UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES E INFORMÁTICOS

TEMA:

ANÁLISIS DE MÉTODOS, TÉCNICAS Y HERRAMIENTAS DE VERIFICACIÓN Y VALIDACIÓN DE SOFTWARE, APLICADOS EN LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN DE LA UNIVERSIDAD TÉCNICA DE AMBATO

Trabajo de Graduación. Modalidad: Proyecto de Investigación, presentado previo la obtención del título de Ingeniero en Sistemas Computacionales e Informáticos

SUBLÍNEA DE INVESTIGACIÓN:

Ingeniería del Software

AUTOR: Daniel Sebastián Jerez Mayorga

TUTOR: Ing. Franklin Oswaldo Mayorga Mayorga, Mg

Ambato - Ecuador

Agosto, 2017

APROBACIÓN DEL TUTOR

En mi calidad de Tutor del Trabajo de Investigación sobre el Tema:

"ANÁLISIS DE MÉTODOS, TÉCNICAS Y HERRAMIENTAS DE VERIFICACIÓN Y VALIDACIÓN DE SOFTWARE, APLICADOS EN LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN DE LA UNIVERSIDAD TÉCNICA DE AMBATO.", del señor Daniel Sebastián Jerez Mayorga, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, considero que el informe investigativo reúne los requisitos suficientes para que continúe con los trámites y consiguiente aprobación de conformidad con el numeral 7.2 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ambato, Agosto de 2017

EL TUTOR

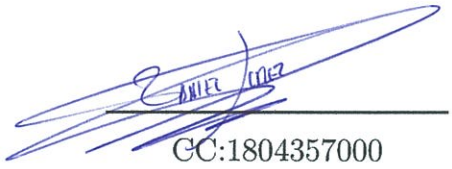
Ing. Franklin Oswaldo Mayorga Mayorga, Mg

AUTORÍA

El presente trabajo de investigación titulado: "ANÁLISIS DE MÉTODOS, TÉCNICAS Y HERRAMIENTAS DE VERIFICACIÓN Y VALIDACIÓN DE SOFTWARE, APLICADOS EN LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN DE LA UNIVERSIDAD TÉCNICA DE AMBATO". Es absolutamente original, auténtico y personal, en tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, Agosto de 2017

Daniel Sebastián Jerez Mayorga



CC:1804357000

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.
Cedo los derechos de mi Trabajo de Titulación, con fines de difusión pública, además autorizo su reproducción dentro de las regulaciones de la Universidad.

Ambato, Agosto de 2017

Daniel Sebastián Jerez Mayorga

CC:1804357000

APROBACIÓN COMISIÓN CALIFICADORES

La Comisión Calificadora del presente trabajo conformada por los señores docentes Ing. Edison Álvarez, Mg. e Ing. Jaime Ruiz, Mg., revisó y aprobó el Informe Final del Proyecto de Investigación titulado"ANÁLISIS DE MÉTODOS, TÉCNICAS Y HERRAMIENTAS DE VERIFICACIÓN Y VALIDACIÓN DE SOFTWARE, APLICADOS EN LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN DE LA UNIVERSIDAD TÉCNICA DE AMBATO", presentado por el señor Daniel Sebastián Jerez Mayorga, de acuerdo al numeral 9.1 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ing. Mg. Elsa Pilar Urrutia Urrutia

PRESIDENTE DEL TRIBUNAL

Ing. Edison Álvarez, Mg.

DOCENTE CALIFICADOR

Ing. Jaime Ruiz, Mg.

DOCENTE CALIFICADOR

DEDICATORIA

A mis padres Mercedes Alicia y Victor Manuel que a pesar de todos los problemas y desaciertos siempre me han apoyado, a mis hermanas Anita, Paulina que siempre me han colaborado cuando lo he necesitado.
A Diana por ser mi compañera de estudios y ahora mi compañera de vida TE AMO.
A Benjamín por traer tanta suerte a mi vida TE AMO hijo.

Daniel Sebastián Jerez Mayorga.

AGRADECIMIENTO

A Dios por darme salud y permitirme despertar cada día sin eso no somos nada, a mis padres por darme la dicha de estar vivo, a mi familia por el apoyo que siempre me brindan, a los docentes que estuvieron presentes en la formación de mi carrera, a Ing. Raúl Peña por darme esa fuerza, motivación y ver ese potencial que puede existir en mi, saludos a la distancia, a todos los jefes y compañeros de trabajado de los cuales he aprendido mucho, a la música la cual me ha influenciado para alcanzar objetivos y ver la vida de otra manera, a Ing. Franklin Mayorga por los conocimientos y tiempo brindado y a Ing. Robert Vaca por permitirme realizar mi estudio en su Dirección.

Daniel Sebastián Jerez Mayorga

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Resumen

Actualmente la calidad del software es un factor escencial para el desarrollo de productos y servicios que cumplan con las necesidades y expectativas de los clientes. Para llegar a alcanzar una calidad deseada se requiere de un proceso de desarrollo de software adecuado, además de la utilización de la verificación y validación de software, la cual se apoya de métodos, técnicas y herramientas que ayudan a cumplir este objetivo.

El presente proyecto propone la aplicación de métodos, técnicas y herramientas de verificación y validación apropiadas para cubrir las necesidades en el desarrollo de software en la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato, para lo cual se identificaron los métodos, técnicas y herramientas utilizadas en la Dirección, así como también el tipo de pruebas que se aplican al software, el proceso, modelos, control y herramientas utilizados actualmente para el desarrollo de software.

Se realizó un análisis basado en el estándar internacional para la evaluación de Software ISO/IEC 9126, en el cual se establecen las características de calidad para productos de software. Se evaluaron los métodos, técnicas y herramientas de V&V de software, conjuntamente con los aplicados en la DITIC. Los métodos y técnicas fueron evaluados según las características y subcaracterísticas del Modelo de Calidad del estándar mencionado, y las herramientas se evaluaron según las Métricas de Calidad en Uso definidas por la Norma en su parte 4 que definen la aceptación del software por parte del usuario final.

Este análisis apoya a la selección de métodos, técnicas y herramientas de verificación y validación que se ajustan a las necesidades en la Universidad Técnica de Ambato, a fin de que ayuden a que el software pueda llegar al cumplimiento de calidad.

Abstract

Currently the quality of software is an essential factor for the development of products and services that meets the needs and expectations of customers. To reach a desired quality requires a proper software development process, in addition to the use of software verification and validation, which relies on methods, techniques and tools that help meet this goal.

The present project proposes the application of verification and validation methods, techniques and tools to cover the needs in software development in the Information and Communication Technology Department of the Technical University of Ambato, for which the methods, techniques and tools used in the Department were identified, as well as the type of tests that are applied to the software, process, models, control and tools currently used for software development.

An analysis was performed based on the international standard for the evaluation of software ISO/IEC 9126, which establishes the quality characteristics for the software products. The V&V software methods, techniques and tools were evaluated together with those applied in the DITIC. The methods and techniques were evaluated according to the characteristics and subcharacteristics of the Quality Model of the standard, and the tools were evaluated according to the Quality Metrics in Use defined by the standard in its part 4 that define the acceptance of the software by the final user.

This analysis supports the selection of methods, techniques and tools of verification and validation that fit the needs of the Technical University of Ambato, to help the software to fulfill the quality compliance.

Glosario de términos

Artefacto.- Un artefacto es una pieza de información que es producida, modificada o usada por el proceso, define un área de responsabilidad para un rol y está sujeta a control de versiones. Un artefacto puede ser un modelo, un elemento de modelo o un documento.

V&V.- Validación y Verificación de Software. Son las pruebas realizadas al software. La verificación tiene lugar en cada paso del ciclo de vida, mientras que la validación ocurre después de que se instala el sistema y antes de ponerlo en servicio.

IEEE.- The Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Eléctricos y Electrónicos).

ISO.- International Organization for Standarization (Organización Internacional para la Estandarización)

Resiliencia.- La resiliencia se define como la capacidad de los seres humanos para adaptarse positivamente a situaciones adversas.

API.- Application Programming Interface (Interfaz de Programación de Aplicaciones). Es un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software.

Stub.- Es un trozo de código usado como sustituto para simular el comportamiento de un código existente.

Error.- Es una equivocación cometida por el desarrollador. Algunos ejemplos de errores son: un error de digitación, una malinterpretación de un requerimiento o de la funcionalidad de un método.

Defecto.- Un defecto se encuentra en un artefacto y puede definirse como una diferencia entre la versión correcta del artefacto y una versión incorrecta.

Falla.- En terminología IEEE, una falla es la discrepancia visible que se produce al ejecutar un programa con un defecto, el cual es incapaz de funcionar correctamente.

BD.- abreviatura para referirnos a una Base de Datos.

Scripts.- Los scripts son un conjunto de instrucciones generalmente almacenadas en un archivo de texto que deben ser interpretados línea a línea en tiempo real para su ejecución.

URL.- Es una secuencia de caracteres que se utiliza para nombrar y localizar recursos, documentos e imágenes en Internet.

Proxy.- Es un programa o dispositivo que realiza una tarea de acceso.

INTRODUCCIÓN

El presente trabajo denominado: "ANÁLISIS DE MÉTODOS, TÉCNICAS Y HERRAMIENTAS DE VERIFICACIÓN Y VALIDACIÓN DE SOFTWARE, APLICADOS EN LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN DE LA UNIVERSIDAD TÉCNICA DE AMBATO", se ha estructurado de la siguiente manera:

CAPÍTULO I denominado "EL PROBLEMA", identifíca el problema que actualmente existe en la Dirección de Tecnología de Información y Comunicación en cuanto a la verificación y validación de software. Además se detalla la justificación y los objetivos que se van a cumplir en la presente investigación.

CAPÍTULO II denominado "MARCO TEÓRICO", muestra las investigaciones previas que sirven de soporte para el desarrollo del proyecto, además la información de estudios similares, asi como los conceptos y fundamentos teóricos que sustenta el tema en general.

CAPÍTULO III denominado "METODOLOGÍA", muestra la modalidad de investigación que ha sido realizada, la recolección de la información, se presenta el tipo de análisis de los datos según el tipo de investigación, y se detalla también el proceso a seguir para el desarrollo de proyecto.

CAPÍTULO IV denominado "DESARROLLO DE LA PROPUESTA", describe los resultados obtenidos en la recolección de información con su respectivo análisis e interpretación, se detalla el proceso, modelos, control, herramientas, y proceso de pruebas utilizados actualmente para el desarrollo de software en la Dirección de Tecnología de Información y Comunicación, así como también el análisis de métodos, técnicas y herramientas apropiadas para cubrir las necesidades en esta Dirección.

CAPÍTULO V denominado “CONCLUSIONES Y RECOMENDACIONES”, presenta las conclusiones obtenidas después de la investigación así como las recomendaciones pertinentes.

CAPÍTULO 1

El problema

1.1. Tema de Investigación

Análisis de métodos, técnicas y herramientas de verificación y validación de software, aplicados en la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato.

1.2. Planteamiento del Problema

En la actualidad la Calidad del Software es un factor esencial para el desarrollo del negocio de una empresa o institución, los proyectos de desarrollo software han padecido tradicionalmente problemas de calidad, tanto en el proceso de desarrollo como en los productos finales, esta problemática tiene su raíz en las fases de desarrollo establecidos en el ciclo de vida del software.

El mundo actual se ve rodeado de muchos desarrollos tecnológicos en el campo de la administración empresarial por procesos, sistemas informáticos que ayudan a la automatización de los mismos, aportando así de gran manera a mejorar las técnicas tradicionales de administración de una empresa.

Ecuador ha surgido como uno de los exportadores de software; esto gracias a que la demanda de este producto a nivel global se ha aumentado a medida que las computadoras y el Internet están presentes en casi todos los aspectos de la sociedad.

Debido a esta demanda, se hace cada vez más necesario para las empresas desarrolladoras de software ecuatorianas la utilización de métodos, técnicas y herramientas que permitan garantizar la calidad de los productos; y así mejorar la competitividad a nivel mundial.

Teniendo en cuenta que los sistemas informáticos que se ofertan en el mercado actual han ayudado a la gestión de las empresas trayéndoles grandes beneficios, la administración educativa superior ha optado también por la utilización de dichos sistemas para la administración y gestión de sus procesos.

La Universidad Técnica de Ambato ha optado por incorporar estos sistemas para mejorar sus procesos administrativos y académicos, productos desarrollados en la Dirección de Tecnología de Información y de Comunicación de la Institución.

La Dirección de Tecnología de Información y Comunicación aplica diferentes métodos, técnicas y herramientas de verificación y validación, cada software es evaluado de diferente manera, por lo que no se cumplen con todos los requerimientos de calidad para así llegar a cubrir con las expectativas del cliente, estos inconvenientes han sido corregidos en su mayoría posterior a la entrega del software al cliente, lo que ha provocado pérdida de recursos y tiempo.

1.3. Delimitación

Área Académica: Ingeniería del Software
Línea de Investigación: Ingeniería del Software
Sublínea de Investigación: Calidad del Software
Delimitación Espacial: Ecuador Tungurahua-Ambato Universidad Técnica de Ambato
Delimitación Temporal: Abril/2016 – Agosto/2017

1.4. Justificación

Este tipo de análisis es indispensable para que en el momento de desarrollo de software se tenga claramente definido cuál es el mejor método, técnica y herramienta que se debe aplicar para así asegurar que se esté construyendo el producto correcto y de la manera correcta.

Debido a los problemas que han surgido en los desarrollos de software, el estudio y mejora de los métodos, técnicas y herramientas de verificación y validación de software conlleva a un progreso valioso para la futura evolución de las tecnologías de la información.

El beneficiario directo es la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato, y beneficiarios indirectos los diferentes departamentos, facultades y usuarios que utilizan sus sistemas.

La realización de este análisis es factible porque existe la información necesaria para realizar el estudio, de la misma manera los conocimientos adquiridos en la formación de la carrera son de gran ayuda para realizar este proyecto de investigación.

1.5. Objetivos

1.5.1. General

- Analizar métodos, técnicas y herramientas de verificación y validación de software, aplicados en la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica Ambato.

1.5.2. Específicos

- Identificar las características y los tipos de pruebas que se aplican al software que se desarrolla en la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato.
- Analizar el proceso de verificación al software que se aplica en la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato.
- Determinar los factores que inciden en el proceso de verificación y validación del software desarrollado en la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato.
- Proponer la aplicación de métodos, técnicas y herramientas de verificación y validación apropiadas para cubrir las necesidades en el desarrollo de software en la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato

CAPÍTULO 2

Marco Teórico

2.1. Antecendentes Investigativos

La importancia de este tema ha conllevado a la realización de varios estudios y proyectos investigativos sobre la Verificación y Validación de software:

En la Universidad Técnica de Ambato se ha realizado un proyecto titulado: "Análisis de las metodologías ágiles y su incidencia en la creación del portafolio de servicio para la Unidad de Extensión Universitaria de la Universidad Técnica del Norte de la Ciudad de Ibarra", en la que el autor Ing. Pedro David Granda Gudiño, detalla que el proceso de desarrollo de software debe ir acompañado de una metodología ágil la cual permite estructurar, definir un marco de trabajo y planificar un proyecto de desarrollo de software para así obtener calidad, mejores costos, tiempos y cumplimiento. [1]

Freddy Tituana Vera de la Escuela Superior Politécnica del Litoral ha realizado un proyecto de investigación titulado: "Análisis de métodos, técnicas y herramientas de verificación y validación de software usados por empresas ecuatorianas desarrolladoras de software", en dicho estudio concluye que las empresas ecuatorianas si utilizan métodos de Verificación y Validación, pero no todas tienen definido en qué etapa de desarrollo utilizarlas, en cuanto a las herramientas las aplican en la etapa de implementación, y las técnicas de igual manera son utilizadas en las últimas etapas de desarrollo; las herramientas de V&V de distribución libre son las preferidas por las empresas mencionadas.[2]

En la Escuela Superior Politécnica del Litoral se ha realizado también un proyecto denominado "Estudio de aplicabilidad y comparativo de un Modelo de Calidad a productos de software con la Norma ISO/IEC 9126", en la que su autor Erick Ortega aplica dicha norma para analizar la calidad de productos desarrollados para empresas nacionales, él concluye que toda evaluación de calidad debe utilizar algún modelo de referencia, como una norma o un modelo propio de la organización. [3]

En la Universidad Carlos III de Madrid, se ha realizado el proyecto de fin de carrera titulado: "Análisis de los procesos de verificación y validación en las organizaciones software", después de realizar un estudio en las empresas del área de tecnología y de desarrollo de software, el autor Jorge Zamora concluye que una organización puede adecuar el desarrollo de su proceso de verificación y validación de software en función de las necesidades de su organización. Además menciona que el proceso de pruebas anteriormente era una actividad que no era considerada importante, hoy en día ha evolucionado, tanto así que se la establece ya como un proceso, con un ciclo de vida propio, y sobre la cual se realizan importantes fases como la de diseño y planificación. [4]

Según Michael S. Deutsch en su libro "Software Verification and Validation: Realistic Project Approaches" establece que: "El desarrollo de sistemas de software implica una serie de actividades de producción en las que las posibilidades de que aparezca el fallo humano son enormes. Los errores pueden empezar a darse desde el primer momento del proceso, en el que los objetivos pueden estar especificados de forma errónea o imperfecta, así como en posteriores pasos de diseño y de desarrollo. Debido a la imposibilidad humana de trabajar y comunicar de forma perfecta, el desarrollo de software ha de ir acompañado de una actividad que garantice la calidad." [5]

2.2. Fundamentación Teórica

2.2.1. Ingeniería del Software

La ingeniería de software es una disciplina formada por un conjunto de métodos, herramientas y técnicas que se utilizan en el desarrollo de software, con el objetivo de poder gestionar la realización de un proyecto para que este pueda ser desarrollado en un tiempo determinado y con el presupuesto previsto. Así como también la de garantizar que un software sea confiable y eficiente.

