proceso de desarrollo?**

   El proceso de Gestión de Desarrollo de Software cuenta con las fases de: Análisis, Diseño, Construcción, Paso a Producción, Mantenimiento y Baja.

3. **¿Existe una fase para las pruebas de software?**

   Una fase como tal no existe, las pruebas de software se encuentran consideradas en la fase de Construcción como una actividad adicional.

4. **¿Se utiliza alguna metodología para la gestión de las pruebas?**

   No, no se utiliza ninguna metodología para la gestión de pruebas, debido al número de integrantes del área de desarrollo.

5. **¿Quién realiza las pruebas de software?**

   Una vez que se cuenta con una versión estable del software, las pruebas son realizadas por el Analista Programador y el Líder de Proyecto o Especialista designado.

6. **¿Se planifican los casos de prueba que van a ser ejecutados?**

   No, el Líder de Proyecto o Especialista prueba el software tomando en cuenta los requisitos solicitados y su conocimiento del proceso implementado.

7. **¿Se utiliza alguna técnica para la realización de las pruebas?**

   No, las pruebas se realizan en base a los requisitos del usuario y el especialista designado, quien es el encargado de probar lo que estime pertinente.

8. **¿Las pruebas de software son documentadas?**

   Existe un formulario para las pruebas de software, donde se registran los requisitos y el resultado satisfactorio o no de la prueba; este formulario es firmado tanto por el Analista Programador y el Especialista que realizaron las pruebas, así como, por el Administrador de Desarrollo, quién es el responsable de todo el proceso.

9. **¿Se registran los defectos de software encontrados en las pruebas?**

   No, solo se comunican al Analista Programador de manera verbal o por correo electrónico para que realice las correcciones de los defectos encontrados.

10. **¿Cuándo se publica el software en producción?**

    El software pasa a producción cuando el Líder del Proyecto da su aceptación final de las pruebas y solicita su instalación en el ambiente de producción.

CAPÍTULO 4. PROPUESTA

4.1. Antecedentes

La Dirección de Tecnología de la COAC Atuntaqui Ltda., tiene establecidos los procesos que rigen la Gestión de Tecnología de la Información, entre estos se encuentra el Proceso de Gestión de Desarrollo de Software.

Figura 4.1. Procesos que integran la Gestión de Tecnología de la Información.

Fuente: Dirección de Tecnología - COAC Atuntaqui Ltda.

4.1.1. Proceso de Gestión de Desarrollo de Software

El objetivo del proceso de Gestión de Desarrollo de Software es definir los procedimientos que permitan gestionar los proyectos de desarrollo de software, durante el ciclo de vida de los sistemas, en alineación con la estrategia y objetivos institucionales.

Las políticas organizacionales establecidas para el proceso son:

1. La planificación de los proyectos de desarrollo de software se realizará en el último trimestre de cada año, con la presencia de Gerencia y Directores Departamentales, presidida por Director de Tecnología.
2. El requisito del proyecto deberá ser realizado por el propietario del proyecto, en el formulario establecido para el efecto, adjuntando el diagrama del proceso.
3. El procedimiento de pruebas del software será responsabilidad del Propietario del proyecto, quien podrá designar a uno de sus colaboradores, garantizando que sea un usuario experto en el proceso correspondiente. El Analista Programador

asignado al proyecto, apoyará durante todo el proceso de pruebas bajo la supervisión del Administrador de Desarrollo.

4. Los sistemas que requieran la confidencialidad de determinados datos deberán mantener mecanismos de encriptación - establecidos en las buenas prácticas de desarrollo seguro de software.
5. Toda aplicación desarrollada por el área de desarrollo deberá cumplir con los lineamientos detallados en la metodología de desarrollo seguro de software.
6. El diseño del software deberá incorporar pistas de auditoría que permitan determinar los eventos suscitados en referencia a la información almacenada.
7. La carga inicial de la información podrá ser insertada en forma directa en la base de datos, proceso que deberá ser ejecutado por Soporte Técnico registrando su usuario, así como la fecha y hora real de la inserción.
8. De forma bienal Director de Tecnología propondrá ante el Comité Informático, la baja de los proyectos de software que no han sido utilizados.

El Proceso de Gestión de Desarrollo de Software está conformado por las fases de Análisis, Diseño, Codificación, Puesta en Producción, Soporte y Baja.

Figura 4.2. Fases que integran el Proceso de Gestión de Desarrollo de Software.

Fuente: Dirección de Tecnología - COAC Atuntaqui Ltda.

En la gráfica se puede observar que la fase de gestión de pruebas de software, no se encuentra considerada dentro del proceso de producción de software como una etapa independiente, esta se define como una actividad de la fase de codificación del software.

4.1.2. Fase de Construcción del Software

El proceso de Gestión de Desarrollo de Software de la Dirección de Tecnología de la COAC Atuntaqui Ltda., no cuenta con una fase independiente para la gestión de pruebas de

software, estas se encuentran integradas en la fase de construcción del software, como una actividad adicional del proceso; es así como, las pruebas de software se realizan de manera informal, sin un modelo de pruebas establecido donde se especifique las tareas, actividades, responsables y entregables de cada una de las etapas.

La Figura 4.3, muestra el flujo establecido para el proceso de construcción del software.

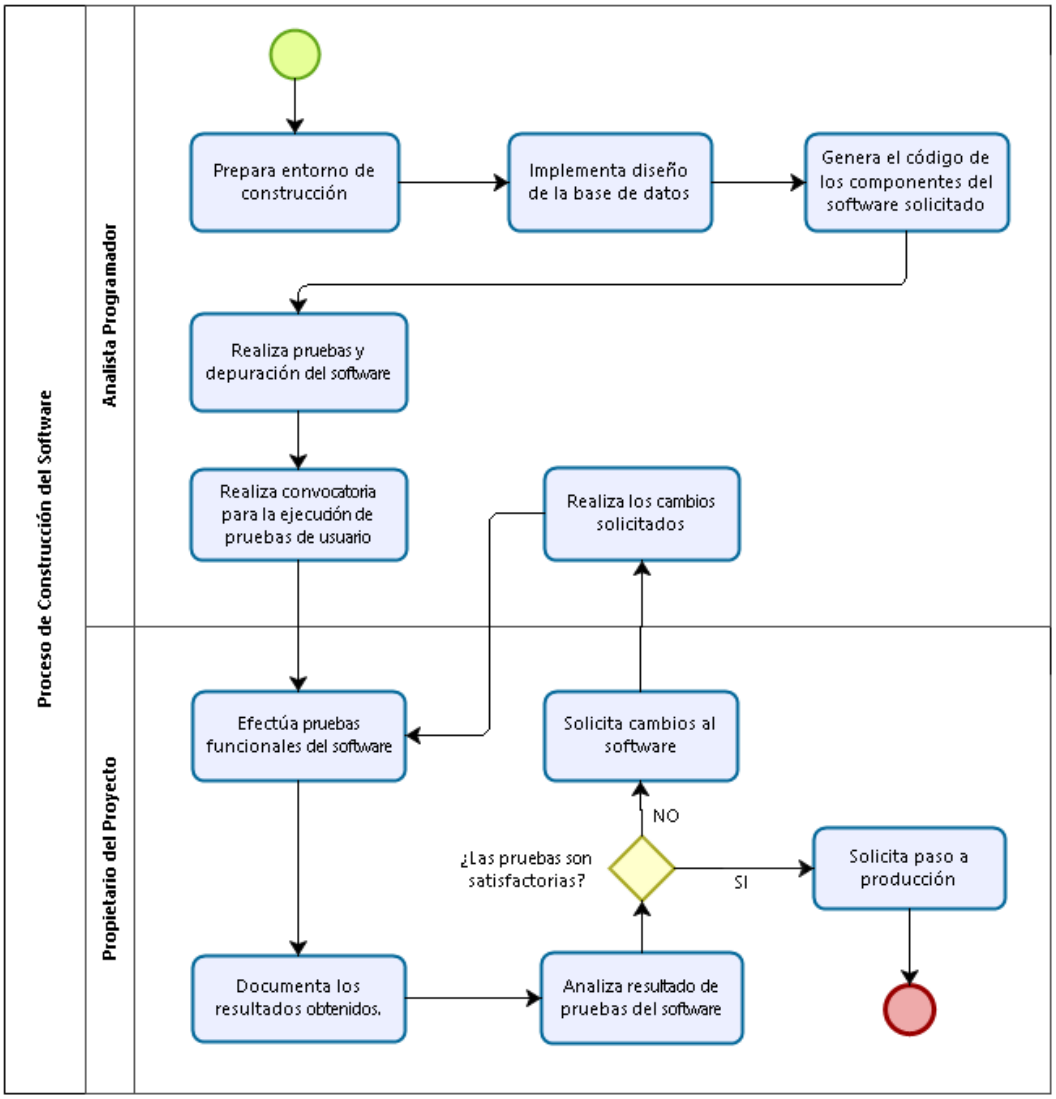


Figura 4.3. Proceso de Construcción del Software.

Fuente: Dirección de Tecnología - COAC Atuntaqui Ltda.

La certificación de las pruebas es realizada por el programador que desarrollo la funcionalidad (módulo o sistema) y el propietario del proyecto que solicito el desarrollo, quienes, por su inexperiencia, no realizan las pruebas suficientes y necesarias para validar la correcta funcionalidad del software.

4.1.3. Proceso de Gestión de Desarrollo de Software Propuesto

Para la implementación del marco de trabajo propuesto, es necesario incorporar la fase de pruebas al proceso de gestión de desarrollo de software, como una fase independiente y previa a la puesta en producción del software, tal y como se muestra en la siguiente figura.

Figura 4.4. Proceso de Gestión del Desarrollo de Software propuesto.

Fuente: Autor

Una vez creada la fase de pruebas como parte del proceso de desarrollo de software, se elabora la presente metodología para pruebas de software, basada en los procesos de pruebas establecidos en la segunda parte del estándar ISO/IEC/IEE 29119, para el proceso de evaluación de los productos de software.

4.2. Metodología de pruebas propuesta

La presente metodología de pruebas tiene como finalidad, formalizar los procesos de pruebas de software en la COAC Atuntaqui Ltda.

La primera fase de la implementación de esta metodología comprende solamente las pruebas funcionales. A medida que se implemente en los proyectos de software y se reciba la retro alimentación respectiva, se pretende que esta metodología evolucione y abarque otros tipos de pruebas.

A través de esta metodología, se podrá:

- Elaborar un plan para la gestión de pruebas de software.
- Establecer la estructura del equipo para las pruebas de software.
- Definir el proceso de ejecución de las pruebas.
- Establecer la documentación y entregables del proceso de pruebas.

Para iniciar la fase de pruebas, se debe disponer de una versión estable del sistema, módulo o funcionalidad a ser probada.

Se entenderá por estable cuando:

- El equipo de desarrollo haya finalizado las pruebas unitarias y de integración.
- Existan los componentes y documentación necesaria para su instalación, que especifique: la configuración de hardware necesaria, los requisitos de software, la parametrización inicial, entre otros.

4.2.1. Alcance

La primera versión de esta metodología comprende solamente las pruebas de tipo funcional. En una primera fase, la ejecución de las pruebas no usará una herramienta de software automática. Se plantea que luego de adquirir suficiente retro alimentación, experiencia y práctica en la ejecución de las pruebas, se analice y evalúe la posibilidad de adquirir una herramienta para la automatización de las pruebas.

A medida que se ejecuten más proyectos y esta metodología se aplique en la ejecución de las pruebas funcionales de los diferentes sistemas desarrollados, se evaluará su efectividad a fin de que evolucione hasta que abarque otros tipos de pruebas.

La metodología no depende del tipo de módulo de software, por lo que en un principio podrá aplicarse a cualquiera de los desarrollos realizados por la Dirección de Tecnología de la COAC Atuntaqui Ltda.

4.2.2. Roles

Para la aplicación de la metodología se establecen los siguientes roles y responsabilidades, para el equipo de trabajo involucrado en el proceso de pruebas de software.

Tabla 4.1. *Roles y responsabilidades del equipo de pruebas.*

Rol	Responsabilidades
Administrador de Desarrollo	Evaluar y mantener la metodología.
Líder de Pruebas	Planificar y controlar las pruebas.
Ingeniero de Pruebas	Construir y ejecutar las pruebas.
Especialista	Apoyar al equipo de pruebas.

Fuente: Autor.

4.2.3. Flujo del Proceso de Pruebas

El flujo del proceso de pruebas de software definido para esta metodología se describe en la siguiente figura.

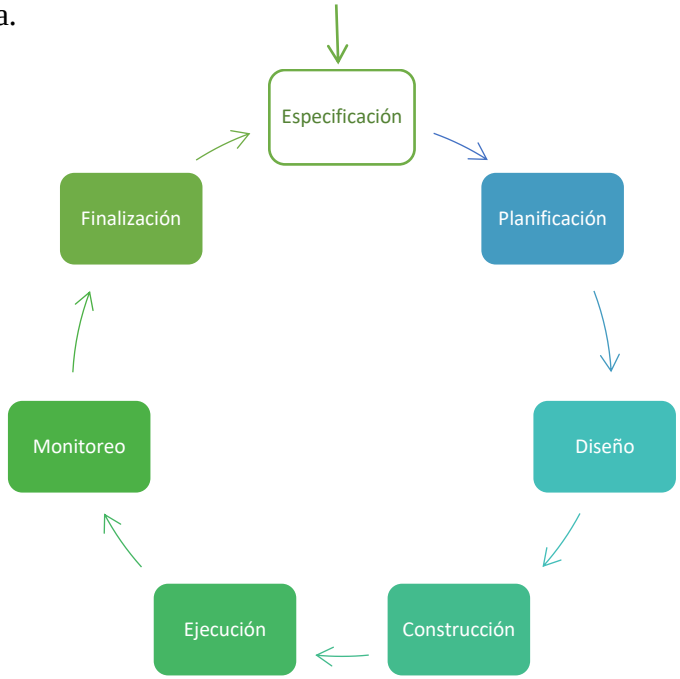


Figura 4.5. Flujo del Proceso de Pruebas de Software.