2.2.2. Desarrollo de Software

En el desarrollo de software se requieren de varios procesos que comprenden desde la creación de un sistema hasta su mantenimiento, a todo esto, se lo denomina como ciclo de vida del software.

2.2.3. Etapas de Desarrollo de Software

- Análisis.

- Diseño.
- Codificación.
- Pruebas.
- Mantenimiento.

2.2.3.1. Análisis de Requisitos

Etapa en la cual el desarrollador o analista reúne los requerimientos funcionales y no funcionales de un sistema, el objetivo de esta fase es el de comprender la naturaleza del software a construir, su aplicación y rendimiento.

Áreas de esfuerzo para el Análisis de Requisitos

1. **Reconocimiento del problema**

   Área de esfuerzo en la cual el analista o desarrollador se dedicada a reconocer los elementos del problema a solucionar como si fuese un usuario, estudia el plan del proyecto y las especificaciones del sistema.

2. **Evaluación y síntesis**

   Área de esfuerzo en la cual el analista o desarrollador evalúa y elabora la estructura y flujo de la información. Establece las características de la interfaz y encentra las limitaciones del diseño.

3. **Modelado**

   Área de esfuerzo en la cual el analista o desarrollador crea modelos del sistema se centra en entender el flujo de datos, de control, el procesamiento funcional, el comportamiento en operación y el contenido de la información.

4. **Especificación**

   Área de esfuerzo en la cual el analista o desarrollador establece los criterios de validación para demostrar que se ha alcanzado una adecuada comprensión de la forma de implementar con éxito el software.

5. **Revisión**

   Área de esfuerzo en la cual el analista o desarrollador realiza una revisión general para determinar si las primeras estimaciones siguen siendo válidas después del conocimiento adicional obtenido en todas las áreas de esfuerzo para el análisis de requisitos.

2.2.3.2. Diseño

La etapa de diseño convierte todos los requisitos en una muestra de software donde se puede evaluar su calidad antes de que comience la codificación, dentro de la cual se agrupan actividades como diseño de las estructuras de datos, la arquitectura general del software, algoritmos y representaciones de interfaz.

2.2.3.3. Codificación

La etapa de codificación consiste en plasmar el diseño mediante la programación de código de una forma clara y legible para la máquina.

2.2.3.4. Pruebas

Etapa que se centra en la realización de pruebas sobre el sistema desarrollado. Pruebas en los procesos lógicos internos del software y en los procesos externos funcionales, con el propósito de detectar defectos, errores y fallas en el mismo. En esta etapa es muy importante que exista un contacto con el solicitante o los interesados del desarrollo de software, de esta manera los objetivos del proyecto se mantendrán vigentes y se tendrá una idea clara de los aspectos que tienen que probarse durante esta etapa.

2.2.3.5. Mantenimiento

Etapa en la que se realizan los cambios que se presentan en un sistema para acoplarse a su entorno externo, porque el cliente requiere mejoras funcionales o de rendimiento. Con una documentación correcta del software se pueden presentar las rutas adecuadas para el mantenimiento y modificación del mismo.

2.2.4. Proceso de Desarrollo de Software

Un proceso de desarrollo de software puede definirse como una manera de dividir el trabajo en distintas actividades o el ciclo de vida del software en distintas fases, con el objetivo de lograr un buen resultado para el proyecto. [6]

El proceso de desarrollo de software tiene como objetivo la elaboración eficaz y eficiente de un producto software que reúna los requisitos del cliente. Ver figura 2.1.

Figura 2.1: Proceso de Desarrollo de Software
Fuente: [6]

El proceso de desarrollo de software es muy difícil de estandarizar en un único modelo, no existe un proceso de desarrollo de software universal que sea efectivo para todos los proyectos de desarrollo de software.

2.2.4.1. Actividades fundamentales que deben estar presentes en el proceso de desarrollo de software

Según Sommerville existen cuatro actividades básicas para el proceso de desarrollo de software[7]:

1. Especificación de software: Es el proceso de comprensión y definición de los requerimientos del sistema y la identificación de las restricciones de funcionamiento para el desarrollo del mismo.

2. Diseño e Implementación: Es el proceso en el cual se diseña y construye el software de acuerdo a las especificaciones.

3. Validación: La verificación y validación del software se utiliza para mostrar que el sistema se ajusta a su especificación y que cumpla las expectativas del usuario final. El proceso de pruebas como las inspecciones y revisiones deben estar en cada etapa del proceso de desarrollo de software desde la definición de requerimientos hasta el desarrollo del programa.

   Etapas del proceso de pruebas:

   a) Prueba de componentes: Se prueban cada uno de los componentes independientemente, los componentes pueden ser funciones, clases u objetos.

   b) Prueba del sistema: Este proceso comprende el encontrar errores que son el resultado de la interacción no prevista entre los componentes y su interfaz. Aquí se valida que el sistema cumpla sus requerimientos funcionales y no funcionales.

c) Pruebas de aceptación: Se prueba con los datos del cliente. Este tipo de prueba puede revelar errores y omisiones en la definición de los requerimientos.

1) Pruebas de aceptación alfa: Se realizan cuando los sistemas son desarrollados para un único cliente

2) Pruebas de aceptación beta: Se realizan cuando los sistemas son desarrollados con fines de comercialización, y va ser utilizados por muchas personas.

Los programadores definen sus propios datos de prueba y de forma incremental prueban el código que se va desarrollando, este enfoque es razonable puesto que el programador es el que mejor conoce los componentes y es por lo tanto la persona indicada para generar los casos de prueba.

4. Evolución: El software debe evolucionar, para adaptarse a los requerimientos cambiantes y necesidades del usuario.

2.2.4.2. Elementos del proceso de desarrollo de software

Según Pressman en un proceso de desarrollo de software deben estar presentes los siguientes elementos[8]:

- Un marco común del proceso, definiendo un pequeño número de actividades del marco de trabajo que son aplicables a todos los proyectos de software, con independencia del tamaño o complejidad.
- Un conjunto de tareas, cada uno es una colección de tareas de ingeniería del software, hitos de proyectos, entregas y productos de trabajo del software, y puntos de garantía de calidad, que permiten que las actividades del marco de trabajo se adapten a las características del proyecto de software y los requisitos del equipo del proyecto.
- Las actividades de protección, tales como garantía de calidad del software, gestión de configuración del software y medición, abarcan el modelo del proceso. Las actividades de protección son independientes de cualquier actividad del marco de trabajo y aparecen durante todo el proceso.

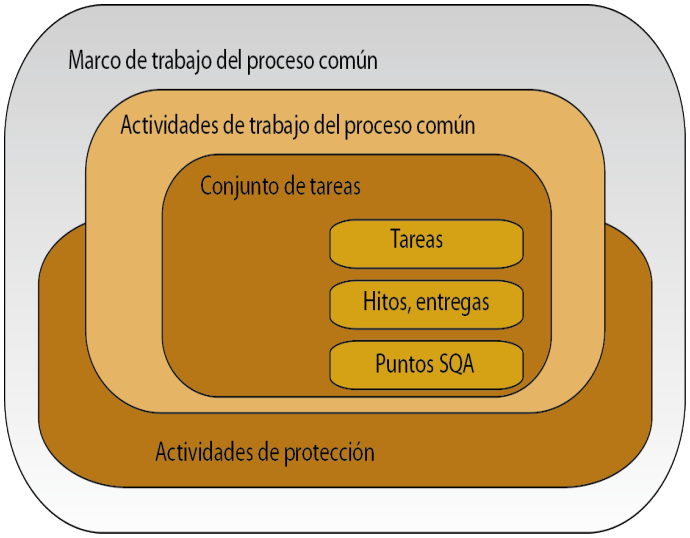


Figura 2.2: Elementos del proceso de desarrollo de software
Fuente: [8]

Otra forma de determinar los elementos del proceso de desarrollo de software según Letelier es estableciendo las relaciones que existen entre ellos[9]:

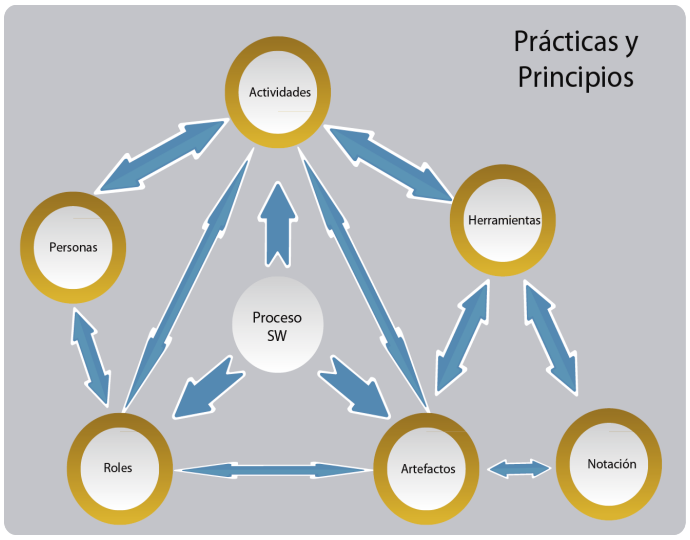


Figura 2.3: Relación entre los elementos del proceso de desarrollo de software
Fuente: [9]

Estas relaciones permiten responder Quién debe hacer Qué, Cuándo y Cómo debe hacerlo.

- **Quién:** Las Personas participantes en el proyecto de desarrollo desempeñando uno o más Roles específicos.

- **Qué**: Un Artefacto es producido por un Rol en una de sus Actividades. Los Artefactos se especifican utilizando Notaciones específicas. Las Herramientas apoyan la elaboración de Artefactos soportando ciertas Notaciones.
- **Cómo y Cuándo:** Las Actividades son una serie de pasos que lleva a cabo un Rol durante el proceso de desarrollo. El avance del proyecto está controlado mediante hitos que establecen un determinado estado de terminación de ciertos Artefactos.

2.2.4.3. Modelos del proceso de desarrollo de software

- Codificar y corregir
- Modelo en Cascada
- Desarrollo Evolutivo
- Desarrollo formal de sistemas
- Desarrollo basado en reutilización
- Desarrollo incremental
- Desarrollo espiral

Codificar y corregir (Code-and-Fix)

Este es el modelo básico utilizado en los inicios del desarrollo de software. Contiene dos pasos:

- Escribir código.
- Corregir problemas en el código. Se trata de primero implementar algo de código y luego pensar acerca de requisitos, diseño, validación, y mantenimiento.

Este modelo tiene tres problemas principales [10]:

- Después de un número de correcciones, el código puede tener una muy mala estructura, hace que los arreglos sean muy costosos.
- Frecuentemente, aún el software bien diseñado, no se ajusta a las necesidades del usuario, por lo que es rechazado o su reconstrucción es muy cara.
- El código es difícil de reparar por su pobre preparación para probar y modificar.

Modelo en cascada

El primer modelo de desarrollo de software que se publicó se derivó de otros procesos de ingeniería[11].

Este modelo toma las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución y las representa como fases separadas del proceso.

El modelo en cascada consta de las siguientes fases:

1. Definición de los requisitos: Los servicios, restricciones y objetivos son establecidos con los usuarios del sistema. Se busca hacer esta definición en detalle.

2. Diseño de software: Se particiona el sistema en sistemas de software o hardware. Se establece la arquitectura total del sistema. Se identifican y describen las abstracciones y relaciones de los componentes del sistema.

3. Implementación y pruebas unitarias: Construcción de los módulos y unidades de software. Se realizan pruebas de cada unidad.

4. Integración y pruebas del sistema: Se integran todas las unidades. Se prueban en conjunto. Se entrega el conjunto probado al cliente.

5. Operación y mantenimiento: Generalmente es la fase más larga. El sistema es puesto en marcha y se realiza la corrección de errores descubiertos. Se realizan mejoras de implementación. Se identifican nuevos requisitos.

La interacción entre fases puede observarse en la Figura 2.4. Cada fase tiene como resultado documentos que deben ser aprobados por el usuario.

Una fase no comienza hasta que termine la fase anterior y generalmente se incluye la corrección de los problemas encontrados en fases previas.

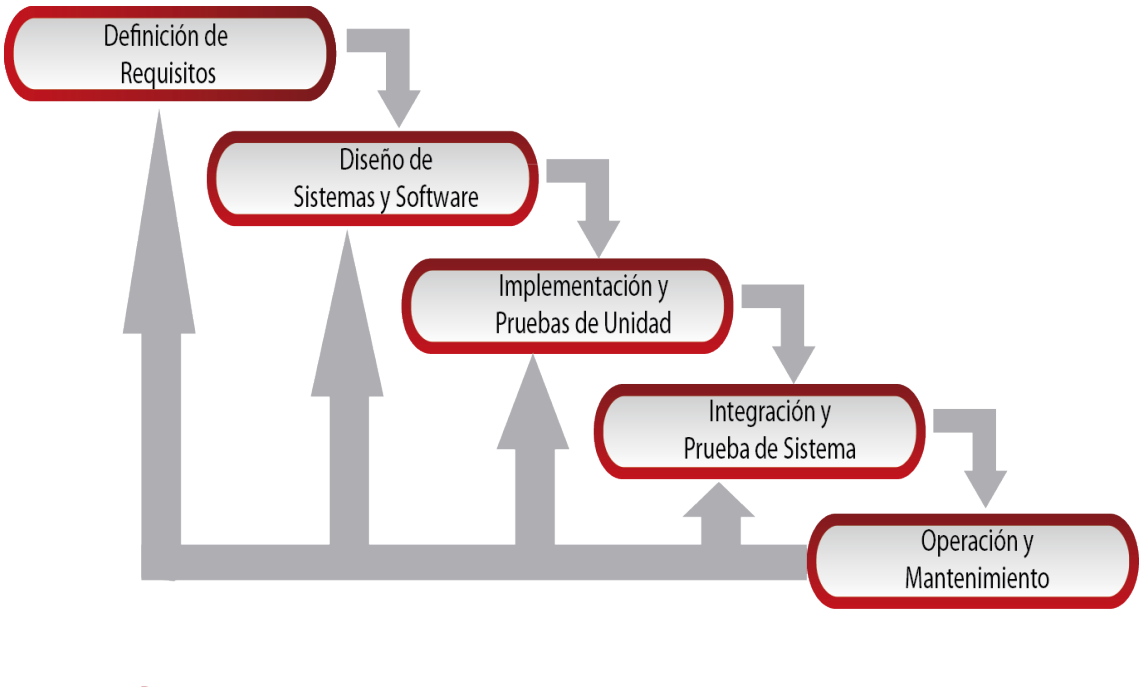


Figura 2.4: Modelo de desarrollo en cascada
Fuente: [9]

En la práctica, este modelo no es lineal, e involucra varias iteraciones e interacción entre las distintas fases de desarrollo.

Algunos problemas que se observan en el modelo de cascada son[9]:

- Las iteraciones son costosas e implican rehacer trabajo debido a la producción y aprobación de documentos.
- Aunque son pocas iteraciones, es normal congelar parte del desarrollo y continuar con las siguientes fases.
- Los problemas se dejan para su posterior resolución, lo que lleva a que estos sean ignorados o corregidos de una forma poco elegante.
- Existe una alta probabilidad de que el software no cumpla con los requisitos del usuario por el largo tiempo de entrega del producto.
- Es inflexible a la hora de evolucionar para incorporar nuevos requisitos.
- Es difícil responder a cambios en los requisitos.

Este modelo sólo debe usarse si se entienden a plenitud los requisitos. Aún se utiliza como parte de proyectos grandes.

Desarrollo evolutivo

Según Letelier la idea detrás de este modelo es el desarrollo de una implantación del sistema inicial, exponerla a los comentarios del usuario, refinarla en N versiones hasta que se desarrolle el sistema adecuado. En la Figura 2.5 se observa cómo las actividades concurrentes: especificación, desarrollo y validación, se realizan durante el desarrollo de las versiones hasta llegar al producto final.

Una ventaja de este modelo es que se obtiene una rápida realimentación del usuario, ya que las actividades de especificación, desarrollo y pruebas se ejecutan en cada iteración.