Fuente: Autor

El proceso de pruebas está compuesto de las siguientes fases:

- **Especificación del software**, entregada por el Líder del Proyecto.
- **Planificación de Pruebas**, ejecutada por el Líder de Pruebas.
- **Diseño de Pruebas**, ejecutada por el Líder de Pruebas.
- **Construcción de Pruebas**, ejecutada por el Ingeniero de Pruebas.
- **Ejecución de Pruebas**, ejecutada por el Ingeniero de Pruebas.
- **Monitoreo y Control de Pruebas**, ejecutada por el Administrador de Desarrollo.
- **Finalización de las pruebas,** ejecutada por el Administrador de Desarrollo.

Sus entradas principales son:

- Manual de Gestión de Desarrollo de Software (Políticas y Estrategias de pruebas).
- El documento final de Requisitos de software (Especificación técnica y funcional)
- Una versión estable del software que se va a probar.

Las principales salidas o entregables del proceso de pruebas son:

- El **Plan de Pruebas de Software,** donde se define: el alcance de las pruebas; sus riesgos, mitigación y contingencia, los recursos necesarios (hardware, software y personas) y el cronograma de actividades.
- La **Especificación del Diseño de Pruebas**, donde se enumeran las características de software que van a ser evaluadas, con su respectiva prioridad de ejecución.
- La **Especificación de los Casos de Prueba**, o combinación de entradas y condiciones que pueden darse para cada característica a ser evaluada.
- Los **Guiones de Prueba**, donde para cada Caso de Prueba, se especifica en detalle: sus condiciones iniciales, sus entradas, los pasos a seguir, los resultados esperados y las condiciones finales.
- Los **Reportes de Defectos**, donde se registra cualquier defecto que se detecte durante la ejecución de las pruebas del software.
- El **Informe de Resultados de las Pruebas**, donde se resumen los resultados obtenidos durante la ejecución de las pruebas.

A continuación, se describe las actividades a desarrollarse en cada una de las fases.

4.2.4. Planificación de Pruebas

La fase de planificación está dividida en cinco actividades:

1. **Identificar los requisitos de prueba**

El objetivo de esta actividad es determinar:

- El alcance de las pruebas,
- Los requisitos o características que se probarán, y
- Las características que quedarán excluidas de las pruebas.

En el alcance, se debe especificar el software a probar y sus principales características, los componentes del software con sus respectivas versiones y el tipo de pruebas a ejecutar.

Para determinar los requisitos o características que se probarán, se debe descomponer el software en aquellas cosas que serán probadas. La lista de Requisitos de Prueba puede desarrollarse a partir de los casos de uso, documentos de especificación (técnica y funcional) u otras fuentes, y se usa como base para las restantes actividades.

Finalmente se debe determinar las características que no se probarán; es decir, aquellas que quedarán excluidas de las pruebas. Para cada una se debe colocar las respectivas razones para su exclusión.

2. Identificar Prioridades

La lista de Requisitos de Prueba debe ordenarse según una prioridad de prueba. Para ello, para cada requisito, se debe evaluar dos factores:

- **Probabilidad de falla**. - expresada en términos de Alta, Media o Baja. Significa cuán probable es que se encuentre un defecto al ejecutar las pruebas del requisito en cuestión.
- **Frecuencia de uso. -** expresada en términos de Alta, Media o Baja, significa con qué frecuencia se hace uso del requisito analizado. Un requisito con un factor de uso alto por lo general conlleva un alto riesgo de falla.

Para determinar el valor de estos factores se puede requerir el apoyo del especialista.

Una vez tabulados estos valores, se debe determinar la Prioridad de Prueba de acuerdo con la siguiente tabla:

Tabla 4.2. *Prioridades de Prueba.*

PRIORIDAD DE PRUEBA		FRECUENCIA DE USO		
		ALTA	MEDIA	BAJA
PROBABILIDAD DE FALLA	ALTA	Alta	Alta	Media
	MEDIA	Alta	Media	*Baja*
	BAJA	Media	*Baja*	*Baja*

Fuente: Autor.

Priorizar, de acuerdo con el riesgo de falla y la frecuencia de uso, ayuda a determinar el orden en el cual las pruebas serán desarrolladas y ejecutadas. Con esto se asegura que, si las pruebas deben terminar por restricciones de tiempo, se probará primero lo que se determinó como más importante.

3. Identificar Recursos

Se debe estimar la cantidad de recursos que se necesitarán para diseñar, construir y ejecutar las pruebas de software.

Los tipos de recursos a identificar son:

- **Datos. –** Corresponde a los datos reales, internos o externos al software evaluado, que serán necesarios para soportar el esfuerzo de pruebas.
- **Ambiente. -** Hardware y software necesario para instalar y probar el sistema. El ambiente debe dimensionarse considerando los requerimientos técnicos del software. Por ejemplo: aplicación de escritorio o web, tipo de dispositivo, etc.
- **Herramientas. -** Entendidas como cualquier herramienta de software que sea necesaria para soportar el proceso de pruebas y que no sea parte del sistema o requerida para su instalación. En este grupo están incluidos, pero no limitados, los generadores de datos, migradores, herramientas para modelación, simuladores, disparadores, etc.
- **Recursos Humanos. -** Equipo humano que incluye, pero no se limita a: Ingenieros de Pruebas, Diseñadores, Implementadores y Administradores.

4. Crear Cronograma

Esta actividad incluye la estimación de tiempos para diseñar, construir y ejecutar las pruebas, basado en experiencias anteriores y métricas. Al momento solo se cuenta como base para esta información con el número y tipo de Requisitos de Prueba establecidos en la actividad Identificar los requisitos de prueba.

Se recomienda que durante el proyecto se definan hitos al finalizar las principales fases del cronograma, con el fin de facilitar el control de avance del cronograma.

5. Generar el Plan de Pruebas

En esta actividad se debe identificar y definir qué entregables se crearán, mantendrán, y estarán disponibles durante la ejecución de las pruebas. Se debe documentar el calendario de entrega de aquellos entregables. También es importante establecer a quiénes se distribuirán tales entregables.

Finalmente se debe combinar todos los datos de los pasos previos y crear un Plan de Pruebas de Software.

4.2.5. Diseño de Pruebas

La fase de diseño de pruebas está compuesta por las siguientes actividades:

- Identificar Casos de Prueba
- Identificar Datos de Prueba

1. Identificar Casos de Prueba

Un Caso de Prueba es una combinación de condiciones y entradas para un Requisito de Prueba específico. Por lo general, para cada Requisito de Prueba se deberá desarrollar más de un Caso de Prueba. Típicamente los Casos de Prueba incluyen casos para verificar:

- El flujo normal de operación;
- Flujos alternativos;
- Flujos de excepción o error;
- Valores mínimos;
- Valores máximos;
- Controles;
- Eventos,
- Cumplimientos de estándares,

- Restricciones de rendimientos,
- Carga,
- Situaciones de estrés,
- Condiciones de alto volumen,
- Seguridad o configuración,
- Instalación, etc.

Por ejemplo, para el Requisito de Prueba: Pagar cheque, podemos definir las siguientes condiciones:

- La cuenta existe o no existe.
- El cheque es válido o no es válido.
- La cuenta tiene fondos o no tiene fondos.

Las cuales generan los siguientes Casos de Prueba:

- Pagar un cheque válido de una cuenta que existe y tiene fondos suficientes.
- Pagar un cheque válido de una cuenta que existe y no tiene fondos suficientes.
- Pagar un cheque no válido de una cuenta que existe.
- Pagar un cheque de una cuenta que no existe.

2. Identificar Datos de Prueba

En este punto se debe identificar todos los datos necesarios para los Casos de Prueba definidos en la actividad anterior.

Los datos de prueba deben cumplir con las siguientes características:

- Que sean datos reales.
- Que la información entre tablas este consistente.
- Que la cantidad de datos en las estructuras sea del mismo volumen o parecido al ambiente en producción.
- Organizar los datos a fin de evitar inestabilidad por pruebas simultáneas.

Para todos los casos, los datos deben ser lo más cercanos o equivalentes a los del ambiente en producción.

Finalmente, se debe considerar en este punto cómo organizar las pruebas y sus datos, de manera que un equipo no afecte la información con la que trabaja otro y cause confusión durante las pruebas.

4.2.6. Construcción de Pruebas

En esta fase, principalmente se desarrollan los Procedimientos de Prueba, y se crean los datos necesarios para las pruebas.

1. **Creación de Procedimientos de Prueba**

Un Procedimiento de Prueba es un conjunto de instrucciones detalladas para preparar, ejecutar y evaluar el resultado de un Caso de Prueba o conjunto de Casos de Pruebas.

Para definir los Procedimientos de Prueba es necesario identificar:

- Las secuencias de navegación o secuencias de eventos necesarios para probar cada Caso de Prueba.
- Las relaciones entre los Procedimientos de Prueba y los Casos de Prueba.

Un Procedimiento de Prueba está compuesto de:

- Condiciones o instrucciones de preparación.
- Condiciones o estado iniciales de la aplicación o del procedimiento.
- Tareas o pasos para ejecutar.
- Valores de entrada.
- Valores o resultados esperados.
- Condiciones o estado finales de la aplicación o del procedimiento.

2. **Crear Datos de Prueba**

En este punto se deben crear los datos necesarios para ejecutar todos y cada uno de los Procedimientos de Prueba. Puede ser tan trivial como partir de la aplicación instalada y

configurada solamente, o más compleja como la generación de un conjunto de datos que cumpla con las características planteadas para la prueba.

También se deben construir todos los programas o procedimientos para cargar los datos, y en el caso de requerirse los procesos para restaurarlos a sus estados iniciales.

4.2.7. Ejecución de Pruebas

La fase de ejecución consiste en tomar los Procedimientos de Prueba y seguir paso a paso lo establecido en ellos.

La secuencia de ejecución de los procedimientos debe estar planeada y diseñada de manera que se haga un uso óptimo de todos los recursos: datos, ambientes y recursos humanos.

Se debe evitar trabajo redundante, maximizar la ejecución paralela de secuencias independientes de procedimientos, y evitar que el trabajo en paralelo cree conflictos durante las pruebas.

Mientras se ejecutan los procedimientos, pueden encontrarse defectos o discrepancias entre los resultados esperados y los obtenidos. En este punto el Ingeniero de Pruebas que encuentra el defecto debe registrar la discrepancia en el formato de Procedimientos de Prueba y reportarlo al Especialista para que asigne la corrección del mismo.

Para el seguimiento de los defectos se usará una herramienta y el flujo descrito a continuación.

1. **Registro y Seguimiento de Defectos**

Cada vez que se encuentre un defecto durante la ejecución de las pruebas, el Ingeniero de Pruebas debe registrar los defectos encontrados, con la siguiente información:

- Código de identificación
- Descripción
- Requisito, Caso y Procedimiento de Prueba relacionados
- Prioridad

- Datos de Entrada
- Resultado Esperado
- Resultado Obtenido

El Líder de Desarrollo debe revisar los defectos, analizarlos y asignarles una prioridad para su corrección, y comunicar esto al equipo de desarrollo. Una vez que el equipo de desarrollo corrija el defecto, el Ingeniero de Pruebas debe validar que el defecto ya no exista. Si es así el defecto pasa a integrar los casos de prueba satisfactorios y se finaliza el incidente. Si la corrección no hace que el defecto desaparezca debe devolverse el incidente, con los respectivos comentarios adicionales. Si al validar la corrección de un defecto, se encuentra otro, debe registrarse como si fuera uno nuevo.

4.2.8. Monitoreo y Control de Pruebas

Una vez finalizado el período de pruebas o incluso mientras dura la fase de ejecución, se debe evaluar los resultados de las pruebas. Para evaluar el estado actual o resultado final de las pruebas, en cada ciclo se han establecido métricas para las pruebas de software.

Una métrica es usada para describir un atributo o también puede definirse como una escala para la medición. Por lo cual, las métricas son usadas para medir la calidad de un proyecto. (Jiménez, 2017).

1. **Cobertura de las Pruebas**

La ***Cobertura de Pruebas*** sirve para evaluar el estado actual de las pruebas, proporciona un indicador de cuantos requisitos se han probado del número total de requisitos especificados. Indica cómo se van cumpliendo los Casos de Prueba especificados, por lo tanto, una mayor Cobertura de Pruebas indica un buen desarrollo de las pruebas.

La Cobertura de Pruebas se define como:

$$CP = \frac{CPE}{CPP}$$

Donde:

CP es el valor de la cobertura de pruebas.

CPE es el número de Casos de Prueba que han sido ejecutados.

CPP es el número total de Casos de Prueba planificados.

La Cobertura de Prueba o CP indica el porcentaje de Casos de Prueba diseñados que han sido ejecutados. Su valor oscila entre 0 y 1, mientras más se acerque a 1 será más eficiente, esto implica que se están ejecutando la mayor cantidad de Casos de Prueba de los propuestos en el Plan de Pruebas, por ejemplo: un valor del 60% indica que, de 10 Casos de Prueba construidos, ya han sido ejecutados 6 y faltan por ejecutarse 4.

Cuando el plazo para las pruebas termine, el índice de Cobertura de Pruebas entregará parte de la información necesaria para medir el riesgo que se corre al entregar el software en el estado actual. Los parámetros para determinar la conveniencia o no de la finalización de las pruebas, deben estar establecidos en el Plan de Pruebas.