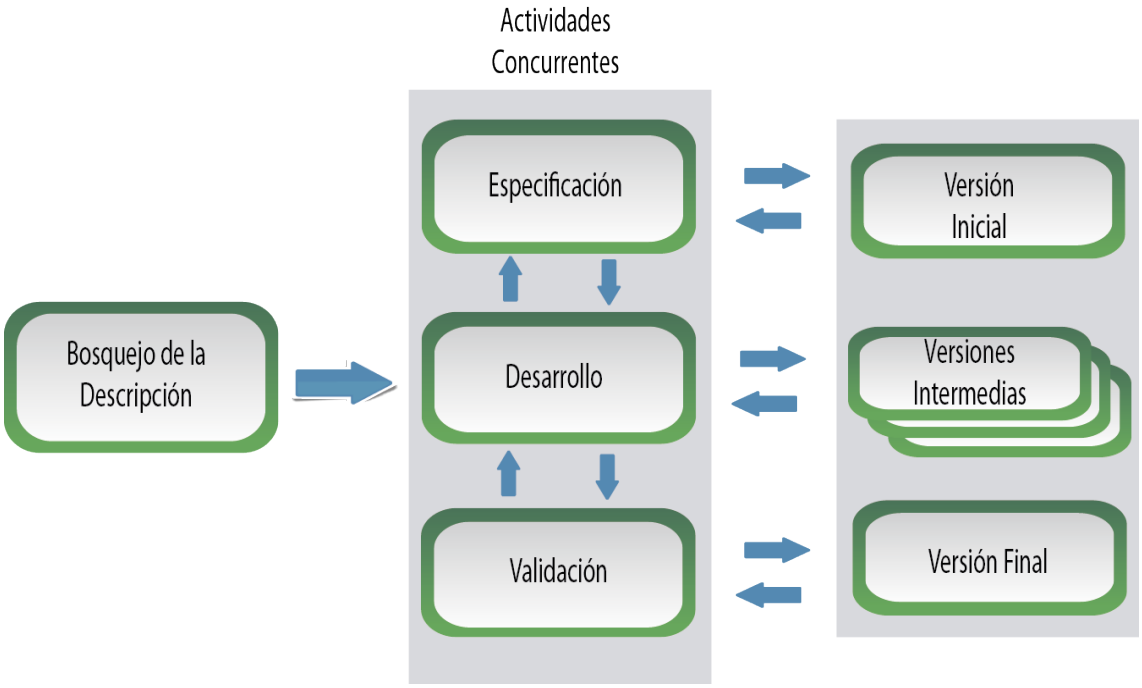


Figura 2.5: Modelo de desarrollo evolutivo.
Fuente: [9]

Existen dos tipos de desarrollo evolutivo:

- Desarrollo Exploratorio: El objetivo de este enfoque es explorar con el usuario los requisitos hasta llegar a un sistema final. El desarrollo comienza con las partes que se tiene más claras. El sistema evoluciona conforme se añaden nuevas características propuestas por el usuario.

- Enfoque utilizando prototipos: El objetivo es entender los requisitos del usuario y trabajar para mejorar la calidad de los requisitos. A diferencia del desarrollo exploratorio, se comienza por definir los requisitos que no están claros para

el usuario y se utiliza un prototipo para experimentar con ellos. El prototipo ayuda a terminar de definir estos requisitos.

Entre los puntos favorables de este modelo están:

- La especificación puede desarrollarse de forma creciente.
- Los usuarios y desarrolladores logran un mejor entendimiento del sistema. Esto se refleja en una mejora de la calidad del software.
- Es más efectivo que el modelo de cascada, ya que cumple con las necesidades inmediatas del cliente.

Desde una perspectiva de ingeniería y administración se identifican los siguientes problemas:

- Proceso no Visible: Los administradores necesitan entregas para medir el progreso. Si el sistema se necesita desarrollar rápido, no es efectivo producir documentos que reflejen cada versión del sistema.
- Sistemas pobremente estructurados: Los cambios continuos pueden ser perjudiciales para la estructura del software haciendo costoso el mantenimiento.
- Se requieren técnicas y herramientas: Para el rápido desarrollo se necesitan herramientas que pueden ser incompatibles con otras o que poca gente sabe utilizar.

Este modelo es efectivo en proyectos pequeños (menos de 100.000 líneas de código) o medianos (hasta 500.000 líneas de código) con poco tiempo para su desarrollo y sin generar documentación para cada versión. Para proyectos largos es mejor combinar lo mejor del modelo de cascada y evolutivo: se puede hacer un prototipo global del sistema y posteriormente re implementarlo con un acercamiento más estructurado. Los subsistemas con requisitos bien definidos y estables se pueden programar utilizando cascada y la interfaz de usuario se puede especificar utilizando un enfoque exploratorio.

Desarrollo formal de sistemas

Este modelo se basa en transformaciones formales de los requisitos hasta llegar a un programa ejecutable.

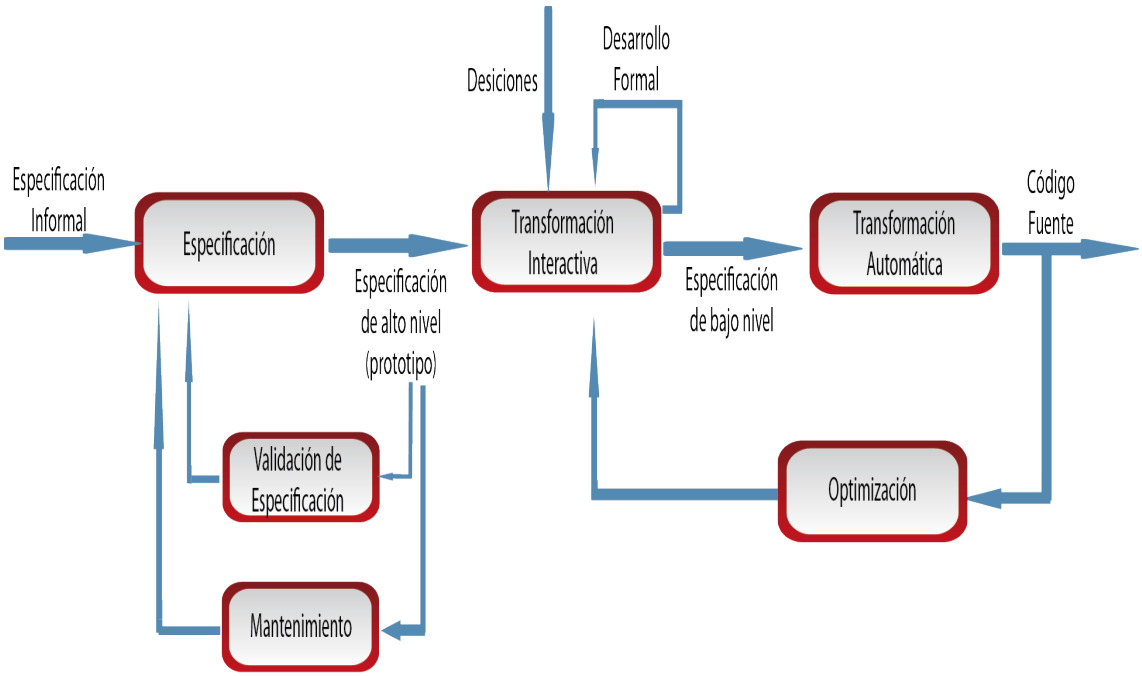


Figura 2.6: Paradigma de programación automática
Fuente: [12]

La Figura 2.6 ilustra un paradigma ideal de programación automática[9]. Se distinguen dos fases globales: especificación (incluyendo validación) y transformación.

Las características principales de este paradigma son: la especificación es formal y ejecutable constituye el primer prototipo del sistema), la especificación es validada mediante prototipación. Posteriormente, a través de transformaciones formales la especificación se convierte en la implementación del sistema, en el último paso de transformación se obtiene una implementación en un lenguaje de programación determinado. , el mantenimiento se realiza sobre la especificación (no sobre el código fuente), la documentación es generada automáticamente y el mantenimiento es realizado por repetición del proceso (no mediante parches sobre la implementación).

Observaciones sobre el desarrollo formal de sistemas:

- Permite demostrar la corrección del sistema durante el proceso de transformación. Así, las pruebas que verifican la correspondencia con la especificación no son necesarias.
- Es atractivo sobre todo para sistemas donde hay requisitos de seguridad y confiabilidad importantes.
- Requiere desarrolladores especializados y experimentados en este proceso para llevarse a cabo.

Desarrollo basado en reutilización

Es un modelo fuertemente orientado a la reutilización[9]. Este modelo consta de 4 fases:

1. Análisis de componentes: Se determina qué componentes pueden ser utilizados para el sistema en cuestión. Casi siempre hay que hacer ajustes para adecuarlos.

2. Modificación de requisitos: Se adaptan los requisitos para concordar con los componentes de la etapa anterior. Si no se puede realizar modificaciones en los requisitos, hay que seguir buscando componentes más adecuados.

3. Diseño del sistema con reutilización: Se diseña o reutiliza el marco de trabajo para el sistema. Se debe tener en cuenta los componentes localizados en la fase 2 para diseñar o determinar este marco.

4. Desarrollo e integración: El software que no puede comprarse, se desarrolla. Se integran los componentes y subsistemas. La integración es parte del desarrollo en lugar de una actividad separada.

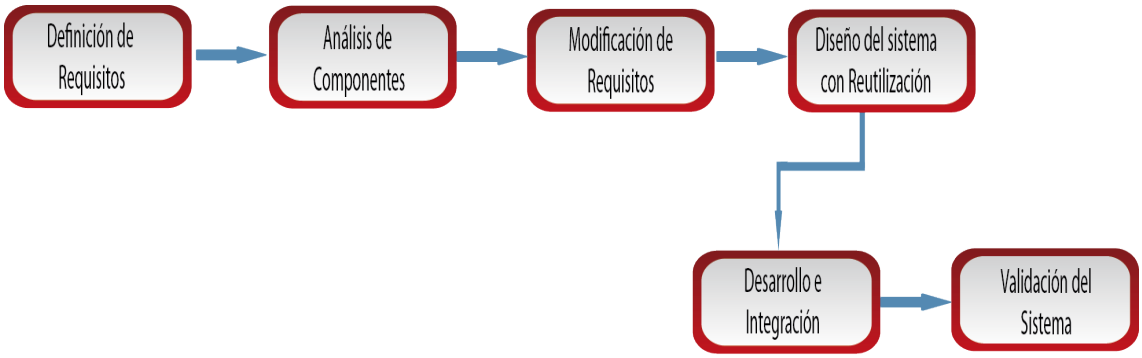


Figura 2.7: Desarrollo basado en reutilización de componentes
Fuente: [9]

Las ventajas de este modelo son:

- Disminuye el costo y esfuerzo de desarrollo.
- Reduce el tiempo de entrega.
- Disminuye los riesgos durante el desarrollo.

Las desventajas de este modelo son:

- Los “compromisos” en los requisitos son inevitables, por lo cual puede que el software no cumpla las expectativas del cliente.
- Las actualizaciones de los componentes adquiridos no están en manos de los desarrolladores del sistema.

Desarrollo incremental

Mills [13] sugirió el enfoque incremental de desarrollo como una forma de reducir la repetición del trabajo en el proceso de desarrollo y dar oportunidad de retrasar la toma de decisiones en los requisitos hasta adquirir experiencia con el sistema como se muestra en la Figura 2.8. Es una combinación del Modelo de Cascada y Modelo Evolutivo.

Reduce el rehacer trabajo durante el proceso de desarrollo y da oportunidad para retrasar las decisiones hasta tener experiencia en el sistema.

Durante el desarrollo de cada incremento se puede utilizar el modelo de cascada o evolutivo, dependiendo del conocimiento que se tenga sobre los requisitos a implementar. Si se tiene un buen conocimiento, se puede optar por cascada, si es dudoso, evolutivo.

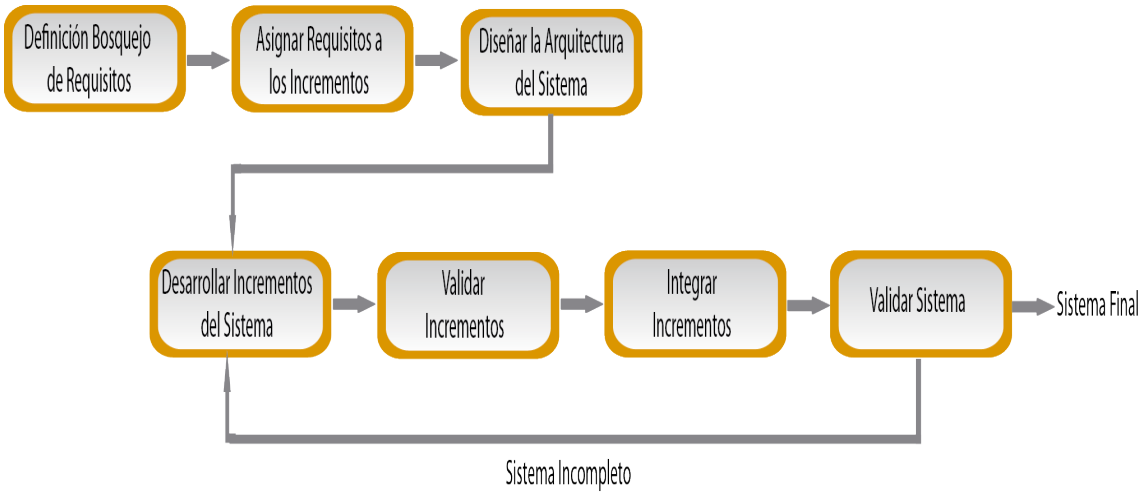


Figura 2.8: Modelo de desarrollo iterativo incremental
Fuente: [13]

Entre las ventajas del modelo incremental se encuentran:

- Los clientes no esperan hasta el fin del desarrollo para utilizar el sistema. Pueden empezar a usarlo desde el primer incremento.

- Los clientes pueden aclarar los requisitos que no tengan claros conforme ven las entregas del sistema.
- Se disminuye el riesgo de fracaso de todo el proyecto, ya que se puede distribuir en cada incremento.
- Las partes más importantes del sistema son entregadas primero, por lo cual se realizan más pruebas en estos módulos y se disminuye el riesgo de fallos.

Algunas de las desventajas identificadas para este modelo son:

- Cada incremento debe ser pequeño para limitar el riesgo (menos de 20.000 líneas).
- Cada incremento debe aumentar la funcionalidad.
- Es difícil establecer las correspondencias de los requisitos contra los incrementos.
- Es difícil detectar las unidades o servicios genéricos para todo el sistema.

Desarrollo en espiral

El modelo de desarrollo en espiral mostrado en la Figura 2.9 es actualmente uno de los más conocidos y fue propuesto por Boehm [10]. El ciclo de desarrollo se representa como una espiral, en lugar de una serie de actividades sucesivas con retrospectiva de una actividad a otra.

Cada ciclo de desarrollo se divide en cuatro fases:

1. Definición de objetivos: Se definen los objetivos. Se definen las restricciones del proceso y del producto. Se realiza un diseño detallado del plan administrativo. Se identifican los riesgos y se elaboran estrategias alternativas dependiendo de estos.
2. Evaluación y reducción de riesgos: Se realiza un análisis detallado de cada riesgo identificado. Pueden desarrollarse prototipos para disminuir el riesgo de requisitos dudosos. Se llevan a cabo los pasos para reducir los riesgos.
3. Desarrollo y validación: Se escoge el modelo de desarrollo después de la evaluación del riesgo. El modelo que se utilizará (cascada, sistemas formales, evolutivo, etc.) depende del riesgo identificado para esa fase.
4. Planificación: Se determina si continuar con otro ciclo. Se planea la siguiente fase del proyecto.

Este modelo a diferencia de los otros toma en consideración explícitamente el riesgo, esta es una actividad importante en la administración del proyecto.

El ciclo de vida inicia con la definición de los objetivos. De acuerdo a las restricciones se determinan distintas alternativas. Se identifican los riesgos al analizar los objetivos contra las alternativas. Se evalúan los riesgos con actividades como análisis detallado, simulación, prototipos, etc. Se desarrolla un poco el sistema. Se planifica la siguiente fase.

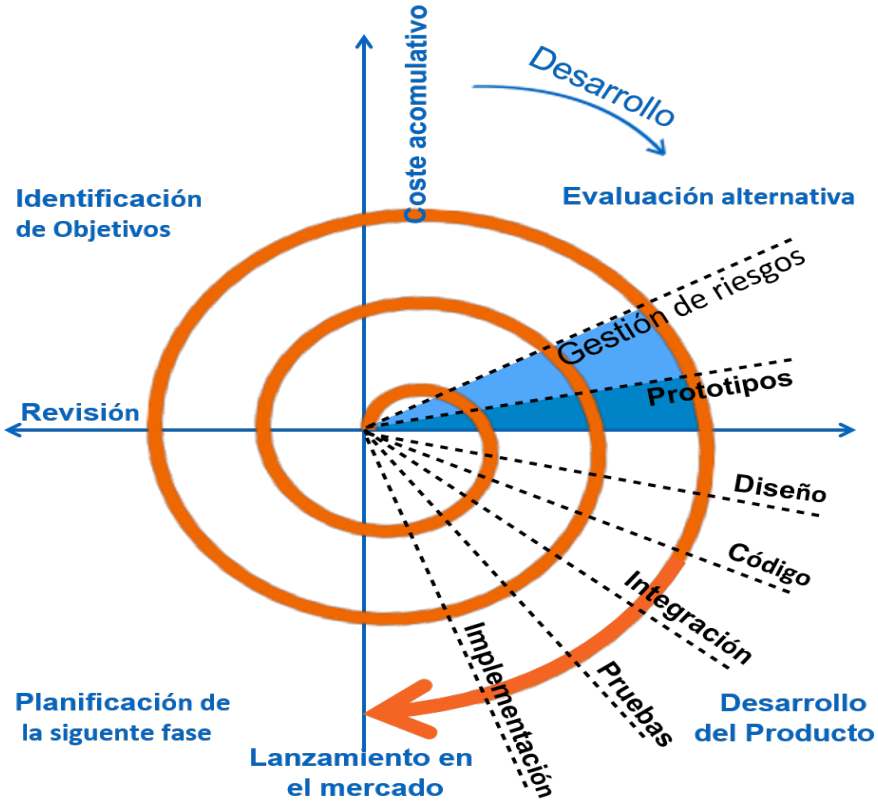


Figura 2.9: Modelo de desarrollo en espiral
Fuente: [10]

Modelo V

El modelo V permite evaluar el software en cada etapa de manera inmersa. Este modelo se conoce también como modelo de validación y verificación, ya que hace que tanto la verificación como la validación vayan en paralelo. En cada etapa, se crea la planificación de las pruebas y los casos de pruebas para verificar y validar el producto según los requisitos de la etapa.