2. Madurez de las Pruebas

La ***Madurez de las Pruebas*** es el indicador del buen desempeño del flujo de trabajo de pruebas, no solo se enfoca en completar la ejecución de todos los Casos de Prueba definidos para cubrir los requisitos, sino que también comprende los Casos de Pruebas que han obtenido resultados satisfactorios.

La Madurez de las Pruebas se calcula como:

$$MP = \frac{CPS}{CPR}$$

Donde:

MP es el valor de la madurez de las pruebas.

CPS es el número de Casos de Prueba con resultado satisfactorio.

CPR es el número de Casos de Prueba diseñados para todos los requisitos.

3. Densidad de defectos

La ***Densidad de Defectos*** ofrece una medida sobre la proporción de defectos en relación con la cantidad de elementos de especificación.

Esta métrica permite realizar análisis estadísticos al finalizar las pruebas para valorar la integridad y madurez del software analizado.

La Densidad de Defectos se calcula como:

$$DD = \frac{TD}{CER}$$

Donde:

DD es la densidad de defectos

TD es el número total de defectos encontrados durante las pruebas.

CER es el número de elementos de especificación revisados.

Es recomendable para una alta calidad del software que la densidad de defectos tenga un valor mínimo.

4. Tendencia de Defectos

La métrica de ***Tendencia de los Defectos*** muestra el número de defectos, en una clasificación dada, como una función del tiempo. Para la clasificación se usará como atributo los estados de los defectos Abierto y Cerrado.

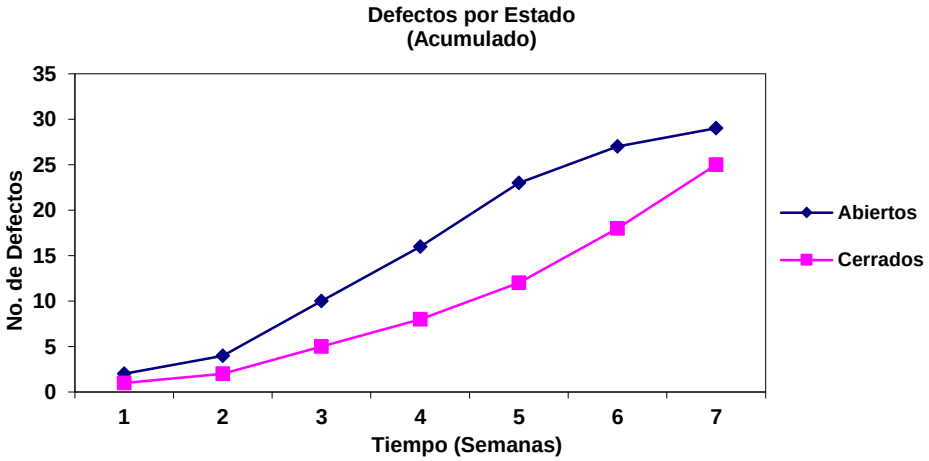


Figura 4.6. Ejemplo de la métrica de Tendencia de los Defectos.

Fuente: Autor.

En el ejemplo de la Figura 4.6, se distinguen tres zonas:

- El inicio (semanas 1 – 2), con una pendiente baja, donde la tasa de defectos es baja.
- La parte media (semanas 3 – 5), donde la pendiente y la tasa de defectos es alta.
- El final (semanas 6 – 7), donde nuevamente la tasa de defectos nuevos es baja. Lo que indica que para hallar un nuevo defecto se requiere más esfuerzo.

La gráfica puede mostrar el valor del número de defectos por período de tiempo o su acumulado, así como, el comportamiento de las pruebas a medida que transcurre el tiempo.

5. Porcentaje de Defectos por Tipo

El ***Porcentaje de Defectos por Tipo*** se emplea para priorizar y categorizar los errores. La prioridad indica la severidad del problema. Las categorías de fallos proporcionan una descripción del error para realizar análisis estadísticos de errores. Permite categorizar los fallos, porque identifica los tipos de defectos más comunes que puedan presentarse en cualquier etapa de desarrollo del software.

El porcentaje de defectos por tipo se calcula como:

$$DT = \frac{NDT}{TDI} x100$$

Donde:

DT es el porcentaje de defectos por tipo.

NDT es el número de defectos por tipo.

TDI es el número total de defectos identificados.

6. Camino Básico

La métrica de ***Camino Básico*** corresponde al porcentaje de caminos básicos independientes probados en relación con el total de ellos, sumando la complejidad ciclomática de todos los módulos del programa, se calcula como:

$$\text{PCB} = \frac{P}{V(G)} x100$$

Donde:

PCB es el porcentaje de caminos básicos.

P es el número de pruebas diseñadas

V(G) es la complejidad ciclomática calculada.

4.2.9. Finalización de las Pruebas

En cualquier punto de la ejecución de las pruebas, se puede evaluar en qué estado se encuentran las pruebas. Las métricas de pruebas como, la *Cobertura de Pruebas* y la *Madurez de las Pruebas* se usan para en un momento dado, decidir si el software evaluado puede ser liberado.

Finalizadas las pruebas, se debe elaborar un Informe de Resultados, el cual debe incluir:

- Los inconvenientes presentados durante las pruebas.
- Un resumen de los tipos de defectos encontrados.
- Los valores obtenidos de las métricas establecidas para las pruebas.
- Las prácticas que han demostrado ser de gran valor durante las pruebas y que deben incorporarse a la metodología.

4.3. Documentación de las pruebas

La parte de documentación de las pruebas determina los formularios y las plantillas de pruebas de software, que pueden ser utilizadas por las organizaciones, proyectos específicos o actividades individuales de prueba (Alaqail, 2018).

Una vez definido los procesos para las pruebas de software, se presenta un conjunto de plantillas guías, basadas en la parte tres del estándar ISO/IEC/IEEE 29119, cuyo objetivo es documentar todas las etapas de las pruebas.

4.3.1. Plan de Pruebas de Software

El Plan de Pruebas es el documento formal de la planificación de las pruebas, es elaborado por el Líder de Pruebas y en él se detallan las actividades, tiempos y responsables.

El Plan de Pruebas debe contener las secciones contenidas en el Anexo 4 y descritas a continuación:

1. **Información general del documento**.

   En esta sección, se establece la identificación única del documento, su emisión, responsables e historial de cambios.

2. **Introducción**

   Describe el proyecto de software dentro del cual se desarrollarán las pruebas y el módulo a probar. Comprende la razón de las pruebas, hechos históricos, documentos relevantes, estándares a verificar, planes de pruebas anteriores, etc.

3. **Objetivo**

   Este apartado se usa para indicar los objetivos de las pruebas enmarcadas en las características del proyecto.

4. **Alcance**

   En esta sección, se describe el alcance que tendrán las pruebas, especificando de manera general las características del software que se van a probar y las que no se van a probar.

   Características para probar

   Identifica las características y composiciones de características que serán probadas, las razones por las cuales se van a probar y su prioridad. Se debe referenciar toda la documentación asociada a las pruebas, como son: política y estrategia organizacional, especificaciones, diseño, entre otros.

   Características para no probar

   Identifica las características y combinaciones de estas que NO serán probadas y las razones por las cuales NO se van a probar. Al igual que en las características a probar se debe referenciar los documentos asociados.

5. Criterios de Paso/Falla/Suspensión/Reanudación

En esta sección, se debe especificar los criterios establecidos para determinar si una determinada prueba es satisfactoria o falla, los criterios establecidos para suspender las pruebas de manera total o parcial, e indicar las actividades que deben ser repetidas en el caso de que las pruebas se suspendan y se deban reanudar.

6. Riesgos y contingencia

Se debe identificar los altos riesgos asociados con el módulo que se va a probar y con la ejecución de las pruebas. Si existen se deben referir y anexar planes de contingencia.

7. Recursos

En este apartado, se debe especificar las características necesarias del ambiente de pruebas, y los recursos humanos necesarios. Debe contener las características del hardware y software; así como, el modo de uso para apoyar a la prueba.

También debe especificar el nivel de la seguridad para la prueba, el software del sistema y los componentes propietarios, tales como: software, datos y hardware. Identificar las herramientas especiales y alguna otra necesidad para la prueba.

Establece los responsables del manejo, diseño, preparación, ejecución de la prueba, así como sus funciones. Indica si son necesarios conocimientos específicos de las personas para el desarrollo de las pruebas o algún entrenamiento especial.

8. Planificación y organización

Incluye el cronograma de actividades para la ejecución de las pruebas, indicando las personas responsables, recursos y el tiempo necesario para realizar las pruebas por cada característica a probar y si existen fases de prueba deben estar especificadas.

9. Firmas de responsabilidad

Por último, se deben registrar las firmas de responsabilidad de las personas que participaron en la elaboración del Plan de Pruebas.

4.3.2. Especificación del Diseño de la Prueba

El Diseño de las Pruebas es elaborado por el Líder de Pruebas y en él se enumeran las características o conjunto de características de software que van a ser evaluadas, con su respectiva prioridad de evaluación, un ejemplo, se muestra en el Anexo 5.

En este documento se describe:

1. **Información general del documento**.

   En esta sección, se establece la identificación única del documento, su emisión, responsables e historial de cambios.

2. **Alcance**

   Se identifica el alcance de la especificación del diseño de pruebas y describe las inclusiones, exclusiones, supuestos y limitaciones de las pruebas.

3. **Características para probar**

   Se listan las características del software a ser evaluadas, con su respectiva prioridad de ejecución, conforme se describe en la Tabla 7 de Prioridades de Prueba**.**

   Cada característica para evaluar debe contener:

   - Identificador único.
   - Descripción breve.
   - Probabilidad de falla
   - Frecuencia de uso
   - Prioridad
   - Comentario

4.3.3. Especificación de los Casos de Prueba

El documento de Casos de Prueba detalla los casos de prueba que se van a ejecutar por cada una de las características de software que va a ser evaluada, y que se encuentran especificadas en el Diseño de la Prueba, como se detalla en el Anexo 6.

Este documento está estructurado por las siguientes partes:

1. **Información general del documento**.

   En esta sección, se establece la identificación única del documento, su emisión, responsables e historial de cambios.

2. **Alcance**

   Se identifica el alcance de la especificación de los casos de pruebas a ser ejecutados y describe las inclusiones, exclusiones, supuestos y limitaciones de las pruebas.

3. **Casos de Prueba**

   En este apartado se enumeran los casos de prueba a realizarse en la etapa de ejecución. Cada caso de prueba se estructura por:

   - Un identificador único.
   - Objetivo.
   - Condiciones previas.
   - Datos de entrada.
   - Resultados esperados.

4.3.4. Procedimiento de Prueba

El Procedimiento de Prueba específica el orden en que deben ejecutarse los casos de prueba, especificando la secuencia de ejecución, restricciones y acciones previas para cada caso, un ejemplo se muestra en el Anexo 7. La prueba se ejecuta utilizando los datos de entrada establecidos en los Casos de Prueba y el resultado obtenido es comparado con el resultado esperado para determinar si la prueba es satisfactoria o no.

El Procedimiento de la Prueba se encuentra estructurado por:

1. **Información general del documento.**

   En esta sección, se establece la identificación única del documento, su emisión, responsables e historial de cambios.

2. **Casos de Prueba**

   En este apartado, se identifican los Casos de Prueba a ser ejecutados, puede ser una copia o una referencia a los Casos de Prueba. Se debe especificar:

   - Identificador del Caso de Prueba
   - Datos de Entrada
   - Resultados Esperados

3. **Pasos por seguir**

   Se detallan los pasos que el Ingeniero de Pruebas debe seguir para ejecutar los Casos de Prueba.

4. **Resultados obtenidos**

   Se describe el resultado obtenido con la ejecución del Caso de Prueba y se determina si la prueba es satisfactoria o no.

4.3.5. Reporte de Defectos

El Reporte de Defectos corresponde al registro de los Casos de Pruebas ejecutados, cuyo resultado obtenido no coincide con el resultado esperado y por lo tanto, la prueba no es satisfactoria; los defectos encontrados se registran en el formulario destinado para el efecto, detallado en el Anexo 8.

4.3.6. Informe de Resultados

El Informe de Resultados es un registro de si una ejecución de Casos de Prueba específica se aprueba o no, es decir, si los resultados reales corresponden a los resultados esperados, si se observan desviaciones o si la ejecución prevista del Caso de Prueba no fue posible.

El resultado para cada Caso de Prueba se registra directamente en el procedimiento de prueba, en un marcador de posición reservado para este propósito. El resultado de la prueba, por tanto, no se considera como un documento independiente.

El Informe de Resultados es un documento gerencial que resume los resultados obtenidos en la ejecución de los casos de prueba, para determinar si el software se encuentra apto para su puesta en producción o si por el contrario todavía falta corregir algunos errores.

4.4. Aplicación del marco de trabajo

Una vez elaborado e implementado el marco de trabajo para las pruebas de software en el proceso de Gestión del Desarrollo de Software de la COAC Atuntaqui Ltda., se procede a aplicar el modelo en el proceso de evaluación del software, con un proyecto real, y posteriormente validar su impacto en la fase de mantenimiento.

La selección del software a evaluar se realiza en coordinación con el Administrador de Desarrollo, quien solicita que se aplique el marco de trabajo de pruebas al proyecto: "*Registro Digital de Declaración Juramentada*", debido a que se encuentra finalizado su desarrollo y está listo para iniciar la fase de pruebas.

4.4.1. Planificación de las pruebas

A continuación, se realiza con la planificación del proceso de pruebas para el proyecto seleccionado, obteniendo el siguiente resultado.

1. Información general del documento

La aplicación del marco de trabajo se usará para evaluar las funcionales del proyecto de software, ***Registro Digital de Declaración Juramentada*** de la COAC Atuntaqui Ltda.

2. Lista de requisitos

Los requisitos funcionales por evaluar se describen en la siguiente tabla.

Tabla 4.3. *Listado de requisitos funcionales del proyecto a evaluar.*

CÓDIGO	DESCRIPCIÓN
REQ-01	El sistema presenta un login, que permite el acceso al sistema, usando las credenciales de usuario del sistema WebCOOP. Al acceder al sistema se toma la cédula del usuario para extraer su información.
REQ-02	El sistema deberá presentar los datos de identificación del declarante.
REQ-03	El sistema debe presentar la información de bienes muebles e inmuebles vigentes para que se actualicen valores, se eliminen los registros que ya no se encuentren vigentes o se ingrese nueva información.
REQ-04	El sistema debe permitir la actualización de valores, el ingreso o eliminación de los registros de bienes inmuebles.
REQ-05	El sistema debe permitir la actualización de valores, el ingreso o eliminación de los registros de bienes muebles, vehículos.
REQ-06	El sistema debe permitir la actualización de valores, el ingreso o eliminación de otros bienes muebles.
REQ-07	El sistema debe permitir la actualización de valores, el ingreso o eliminación de dinero en efectivo, bancos, cooperativas u otros:
REQ-08	El sistema debe permitir la actualización de valores, el ingreso o eliminación de inversiones.
REQ-09	El sistema debe permitir la actualización de valores, el ingreso o eliminación de establecimientos o negocios adquiridos propios o de la sociedad conyugal.
REQ-10	El sistema debe permitir la actualización de valores, el ingreso o eliminación de información de deudas contraídas.
REQ-11	El sistema debe permitir la actualización de valores, el ingreso o eliminación del detalle de tarjetas de crédito.
REQ-12	Se solicite el ingreso de datos de separación de bienes/ capitulaciones matrimoniales (S/N), Liquidación de sociedad conyugal (S/N)
REQ-13	Realice cálculo del total de activos, pasivos y patrimonio y almacene la información, todos los registros asociados a la nueva declaración
REQ-14	Los registros deben contener, usuario que realiza el ingreso, modificación o eliminación, fecha del primer registro y fecha de actualización, estado de registro (Vigente, No vigente, Anulado)
REQ-15	Consulte datos del domicilio del declarante, datos de identificación del cónyuge o conviviente y datos institucionales y genere formulario de declaración juramentada
REQ-16	Si el usuario no presenta algún tipo de bien, la declaración debe presentar el mensaje “Ninguno” en las casillas correspondiente
REQ-17	Presente un mensaje que solicite al usuario la verificación de datos.
REQ-18	Permita la impresión de la declaración juramentada. Permita buscar las declaraciones juramentadas de todos los empleados por período.
REQ-19	Genere reporte de empleados vigentes con y sin declaración del período solicitado.
REQ-20	Permita notificar en forma automática a los empleados que no han ingresado la declaración juramentada.
REQ-21	Genere reporte detallado de bienes de bienes declarados por los empleados en los últimos tres períodos, mostrando los datos en forma comparativa.
REQ-22	Genere reporte de los totales generados por todos los empleados en los últimos tres años, mostrando los datos en forma comparativa.

Fuente: Autor.

3. Prioridad de requisitos

En compañía del especialista, se establece la prioridad de ejecución de las pruebas para cada uno de los requisitos funcionales. A continuación, se muestra la clasificación de los requisitos de acuerdo con la Tabla 7 de Prioridades de Prueba.

Tabla 4.4. *Prioridad de Prueba por requisito funcional.*

Código	Probabilidad de falla	Frecuencia de uso	Prioridad
REQ-01	Baja	Alta	**Media**
REQ-02	Baja	Alta	**Media**
REQ-03	Media	Media	**Media**
REQ-04	Alta	Alta	**Alta**
REQ-05	Alta	Media	**Alta**
REQ-06	Alta	Alta	**Alta**
REQ-07	Alta	Alta	**Alta**
REQ-08	Alta	Media	**Alta**
REQ-09	Alta	Media	**Alta**
REQ-10	Alta	Alta	**Alta**
REQ-11	Alta	Alta	**Alta**
REQ-12	Alta	Alta	**Alta**
REQ-13	Baja	Alta	**Media**
REQ-14	Media	Media	**Media**
REQ-15	Media	Media	**Media**
REQ-16	Media	Media	**Media**
REQ-17	Baja	Media	**Baja**
REQ-18	Baja	Media	**Baja**
REQ-19	Baja	Baja	**Baja**
REQ-20	Media	Media	**Media**
REQ-21	Baja	Media	**Baja**
REQ-22	Baja	Media	**Baja**

Fuente: Autor.

Tabla 4.5. *Consolidado de características a probar por prioridad de prueba*.

Prioridad	Probabilidad de falla	Frecuencia de uso	Nro. de Funcionalidades
Alta	Alta	Alta	**6**
Alta	Alta	Media	**3**
Media	Media	Media	**5**
Media	Baja	Alta	**3**
Baja	Baja	Media	**4**
Baja	Baja	Baja	**1**
	Total		**22**

Fuente: Autor.

4. Recursos de Prueba

Para la ejecución de las pruebas son necesarios los siguientes recursos.

Tabla 4.6. *Recursos necesarios para las pruebas.*

Tipo de Recursos	Descripción
Ambiente	- Servidor de base de datos Sybase 15.7. - Servidor de Internet Information Server 7. - Versión estable del software. - Computador portátil.
Herramientas	- Microsoft Office: Word, Excel. - Formatos de Pruebas establecidos
Datos	- Credenciales para el servidor de pruebas.
Recursos Humanos	- Líder de Pruebas - Ingeniero de Pruebas - Especialista (Abogado)

Fuente: Autor.

5. Cronograma de actividades

Se define el cronograma de actividades a realizar durante la fase de pruebas.

Tabla 4.7. *Cronograma de actividades.*

Actividades / Semanas	1	2	3	4	5	6
Diseño de la Prueba						
Construcción de Casos de Prueba						
Ejecución de Casos de Prueba						
Evaluación de las Pruebas						

Fuente: Autor.

4.4.2. Diseño de la Prueba

A continuación, el Líder de Pruebas identifica los Casos de Prueba a realizar durante la ejecución de las pruebas, tomando en cuenta la Prioridad de Prueba establecida para cada uno de los requisitos, en el Plan de Pruebas.

Tabla 4.8. *Casos de Prueba diseñados.*

Requisito	Caso de Prueba	Detalle del caso de prueba
REQ-04: El sistema debe permitir la actualización de valores, el ingreso o eliminación de los registros de bienes inmuebles.	CP-001	Ingresar bien inmueble con valores vacíos.
	CP-002	Ingresar bien inmueble con valores numéricos en cero.
	CP-003	Ingresar bien inmueble con valores numéricos negativos.
	CP-004	Ingresar bien inmueble con fechas incorrectas.
	CP-005	Ingresar bien inmueble con valores correctos.
	CP-006	Actualizar datos de un bien inmueble.
	CP-007	Eliminar datos de un bien inmueble.
	CP-008	Validar los catálogos del formulario.
REQ-06: El sistema debe permitir la actualización de valores, el ingreso o eliminación de otros bienes muebles.	CP-009	Ingresar bien mueble con valores vacíos.
	CP-010	Ingresar bien mueble con valores numéricos en cero.
	CP-011	Ingresar bien mueble con valores numéricos negativos.
	CP-012	Ingresar bien mueble con valores correctos.
	CP-013	Actualizar datos de un bien mueble.
	CP-014	Eliminar datos de un bien mueble.
	CP-015	Validar los catálogos del formulario.
REQ-07: El sistema debe permitir la actualización de valores, el ingreso o eliminación de dinero en efectivo, bancos, cooperativas u otros.	CP-016	Ingresar cuenta con valores vacíos.
	CP-017	Ingresar cuenta con valores numéricos en cero.
	CP-018	Ingresar cuenta con valores numéricos negativos.
	CP-019	Ingresar cuenta con valores correctos.
	CP-020	Actualizar datos de una cuenta.
	CP-021	Eliminar datos de una cuenta.
	CP-022	Validar los catálogos del formulario.
REQ-10: El sistema debe permitir la actualización de valores, el ingreso o eliminación de información de deudas contraídas.	CP-023	Ingresar deuda con valores vacíos.
	CP-024	Ingresar deuda con valores numéricos en cero.
	CP-025	Ingresar deuda con valores numéricos negativos.
	CP-026	Ingresar deuda con valores correctos.
	CP-027	Actualizar datos de deuda.
	CP-028	Eliminar datos de deuda.
	CP-029	Validar los catálogos del formulario.

REQ-11: El sistema debe permitir la actualización de valores, el ingreso o eliminación del detalle de tarjetas de crédito.	CP-030	Ingresar tarjeta de crédito con valores vacíos.
	CP-031	Ingresar tarjeta de crédito con valores numéricos en cero.
	CP-032	Ingresar tarjeta de crédito con valores numéricos negativos.
	CP-033	Ingresar tarjeta de crédito con fechas incorrectas.
	CP-034	Ingresar tarjeta de crédito con número de tarjeta incorrecto.
	CP-035	Ingresar tarjeta de crédito con valores correctos.
	CP-036	Actualizar datos de tarjeta de crédito.
	CP-037	Eliminar datos de tarjeta de crédito.
	CP-038	Validar los catálogos del formulario.
REQ-12: Se solicite el ingreso de datos de separación de bienes/capitulaciones matrimoniales (S/N), liquidación de sociedad conyugal (S/N).	CP-039	Ingresar formulario con valores vacíos.
	CP-040	Ingresar separación de bienes con valor N o S.
	CP-041	Ingresar separación de bienes con valor diferente a N o S.
	CP-042	Ingresar liquidación de sociedad con valor N o S.
	CP-043	Ingresar liquidación de sociedad con valor diferente a N o S.
REQ-05: El sistema debe permitir la actualización de valores, el ingreso o eliminación de los registros de bienes muebles, vehículos.	CP-044	Ingresar vehículo con valores vacíos.
	CP-045	Ingresar vehículo con valores numéricos en cero.
	CP-046	Ingresar vehículo con valores numéricos negativos.
	CP-047	Ingresar vehículo con placa incorrecta.
	CP-048	Ingresar vehículo con valores correctos.
	CP-049	Actualizar datos del vehículo.
	CP-050	Eliminar datos del vehículo.
	CP-051	Validar los catálogos del formulario.
REQ-08: El sistema debe permitir la actualización de valores, el ingreso o eliminación de inversiones.	CP-052	Ingresar inversión con valores vacíos.
	CP-053	Ingresar inversión con valores numéricos en cero.
	CP-054	Ingresar inversión con valores numéricos negativos.
	CP-055	Ingresar inversión con fechas incorrectas.
	CP-056	Ingresar inversión con valores correctos.
	CP-057	Actualizar datos de una inversión.
	CP-058	Eliminar datos de una inversión.
	CP-059	Validar los catálogos del formulario.
REQ-09: El sistema debe permitir la actualización de valores, el ingreso o eliminación de establecimientos o negocios adquiridos propios o de la sociedad conyugal.	CP-060	Ingresar formulario con valores vacíos.
	CP-061	Ingresar negocio con valores numéricos en cero.
	CP-062	Ingresar negocio con valores numéricos negativos.
	CP-063	Ingresar negocio con RUC incorrecto.
	CP-064	Ingresar negocio con valores correctos.
	CP-065	Actualizar datos de un negocio.
	CP-066	Eliminar datos de un negocio.
	CP-067	Validar los catálogos del formulario.

REQ-03: El sistema debe presentar la información de bienes muebles e inmuebles vigentes para que se actualicen valores, se eliminen los registros que ya no se encuentren vigentes o se ingrese nueva información.	CP-068	Validar que la información presentada sea de registros vigentes.
	CP-069	Ingresar datos de un nuevo bien.
	CP-070	Actualizar los datos de un bien existente.
	CP-071	Eliminar datos de un bien existente.
REQ-14: Los registros deben contener, usuario que realiza el ingreso, modificación o eliminación, fecha del primer registro y fecha de actualización, estado de registro (Vigente, No vigente, Anulado).	CP-072	Ingresar un registro y validar su información de control.
	CP-073	Modificar un registro y validar su información de control.
	CP-074	Eliminar un registro y validar su información de control.
REQ-15: Consulte datos del domicilio del declarante, datos de identificación del cónyuge o conviviente y datos institucionales y genere formulario de declaración juramentada.	CP-075	Generar y validar la información del formulario de declaración juramentada, contenga los datos correctos ingresados.
REQ-16: Si el usuario no presenta algún tipo de bien, la declaración debe presentar el mensaje “Ninguno” en las casillas correspondiente.	CP-076	Generar y validar la información del formulario de declaración juramentada de un colaborador sin bienes ingresados
REQ-20: Permita notificar en forma automática a los empleados que no han ingresado la declaración juramentada.	CP-077	Verificar las notificaciones de los colaboradores que aún no registran la declaración juramentada.
REQ-01: Al acceder al sistema se toma la cedula del usuario para extraer su información	CP-078	Ingresar con credenciales vacías.
	CP-079	Ingresar con credenciales incorrectas.
	CP-080	Ingresar con credenciales correctas
REQ-02: Una vez ingresada la cédula, el sistema deberá presentar los datos de identificación del declarante.	CP-081	Ingresar con usuario sin datos de identificación ingresada.
	CP-082	Ingresar con usuario de estado civil Soltero.
	CP-083	Ingresar con usuario de estado civil Casado.
REQ-13: Realice cálculo del total de activos, pasivos y patrimonio y almacene la información, todos los registros asociados a la nueva declaración.	CP-084	Consultar y verificar que la declaración juramentada con activos y pasivos sea correcta.
	CP-085	Consultar y verificar que la declaración juramentada sin activos y pasivos sea correcta.
REQ-17: Presente un mensaje que solicite al usuario la verificación de datos.	CP-086	Ingresar declaración juramentada y validar el mensaje de confirmación.
REQ-18: Permita la impresión de la declaración juramentada.	CP-087	Consultar e imprimir la declaración juramentada.
	CP-088	Consultar las declaraciones de los empleados en un periodo de tiempo.