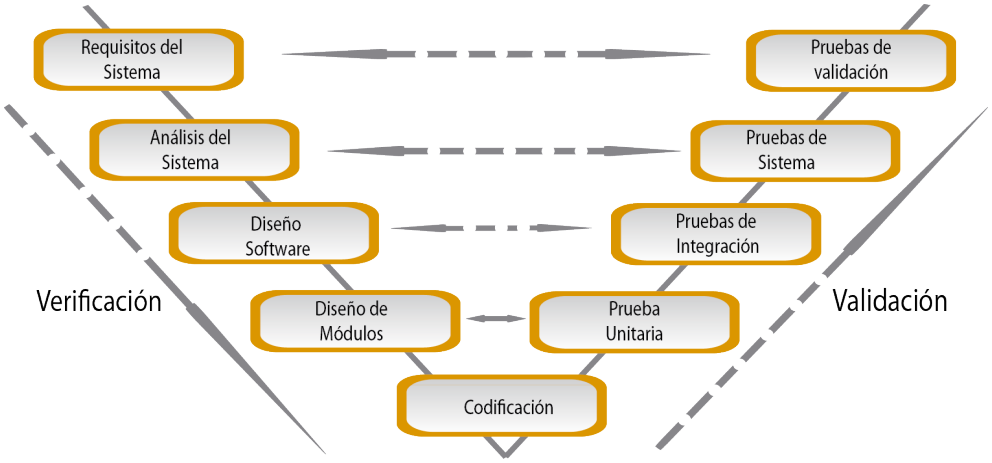


Figura 2.10: Modelo V
Fuente: [7]

Ventajas:

- El nexo que existe entre las etapas de desarrollo y los tipos de pruebas facilitan encontrar fallos.
- La curva de aprendizaje de este modelo es sencilla.
- Este método revisa cada una de las etapas del método en cascada.
- Es un método más robusto y completo que el método en cascada.
- La iteración y trabajo que hay que revisar es de fácil comprensión.
- Especifica bien los roles de los distintos tipos de pruebas a realizar.
- Involucra al usuario en las pruebas.

Desventajas:

- Difícilmente el cliente expone los requisitos claramente.
- El modelo no permite retornar a etapas anteriores.

2.2.5. Normas de Control Interno de la Contraloría General Del Estado

NORMAS DE CONTROL INTERNO PARA LAS ENTIDADES, ORGANISMOS DEL SECTOR PUBLICO Y PERSONAS JURÍDICAS DE DERECHO PRIVADO QUE DISPONGAN DE RECURSOS PÚBLICOS

COMPONENTE: 400 ACTIVIDADES DE CONTROL

SUBCOMPONENTE: 410 TECNOLOGÍA DE LA INFORMACIÓN

410-06 Administración de proyectos tecnológicos

La Unidad de Tecnología de Información definirá mecanismos que faciliten la administración de todos los proyectos informáticos que ejecuten las diferentes áreas que conformen dicha unidad. Los aspectos a considerar son:

1. Descripción de la naturaleza, objetivos y alcance del proyecto, su relación con otros proyectos institucionales, sobre la base del compromiso, participación y aceptación de los usuarios interesados.

2. Cronograma de actividades que facilite la ejecución y monitoreo del proyecto que incluirá el talento humano (responsables), tecnológicos y financieros además de los planes de pruebas y de capacitación correspondientes.

3. La formulación de los proyectos considerará el Costo Total de Propiedad CTP; que incluya no sólo el costo de la compra, sino los costos directos e indirectos, los beneficios relacionados con la compra de equipos o programas informáticos, aspectos del uso y mantenimiento, formación para el personal de soporte y usuarios, así como el costo de operación y de los equipos o trabajos de consultoría necesarios.

4. Para asegurar la ejecución del proyecto se definirá una estructura en la que se nombre un servidor responsable con capacidad de decisión y autoridad y administradores o líderes funcionales y tecnológicos con la descripción de sus funciones y responsabilidades.

5. Se cubrirá, como mínimo las etapas de: inicio, planeación, ejecución, control, monitoreo y cierre de proyectos, así como los entregables, aprobaciones y compromisos formales mediante el uso de actas o documentos electrónicos legalizados.

6. El inicio de las etapas importantes del proyecto será aprobado de manera formal y comunicado a todos los interesados.

7. Se incorporará el análisis de riesgos. Los riesgos identificados serán permanentemente evaluados para retroalimentar el desarrollo del proyecto, además de ser registrados y considerados para la planificación de proyectos futuros.

8. Se deberá monitorear y ejercer el control permanente de los avances del proyecto.

9. Se establecerá un plan de control de cambios y un plan de aseguramiento de calidad que será aprobado por las partes interesadas.

10. El proceso de cierre incluirá la aceptación formal y pruebas que certifiquen la calidad y el cumplimiento de los objetivos planteados junto con los beneficios obtenidos.

410-07 Desarrollo y adquisición de software aplicativo

La Unidad de Tecnología de Información regulará los procesos de desarrollo y adquisición de software aplicativo con lineamientos, metodologías y procedimientos. Los aspectos a considerar son:

1. La adquisición de software o soluciones tecnológicas se realizarán sobre la base del portafolio de proyectos y servicios priorizados en los planes estratégico y operativo previamente aprobados considerando las políticas públicas establecidas por el Estado, caso contrario serán autorizadas por la máxima autoridad previa justificación técnica documentada.

2. Adopción, mantenimiento y aplicación de políticas públicas y estándares internacionales para: codificación de software, nomenclaturas, interfaz de usuario, interoperabilidad, eficiencia de desempeño de sistemas, escalabilidad, validación contra requerimientos, planes de pruebas unitarias y de integración.

3. Identificación, priorización, especificación y acuerdos de los requerimientos funcionales y técnicos institucionales con la participación y aprobación formal de las unidades usuarias. Esto incluye, tipos de usuarios, requerimientos de: entrada, definición de interfaces, archivo, procesamiento, salida, control, seguridad, plan de pruebas y trazabilidad o pistas de auditoría de las transacciones en donde aplique.

4. Especificación de criterios de aceptación de los requerimientos que cubrirán la definición de las necesidades, su factibilidad tecnológica y económica, el análisis de riesgo y de costo-beneficio, la estrategia de desarrollo o compra del

software de aplicación, así como el tratamiento que se dará a aquellos procesos de emergencia que pudieran presentarse.

5. En los procesos de desarrollo, mantenimiento o adquisición de software aplicativo se considerarán: estándares de desarrollo, de documentación y de calidad, el diseño lógico y físico de las aplicaciones, la inclusión apropiada de controles de aplicación diseñados para prevenir, detectar y corregir errores e irregularidades de procesamiento, de modo que éste, sea exacto, completo, oportuno, aprobado y auditable. Se considerarán mecanismos de autorización, integridad de la información, control de acceso, respaldos, diseño e implementación de pistas de auditoría y requerimientos de seguridad. La especificación del diseño considerará las arquitecturas tecnológicas y de información definidas dentro de la organización.

6. En caso de adquisición de programas de computación (paquetes de software) se preverán tanto en el proceso de compra como en los contratos respectivos, mecanismos que aseguren el cumplimiento satisfactorio de los requerimientos de la entidad. Los contratos tendrán el suficiente nivel de detalle en los aspectos técnicos relacionados, garantizar la obtención de las licencias de uso y/o servicios, definir los procedimientos para la recepción de productos y documentación en general, además de puntualizar la garantía formal de soporte, mantenimiento y actualización ofrecida por el proveedor.

7. En los contratos realizados con terceros para desarrollo de software deberá constar que los derechos de autor serán de la entidad contratante y el contratista entregará el código fuente. En la definición de los derechos de autor se aplicarán las disposiciones de la Ley de Propiedad Intelectual. Las excepciones serán técnicamente documentadas y aprobadas por la máxima autoridad o su delegado.

8. La implementación de software aplicativo adquirido incluirá los procedimientos de configuración, aceptación y prueba personalizados e implantados. Los aspectos a considerar incluyen la validación contra los términos contractuales, la arquitectura de información de la organización, las aplicaciones existentes, la interoperabilidad con las aplicaciones existentes y los sistemas de bases de datos, la eficiencia en el desempeño del sistema, la documentación y los manuales de usuario, integración y planes de prueba del sistema.

9. Los derechos de autor del software desarrollado a la medida pertenecerán a la entidad y serán registrados en el organismo competente. Para el caso de

software adquirido se obtendrá las respectivas licencias de uso.

10. Formalización con actas de aceptación por parte de los usuarios, del paso de los sistemas probados y aprobados desde el ambiente de desarrollo/prueba al de producción y su revisión en la post-implantación.

11. Elaboración de manuales técnicos, de instalación y configuración; así como de usuario, los cuales serán difundidos, publicados y actualizados de forma permanente.

COMPONENTE: 500 INFORMACIÓN Y COMUNICACIÓN

La máxima autoridad y los directivos de la entidad, deben identificar, capturar y comunicar información pertinente y con la oportunidad que facilite a las servidoras y servidores cumplir sus responsabilidades.

El sistema de información y comunicación, está constituido por los métodos establecidos para registrar, procesar, resumir e informar sobre las operaciones técnicas, administrativas y financieras de una entidad. La calidad de la información que brinda el sistema facilita a la máxima autoridad adoptar decisiones adecuadas que permitan controlar las actividades de la entidad y preparar información confiable.

El sistema de información permite a la máxima autoridad evaluar los resultados de su gestión en la entidad versus los objetivos predefinidos, es decir, busca obtener información sobre su nivel de desempeño.

La comunicación es la transmisión de información facilitando que las servidoras y servidores puedan cumplir sus responsabilidades de operación, información financiera y de cumplimiento.

Los sistemas de información y comunicación que se diseñen e implanten deberán concordar con los planes estratégicos y operativos, debiendo ajustarse a sus características y necesidades y al ordenamiento jurídico vigente.

La obtención de información interna y externa, facilita a la alta dirección preparar los informes necesarios en relación con los objetivos establecidos.

El suministro de información a los usuarios, con detalle suficiente y en el momento preciso, permitirá cumplir con sus responsabilidades de manera eficiente y eficaz.

500-01 Controles sobre sistemas de información

Los sistemas de información contarán con controles adecuados para garantizar confiabilidad, seguridad y una clara administración de los niveles de acceso a la información y datos sensibles.

En función de la naturaleza y tamaño de la entidad, los sistemas de información serán manuales o automatizados, estarán constituidos por los métodos establecidos para registrar, procesar, resumir e informar sobre las operaciones administrativas y financieras de una entidad y mantendrán controles apropiados que garanticen la integridad y confiabilidad de la información.

La utilización de sistemas automatizados para procesar la información implica varios riesgos que necesitan ser considerados por la administración de la entidad. Estos riesgos están asociados especialmente con los cambios tecnológicos por lo que se deben establecer controles generales, de aplicación y de operación que garanticen la protección de la información según su grado de sensibilidad y confidencialidad, así como su disponibilidad, accesibilidad y oportunidad.

Las servidoras y servidores a cuyo cargo se encuentre la administración de los sistemas de información, establecerán los controles pertinentes para que garanticen razonablemente la calidad de la información y de la comunicación.

2.2.6. Calidad del Software

La calidad del software se refiere a la eficiencia, confiabilidad e integridad de un sistema, al grado de cumplimiento de los requerimientos establecidos en un inicio.

Según el estándar 610 de la IEEE la calidad del software es "el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario". [14] La calidad debe controlarse en todas las etapas de desarrollo de software.

2.2.7. Estándares de Calidad

Los Estándares de Calidad forman parte de la Ingeniería del Software, son un conjunto de criterios que guían la forma en que se aplican procedimientos y metodologías, con el objetivo de que permitan unificar la filosofía de trabajo, lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, elevando así la productividad tanto para el desarrollo como para el control de la calidad del software.

2.2.7.1. ISO/IEC 9126

ISO/IEC 9126 es un estándar internacional para la evaluación de software, en el cual se establecen las características de calidad para productos de software.

Está compuesto por 4 partes:

1. Modelo de calidad (FDIS 9126-1), describe un modelo de dos partes para calidad de productos de software:
   a) Calidad interna y externa, y
   b) Calidad en uso
2. Métricas externas (PDTR 9126-2)
3. Métricas internas (PDTR 9126-3)
4. Métricas de calidad en uso (PDTR 9126-4)

El modelo de calidad del producto software de esta norma incluye:

- Calidad interna: medible a partir de las características intrínsecas, como el código fuente.
- Calidad externa: medible en el comportamiento del producto, como en una prueba.
- Calidad en uso: durante la utilización efectiva por parte del usuario.

Calidad Interna y Externa

El modelo de calidad del estándar ISO-9126 establece seis características para la calidad interna y externa, cada una de las cuales se detalla a través de un conjunto de subcaracterísticas que permiten profundizar en la evaluación de la calidad de productos de software.

Ver Figura 2.11.

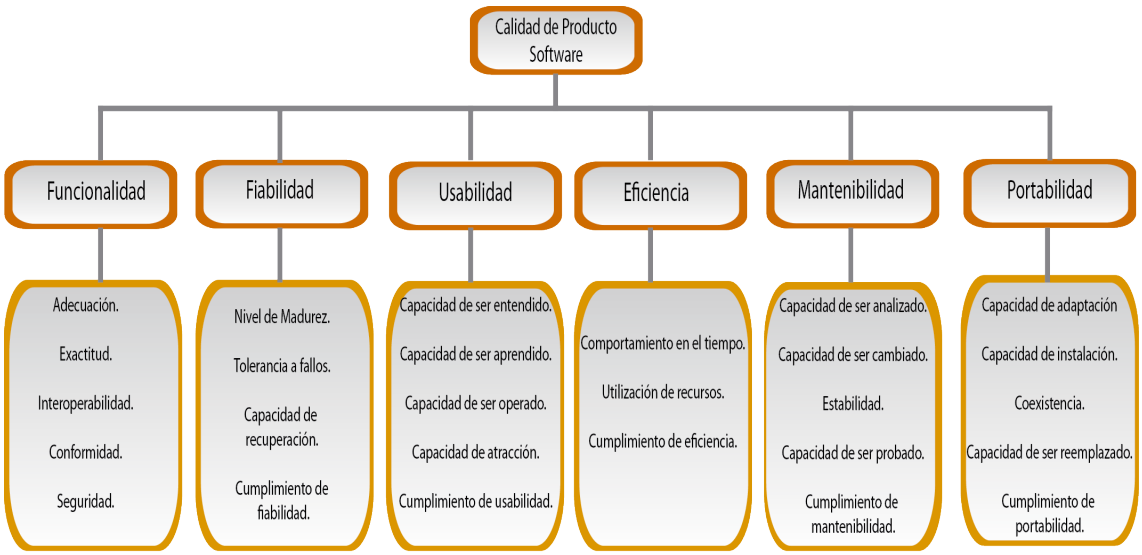


Figura 2.11: Características y subcaracterísticas de la calidad de un producto software

Fuente: [15]

Definición de las características y subcaracterísticas según el modelo de calidad del estándar ISO-9126:

1. **Funcionalidad**

   La funcionalidad reúne un conjunto de atributos que permiten calificar si un producto de software maneja en forma adecuada el conjunto de funciones que satisfagan las necesidades para las cuales fue diseñado.

   Subcategorías:

   - Adecuación

     Se enfoca a evaluar si el software cuenta con un conjunto de funciones apropiadas para efectuar las tareas que fueron especificadas en su definición.

   - Exactitud

     Este atributo permite evaluar si el software presenta resultados o efectos acordes a las necesidades para las cuales fue creado.

   - Interoperabilidad

     Permite evaluar la habilidad del software de interactuar con otros sistemas previamente especificados.

- Conformidad

  Evalúa si el software se adhiere a estándares, convenciones o regulaciones en leyes y prescripciones similares.

- Seguridad

  Se refiere a la habilidad de prevenir el acceso no autorizado, ya sea accidental o premeditado, a los programas y datos.