REQ-21: Genere reporte detallado de bienes declarados por los empleados en los últimos tres períodos, mostrando los datos en forma comparativa.	CP-089	Generar la consulta de los bienes declarados por los empleados en los tres últimos periodos.
	CP-090	Generar la consulta de comparación de los datos históricos de los últimos tres periodos.
REQ-22: Genere reporte de los totales generados por todos los empleados en los últimos tres años, mostrando los datos en forma comparativa.	CP-091	Generar la consulta de totales de todos los empleados en los tres últimos periodos.
	CP-092	Generar la consulta de comparación de totales de todos los empleados en los tres últimos periodos.
REQ-19: Genere reporte de empleados vigentes con y sin declaración del período solicitado.	CP-093	Generar y validar la consulta de los empleados vigentes con y sin declaración juramentada.

Fuente: Autor.

4.4.3. Construcción de las Pruebas

Una vez definidos los Casos de Prueba a realizar durante el proceso de ejecución de las pruebas, el Ingeniero de Pruebas construye los Casos de Prueba estableciendo los datos de entrada y los resultados esperados en cada uno de los casos planteados por el Líder de Pruebas.

Tabla 4.9. *Construcción de los Casos de Prueba.*

Caso de Prueba	Datos de Entrada	Resultado esperado
CP-001: Ingresar bien inmueble con valores vacíos.	*Ninguno*	**Mensaje:** Debe ingresar un valor.
CP-002: Ingresar bien inmueble con valores numéricos en cero.	Fecha Adquisición: 01/01/2017 Fecha Inscripción: 01/01/2017 Ciudad: Ibarra Tipo de Bien: Vivienda Dirección: Princesa Paccha CS3 Clave Catastral: 100101010508037 Valor del bien: 0	**Mensaje:** El valor del bien inmueble debe ser mayor a cero.
CP-003: Ingresar bien inmueble con valores numéricos negativos.	Fecha Adquisición: 01/01/2017 Fecha Inscripción: 01/01/2017 Ciudad: Ibarra Tipo de Bien: Vivienda Dirección: Princesa Paccha CS3 Clave Catastral: 100101010508037 Valor del bien: -100	**Mensaje:** El valor del bien inmueble debe ser mayor a cero.
CP-004: Ingresar bien inmueble con fechas incorrectas.	Fecha Adquisición: 10/01/2020 Fecha Inscripción: 09/01/2020 Ciudad: Ibarra Tipo de Bien: Vivienda Dirección: Princesa Paccha CS3	**Mensaje:** Las fechas deben ser menores a la fecha actual.

	Clave Catastral: 100101010508037 Valor del bien: 100	
CP-005: Ingresar bien inmueble con datos correctos.	Fecha Adquisición: 01/01/2017 Fecha Inscripción: 01/01/2017 Ciudad: Ibarra Tipo de Bien: Vivienda Dirección: Princesa Paccha CS3 Clave Catastral: 100101010508037 Valor del bien: 100	**Mensaje:** El bien inmueble se ingresó satisfactoriamente.
CP-006: Actualizar datos de un bien inmueble.	Código del bien inmueble: ingresado en el caso de prueba CP-005. Valor del bien: 10000	Mensaje: Los datos del bien inmueble se actualizaron satisfactoriamente.
CP-007: Eliminar datos de un bien inmueble.	Código del bien inmueble: ingresado en el caso de prueba CP-005.	Mensaje: El bien inmueble se eliminó correctamente.
CP-008: Validar los catálogos del formulario.	Ciudad: Catálogo de ciudades Tipo de Bien: Catálogo de tipo de bien	Validar que los catálogos de ciudad y tipo de bien muestren solo los registros vigentes.
CP-009: Ingresar bien mueble con valores vacíos.	Ninguno	Mensaje: Debe ingresar un valor.
CP-010: Ingresar bien mueble con valores numéricos en cero.	Tipo de bien: Equipo de oficina Valor del bien: 0	Mensaje: El valor del bien debe ser mayor a cero.
CP-011: Ingresar bien mueble con valores numéricos negativos.	Tipo de bien: Equipo de oficina Valor del bien: -1000	Mensaje: El valor del bien debe ser mayor a cero.
CP-012: Ingresar bien mueble con valores correctos.	Tipo de bien: Equipo de oficina Valor del bien: 1000	Mensaje: El bien mueble se ingresó satisfactoriamente.
CP-013: Actualizar datos de un bien mueble.	Código del bien mueble: ingresado en el caso de prueba CP-012. Valor del bien: 3000	Mensaje: Los datos del bien mueble se actualizaron satisfactoriamente.
CP-014: Eliminar datos de un bien mueble.	Código del bien mueble: ingresado en el caso de prueba CP-012.	Mensaje: El bien mueble se eliminó correctamente.
CP-015: Validar los catálogos del formulario.	Tipo de bien: Catálogo de tipo de bienes muebles	Validar que el catálogo de tipo de bien muestre solo los registros vigentes.
CP-016: Ingresar cuenta con valores vacíos.	Ninguno	Mensaje: Debe ingresar un valor.
CP-017: Ingresar cuenta con valores numéricos en cero.	Ciudad: Ibarra Institución: COAC Atuntaqui Ltda. Tipo de cuenta: Ahorros Número de cuenta: 401110123456 Saldo: 0	Mensaje: El saldo de la cuenta debe ser mayor a cero.
CP-018: Ingresar cuenta con valores numéricos negativos.	Ciudad: Ibarra Institución: COAC Atuntaqui Ltda. Tipo de cuenta: Ahorros	Mensaje: El saldo de la cuenta debe ser mayor a cero.

	Número de cuenta: 401110123456 Saldo: -100	
CP-019: Ingresar cuenta con valores correctos.	Ciudad: Ibarra Institución: COAC Atuntaqui Ltda. Tipo de cuenta: Ahorros Número de cuenta: 401110123456 Saldo: 100	Mensaje: La cuenta bancaria se ingresó satisfactoriamente.
CP-020: Actualizar datos de una cuenta.	Código de la cuenta: ingresada en el caso de prueba CP-019. Saldo: 1000	Mensaje: Los datos de la cuenta bancaria se actualizaron satisfactoriamente.
CP-021: Eliminar datos de una cuenta.	Código de la cuenta: ingresada en el caso de prueba CP-019.	Mensaje: La cuenta bancaria se eliminó correctamente.
CP-022: Validar los catálogos del formulario.	Ciudad: Catálogo de ciudades Institución: Catálogo de instituciones financieras. Tipo de cuenta: Catálogo de tipos de cuentas	Validar que los catálogos de ciudad, institución financiera y tipo de cuenta muestren solo los registros vigentes.
CP-023: Ingresar deuda con valores vacíos.	Ninguno	Mensaje: Debe ingresar un valor.
CP-024: Ingresar deuda con valores numéricos en cero.	Tipo de crédito: Consumo Institución: COAC Atuntaqui Ltda. Garantía: Hipotecaria Monto: 0 Saldo: 0	Mensaje: El monto y saldo del crédito deben ser mayores a cero.
CP-025: Ingresar deuda con valores numéricos negativos.	Tipo de crédito: Consumo Institución: COAC Atuntaqui Ltda. Garantía: Hipotecaria Monto: -10000 Saldo: -5000	Mensaje: El monto y saldo del crédito deben ser mayores a cero.
CP-026: Ingresar deuda con valores correctos.	Tipo de crédito: Consumo Institución: COAC Atuntaqui Ltda. Garantía: Hipotecaria Monto: 10000 Saldo: 5000	Mensaje: Los datos del crédito se registraron satisfactoriamente.
CP-027: Actualizar datos de deuda.	Código del crédito: ingresado en el caso de prueba CP-026. Monto: 20000	Mensaje: Los datos del crédito se actualizaron satisfactoriamente.
CP-028: Eliminar datos de deuda.	Código del crédito: ingresado en el caso de prueba CP-026.	Mensaje: El crédito se eliminó correctamente
CP-029: Validar los catálogos del formulario.	Tipo de crédito: Catálogo Tipos de Crédito Institución: Catálogo de instituciones financieras. Garantía: Catálogo de Tipos de Garantías	Validar que los catálogos de tipo de crédito, institución financiera y tipo de garantía muestren solo los registros vigentes.
CP-030: Ingresar tarjeta de crédito con valores vacíos.	Ninguno	Mensaje: Debe ingresar un valor.

CP-031: Ingresar tarjeta de crédito con valores numéricos en cero.	Emisor: Banco del Pacífico Número: 1234567890123456 Fecha expedición: 01/01/2018 Cupo: 0 Saldo: 0	Mensaje: El cupo de la tarjeta de crédito debe ser mayor a cero.
CP-032: Ingresar tarjeta de crédito con valores numéricos negativos.	Emisor: Banco del Pacífico Número: 1234567890123456 Fecha expedición: 01/01/2018 Cupo: -1000 Saldo: -100	Mensaje: El cupo y saldo de la tarjeta de crédito deben ser mayores a cero.
CP-033: Ingresar tarjeta de crédito con fechas incorrectas.	Emisor: Banco del Pacífico Número: 1234567890123456 Fecha expedición: 01/01/2020 Cupo: 1000 Saldo: 100	Mensaje: La fecha de expedición debe ser menor a la fecha actual.
CP-034: Ingresar tarjeta de crédito con número de tarjeta incorrecta.	Emisor: Banco del Pacífico Número: 1234 Fecha expedición: 01/01/2020 Cupo: 1000 Saldo: 100	Mensaje: El número de tarjeta no es correcto.
CP-035: Ingresar tarjeta de crédito con valores correctos.	Emisor: Banco del Pacífico Número: 1234567890123456 Fecha expedición: 01/01/2017 Cupo: 1000 Saldo: 100	Mensaje: La tarjeta de crédito se ingresó correctamente.
CP-036: Actualizar datos de tarjeta de crédito.	Código de la tarjeta: ingresada en el caso de prueba CP-035. Saldo: 500	Mensaje: Los datos de la tarjeta de crédito se actualizaron correctamente.
CP-037: Eliminar datos de tarjeta de crédito.	Código de la tarjeta: ingresada en el caso de prueba CP-035.	Mensaje: La tarjeta de crédito se eliminó correctamente.
CP-038: Validar los catálogos del formulario.	Emisor: Catálogo de instituciones financieras.	Validar que el catálogo de instituciones financieras muestre solo los registros vigentes.
CP-039: Ingresar formulario con valores vacíos.	Ninguno	Mensaje: Debe ingresar un valor.
CP-040: Ingresar separación de bienes con valor N o S.	Separación de bienes: S	Mensaje: Declaración ingresada correctamente.
CP-041: Ingresar separación de bienes con valor diferente a N o S.	Separación de bienes: X	Mensaje: El valor de la Separación de bienes debe ser S o N.
CP-042: Ingresar liquidación de sociedad con valor N o S.	Liquidación de sociedad: S	Mensaje: Declaración ingresada correctamente.
CP-043: Ingresar liquidación de sociedad con valor diferente a S o N.	Liquidación de sociedad: X	Mensaje: El valor de la Liquidación de la sociedad debe ser S o N.

CP-044: Ingresar vehículo con valores vacíos.	Ninguno	Mensaje: Debe ingresar un valor.
CP-045: Ingresar vehículo con valores numéricos en cero.	Clase: Auto, Marca: Renault Modelo: Sandero, Año: 2012 Placa: PBV0124, Avalúo: 0	Mensaje: El valor del avalúo debe ser mayor a cero.
CP-046: Ingresar vehículo con valores numéricos negativos.	Clase: Auto, Marca: Renault Modelo: Sandero, Año: 2012 Placa: PBV0124, Avalúo: -10000	Mensaje: El valor del avalúo debe ser mayor a cero.
CP-047: Ingresar vehículo con placa incorrecta.	Clase: Auto, Marca: Renault Modelo: Sandero, Año: 2012 Placa: 1234, Avalúo: 10000	Mensaje: El número de la placa es incorrecto.
CP-048: Ingresar vehículo con valores correctos.	Clase: Auto, Marca: Renault Modelo: Sandero, Año: 2012 Placa: PVB0124, Avalúo: 10000	Mensaje: Los datos del vehículo fueron ingresados correctamente.
CP-049: Actualizar datos del vehículo.	Código del vehículo: ingresado en el caso de prueba CP-050 Avalúo: 15000	Mensaje: Los datos del vehículo fueron actualizados correctamente.
CP-050: Eliminar datos del vehículo.	Código del vehículo: ingresado en el caso de prueba CP-050	Mensaje: El vehículo se eliminó correctamente.
CP-051: Validar los catálogos del formulario.	Clase: Catálogo de tipo de vehículo. Marca: Catálogo de marcas. Modelo: Catálogo de modelos.	Validar que los catálogos de tipo de vehículo, marca y modelos muestren los registros vigentes. Verificar que el catálogo de Modelos este atado al catálogo de Marcas.
CP-052: Ingresar inversión con valores vacíos.	Ninguno	Mensaje: Debe ingresar un valor.
CP-053: Ingresar inversión con valores numéricos en cero.	Ciudad: Ibarra Institución: COAC Atuntaqui Ltda. Tipo de inversión: Plazo Fijo Fecha Inversión: 01/01/2018 Saldo: 0	Mensaje: El valor de la inversión debe ser mayor a cero.
CP-054: Ingresar inversión con valores numéricos negativos.	Ciudad: Ibarra Institución: COAC Atuntaqui Ltda. Tipo de inversión: Plazo Fijo Fecha Inversión: 01/01/2018 Saldo: -10000	Mensaje: El valor de la inversión debe ser mayor a cero.
CP-055: Ingresar inversión con fechas incorrectas.	Ciudad: Ibarra Institución: COAC Atuntaqui Ltda. Tipo de inversión: Plazo Fijo Fecha Inversión: 01/01/2021 Saldo: 10000	Mensaje: La fecha de la inversión debe ser menor a la fecha actual.
CP-056: Ingresar inversión con valores correctos.	Ciudad: Ibarra Institución: COAC Atuntaqui Ltda. Tipo de inversión: Plazo Fijo Fecha Inversión: 01/01/2018	Mensaje: La inversión se ingresó correctamente.

	Saldo: 10000	
CP-057: Actualizar datos de una inversión.	Código de la inversión: ingresada en el caso de prueba CP-056. Saldo: 20000	Mensaje: Los datos de la inversión se actualizaron correctamente.
CP-058: Eliminar datos de una inversión.	Código de la inversión: ingresada en el caso de prueba CP-056.	Mensaje: La inversión se eliminó correctamente.
CP-059: Validar los catálogos del formulario.	Ciudad: Catálogo de ciudades Institución: Catálogo de instituciones financieras. Tipo de inversión: Catálogo de tipos de inversiones.	Validar que los catálogos de ciudad, instituciones financieras y tipos de inversiones muestren solo los registros vigentes.
CP-060: Ingresar negocio con valores vacíos.	Ninguno	Mensaje: Debe ingresar un valor.
CP-061: Ingresar negocio con valores numéricos en cero.	Tipo de contribuyente: Jurídico RUC: 1003090733001 Act. Económica: Venta de Software Monto Inversión: 0 Utilidad Anual: 0	Mensaje: El monto de la inversión debe ser mayor a cero.
CP-062: Ingresar negocio con valores numéricos negativos.	Tipo de contribuyente: Jurídico RUC: 1003090733001 Act. Económica: Venta de Software Monto Inversión: -1000 Utilidad Anual: 0	Mensaje: El monto de la inversión debe ser mayor a cero.
CP-063: Ingresar negocio con RUC incorrecto.	Tipo de contribuyente: Jurídico RUC: 12345 Act. Económica: Venta de Software Monto Inversión: 1000 Utilidad Anual: 0	Mensaje: El valor del RUC es incorrecto.
CP-064: Ingresar negocio con valores correctos.	Tipo de contribuyente: Jurídico RUC: 1003090733001 Act. Económica: Venta de Software Monto Inversión: 1000 Utilidad Anual: 0	Mensaje: Los datos del negocio se ingresaron correctamente.
CP-065: Actualizar datos de un negocio.	Código del negocio: ingresado en el caso de prueba CP-064. Monto Inversión: 20000	Mensaje: Los datos del negocio fueron actualizados correctamente.
CP-066: Eliminar datos de un negocio.	Código del negocio: ingresado en el caso de prueba CP-064.	Mensaje: El negocio fue eliminado correctamente.
CP-067: Validar los catálogos del formulario.	Tipo de contribuyente: Catálogo de tipo de contribuyente	Verificar que el catálogo de tipo de contribuyente solo muestre los registros vigentes.
CP-068: Validar que la información presentada sea de registros vigentes.	Cédula: 1003090733	Verificar que la información presentada en la declaración sea correcta.
CP-069: Ingresar datos de un nuevo bien.	Fecha Adquisición: 01/01/2018 Fecha Inscripción: 01/01/2018	Mensaje: El bien fue ingresado correctamente.

	Ciudad: Atuntaqui Tipo de Bien: Terreno Dirección: Natabuela Clave Catastral: 100101024578401 Valor del bien: 1000	
CP-070: Actualizar los datos de un bien existente.	Código del bien: ingresado en el caso de prueba CP-069. Clave Catastral: 100101032456802 Valor del bien: 1000	Mensaje: El bien fue modificado satisfactoriamente.
CP-071: Eliminar datos de un bien existente.	Código del bien: ingresado en el caso de prueba CP-069.	Mensaje: El bien se eliminó correctamente.
CP-072: Ingresar un registro y validar su información de control.	Fecha Adquisición: 01/01/2018 Fecha Inscripción: 01/01/2018 Ciudad: Atuntaqui Tipo de Bien: Terreno Dirección: Natabuela Clave Catastral: 100101024578401 Valor del bien: 1000	Verificar que el usuario, terminal y fecha del ingreso se registren en la base de datos.
CP-073: Modificar un registro y validar su información de control.	Código del bien: ingresado en el caso de prueba CP-072. Clave Catastral: 100101032456802 Valor del bien: 1000	Verificar que el usuario, terminal y fecha de la modificación se registren en la base de datos.
CP-074: Eliminar un registro y validar su información de control.	Código del bien: ingresado en el caso de prueba CP-072.	Verificar que el usuario, terminal y fecha de eliminación se registren en la base de datos.
CP-075: Generar y validar la información del formulario de declaración juramentada, contenga los datos correctos ingresados.	Cédula: 1003090733	Verificar que los datos de la declaración juramentada sean correctos.
CP-076: Generar y validar la información del formulario de declaración juramentada de un colaborador sin bienes ingresados.	Cédula: 1002722229	Verificar que en los tipos de bienes donde no se encuentra ingresado ningún bien se encuentre la palabra NINGUNO.
CP-077: Verificar las notificaciones de los usuarios que aún no registran la declaración juramentada.	Opción: Enviar notificaciones	Verificar el envío de las notificaciones a los colaboradores que no tienen registrada la declaración.
CP-078: Ingresar con credenciales vacías.	Ninguno	Mensaje: Credenciales incorrectas.
CP-079: Ingresar con credenciales incorrectas.	Usuario: wcardenas Clave: wcardenas	Mensaje: Credenciales incorrectas.
CP-080: Ingresar con credenciales correctas	Usuario: wcardenas Clave: 49004900	Mensaje: Bienvenido Wilson
CP-081: Ingresar con usuario sin datos de identificación.	Usuario: sportilla Clave: 49004900	Mensaje: Primero debe registrar su información personal.

CP-082: Ingresar con usuario de estado civil Soltero.	Usuario: dmedina Clave: 49004900	Verificar que no solicite ingresar datos del cónyuge.
CP-083: Ingresar con usuario de estado civil Casado.	Usuario: wcardenas Clave: 49004900	Verificar la información del cónyuge.
CP-084: Consultar y verificar que la declaración juramentada sea correcta.	Cédula: 1003090733	Verificar que la información de activos y pasivos ingresada sea correcta.
CP-085: Consultar y verificar que la declaración juramentada sin activos y pasivos sea correcta.	Cédula: 1002722229	Verificar que la información de la declaración juramentada no presente información ingresada. Presente el texto NINGUNA.
CP-086: Ingresar declaración juramentada y validar el mensaje de confirmación.	Acción: Guardar declaración	Mensaje: La declaración fue generada satisfactoriamente.
CP-087: Consultar e imprimir la declaración juramentada.	Cédula: 1003090733	Generar la declaración y verificar que esté de acuerdo con el formato establecido.
CP-088: Consultar las declaraciones de los empleados por tiempo.	Fecha inicio: 01/01/2018 Fecha fin: 31/12/2018	Generar el reporte y verificar que muestre todas las declaraciones ingresadas durante el periodo.
CP-089: Generar la consulta de los bienes declarados por los empleados en los tres últimos periodos.	Cédula: 1003090733	Verificar que se genere el reporte y muestre los bienes declarados en los últimos tres periodos.
CP-090: Generar la consulta de comparación de los datos históricos de los últimos tres periodos.	Cédula: 1003090733	Verificar que se genere el reporte y muestre una comparativa de las declaraciones ingresadas en los últimos tres periodos.
CP-091: Generar la consulta de totales de todos los empleados en los tres últimos periodos.	Opción: Últimas Declaraciones	Verificar que se genere el reporte y muestre las últimas tres declaraciones de todos los empleados.
CP-092: Generar la consulta de comparación de totales de todos los empleados en los tres últimos periodos.	Opción: Comparar Declaraciones	Verificar que se genere el reporte y muestre una comparación de las declaraciones ingresadas en los últimos tres periodos por todos los empleados.
CP-093: Generar y validar la consulta de los empleados vigentes con y sin declaración juramentada.	Opción: Declaraciones por empleado	Verificar que se genere y muestre un listado de todos los empleados indicando si ya registro o no la declaración del periodo actual.

Fuente: Autor.

4.4.4. Ejecución de las Pruebas

En la etapa de ejecución de las pruebas, el Ingeniero de Pruebas ejecuta cada uno de los casos de prueba planteados, registrando el resultado satisfactorio o no de la prueba, tomando en cuenta el resultado esperado.

A continuación, se detalla el resultado obtenido en las pruebas realizadas.

Tabla 4.10. *Resultados obtenidos con la ejecución de los Casos de Prueba.*

Caso de Prueba	Ciclo 1	Ciclo 2	Ciclo 3
CP-001	✓	✓	-
CP-002	✗	✓	✓
CP-003	✓	✓	-
CP-004	✗	✓	✓
CP-005	✓	✓	✓
CP-006	✓	✓	✓
CP-007	✓	✓	✓
CP-008	✓	✓	-
CP-009	✓	✓	-
CP-010	✗	✓	✓
CP-011	✓	✓	-
CP-012	✓	✓	✓
CP-013	✓	✓	✓
CP-014	✓	✓	✓
CP-015	✓	✓	-
CP-016	✓	✓	-
CP-017	✗	✓	✓
CP-018	✓	✓	-
CP-019	✓	✓	✓
CP-020	✓	✓	✓
CP-021	✓	✓	✓
CP-022	✗	✓	✓
CP-023	✓	✓	-
CP-024	✗	✓	✓
CP-025	✓	✓	-
CP-026	✓	✓	✓
CP-027	✓	✓	✓
CP-028	✓	✓	✓
CP-029	✓	✓	-
CP-030	✓	✓	-
CP-031	✗	✓	✓
CP-032	✓	✓	-
CP-033	✗	✓	✓
CP-034	✗	✓	✓
CP-035	✓	✓	✓
CP-036	✓	✓	✓
CP-037	✓	✓	✓
CP-038	✗	✓	✓
CP-039	✓	✓	-
CP-040	✓	✓	✓
CP-041	✓	✓	-
CP-042	✓	✓	✓

CP-043	✓	✓	-
CP-044	✓	✓	-
CP-045	✗	✓	✓
CP-046	✓	✓	-
CP-047	✗	✓	✓
CP-048	✓	✓	✓
CP-049	✓	✓	✓
CP-050	✓	✓	✓
CP-051	✗	✓	✓
CP-052	✓	✓	-
CP-053	✗	✓	✓
CP-054	✓	✓	-
CP-055	✗	✓	✓
CP-056	✓	✓	✓
CP-057	✓	✓	✓
CP-058	✓	✓	✓
CP-059	✓	✓	-
CP-060	✓	✓	-
CP-061	✗	✓	✓
CP-062	✓	✓	-
CP-063	✗	✗	✓
CP-064	✗	✓	✓
CP-065	✓	✓	✓
CP-066	✓	✓	✓
CP-067	✓	✓	-
CP-068	✓	✓	✓
CP-069	✓	✓	-
CP-070	✓	✓	-
CP-071	✓	✓	-
CP-072	✓	✓	-
CP-073	✓	✓	-
CP-074	✓	✓	-
CP-075	✓	✓	-
CP-076	✓	✓	-
CP-077	✗	✓	✓
CP-078	✓	✓	-
CP-079	✓	✓	-
CP-080	✓	✓	-
CP-081	✗	✓	✓
CP-082	✗	✓	✓
CP-083	✓	✓	-
CP-084	✓	✓	✓
CP-085	✓	✓	✓
CP-086	✓	✓	✓

CP-087	✓	✓	✓
CP-088	✓	✓	-
CP-089	✓	✓	-
CP-090	✓	✓	-
CP-091	✓	✓	-
CP-092	✓	✓	-
CP-093	✓	✓	-

Fuente: Autor.

4.4.5. Monitoreo y Control de las Pruebas

En cada uno de los ciclos de pruebas ejecutados, se realiza el análisis de los resultados obtenidos en las pruebas, para determinar si el resultado de estas cumple con los parámetros de satisfacción establecidos en el Plan de Pruebas y proceder con la publicación del software.

En el caso de estudio presentado, se obtuvieron los siguientes resultados.

Tabla 4.11. *Métricas obtenidas en el caso de estudio.*

Métricas	Ciclo 1	Ciclo 2	Ciclo 3
Especificaciones de Prueba EP	22	22	22
Casos de Prueba Planificados CPP	93	93	51
Casos de Prueba Ejecutados CPE	93	93	51
Casos de Prueba Satisfactorios CPS	72	92	51
Casos de Prueba con Defectos CPD	21	1	0
Cobertura de Pruebas (CPE/CPP)	100 %	100 %	100 %
Madurez de las Pruebas (CPS/CPP)	77 %	99 %	100 %
Densidad de Defectos (CPD/EP)	95 %	5 %	0 %

Fuente: Autor.