2. **Confiabilidad (Fiabilidad)**

   La confiabilidad reúne un conjunto de atributos los cuales se refieren a la capacidad del software de mantener su nivel de ejecución bajo condiciones normales en un periodo de tiempo establecido.

   Subcategorias:

   - Nivel de madurez

     Permite medir la frecuencia de falla por errores en el software.

   - Tolerancia a fallos

     Se refiere a la habilidad de mantener un nivel específico de funcionamiento en caso de fallas del software o de cometer infracciones de su interfaz específica.

   - Capacidad de recuperación

     Se refiere a la capacidad de restablecer el nivel de operación y recobrar los datos que hayan sido afectados directamente por una falla, así como al tiempo y el esfuerzo necesarios para lograrlo.

   - Cumplimiento de la fiabilidad

     Se refiere a la capacidad del producto software para adherirse a normas, convenciones o regulaciones relacionadas con al fiabilidad.

3. **Usabilidad**

   La usabilidad reúne un conjunto de atributos que permiten evaluar el esfuerzo necesario que deberá invertir el usuario para utilizar el sistema.

   Subcategorias:

   - Capacidad de ser entendido

     Se refiere a la capacidad del producto de software para permitir al usuario entender si el software es adecuado, y cómo se puede utilizar para determinadas tareas y condiciones de uso.

- Capacidad de ser aprendido

  Se refiere a la capacidad del producto de software para permitir al usuario aprender su aplicación.

- Capacidad de ser operado

  Se refiere a la capacidad del producto de software para permitir al usuario operarlo y controlarlo.

- Capacidad de atracción

  Se refiere a la capacidad del producto de software para ser atractivo para el usuario.

- Cumplimiento de usabilidad

  Se refiere a la capacidad del producto de software a que se adhiera a las normas, convenciones o regulaciones en leyes y reglamentos relativos a la usabilidad.

4. **Eficiencia**

   La eficiencias reúne un conjunto de atributos que permiten evaluar la relación entre el nivel de funcionamiento del software y la cantidad de recursos usados.

   Subcategorias:

   - Comportamiento en el tiempo

     Se refiere a la capacidad del producto de software para proporcionar una respuesta adecuada, tiempos de procesamiento y tasas de rendimiento en el desempeño de su función bajo condiciones establecidas.

   - Utilización de recursos

     Se refiere a la capacidad del producto de software para utilizar cantidades y tipos apropiados de los recursos cuando el software realiza su función bajo condiciones establecidas.

   - Cumplimiento de Eficiencia

     Se refiere a la capacidad del producto de software a que se adhiera a las normas, convenciones o regulaciones en leyes y reglamentos relativos a la eficiencia

5. **Mantenibilidad**

   La mantenibilidad reúne un conjunto de atributos que permiten medir el esfuerzo necesario para realizar modificaciones al software, ya sea por la

corrección de errores, mejoras, cambios en los requisitos o por el incremento de funcionalidad

Subcategorias:

- Capacidad de ser analizado

  La capacidad del producto de software para ser diagnosticado por deficiencias o causas de las fallas en el software, o para identificar las partes que han de ser modificadas.

- Capacidad de ser cambiado

  La capacidad del producto de software para permitir una modificación especifica a ser implementada.

- Estabilidad

  La capacidad del producto de software para evitar los efectos inesperados de las modificaciones del software.

- Capacidad de ser probado

  La capacidad del producto de software para permitir la validación de las modificaciones realizadas.

- Cumplimiento de Mantenibilidad

  La capacidad del producto de software a que se adhiera a las normas, convenciones o regulaciones en leyes y reglamentos relativos a la Mantenibilidad.

6. **Portabilidad**

La portabilidad reúne un conjunto de atributos que permiten medir la capacidad del producto de software para ser transferido de un medio a otro, un ambiente puede incluir organización, hardware o software.

Subcategorias:

- Capacidad de adaptación

  La capacidad del producto de software de adaptarse a diferentes entornos específicos, sin aplicar acciones o medios distintos de los previstos para este fin.

- Capacidad de Instalación

  La capacidad del producto de software de ser instalado en un ambiente específico.

- Coexistencia

  La capacidad del producto de software para coexistir con otro software independiente en un entorno común compartiendo recursos comunes.

- Capacidad de ser reemplazado

  La capacidad del producto de software de ser utilizado en lugar de otro producto de software para el mismo propósito en el mismo entorno.

- Cumplimiento de portabilidad

  La capacidad del producto de software a que se adhiera a las normas, convenciones o regulaciones en leyes y reglamentos relativos a la portabilidad.

Calidad en uso

La calidad en uso se refiere a la aceptación por parte del usuario final. El modelo de calidad del estándar ISO-9126 establece los siguientes atributos para la calidad en uso:

1. Efectividad: Capacidad del producto software para permitir a los usuarios alcanzar objetivos especificados con exactitud y completitud, en un contexto de uso especificado.

2. Productividad: Capacidad del producto software para permitir a los usuarios gastar una cantidad adecuada de recursos con relación a la efectividad alcanzada, en un contexto de uso especificado.

3. Seguridad física: Capacidad del producto software para alcanzar niveles aceptables del riesgo de hacer daño a personas, al negocio, al software, a las propiedades o al medio ambiente en un contexto de uso especificado.

4. Satisfacción: Capacidad del producto software para satisfacer a los usuarios en un contexto de uso especificado.

2.2.8. Verificación de Software

La verificación de software consiste en determinar si los sistemas procedentes de un ciclo de vida de software están de acuerdo con sus especificaciones y cumplen los requerimientos funcionales y no funcionales que se le han especificado, es decir que este libre de errores.

2.2.8.1. Verificación Estática

Se ocupa del análisis de representaciones estáticas del sistema para descubrir problemas. Pueden ser complementadas por documentos basados en herramientas y análisis de código [16].

2.2.8.2. Verificación Dinámica

Se ocupa de la ejercitación y la observación del comportamiento del producto. Permite que la detección del defecto sea combinada con otros chequeos de la calidad [7].

Este tipo de verificación se centra en la observación y comportamiento del producto final.

2.2.9. Validación de Software

La validación de software consiste en garantizar que el software desarrollado cumpla las expectativas del cliente, esto quiere decir que la validación va más allá de comprobar si el sistema está acorde con sus especificaciones. La validación de software se centra más en probar que el software hace lo que el usuario espera y no solamente comprobar lo que se ha especificado.

2.2.9.1. Validación Simple

La validación simple centra su trabajo en inspecciones mediante procesos establecidos para validar el cumplimiento de los requerimientos establecidos en el desarrollo.

2.2.9.2. Validación Cruzada

La Validación cruzada es una técnica utilizada para evaluar resultados de un análisis estadísticos. Utilizada en proyectos de Inteligencia Artificial, para validar modelos generados.

2.2.10. Métodos de Verificación y Validación de Software

- Método de Caja Negra
- Método de Caja Blanca
- Método Top-Down
- Método Act-like-a-customer

- Método ATAM
- Método ADR(Active Design Reviews)
- Método ARID(Active Reviews for Intermediate Designs)

2.2.10.1. Método de Caja Negra

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se ajusta a su especificación. Por ello se denominan pruebas funcionales, el tester se limita a ingresar datos como entradas y estudiar las salidas, sin preocuparse de lo que pueda estar haciendo el módulo por dentro[16].

Las pruebas de caja negra intentan encontrar errores como:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en acceso a BD externas.
- Errores de rendimiento.

2.2.10.2. Método de Caja Blanca

Conocido como pruebas de Cobertura en la cual se conoce el diseño interno del Software, como ha sido desarrollado a diferencia del método de caja negra en este método se hace énfasis en el detalle de implementación.

El principal objetivo es probar la lógica del programa desde el punto de vista algorítmico[16].

La idea de este método es el de diseñar un plan de pruebas en las que se vaya ejecutando sistemáticamente el código hasta que haya corrido todo o la gran mayoría de el, esto se vuelve un procedimiento complicado cuando el programa contiene código de difícil alcance.

2.2.10.3. Método Top-Down

Cada parte del sistema se prueba con mayor detalle a partir de atributos de calidad y patrones, hasta que la especificación completa es lo suficientemente detallada para validar el modelo. El modelo Top Down se diseña con ayuda de cajas negras.[16]

Una de las desventajas de este método es que no permite detectar la mayor cantidad de errores.

2.2.10.4. Método Act-like-a-customer

Es un método de prueba en el cual las pruebas se desarrollan basadas en el conocimiento de cómo los clientes usan el Software, es decir se enfocan en encontrar los defectos que los clientes pueden encontrar [16].

Una de las desventajas es que los productos software complejos tienen muchos errores y los clientes regularmente encuentran un pequeño porcentaje de estos.

2.2.10.5. Método ATAM

Architecture Tradeoff Analysis Method es un método que nos ayuda a elegir una arquitectura adecuada para un sistema de software que satisfaga las necesidades de calidad . El método ATAM resulta útil cuando se lo realiza temprano en el desarrollo de software, cuando el costo de cambiar las arquitecturas es mínimo.

2.2.10.6. Método ADR

El método Revisión de Diseño Activo al igual que el método ATAM asegura la calidad del producto software, se usa ADR para evaluar unidades de software, como módulos o componentes.

En este método se realizan preguntas que tienden a dirigirse hacia:

- La calidad e integridad de la documentación.
- La suficiencia de los servicios proporcionados por el diseño.

2.2.10.7. Método ARID

El método de Revisión Activa para Diseño Intermedio, es la combinación entre el método ATAM y el método ADR el cual saca provecho de las ventajas de cada uno de estos métodos.

Entre las ventajas de utilizar ARID se encuentran las siguientes:

- Este método es adecuado para realizar la evaluación de diseños parciales en las etapas tempranas del desarrollo.
- Permite el descubrimiento de errores a tiempo lo que ayuda a saber si el diseño es viable.

2.2.11. Proceso de Pruebas

2.2.11.1. Niveles de Pruebas

Rakitin detalla los siguientes niveles de prueba[16]:

Pruebas Unitarias

Verifican la funcionalidad y estructura de cada componente individualmente una vez que ha sido codificado.

Pruebas de Integración

Verifican el correcto ensamblaje entre los distintos componentes de un sistema, una vez éstos han sido probados de forma unitaria, con especial atención a los interfaces, tanto internos como externos.

Pruebas de Sistema

Prueban a fondo el sistema,comprobando su funcionalidad e integridad globalmente, en un entorno lo más parecido posible al entorno final de producción.

Pruebas de Implantación

Comprueban el correcto funcionamiento del sistema dentro del entorno real de producción.

Pruebas de Aceptación

Verifican que el sistema cumple con todos los requisitos indicados y permite que los usuarios del sistema den el visto bueno definitivo.

Pruebas de Regresión

Comprueba que los cambios sobre un componente del sistema, no introducen un comportamiento no deseado o errores adicionales en otros componentes no modificados.

Pruebas de Valores de Frontera

La prueba de Frontera o también conocido como Boundary test, es un método para diseñar casos de prueba que se enfocan en pruebas cerca de los valores límites permitidos para ver el comportamiento del sistema.

2.2.11.2. Métodos y tipos de pruebas

Funcional

El objetivo de este tipo de prueba es el determinar si una función específica trabaja como lo especificado.

Algorítmicos

El objetivo de este tipo de prueba es el determinar si un algoritmo ha sido correctamente implementado.

Positivos

El objetivo de este tipo de prueba es el determinar si los resultados de las pruebas son consistentes cuando se enfrenta con las entradas adecuadas.

Negativos

El objetivo de este tipo de prueba es el determinar si el comportamiento del software es estable cuando se enfrenta con entradas inválidas o acciones con operadores inesperados.

Usables

Se enfoca en el usuario, a través de un conjunto de actividades y técnicas, cuyo objetivo es crear un sistema usable para usuarios expertos o inexpertos.

2.2.12. Técnicas de Verificación y Validación de Software

2.2.12.1. Análisis y Pruebas Funcionales

Esta técnica consiste en ejecutar parte o todo el sistema para validar que los requisitos del usuario se encuentren cubiertos[16].

Las técnicas de pruebas funcionales y análisis son las siguientes:

- Pruebas de aceptación.
- Pruebas alfa y beta.
- Pruebas de usuario.
- Pruebas en pares.

2.2.12.2. Análisis y Pruebas Estructurales

Esta técnica permite examinar la lógica de las unidades y puede usarse para apoyar los requerimientos del software para una prueba de cobertura, es decir, cuando el programa se ha ejecutado[16].

Las técnicas de pruebas estructurales y análisis son las siguientes:

- Basadas en el flujo de control.
- Basadas en el flujo de los datos.

2.2.12.3. Error-Oriented

Son técnicas que se enfocan en evaluar la presencia o ausencia de errores en el proceso de programación. Existen tres tipos:

1. Basadas en el error

   Encuentra errores en el proceso de programación. Estas pruebas pueden ser conducidas por historias de errores del programador y por las medidas de complejidad del software. Son aplicadas en la etapa de desarrollo.

2. Basadas en los fallos

   Este tipo de pruebas tiene como objetivo demostrar que ciertas fallas no necesariamente están en el código.

3. Probable corrección

   Aplicar este tipo de prueba ayuda a disminuir la probabilidad de que existan fallos en el programa. Se ejecutan pruebas en métodos, funciones con el objetivo de determinar que las mismas están o no implementadas de forma correcta.

2.2.12.4. Testing and Analysis

Las actividades de pruebas y análisis ocurren a lo largo del desarrollo y evolución de los sistemas de software, desde la entrega de requerimientos hasta la posterior evolución. La calidad depende de cada parte del proceso de software, no sólo en análisis y pruebas de software.

No hay una cantidad específica de pruebas y análisis que puedan compensar la mala calidad que se deriva de otras actividades. Por otra parte, una característica esencial de los procesos de software que producen productos de alta calidad es que la prueba y el análisis de software están completamente integrados[17].

2.2.12.5. Estrategias de Integración

Consiste en descubrir errores asociados con las interfaces de los módulos. Su objetivo es tomar los módulos individuales de manera sistemática para formar subsistemas y al final un sistema completo.

Las técnicas de las estrategias de integración son las siguientes:

- Incremental: se prueba el programa en pequeñas porciones en las que los fallos son más fáciles de detectar. Puede ser ascendente o descendente.
- No Incremental: se combinan todos los módulos para probar todo el programa, es más difícil identificar de qué módulo provienen los errores.

2.2.12.6. Hybrid Approaches (Enfoques Híbridos)

Combina más de una técnica, su objetivo es aprovechar las ventajas de cada una de ellas, contribuyendo de esta manera a elaborar un software de calidad[16].

2.2.12.7. Análisis de Flujo de Transacción

El flujo de transacción se refiere al movimiento de datos a través de un camino de llegada que convierte la información del mundo exterior en una transacción. Se evalúa la transacción y de acuerdo a su valor, el flujo sigue por uno de los caminos de acción[2].

2.2.12.8. Stress Testing (Análisis de Estrés)

Este tipo de prueba es usado para determinar la estabilidad de un sistema, pone énfasis en la robustez, disponibilidad y manejo de errores bajo condiciones desfavorables o una carga pesada. Su objetivo es asegurar que el software no se bloquee en caso de existir recursos insuficientes como memoria o espacio en disco.

2.2.12.9. Análisis de Falla

Las pruebas basadas en fallas utilizan un modelo de fallas directamente para hipotetizar fallas potenciales en un programa bajo prueba, así como para crear o evaluar conjuntos de pruebas basados en su eficacia en la detección de esas fallas hipotéticas. Se usan comúnmente en pruebas funcionales y estructurales, por ejemplo cuando se identifican valores de error para las características de parámetros en la prueba de partición de categoría o cuando se rellenan catálogos con valores erróneos[17].

2.2.12.10. Análisis de Concurrencia

Se lo conoce también como multi-user testing. Este tipo de prueba busca detectar defectos en el sistema cuando múltiples usuarios realizan la misma acción y al mismo tiempo en una aplicación, módulo o base de datos.

2.2.12.11. Análisis de Algoritmos

El Análisis de Algoritmos es considerado un método automatizado muy útil en la creación de pruebas que son mucho más potentes en términos de cobertura de los requisitos y la cobertura de datos de prueba y también más rápido de crear. Este tipo de prueba resuelve el problema de los cambios frecuentes en los requisitos y lenta creación de casos de prueba manual.

2.2.12.12. Análisis de Cobertura

Las pruebas de cobertura miden la cantidad de pruebas realizadas por un conjunto de casos de prueba.

Ventajas:

- Crea casos de prueba adicionales para aumentar la cobertura.
- Ayuda a encontrar áreas de un programa que no han sido cubiertas por un conjunto de casos de prueba.
- Ayuda a determinar una medida cuantitativa de la cobertura de código, que indirectamente mide la calidad de la aplicación o producto.

2.2.12.13. Análisis de Flujo de Datos

El ejercicio de cada declaración o rama con casos de prueba es una meta práctica, pero el ejercicio de cada camino es imposible. Incluso el número de rutas simples (es decir, sin bucles) puede ser exponencial en el tamaño del programa. Por lo tanto, los criterios de selección y adecuación orientados a la trayectoria deben seleccionar una pequeña fracción de las rutas de flujo de control. Los criterios de adecuación de la prueba de flujo de datos mejoran con respecto a los criterios de flujo de control puro, seleccionando caminos basados en cómo un elemento sintáctico puede afectar el cálculo de otro[17].

2.2.12.14. Inspecciones

Las inspecciones de software son revisiones manuales y colaborativas que se pueden aplicar a cualquier artefacto de software desde documentos de requisitos hasta código fuente. La inspección complementa las pruebas ayudando a controlar muchas propiedades que son difíciles o imposibles de verificar dinámicamente. Su flexibilidad hace que la inspección sea particularmente valiosa cuando otros análisis más automatizados no son aplicables[17].

2.2.12.15. Model Checking (Verificación de Modelos)

Los modelos se utilizan a menudo para expresar requisitos e integran tanto la estructura como la información de fallos que pueden ayudar a generar especificaciones de casos de prueba. El flujo de control y las pruebas de flujo de datos se basan en modelos extraídos del código de programa. Los modelos también se pueden extraer de las especificaciones y el diseño, lo que nos permite hacer uso de información adicional sobre el comportamiento previsto.

Las pruebas basadas en modelos consisten en usar modelos de comportamiento esperado para producir especificaciones de casos de prueba que pueden revelar discrepancias entre el comportamiento real del programa y el modelo[17].

2.2.13. Herramientas de Verificación y Validación de Software

2.2.13.1. Herramientas Open Source

Bugzilla Testopia

Es un administrador de casos de prueba, interactúa con el sistema Bugzilla mediante extensiones. Bugzilla es una herramienta que detecta defectos o bugs en sitios web o aplicaciones.

Testopia permite realizar pruebas de software, reportes de defectos encontrados además de seguimientos y resultados de los casos de prueba.

Características:

1. Se encuentra en la versión 2.5.
2. Se integra con Bugzilla y todos sus componentes, productos y versiones.
3. Permite a los usuarios logearse en una herramienta y usar los permisos de grupo de Bugzilla para limitar el acceso a la modificación de los objetos testeados.

4. Permite a los usuarios adjuntar los fallos encontrados a los resultados de un caso de prueba para una administración centralizada del proceso de ingeniería del software.

qaBook

Son un conjunto de herramientas profesionales de administración de pruebas. Ofrece los siguientes productos:

1. **QABook Agile:**

   Solución de administración de proyectos, pruebas y defectos. Diseñada para las metodologías ágiles Scrum y Kanban. Ofrece versión de prueba.

   Características:

   a) Pizarra ágil para las metodologías Scrum y Kanban

   b) Panel de control personalizable en tiempo real

   c) El seguimiento de errores permite crear, gestionar, evaluar y priorizar errores.

   d) Permite visualizar mediante gráficos los datos obtenidos de los diferentes módulos.

   e) Permite la integración con JIRA.

2. **QABook Enterprise:**

   Herramienta de gestión de pruebas a gran escala. Es una aplicación basada en la web tiene un corredor de pruebas completo, la capacidad de capturar y editar capturas de pantalla y un tablero de control personalizable en tiempo real. QABook Enterprise es una herramienta adecuada para las metodologías en cascada y modelo en V. Ofrece versión de prueba.

   Características:

   a) Permite definir las necesidades y establecer prioridades, rastrea las pruebas y los errores y permite visualizarlos mediante gráficos.

   b) Divide el proyecto en unidades más pequeñas, como áreas de funcionalidad, versiones o subsistemas.

   c) Al ejecutar las pruebas, permite especificar qué entornos se están probando.

   d) Crea y administra fallas o errores que están causando que un sistema de software se bloquee o produzca salidas no válidas.

e) Permite la integración con JIRA.

3. **QABook BugReach:**

   QABook Bugreach es un sistema de gestión que proporciona una forma flexible de detectar defectos en múltiples proyectos. Adecuado para cualquier tipo y tamaño de proyecto incluyendo aplicaciones web, móviles y empresariales. Ofrece versión de prueba.

   Características:

   a) Administra todos los errores en una pantalla de administración de defectos.

   b) Puede usarse como una herramienta independiente o junto con QABook Enterprise.

   c) Permite la integración con JIRA.

   d) Genera gráficos que muestran defectos por estado, prioridad, gravedad y asignación.

   e) Permite trabajar en múltiples equipos y múltiples ubicaciones, los desarrolladores pueden ocuparse de los bugs utilizando BugReach, mientras que los testers pueden utilizar QABook Enterprise o Desktop para acceder a requisitos y casos de prueba.

4. **QABook Desktop:**

   Es la versión de escritorio cliente-servidor de QABook Enterprise. Es gratuito.

   Características

   a) Integración con Jira

   b) Adecuado para las metodologías en cascada y modelo en V.

   c) Permite definir las necesidades y establecer prioridades, rastrea las pruebas y los errores y permite visualizarlos mediante gráficos.

   d) Divide el proyecto en unidades más pequeñas, como áreas de funcionalidad, versiones o subsistemas.

   e) Cuenta con un repositorio centralizado que permite al equipo trabajar en el mismo proyecto y datos simultáneamente. Mediante la opción exportación es posible compartir datos con compañeros o para uso sin conexión.

f) Crea y administra fallas o errores que están causando que un sistema de software se bloquee o produzca salidas no válidas.

g) Permite utilizarlo fuera de la oficina para realizar las pruebas de campo/sitio/usuario, para luego importar los datos al repositorio centralizado.

h) Cuenta con un Editor de Imagen para la edición rápida de las capturas de pantalla tomadas durante la ejecución de la prueba o al adjuntar a los registros.

i) Los permisos son fáciles de manejar y pueden configurarse en módulos, campos e incluso valores desplegables. Además, puede especificar solamente permisos de propietario.

j) Permite instalar scripts de prueba con acción y resultado esperado.

k) Cuenta con un corredor de prueba avanzado. Para cada ejecución de prueba, se crea un historial de ejecución que contiene un informe de prueba detallado. El historial de ejecución mantiene un registro de todas las ejecuciones de prueba históricas.

Selenium

Es una herramienta para la automatización de pruebas para aplicaciones Web.

Selenium es un conjunto de diferentes herramientas de software cada una con un enfoque diferente para apoyar la automatización de pruebas.

Todo el conjunto de herramientas resulta en un rico conjunto de funciones de pruebas específicamente orientadas a las necesidades de pruebas de aplicaciones web de todo tipo. Estas operaciones son altamente flexibles, permitiendo muchas opciones para localizar elementos de la interfaz de usuario y comparar los resultados de las pruebas esperadas con el comportamiento real de la aplicación.

Una de las características clave de Selenium es el soporte para ejecutar sus pruebas en múltiples plataformas de navegación.

La mayor fortaleza de Selenium es que existen muchas maneras de agregar funcionalidad a los scripts de prueba y al marco de Selenium para personalizar su automatización de prueba.

La automatización de pruebas permite:

- Prueba de regresión frecuente
- Retroalimentación rápida a los desarrolladores

- Iteraciones virtualmente ilimitadas de la ejecución de casos de prueba
- Apoyo a metodologías ágiles y de desarrollo extremo
- Documentación disciplinada de los casos de prueba
- Informes personalizados sobre defectos
- Detección de defectos perdidos por pruebas manuales

Selenium está compuesto por múltiples herramientas:

1. Selenium 2 (Selenium WebDriver)

   Incluye una API orientada a objetos. Es compatible con la API WebDriver y la tecnología Selenium 1(predecesora de Selenium 2, también llamada Selenium RC o Remote Control, ahora obsoleta).

2. Selenium IDE

   Es un complemento de Firefox que graba y reproduce las interacciones del usuario con el navegador. Este complemento permite crear scripts sencillos o ayudar en pruebas exploratorias. También puede exportar secuencias de comandos de control remoto o WebDriver, aunque tienden a ser algo frágiles y deben revisarse en algún tipo de estructura de objeto de página para cualquier tipo de resiliencia.

   Ideal para usuarios que no han experimentado con un lenguaje de programación o de secuencias de comandos.

3. Selenium-Grid

   Permite que la solución Selenium RC sea escalable para grandes suites de prueba y para aquellas que deben ejecutarse en múltiples entornos.

   Selenium Grid le permite ejecutar sus pruebas en paralelo, es decir, diferentes pruebas se pueden ejecutar al mismo tiempo en diferentes máquinas remotas. Esto tiene dos ventajas:

   - En primer lugar, si dispone de una suite de pruebas grande o una suite de pruebas de ejecución lenta, puede aumentar su rendimiento de forma sustancial utilizando Selenium Grid para dividir su suite de pruebas para ejecutar diferentes pruebas al mismo tiempo utilizando esas diferentes máquinas.

- Además, si debe ejecutar su suite de pruebas en varios entornos, puede tener distintas máquinas remotas soportando y ejecutando sus pruebas en ellas al mismo tiempo. En cada caso, Selenium Grid mejora en gran medida el tiempo que tarda en ejecutar su suite haciendo uso de procesamiento paralelo.

SoapUI

SoapUI es una herramienta de prueba para las APIs de SOAP y REST, se enfoca en asegurar la calidad al desarrollar APIs y Servicios Web.

SoapUI ofrece:

- Pruebas funcionales SOAP Web Services
- Pruebas funcionales REST API
- Cobertura WSDL
- Pruebas de aserción de mensajes
- Refactorización de pruebas

  Características:

  - Permite crear pruebas funcionales ad hoc
  - Pruebas de rendimiento: Prueba rápida de carga de API para principiantes
  - Pruebas de seguridad: Evalúa las vulnerabilidades del API y valida la seguridad
  - Simula el comportamiento de la API estática durante las pruebas

Open Load Tester

OpenLoad es una solución de optimización de rendimiento rápido basada en el navegador, para la carga y pruebas de tensión de sitios web dinámicos. Utiliza IBM WebSphere y DB2 Universal Database y está completamente integrado con IBM WebSphere Studio Application Developer.

Características:

- Reduce sustancialmente el tiempo y el conjunto de habilidades requeridas para optimizar el rendimiento de las aplicaciones basadas en Web.
- Simplifica el proceso de construcción de escenarios de usuarios reales.

- Verifica el comportamiento funcional esperado.
- Señala cuellos de botella de rendimiento dentro de las aplicaciones Web y la infraestructura de TI.

FunkLoad

FunkLoad es una herramienta de pruebas funcional y de carga en la web, escrito en Python.

Los principales casos de uso son:

- Pruebas funcionales de proyectos web y, por tanto, pruebas de regresión también.
- Pruebas de rendimiento: al cargar la aplicación web y supervisar sus servidores, le ayuda a identificar cuellos de botella, proporcionando un informe detallado de la medición del rendimiento.
- Herramienta de prueba de carga para exponer los errores que no surgen en pruebas superficiales, como pruebas de volumen o pruebas de longevidad.
- Herramienta de prueba de estrés para abrumar los recursos de la aplicación web y probar la recuperación de la aplicación.
- Escribir agentes web mediante scripting de cualquier tarea web repetitiva.

  Características:

  - Las pruebas funcionales son scripts puros de Python usando el framework pyUnit como prueba de unidad normal. Python permite escenarios complejos para manejar aplicaciones del mundo real.
  - Emula un navegador web (single-threaded) usando una webunit mejorada de Richard Jones
  - Corredor de pruebas avanzado con muchas opciones de línea de comandos
  - Convierte una prueba funcional en una prueba de carga: basta con invocar el corredor de banco para identificar problemas de escalabilidad y rendimiento. Si es necesario, el banco puede distribuirse en un grupo de máquinas de trabajo.
  - Informes de banco detallados en ReST, HTML, Org-mode, PDF (utilizando LaTeX / PDF Org-mode export)

- Reportes diferenciales para comparar 2 reportes de benchmark dando una rápida visión general de escalabilidad y cambios de velocidad.
- Informes de tendencias para ver la evolución del rendimiento.
- Fácil personalización de pruebas con un archivo de configuración o mediante línea de comandos.
- Fácil creación de pruebas usando TCPWatch incorporado como registrador de proxy, por lo que puede utilizar su navegador web y producir una prueba de FunkLoad automáticamente, incluida la carga de archivos o cualquier llamada ajax.
- Proporciona ayudantes de aserción web para verificar los resultados esperados en las respuestas.
- Proporciona ayudantes para recuperar contenido en la página de respuestas utilizando DOM.

FWPTT

Fwptt es una herramienta de testeo para pruebas de carga en aplicaciones Web. Puede registrar las solicitudes normales y ajax.

Características:

- Permite importar las sesiones de navegación recodificadas realizadas con Fiddler. Estas solicitudes http se pueden importar y utilizar para generar una clase c# que llame a toda la solicitud http que el usuario haya grabado previamente.
- Incorpora un corredor de pruebas mediante el cual el usuario será capaz de ejecutar el código de su clase de prueba c#.
- Es posible grabar las acciones de navegación usando cualquier navegador.
- La clase tiene funciones para manejar los parámetros de consulta / post que se puede utilizar para agregar o eliminar más fácilmente algunos de los ya existentes, además se pueden utilizar todas las clases .net para conectarse a una base de datos abierta.

LoadUI

Es una herramienta de testeo de pruebas de carga para API REST & SOAP. Trabaja de la mano con la herramienta de pruebas funcionales SoapUI. Ofrece una versión de prueba de 14 días.

Características:

- Permite ejecutar rápidamente pruebas de carga de la API, ya sea contra un punto final de servicio web único o basado en una prueba de API funcional existente creada en SoapUI.
- Prueba la velocidad y la escalabilidad de los nuevos cambios a una API
- Ofrece una vista previa de los comportamientos de rendimiento de la API antes de liberarlos a entornos de producción
- Permite lanzar simultáneamente múltiples cargas variables de tráfico en la API
- Graba el tráfico de dispositivos móviles y los reproduce a gran volumen
- Visualiza los efectos de una prueba de carga en su servidor y en los recursos de red
- Escala cualquier prueba de carga que construya indefinidamente
- Genera carga desde nodos basados en Windows o Linux
- Envía una mezcla de tráfico local o fuera de las instalaciones

La familia de estas herramientas SmartBear ofrece además otras herramientas de Prueba de Software:

- **TestComplete:**

  Herramienta para prueba automatizadas

- **TestLeft**

  Pruebas funcionales para desarrolladores

- **TestExecute**

  Escala las herramientas TestComplete o TestLeft

- **QAComplete**

  Administración de pruebas

- **LoadComplete**

  Pruebas de carga

- **CrossBrowserTesting**

  Plataforma para pruebas en la nube

Sahi Open Source

Sahi es una herramienta gratuita y de código abierto para la automatización de las pruebas de aplicaciones web. Sahi permite una fácil automatización incluso de complejas aplicaciones web 2.0 con gran cantidad de contenido AJAX.

Características:

- Permite realizar pruebas a través de varias combinaciones de navegador y Sistema Operativo.
- Ofrece las opciones de grabado, identificación inteligente de objetos, scripting simple, espera automática e informes incorporados.
- Informes de reproducción en HTML, JUnit Style.
- Trabaja en Internet Explorer, Firefox, Chrome, Safari, Opera, etc. en Windows, Mac y Linux.
- Funciona en cualquier navegador que soporte un proxy y ejecute Javascript, lo que significa que soporta todos los navegadores desde IE6.

2.2.13.2. Herramientas Comerciales

QA Complete

Esta herramienta ofrece al equipo de prueba una aplicación para administrar casos de prueba, entornos de prueba, pruebas automatizadas, defectos y tareas de proyectos de prueba.

QA Complete es una solución integral que ofrece visibilidad del proceso de gestión de pruebas y garantiza la entrega de software de alta calidad.

Características:

- Gestión de casos de prueba
- Manejo del entorno de prueba
- Gestión de defectos y problemas
- Integración de la automatización de pruebas
- Integración del tracker de errores
- Gestión de proyectos

- Documentos compartidos
- Gestión de requisitos
- Gestión de casos de prueba
- Manejo del entorno de prueba
- Gestión de defectos y problemas
- Integración de la automatización de pruebas
- Integración del tracker de errores
- Gestión de proyectos
- Documentos compartidos
- Gestión de requisitos

Microsoft TeamSystem

Son un conjunto de herramientas de prueba que se integran con Visual Studio. Visual Studio Team System Edition trabaja no sólo en su propio marco de pruebas, sino también dentro de un marco más amplio de herramientas de ciclo de vida del software.

Test Edition permite crear, administrar, editar y ejecutar pruebas, y también obtener y almacenar resultados de pruebas. Varios tipos de prueba, incluyendo unidades, Web, carga y pruebas manuales, están integrados en Visual Studio. Además, se incluye la medición de la cobertura del código.

Las pruebas se ejecutan mediante el IDE de Visual Studio. Además, puede ejecutar grupos de pruebas o cualquier prueba distinta del tipo de prueba manual desde una línea de comandos.

Debido a que las herramientas de prueba están integradas con las otras partes de Visual Studio Team System, puede publicar resultados en una base de datos, generar informes históricos, comparar diferentes tipos de datos y ver cuántos y qué errores se encontraron

Características:

- Se pueden publicar los resultados de pruebas y datos de cobertura de código en Team Foundation Server.