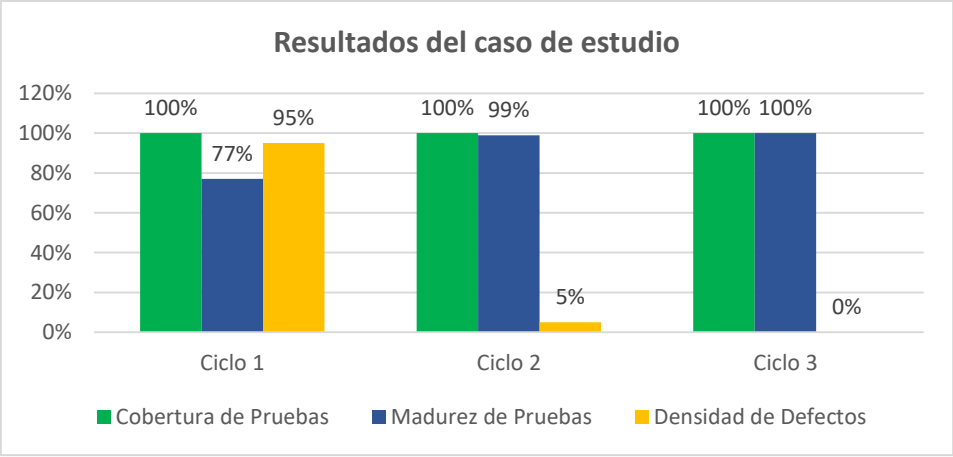


Figura 4.7. Resultados obtenidos en el caso de estudio presentado.
Fuente: Autor

Como se puede observar en la Figura 4.7, la *Cobertura de Pruebas* en los tres ciclos de pruebas es del cien por ciento, es decir, en los tres ciclos se ejecutaron todos los casos de pruebas especificados.

La métrica de *Madurez de las Pruebas* inicia con un valor del 77% en el primer ciclo, incrementa al 99% en el segundo ciclo y finaliza con un valor del 100% en el tercer ciclo, lo que significa que, conforme se ejecutaban las pruebas, los errores de software detectados fueron corregidos y al final todos los casos de pruebas diseñados obtuvieron un resultado satisfactorio.

La *Densidad de Defectos* alcanzada en el primer ciclo fue del 95%, con la corrección de los errores detectados, en el segundo ciclo se disminuye al 5% y en el tercer ciclo se termina con un valor del 0%, lo que significa que todos los errores de software fueron corregidos.

4.4.6. Finalización de las Pruebas

Por último, el Administrador de Desarrollo tomando en cuenta los resultados obtenidos en los diferentes ciclos de ejecución de las pruebas y en coordinación con el Director de Tecnología y Líder del Proyecto pueden dar por finalizado el proceso de ejecución de las pruebas y definir si el software está listo para publicarse en el ambiente de producción.

En el caso de estudio, fueron necesario tres ciclos de pruebas para obtener un resultado satisfactorio de las pruebas, y el software se encuentre listo para su liberación.

4.5. Resultados obtenidos con la propuesta

Una vez implementado el marco de trabajo para pruebas de software en el proceso de Gestión de Desarrollo de Software, de la Dirección de Tecnología de la COAC Atuntaqui Ltda., y aplicado en el caso de estudio, se pretende determinar el impacto obtenido en las fases de pruebas y mantenimiento del software.

4.5.1. Impacto de la propuesta en el proceso de evaluación del software

Para determinar el porcentaje de aceptación que alcanza la propuesta para el proceso de evaluación del software, se realiza una última encuesta al personal de desarrollo.

Los resultados de la encuesta se presentan en la siguiente gráfica.

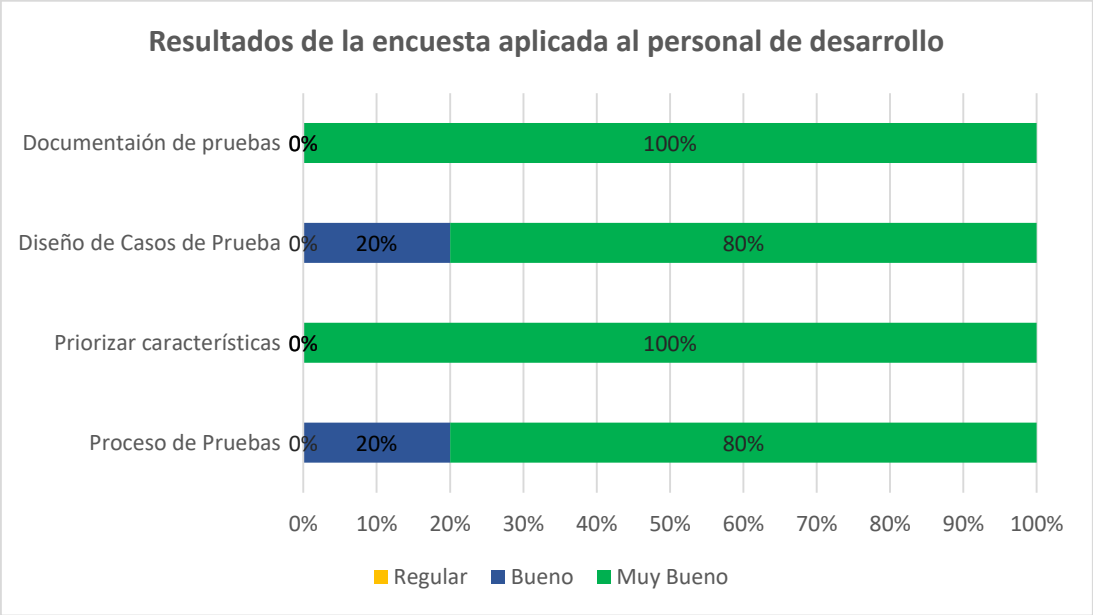


Figura 4.8. Resultados de la segunda encuesta aplicada al personal de desarrollo.

Fuente: Autor

Se observa que el 80% de los encuestados califican al ***proceso de pruebas de software*** como *Muy Bueno* y el 20% lo califica como *Bueno*. El 80% de los encuestados, piensa que el proceso de ***diseño de casos de prueba*** es *Muy Bueno*, mientras que un 20% cree que es *Bueno*. Por último, todos los encuestados, califican al proceso de ***priorización de las características a probar*** y al ***proceso de documentación*** de las pruebas como *Muy Bueno*.

La percepción final, por parte del personal de desarrollo, al proceso de pruebas de software, con la implementación del marco de trabajo es: un 0% lo considera *Regular*, un 10% cree que es *Bueno* y el 90% piensa que es *Muy Bueno*.

Con los datos obtenidos en la encuesta, se determina que el porcentaje de aceptación de la propuesta alcanza un 90%, logrando una mejora considerable en la apreciación de la fase de pruebas de software, por parte del personal de desarrollo de la cooperativa.

4.5.2. Impacto de la propuesta en la fase de mantenimiento

Para medir el impacto conseguido en la fase de mantenimiento, se analiza la cantidad de fallas de software reportadas en ambientes de producción, de tres proyectos implementados

sin usar el marco de trabajo y de tres proyectos que si usaron el marco de trabajo propuesto. La selección de los proyectos de software se realizó con el asesoramiento del Administrador de Desarrollo, considerando el número de requisitos funcionales y el tiempo de desarrollo establecidos para cada proyecto.

Tabla 4.12. *Proyectos seleccionados para analizar el impacto de la propuesta*.

Proyectos	**Descripción del Proyecto**	**Nro. de requisitos**	**Tiempo de desarrollo**	**Marco de Pruebas**
Proyecto A	Sistema de Service Desk	20	3 meses	No
Proyecto B	Sistema de Rifas y Sorteos	21	3 meses	No
Proyecto C	Sistema de Crédito Comunal	32	4 meses	No
Proyecto D	Sistema de Declaración Juramentada	22	3 meses	Si
Proyecto E	Sistema de Gestión de Cheques	18	3 meses	Si
Proyecto F	Sistema de Microcrédito Externo	30	4 meses	Si

Fuente: Departamento de Tecnología, COAC Atuntaqui Ltda.

Los Proyectos A, B y C corresponden a los proyectos de software desarrollados antes de la implementación del marco de trabajo y los Proyectos D, E y F corresponden a proyectos evaluados con el marco de trabajo propuesto, incluido el caso de estudio presentado.

A continuación, se muestra el número de casos de soporte ingresados en la herramienta *Service Desk* para cada uno de los proyectos, los cuales corresponden a fallas de software reportadas en ambientes de producción.

Tabla 4.13. *Fallas de software reportadas para cada proyecto.*

Proyectos	Errores en funcionalidades	Errores en datos	Casos totales
Proyecto A	6	2	8
Proyecto B	6	4	10
Proyecto C	8	6	14
Total de fallas sin usar la propuesta			**32**
Proyecto D	1	0	1
Proyecto E	1	0	1
Proyecto F	2	0	2
Total de fallas usando la propuesta			**4**

Fuente: Mesa de Ayuda - COAC Atuntaqui Ltda.

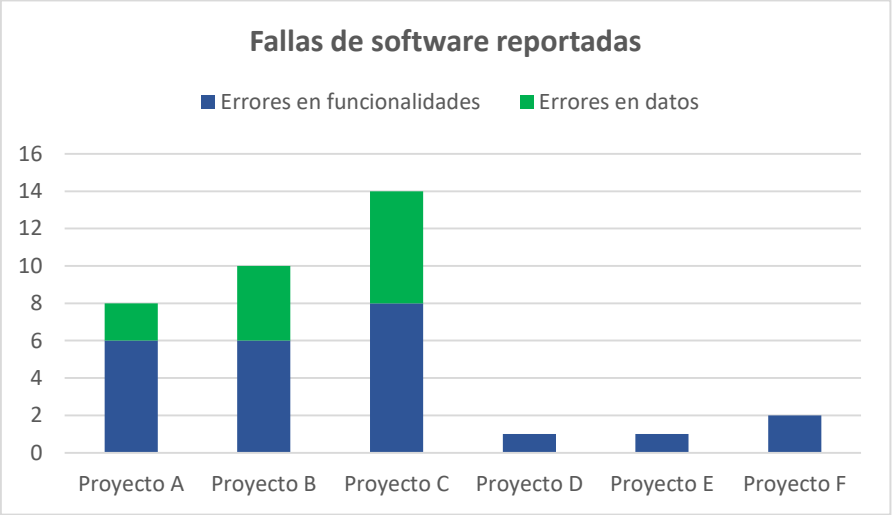


Figura 4.9. Cantidad de fallas de software reportadas en cada proyecto de software.

Fuente: Autor.

Como se puede observar en la gráfica, la cantidad de fallas de software reportadas en los proyectos que no usaron la propuesta es superior, a la cantidad de fallas de software de los proyectos que si usaron la propuesta.

Por lo tanto, el impacto de la propuesta corresponde al porcentaje de fallas de software alcanzado, en relación con la cantidad de fallas de software reportadas sin usar la propuesta.

$$\%\ de\ fallas\ alcanzado = \frac{cantidad\ de\ fallas\ con\ la\ propuesta}{cantidad\ de\ fallas\ \text{sin}\ la\ propuesta}\ x\ 100$$

$$\%\ de\ fallas\ alcanzado = \frac{4}{32}\ x\ 100$$

$$\%\ de\ fallas\ alcanzado = 12{,}5\ \%$$

Se determina que el impacto causado por el marco de trabajo de pruebas de software, en la fase de mantenimiento, es la reducción al 12,5% de las fallas de software reportadas anteriormente en ambientes de producción; además, ninguna de las fallas reportadas corresponde a errores en los datos, evitando la afectación directa a la base de datos.

CAPÍTULO 5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- La implementación del marco de trabajo para la gestión de pruebas, en la COAC Atuntaqui Ltda., fue una experiencia enriquecedora y exitosa, con el caso de estudio presentado se pudo comprobar que la metodología propuesta reduce la cantidad de fallas de software en ambientes de producción a un 12,5%, mejorando la calidad de los productos de software desarrollados en la cooperativa.
- El personal técnico de la Dirección de Tecnología de la COAC Atuntaqui Ltda., tuvo una aceptación favorable, del 90%, al marco de trabajo propuesto, pues les permite contar con una guía especializada para la gestión de las pruebas de software.
- El equipo de desarrollo dispone de mayor tiempo para la implementación de nuevos proyectos, porque el marco de trabajo de pruebas propuesto permite identificar defectos de software, que con las prácticas anteriores no eran posible; esto se puede evidenciar con los 22 defectos de software encontrados en el caso de estudio presentado.

5.2. Recomendaciones

- Con el objetivo de mejorar la calidad de los productos de software, se recomienda utilizar el marco de trabajo de pruebas, en todos los proyectos de software desarrollados por la institución.
- Para mejorar la metodología de pruebas propuesta, y que esta sea aplicable a otro tipo de pruebas, se recomienda su retroalimentación con las buenas prácticas utilizadas en el proceso de evaluación del software.
- Cuando se tenga la experticia suficiente en el proceso de pruebas de software, se recomienda analizar la posibilidad de adquirir o implementar una herramienta para automatizar las pruebas de regresión, que permita reducir el tiempo y costo de la fase de pruebas.
- Como se encuentra establecido en la metodología de pruebas propuesta, se recomienda que el equipo de pruebas sea diferente al equipo de desarrollo, o que por lo menos, el personal que evalúe el software sea ajeno al equipo que desarrollo el software.

REFERENCIAS BIBLIOGRÁFICAS

Alaqail, H & Ahmed, S. (2018). Overview of Software Testing Standard ISO/IEC/IEEE 29119. *International Journal of Computer Science and Network Security. 18*(2), 112-116.

Bruegge, B. & Dutoit, A. (2012). *Ingeniería de Software Orientado a Objetos (1ra. ed.).* México D.F., México: Pearson Educación de México S.A.

Cooperativa de Ahorro y Crédito Atuntaqui Ltda. (2018). Manual del Proceso de Gestión de Desarrollo de Software. Atuntaqui: Autor.

Gallesdic, I.; Killiospy, G. (2013). La Prueba del Software como Ciencia. *Revista Antioqueña de las Ciencias Computacionales y la Ingeniería de Software*. *3* (1), 33-37.