- Después de que los resultados de las pruebas se publiquen, pueden ser vistos y analizados por el resto de su equipo.
- Puede crear un error en un resultado de prueba.
- Puede descargar los resultados de las pruebas publicadas desde una compilación.
- Después de haber creado una lista de pruebas, puede volver a utilizarla cuando una persona que trabaja en el laboratorio de compilación utilizando Team Foundation Build crea un tipo de compilación, como el proceso de ejecución de las pruebas de verificación de compilación.
- Se puede asociar una lista de pruebas con una política de registro.
- Un director de proyecto puede especificar que, antes de que se verifique el código en particular, debe probarse mediante un conjunto específico de pruebas..

IBM Rational

Es un software de pruebas funcionales y de regresión automatizadas. Este software proporciona funciones de pruebas automatizadas para pruebas funcionales, de regresión, de GUI y basadas en los datos. Rational Function Tester da soporte a diversas aplicaciones, como: web, .Net, Java, Siebel, SAP, basadas en emulador de terminal, PowerBuilder, Ajax, Adobe Flex, Dojo Toolkit, GEF, documentos Adobe PDF, zSeries, iSeries y pSeries.

Características:

- Prueba de guión gráfico: simplifica la visualización y la edición de pruebas mediante el lenguaje natural y las capturas de pantalla.
- Pruebas automatizadas: permite a los probadores automatizar pruebas de forma flexible cuando se realizan cambios frecuentes en la interfaz de usuario de las aplicaciones con la tecnología ScriptAssure.
- Pruebas basadas en datos: le permite realizar las mismas series de acciones de pruebas con un conjunto variado de datos de pruebas. Script de pruebas: combina un grabador de acciones de usuario con varias opciones de personalización y funciones de mantenimiento de scripts inteligentes.

- Integraciones: se integra con IBM Rational Team Concert e IBM Rational Quality Manager para proporcionar acceso a elementos de trabajo y a soporte de activos de pruebas de la SCM lógicas o compuestas.

Sahi Pro

Sahi es un conjunto de herramientas para pruebas de automatización para aplicaciones web y de escritorio. Ofrece una automatización rápida y confiable. Sahi está disponible como un producto libre, de código abierto y como una versión comercial "Sahi Pro".

Sahi Pro es adecuado para pruebas cross-browser/multi-browser de aplicaciones web 2.0 complejas con contenido de AJAX y contenido dinámico. Sahi corre sobre cualquier navegador moderno que admita javascript.

Sahi funciona bien en entornos de desarrollo Ágiles, lo que permite una rápida automatización y mantenimiento con una fácil integración de la versión Sistemas. Sahi ahorra tiempo y esfuerzo con un desarrollo más rápido, menos mantenimiento y una rápida reproducción distribuida.

Por defecto, Sahi Pro soporta la aplicación web y la automatización REST API.

Características:

- Excelente grabador y objeto espía que funciona en Internet Explorer, Firefox, Chrome, Safari, Opera.
- Reproducción en cualquier navegador de escritorio e incluso en navegadores móviles.
- Súper simple y robusto mecanismo de identificación de objetos que funciona a través de los navegadores.
  - No utiliza selectores XPaths o css. Tiene sus propias envolturas alrededor del Javascript DOM.
  - Funciona incluso cuando los elementos no tienen ids.
  - Recorre automáticamente los marcos y los iframes.
- Espera automáticamente las cargas de página y la actividad de ajax. No hay necesidad de agregar declaraciones de espera en 95 % de casos. Esto reduce el tamaño del código de base en un 50 % en comparación con otras herramientas y hace que los scripts sean robustos y fácilmente mantenibles.
- No necesita el navegador para estar en foco.

- Puede reproducir múltiples secuencias simultáneamente reduciendo el tiempo de reproducción.
- Crea automáticamente informes enriquecidos sin agregar ningún código adicional. Esto mantiene el script simple sin ningún tipo de desorden.
- Admite etiquetas personalizadas HTML5 y DOM de sombra.
- Tiene una interfaz de usuario fácil para crear y probar las API de REST.
- Puede medir la cobertura de código JavaScript en sus aplicaciones web.
- Sahi Pro Desktop Add-on admite la automatización de aplicaciones de escritorio de Windows y aplicaciones Java desde el mismo script.

Parasoft

Los productos de Parasoft se enfocan en entregar software de alta calidad de manera eficiente y consistente.La familia de soluciones de calidad de software de Parasoft ofrece capacidades de prueba de extremo a extremo que abarcan pruebas funcionales, gestión de laboratorio de pruebas, pruebas de desarrollo y prevención automatizada de defectos.

Objetivos:

- Simular ambientes de prueba realistas y completos bajo demanda
- Garantizar la integridad de las transacciones API
- Aplicar las actividades de prevención de defectos de forma consistente y continua

Está formado por las siguientes herramientas:

1. **Parasoft Virtualize**

   Crear, implementar y administrar entornos simulados de dev / test para permitir el acceso bajo demanda a entornos completos.

2. **SOAtest**

   Automatiza pruebas completas de extremo a extremo para transacciones comerciales y de seguridad crítica. Permite probar a fondo las aplicaciones compuestas con un soporte robusto para REST y servicios web.

3. **Plataforma de pruebas de desarrollo**

   Permite la Prueba Continua. Aprovechando las políticas, aplica sistemáticamente las prácticas de calidad de software a través de los equipos y a lo largo del SDLC. Ofrece una plataforma para la prevención de defectos automatizados y la medición uniforme de riesgo.

4. **C/C++test**

   Es una solución de pruebas de desarrollo integrada para automatizar una amplia gama de pruebas, utiliza prácticas que mejoran la productividad del equipo de desarrollo y la calidad del software.

5. **Jtest**

   Jtest se integra perfectamente con Parasoft SOAtest, que permite la comprobación funcional y de carga de extremo a extremo para aplicaciones y transacciones complejas distribuidas .

6. **dotTEST**

   DotTEST se integra perfectamente con Parasoft SOAtest, que permite realizar pruebas completas de carga y carga para aplicaciones y transacciones complejas distribuidas.

7. **Insure++**

   Identifica los errores de programación y de acceso a la memoria difíciles de realizar, como corrupción de memoria, pérdidas de memoria, acceso fuera de los límites de la matriz, punteros no válidos y más para garantizar la integridad del uso de memoria de las aplicaciones.

Test Complete

Es una de las herramientas de prueba de la familia SmartBear. Automatiza la interfaz de usuario. Utiliza un algoritmo de identificación de objetos de la clase de TestComplete para construir pruebas de interfaz de usuario escalables y estables. TestComplete tiene una arquitectura abierta y flexible que hace que la creación, mantenimiento y ejecución de pruebas automatizadas en aplicaciones de escritorio, web y móviles sean fáciles, rápidas y rentables.

Características:

- Soporte para múltiples lenguajes de secuencias de comandos

- La capacidad de registrar pruebas robustas automatizadas sin conocimientos de secuencias de comandos
- Pruebas de regresión que no fallan cuando la interfaz de usuario cambia
- Pruebas basadas en datos
- Plugins y extensiones personalizadas

NeoLoad

NeoLoad es una herramienta de prueba de carga y rendimiento que simula de forma realista la actividad del usuario y supervisa el comportamiento de la infraestructura para eliminar cuellos de botella en aplicaciones web y móviles.

Características:

- NeoLoad soporta las últimas tecnologías web como HTML5, Push, WebSocket, AngularJS, Oracle Forms y muchos más.
- Los bucles, las condiciones y otros controles de "arrastrar y soltar" simplifican la creación del diseño de prueba, y Javascript se puede usar en casos extremadamente avanzados.
- Permite definir fácilmente las reglas avanzadas de extracción de datos y las valida con el contenido grabado antes de ejecutar la prueba.
- Extracción inteligente de datos y enlaces dinámicos desde páginas para la variabilización en el escenario de pruebas de rendimiento, sin necesidad de scripts:
  - Reglas predefinidas para los marcos más comunes: .Net, JSF, Oracle E-Business, SAP Web y más
  - Detección automática y manejo de parámetros específicos de la aplicación
- Comprueba rápidamente una ruta de acceso de usuario con una nueva versión de la aplicación bajo prueba para identificar los cambios que pueden causar errores.
- Permite grabar cualquier aplicación móvil (navegador nativo, híbrido o móvil) directamente desde cualquier dispositivo móvil con modo proxy o modo de túnel DNS.

WebLOAD Professional

Características:

- Prueba de creación: Permite construir escenarios de prueba de carga más fácil y eficientemente con WebLOAD.
- Correlación: El motor de correlación de WebLOAD identifica y reemplaza valores dinámicos que son únicos para cada ejecución del script, como ID de sesión, hora y muchos otros, y los reemplaza automáticamente.
- Ejecución de pruebas: Crear escenarios de carga realistas.Ofrece la posibilidad de simular condiciones de carga realistas y variadas para un número ilimitado de usuarios. Puede definir una variedad de scripts, navegadores y características de red en una sola ejecución de prueba. Mediante el programador, puede controlar la acumulación de carga de diferentes maneras que imitan las condiciones de la vida real, incluidas las lineales, los pasos y los incrementos. Puede cambiar el tamaño de carga durante la ejecución con un control intuitivo del acelerador, detener el análisis y reiniciar en cualquier momento.
- Analítica: Ayuda a identificar cuellos de botella de rendimiento en su sistema con más de 80 informes y gráficos que le permiten profundizar en cualquier ángulo y nivel granular para identificar problemas.
- Panel de control WebLOAD: Permite ver y analizar sesiones de carga desde cualquier navegador web mientras se trabaja con varios miembros del equipo y comparte datos.
- Rendimiento del lado del servidor: recopila datos de rendimiento del servidor de los sistemas operativos, servidores web, servidores de aplicaciones y servidores de bases de datos para ayudarle a identificar la causa raíz de los problemas.
- JavaScript: Usa JavaScript como su lenguaje de script de escenarios de prueba de carga, que proporciona un entorno estándar accesible para extender las pruebas de carga.
- Prueba de carga móvil: Puede crear scripts de prueba para dispositivos móviles y proporciona varias maneras de cargarlas al sistema de prueba con el uso del dispositivo móvil.

- Pruebas de rendimiento en la nube: Permite ejecutar pruebas de rendimiento desde la nube generando carga de Amazon EC2. Con cero instalación o configuración, puede generar carga desde múltiples ubicaciones en el mundo, distribuir carga entre las máquinas en la nube y en las premisas.

Ofrece las siguientes herramientas:

- Website Load Testing
- Java Load Testing
- .NET Load Testing
- Oracle Forms
- Web Services And APIs
- Selenium
- Continuous Integration (CI)
- CRM And ERP
- Ellucian

Load Impact

Es una herramienta de pruebas de carga

Características:

- Simula el comportamiento de usuarios y el tráfico exactamente cómo sucedería en la vida real.
- Generación simultánea de carga geográfica: En una sola prueba, genera carga de hasta 10 ubicaciones diferentes simultáneamente y es posible añadir aún más ubicaciones a petición.
- Secuencias de comandos automático y avanzado: Escanea escenarios de usuario en Lua, usando el IDE con terminación de código, JSON y análisis XML, parametrización de datos. Analiza una página web y genera un script de forma automática - no requiere programación.
- Graba una sesión HTTP utilizando el registrador de proxy o nuestra extensión de Chrome y permite que los usuarios simulados realicen las mismas acciones durante la prueba.

- Localiza los problemas de rendimiento, supervisan los respaldos mientras prueba con los agentes de servidor New Relic y/o Load Impact. Recopila métricas como:
  - Uso de CPU
  - Uso de memoria
  - APDEX
  - E/S de disco
  - E/S de red
- Permite configurar las pruebas de carga para que se ejecuten en la mitad de la noche o una vez a la semana. También permite incluir Load Impact como parte del proceso de entrega continua con el uso de los complementos Jenkins y TeamCity, así como la API abierta y varios SDK.
- Permite hacer pruebas basadas en datos. Simular usuarios reales incluyendo varias fuentes de datos parametrizados en guiones de escenario. Simplemente se cargan archivos CSV que contengan los datos que se desee utilizar, como credenciales de inicio de sesión, ID de producto, URL, etc., y se asocian con el script deseado.
- Emite una notificación cuando se hayan completado las pruebas importantes. Es un webhook simple, por lo que puede obtener la notificación en el medio que prefiera - Slack, Hipchat, correo electrónico.

2.3. Propuesta de Solución

Se propone la realización de un análisis y presentación de métodos, técnicas y herramientas existentes de verificación y validación del software apropiado para la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato, para mejorar la calidad en el proceso de desarrollo de software.

CAPÍTULO 3

Metodología

3.1. Modalidad Básica de la Investigación

El presente trabajo tiene las siguientes modalidades:

Modalidad de Campo: Se considera esta modalidad ya que el investigador acudirá al lugar en donde se producen los hechos para obtener información relacionada con los objetivos del trabajo de grado. Las técnicas a ser utilizadas serán: entrevistas, encuesta y la observación.

Modalidad Bibliográfica o Documentada: Se considera esta modalidad ya que se recurre a diferentes fuentes obtenidas de libros, artículos científicos, tesis desarrolladas en Universidades para profundizar enfoques con respecto al tema de la investigación.

Modalidad aplicada: Por la utilización de los conocimientos adquiridos a lo largo de la carrera universitaria.

3.2. Población y Muestra

La presente investigación por su característica no requiere población ni muestra.

3.3. Recolección de la Información

Se recolectará la información realizando entrevistas al personal encargado de desarrollo de software en la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato, así como también se buscará en Internet información virtual utilizando documentos técnicos, tesis, libros, todo esto para alcanzar los objetivos planteados.

También se realizará la recolección de información mediante observación en el departamento respectivo.

3.4. Procesamiento y Análisis de Datos

Los datos se los recogerán y se seleccionarán de acuerdo al área donde se maneje información requerida de la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato, y se pasará a tabular a fin de obtener resultados útiles para la verificación de los métodos, técnicas y herramientas utilizados.

3.5. Desarrollo del Proyecto

Para el desarrollo del proyecto se procederá de la siguiente forma:

- Identificación de las características principales de Verificación y Validación.
- Análisis del proceso de verificación al software que se aplica en la Dirección de Tecnología de Información y Comunicación.
- Identificación de las etapas del proceso de desarrollo.
- Identificación de las etapas del proceso de desarrollo de software en las que se aplican métodos, técnicas y herramientas para la verificación y validación de Software.
- Identificación de los factores que inciden en el proceso de verificación del software.
- Identificación de los factores que inciden en el proceso de validación del software.
- Análisis de métodos, técnicas y herramientas de verificación y validación de software.
- Propuesta de métodos, técnicas y herramientas de verificación y validación apropiadas.

CAPÍTULO 4

Desarrollo de la propuesta

4.1. Recolección de Información

Para la realización del estudio se empleó una encuesta para conseguir la información necesaria, ya que esta técnica es muy práctica la cual consta de una serie de preguntas que permiten reunir datos de modo rápido y eficaz. La encuesta se la realizó de manera online; las razones por las que se aplicó de esta forma fueron:

- Aprovechar la aplicación Forms que viene incluida en el paquete de office365 de la Institución.
- Asegurar la recopilación de la información de manera rápida.
- Facilitar la interpretación de los datos ya que se muestran de manera organizada.

4.2. Análisis y Presentación de Resultados

La encuesta se realizó al personal del área de desarrollo de software de la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato con el propósito de identificar cuales son los Métodos, Técnicas y Herramientas de Verificación y Validación de Software aplicados en la Dirección de Tecnología de Información y Comunicación, obteniendo los siguientes resultados:

Pregunta1:

¿Utiliza usted técnicas de Verificación y Validación de software para realizar un proyecto?

Opción	Total	Porcentaje
Si	4	66.67 %
No	2	33.32 %

Tabla 4.1: Resultados Pregunta 1

Elaborado por: Daniel Jerez

Del total de las personas del área de desarrollo de la DITIC encuestadas, el 66.67 % afirmó que utiliza técnicas de Verificación y Validación, mientras que un 33.32 % respondieron que no utilizan técnicas de Verificación y Validación para realizar un proyecto de desarrollo de software. A continuación se muestra la gráfica de esta pregunta:

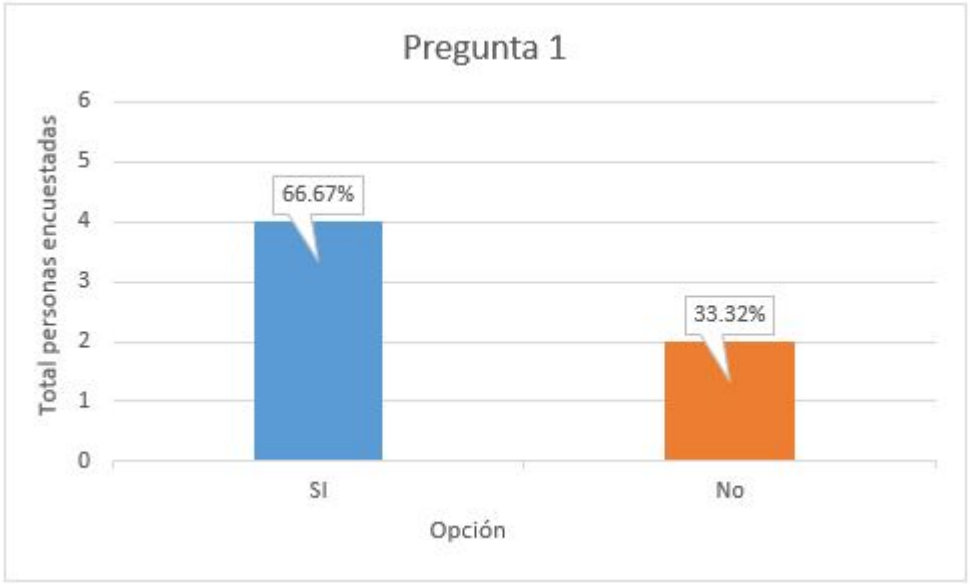


Figura 4.1: Gráfico Pregunta 1
Elaborado por: Daniel Jerez

Interpretación:

Un 33.32 % no utiliza técnicas de Verificación y Validación de software para realizar un proyecto de desarrollo lo que refleja que este pequeño porcentaje desconoce o no aplica dichas técnicas, además de eso se comentó que en el área de desarrollo a pesar de que no todos los miembros utilizan técnicas de Verificación y Validación y que no usaban una o más técnicas establecidas, estaban considerando

establecer en los próximos meses el uso de una o varias técnicas que se adapten a su entorno.