GIIS Grupo de Investigación en Ingeniería del Software (2017). *Grupo de Trabajo AEN/CTN71/SC7/GT26 Pruebas de Software. ISO/IEC/IEEE 29119 Software Testing Standard.* Asturias, España. Recuperado de http://in2test.lsi.uniovi.es/gt26/?lang=es.

González, L. (2012). *Introducción al Software Testing.* Vigo, España: Universidad de Vigo.

Guerra, L. & Bedini, A. (2006). *Gestión de Proyectos de Software (1ra. ed.).* Valparaíso, Chile: Universidad Técnica Federico Santa María.

IEEE Computer Society (2014). *SWEBOK Guide to the Software Engineering Body of Knowledge (v 3.0).* New York, United States of America: IEEE.

ISO/IEC/IEEE (2013). *ISO/IEC/IEEE 29119-1:2013 Software and systems engineering - Software testing – Part 1: Concepts and definitions*, Ginebra, Suiza: ISO.

ISO/IEC/IEEE (2013). *ISO/IEC/IEEE 29119-2:2013 Software and systems engineering - Software testing – Part 2: Test processes*, Ginebra, Suiza: ISO.

ISO/IEC/IEEE (2013). *ISO/IEC/IEEE 29119-3:2013 Software and systems engineering - Software testing – Part 3: Test documentation*, Ginebra, Suiza: ISO.

ISO/IEC/IEEE (2015). *ISO/IEC/IEEE 29119-4:2015 Software and systems engineering - Software testing – Part 4: Test techniques*, Ginebra, Suiza: ISO.

ISO/IEC/IEEE (2017). *ISO/IEC/IEEE 24765:2017 Systems and software engineering – Vocabulary*, Ginebra, Suiza: ISO.

INTECO (Instituto Nacional de Tecnologías de la comunicación) (2009). *Curso de Introducción a la Ingeniería de Software.* Madrid, España: INTECO.

Jiménez, O. (2017). *Pruebas de Calidad Aplicadas al Sitio Web Allison* (Tesis de Maestría en Sistemas Computacionales). Tecnológico Nacional de México, Villa de Álvarez, Colombia.

Jorgensen, P. (2013). *Software Testing: A Craftsman's Approach (4th ed.).* Florida, United States of America: Auerbach Publications.

Lerche-Jensen, S. (2015). *Test Processes ISO 29119.* New York, United States of America: Kindle Edition.

Mera-Paz, J. (2016). Análisis del Proceso de Pruebas de Calidad de Software. *Ingeniería Solidaria*. *12* (20), 163-176. doi: http://dx.doi.org/10.16925/in.v12i20.1482.

Páez, J. (2011). *Diseño de un modelo para evaluación/pruebas del software en base a Ingeniería de pruebas aplicando el estándar ISO/IEC 29119 en la empresa Omnisoft de la ciudad de Quito* (Tesis de Maestría). Escuela Politécnica del Ejército, Latacunga, Ecuador.

Pressman, R. (2014). *Ingeniería de Software un enfoque práctico (8va. ed.).* México D.F., México: The McGraw-Hill Companies, Inc.

Sánchez, J. (2015). *Pruebas de Software. Fundamentos y Técnicas.* (Tesis de Maestría). Universidad Politécnica de Madrid, Madrid, España.

Schaull, S. (2011). El desarrollo de software como ingeniería de software. *Columbia University, NY. 2* (2), 6-9.

Sommerville, I. (2011). *Ingeniería de software (9na. ed.).* México D.F., México: Pearson Education, Inc.

Reid, S. (2012). *ISO/IEC/IEEE 29119: The New International Software Testing Standards*. Londres, Reino Unido: EuroSTAR 2012.

Reid, S. (2017). *ISO/IEC/IEEE 29119 Software Testing Standards: A Practitioner's Guide.* New York, United States of America: Kindle Edition.

Villarreal, D.; Gamboa, S; Gómez, L. (2015). Estudio sobre realización y documentación de pruebas software. *Maskana. 1* (1), 219 -225.

Veenendaal, E. (2016). *TMMi and ISO/IEC 29119: Friends or Foes?* Leinster, Irlanda: TMMi Foundation.

ANEXOS

ANEXO 1. Encuesta para el Área de Desarrollo

La presente encuesta tiene como objetivo evaluar la percepción que usted tiene a cerca del proceso de pruebas de software que actualmente realiza la institución. Los datos obtenidos en la misma tienen un fin académico, por lo que se solicita responder las preguntas de manera objetiva y sincera.

Como califica usted, los siguientes aspectos:

1. ¿El proceso de pruebas de software es?
    a. Muy Bueno
    b. Bueno
    c. Regular
2. ¿La priorización de las características a evaluar es?
    a. Muy Buena
    b. Buena
    c. Regular
3. ¿La selección de los casos de pruebas a ejecutar es?
    a. Muy Buena
    b. Bueno
    c. Regular
4. ¿La documentación resultante del proceso de pruebas es?
    a. Muy Buena
    b. Buena
    c. Regular

ANEXO 2. Encuesta para el Área de Soporte

La presente encuesta tiene como objetivo evaluar la percepción que usted tiene a cerca del software desarrollado en la institución. Los datos obtenidos en la misma tienen un fin educativo, por lo que se solicita responder las preguntas de manera objetiva y sincera.

Como califica usted, los siguientes aspectos:

1. ¿La calidad del software desarrollado en la institución?
    a. Muy Bueno
    b. Bueno
    c. Regular
2. ¿La ausencia de fallas de software?
    a. Muy Bueno
    b. Bueno
    c. Regular

ANEXO 3. Cuestionario base para entrevista con Administrador de Desarrollo

El siguiente cuestionario, es parte de la entrevista realizada con el Administrador de Desarrollo de la COAC Atuntaqui Ltda., y su objetivo es conocer más a detalle el proceso de pruebas ejecutado en la institución.

1. ¿La institución cuenta con un proceso de desarrollo de software establecido?
2. ¿Cuáles son las fases del proceso de desarrollo?
3. ¿Existe una fase para las pruebas de software?
4. ¿Se utiliza alguna metodología para la gestión de las pruebas?
5. ¿Quién realiza las pruebas de software?
6. ¿Se planifican los casos de prueba que van a ser ejecutados?
7. ¿Se utiliza alguna técnica para la realización de las pruebas?
8. ¿Las pruebas de software son documentadas?
9. ¿Se registran los defectos de software encontrados en las pruebas?
10. ¿Cuándo se publica el software en producción?

ANEXO 4. Plantilla Plan de Pruebas

COOPERATIVA DE AHORRO Y CRÉDITO "ATUNTAQUI" LTDA.

GESTIÓN DE DESARROLLO DE SOFTWARE

PLAN DE PRUEBAS DE SOFTWARE

[Nombre del proyecto]

PROPIETARIO

Administrador de Desarrollo

1. INFORMACIÓN GENERAL DEL DOCUMENTO

Historial de versiones

Fecha	Versión	Responsable	Descripción

Información del proyecto

Empresa	
Proyecto	
Fecha de elaboración	
Área / Departamento	
Líder del proyecto	
Líder de pruebas	

2. INTRODUCCIÓN

Describe el proyecto dentro del cual se desarrollan las pruebas y el módulo a probar. Comprende la razón de las pruebas, hechos históricos, documentos relevantes, estándares a verificar, planes de pruebas anteriores, etc.

3. OBJETIVOS

Esta sección es usada para indicar los objetivos de las pruebas enmarcadas en las características del proyecto.

4. ALCANCE

En esta sección se describe el alcance de la prueba, para lo cual es necesario especificar de manera general las características que se van a probar y las que no se van a probar.

Características a probar

Identifica las características y combinaciones de estas que serán probadas y las razones por las cuales se van a probar y su prioridad. Referenciar los documentos (especificaciones, diseño, etc.) asociados y de soporte.

Características a NO probar

Identifica las características y combinaciones de estas que NO serán probadas y las razones por las cuales NO se van a probar.

5. CRITERIOS DE ACEPTACIÓN, RECHAZO, SUSPENSIÓN Y REANUDACIÓN

Especifica los criterios para determinar si el resultado de la prueba es aceptado o rechazado. Así como, los criterios para suspender las pruebas de manera parcial o total y las condiciones necesarias para la reanudación de estas.

Criterios de aceptación o rechazo

Son los criterios que serán considerados para dar por completado el plan de pruebas de software, por ejemplo: completar las pruebas unitarias, cierto porcentaje de casos exitosos, cobertura de todos los componentes y líneas de código, porcentaje de defectos corregidos, entre otros.

Criterios de suspensión

Establece claramente en qué condiciones se detienen un conjunto de casos de pruebas, por ejemplo, en caso de existir defectos que impidan la ejecución de más casos de pruebas, cierto porcentaje de casos fallidos, o cualquier otro que se especifique.

Criterios de reanudación

Luego de haber suspendido las pruebas, aquí se establece bajo qué criterios se reanudarán.

6. RIESGOS Y CONTINGENCIAS

Identifica los riesgos asociados con el proceso de pruebas de software, por ejemplo, algunas fuentes de riesgos suelen ser:

- Dependencias con desarrollos.
- Dependencias con otros proyectos.
- Disponibilidad de recursos.
- Restricciones de tiempo.
- Premisas que resulten no ser ciertas.

Los riesgos se pueden clasificar en función de su probabilidad e impacto, cada uno debe contemplar un plan de mitigación para evitar que ocurra o plan de contingencia cuando el riesgo no puede mitigarse y tiene que aceptarse.

7. RECURSOS

Especifica las características, que debe tener el ambiente de pruebas y los recursos humanos necesarios.

Requerimientos de hardware

Lista de los requerimientos de equipos, hardware y red necesarios para completar las actividades del plan de pruebas de software. Incluye servidores de aplicación, bases de datos, equipos que necesitan los ingenieros de pruebas, conectividad a la red (incluyendo accesos), entre otros.

Requerimientos de software

Lista de los requerimientos de software necesarios para completar las actividades de prueba, puede incluir accesos a sistemas (en entorno de pruebas) y bases de datos, así como instalación de software en los computadores asignados a los ingenieros de pruebas.

Herramientas de pruebas requeridas

Especifica las herramientas de software, metodologías o técnicas especiales empleadas en las pruebas, por ejemplo, herramientas de automatización de pruebas, software de gestión de pruebas, entre otros.

Recursos humanos

Lista el personal necesario para completar las actividades de pruebas, especificando sus roles, por ejemplo: un líder de pruebas, dos ingenieros de pruebas, entre otros.

8. PLANIFICACIÓN Y ORGANIZACIÓN

Especifica los procedimientos o metodología de pruebas a emplear durante la ejecución del plan de pruebas de software.

Matriz de responsabilidades

Lista cada una de las personas integrantes del equipo de pruebas y sus responsabilidades.

Cronograma

Se establecen las actividades a realizar para la ejecución de las pruebas, especificando las dependencias (actividades predecesoras) y demás aspectos de un cronograma. Se debe identificar claramente las personas responsables, recursos y tiempo necesario para realizar las pruebas.

9. APROBACIONES

Contiene los datos y firmas de las personas responsables de la elaboración, revisión y aprobación del Plan de Pruebas.

Elaborado	Revisado	Aprobado
Firma:	Firma:	Firma:
Nombre: _____ Cargo: _____ Fecha: _____	Nombre: _____ Cargo: _____ Fecha: _____	Nombre: _____ Cargo: _____ Fecha: _____

ANEXO 5. Especificación de Prioridad de Pruebas

DIRECCIÓN DE TECNOLOGÍA

Especificación del Prioridad de Pruebas

GTI – GDS – GPS - EPP

Módulo: _____

Versión: _____

Id Req.	Descripción	Probabilidad de falla	Frecuencia de uso	Prioridad	Comentario

Aprobaciones

Elaborado	Revisado	Aprobado
Firma:	Firma:	Firma:
Nombre: _____ Cargo: _____ Fecha: _____	Nombre: _____ Cargo: _____ Fecha: _____	Nombre: _____ Cargo: _____ Fecha: _____

ANEXO 6. Especificación de Casos de Prueba

DIRECCIÓN DE TECNOLOGÍA

Especificación de Casos de Prueba

GTI – GDS – GPS - ECP

Módulo: _____

Versión: _____

ID Caso Prueba	Descripción	Condiciones previas	Datos de Entrada	Resultado esperado

Aprobaciones

Elaborado	Revisado	Aprobado
Firma:	Firma:	Firma:
Nombre: _____ Cargo: _____ Fecha: _____	Nombre: _____ Cargo: _____ Fecha: _____	Nombre: _____ Cargo: _____ Fecha: _____

ANEXO 7. Procedimiento de Pruebas

DIRECCIÓN DE TECNOLOGÍA

Procedimiento de Pruebas

GTI - GDS - GPS - PP

Módulo: _____

Versión: _____

ID Caso Prueba	Datos de Entrada	Resultado Esperado	Pasos a seguir	Resultado Obtenido

Aprobaciones

Elaborado	Revisado	Aprobado
Firma:	Firma:	Firma:
Nombre: _____ Cargo: _____ Fecha: _____	Nombre: _____ Cargo: _____ Fecha: _____	Nombre: _____ Cargo: _____ Fecha: _____

ANEXO 8. Reporte de Defectos

DIRECCIÓN DE TECNOLOGÍA

Reporte de Defectos

GTI - GDS - GPS - RD

Módulo: _____

Versión: _____

Id Caso de Prueba		
Descripción Caso de Prueba		
Datos de Entrada	**Resultado Esperado**	**Resultado Obtenido**
Procedimiento ejecutado		
Observaciones		

Aprobaciones

Elaborado	**Revisado**	**Aprobado**
Firma:	Firma:	Firma:
Nombre: _____ Cargo: _____ Fecha: _____	Nombre: _____ Cargo: _____ Fecha: _____	Nombre: _____ Cargo: _____ Fecha: _____