Pregunta 2:

Seleccione el tipo de Técnicas que utiliza

- Análisis y Pruebas Funcionales
- Análisis y Pruebas Estructurales
- Estrategias de Integración
- Testing and Análisis
- Error-Oriented
- Hybrid Approaches
- Análisis de Flujo de Transacción
- Análisis de Algoritmos
- Análisis de Concurrencia
- Análisis de Cobertura
- Análisis de Stress
- Análisis de Falla
- Análisis de Flujo de Datos
- Model Checking
- Pruebas de Desempeño
- Inspecciones
- Otro
- Ninguna

Del total de las personas del área de desarrollo de la DITIC encuestadas, el 25 % respondió Análisis y Pruebas Funcionales, el 12.5 % respondió Estrategias de Integración, las técnicas Análisis y Pruebas Estructurales, Testing and Análisis, Error-Oriented, Análisis de Flujo de Transacción, Análisis de Algoritmos, Análisis

de Concurrencia, Análisis de Falla y Análisis de Flujo de Datos fueron seleccionadas por un 6,3 % de los encuestados, mientras que un 12.5 % seleccionó que no utiliza ninguna, el resto de técnicas obtuvieron un valor de 0 %. A continuación se muestra la gráfica de esta pregunta:

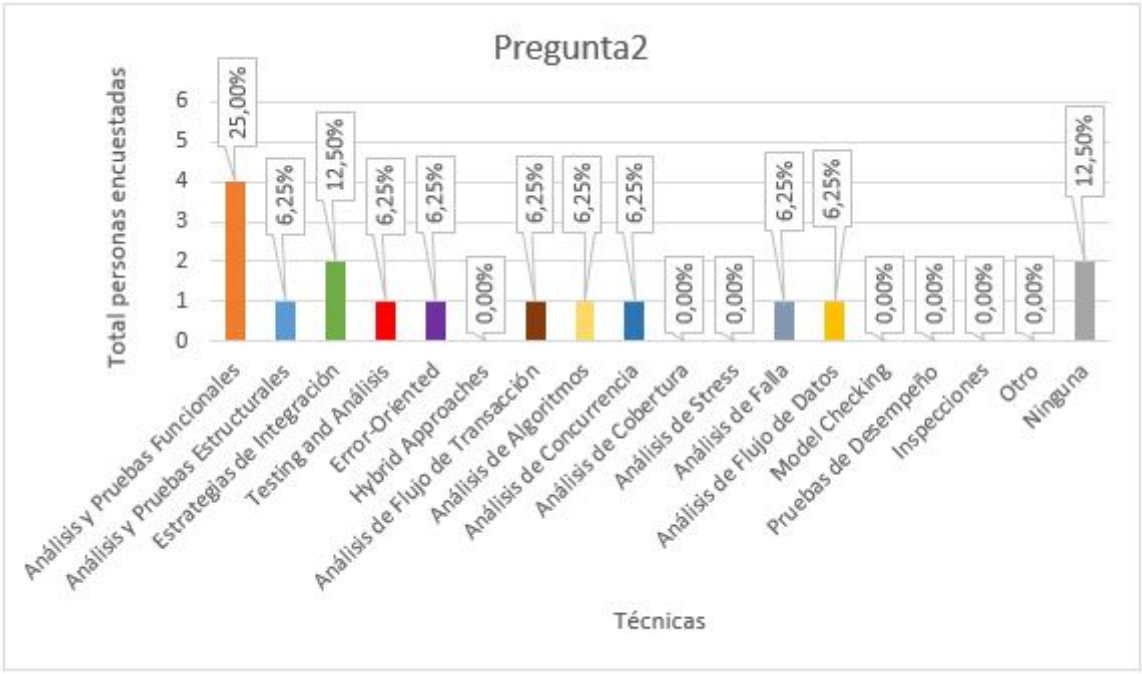


Figura 4.2: Gráfico Pregunta 2
Elaborado por: Daniel Jerez

Interpretación:

Con estos resultados se puede constatar que la mayoría utiliza sea una o varias técnicas de V&V de software, pero cabe mencionar que aún existe un porcentaje del 12.50 % que no hace uso de ninguna de ellas lo que corrobora la interpretación de la pregunta anterior, en la que se menciona que un pequeño porcentaje desconoce o no aplica dichas técnicas.

Pregunta 3:

¿Utiliza usted métodos de Verificación y Validación de software para realizar un proyecto?

Opción	Total	Porcentaje
Si	3	50 %
No	3	50 %

Tabla 4.2: Resultados Pregunta 3

Elaborado por: Daniel Jerez

Del total de las personas del área de desarrollo de la DITIC encuestadas, el 50 % afirmo que utiliza métodos de Verificación y Validación, mientras que un 50 % respondieron que no utilizan métodos de Verificación y Validación para realizar un proyecto de desarrollo de software. Se comentó que en el área a pesar de que no todos los miembros utilizan métodos de Verificación y Validación y que no usaban uno o más métodos establecidos, estaban considerando establecer en los próximos meses el uso de uno o varios métodos que se adapten a su entorno, ya que conocen de la importancia de las mismos. A continuación, se muestra la gráfica de esta pregunta:

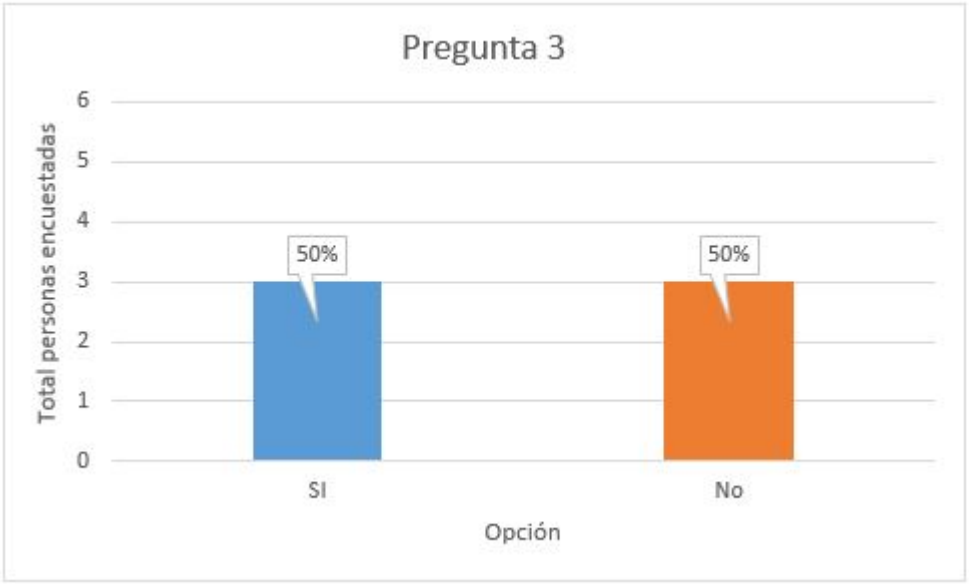


Figura 4.3: Gráfico Pregunta 3
Elaborado por: Daniel Jerez

Interpretación:

Al ser importante la utilización de métodos en la realización de un proyecto de desarrollo de software, se considera que un 50 % que no utiliza métodos es un

porcentaje alto, lo que puede verse reflejado en no garantizar la calidad del producto final.

Pregunta 4:

Seleccione el tipo de método que utiliza:

- Caja Negra
- Caja Blanca
- Top-Down
- Act-like-a-customer
- ATAM
- ADR
- ARID
- Otro
- Ninguna

Del total de las personas del área de desarrollo de la DITIC encuestadas, el 57.14 % respondió Caja Negra, las opciones Caja Blanca, Otro y Ninguna obtuvieron un 14.29 % respectivamente, mientras que el resto de métodos obtuvieron un valor de 0 %. A continuación, se muestra la gráfica de esta pregunta:

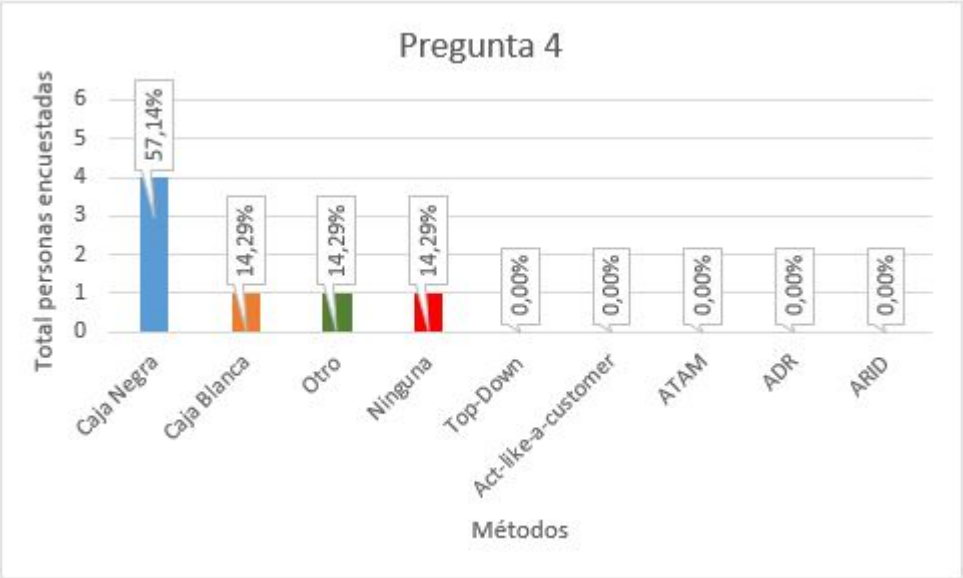


Figura 4.4: Gráfico Pregunta 4
Elaborado por: Daniel Jerez

Interpretación:

Con estos resultados se puede constatar que aunque hay un porcentaje elevado que selecciono el método de caja negra, caja blanca u otros aún existe un 14,29 % que es un valor representativo que no utiliza dichos métodos.

Pregunta 5:

¿Utiliza usted herramientas de Verificación y Validación de software para realizar un proyecto?

Opción	Total	Porcentaje
Si	0	0 %
No	6	100 %

Tabla 4.3: Resultados Pregunta 5

Elaborado por: Daniel Jerez

Del total de las personas del área de desarrollo de la DITIC encuestadas, el 100 % afirmo que no utiliza herramientas de Verificación y Validación, para realizar un proyecto de desarrollo de software. A continuación, se muestra la gráfica de interpretación de esta pregunta:

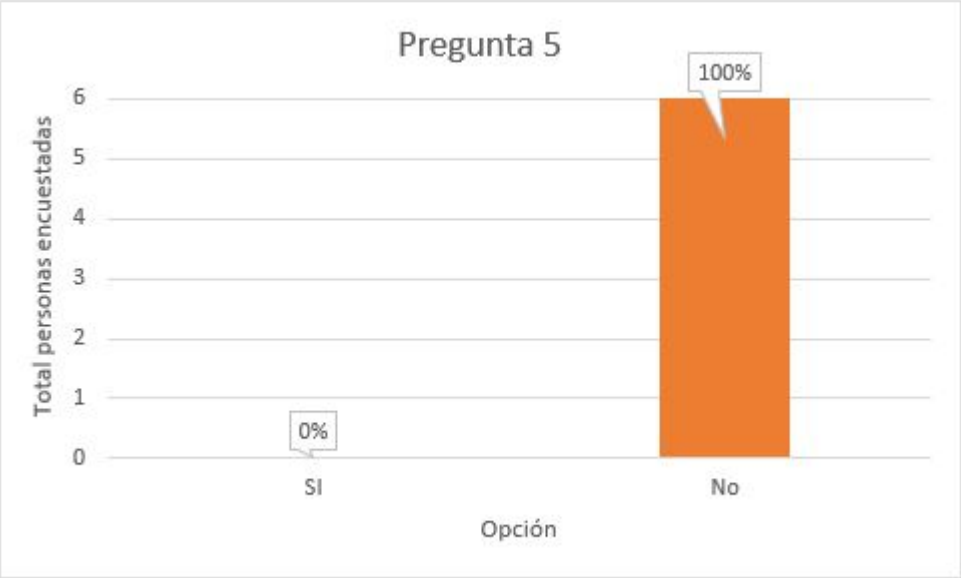


Figura 4.5: Gráfico Pregunta 5
Elaborado por: Daniel Jerez

Interpretación:

El resultado refleja que el 100 % de los encuestados no utiliza herramientas de V&V de software con esto se puede constatar que desconocen del uso y la importancia de las mismas razones como estas son las que motivan realizar la presente investigación.

Pregunta 6:

¿Utiliza usted alguna metodología para el proceso de desarrollo de software?

Opción	Total	Porcentaje
Si	5	83.33 %
No	1	16.67 %

Tabla 4.4: Resultados Pregunta 6

Elaborado por: Daniel Jerez

Del total de las personas del área de desarrollo de la DITIC encuestadas, el 83.33 % afirmo que utiliza alguna metodología, mientras que un 16.67 % respondieron que no utilizan alguna metodología para el proceso de desarrollo de software. A continuación, se muestra la gráfica de interpretación de esta pregunta:

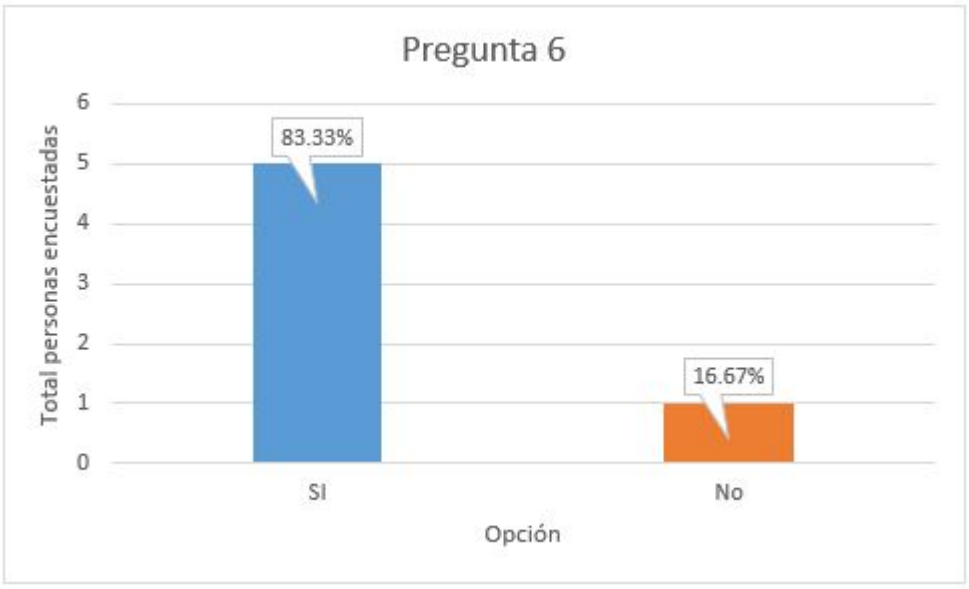


Figura 4.6: Gráfico Pregunta 6
Elaborado por: Daniel Jerez

Interpretación:

Escasamente un 16.67 % de los encuestados no utiliza metodologías en el proceso de desarrollo lo que refleja que un gran porcentaje hace uso de las metodologías existentes para realizar un proyecto de desarrollo de software.

Pregunta 7:

¿Qué metodología utiliza en el proceso de desarrollo de software, en caso de utilizarla?

Respuesta	Analista Programador	Porcentaje
Cascada	3	50 %
Xp	0	0 %
Scrum	1	16.67 %
RAD	1	16.67 %
Ninguna	1	16.67 %

Tabla 4.5: Resultados Pregunta 7

Elaborado por: Daniel Jerez

Del total de las personas del área de desarrollo de la DITIC encuestadas, el 50 % respondió Cascada, el 16.67 % respondió Scrum, RAD y Ninguna, mientras que nadie utiliza metodología XP, A continuación, se muestra la gráfica de interpretación de esta pregunta:

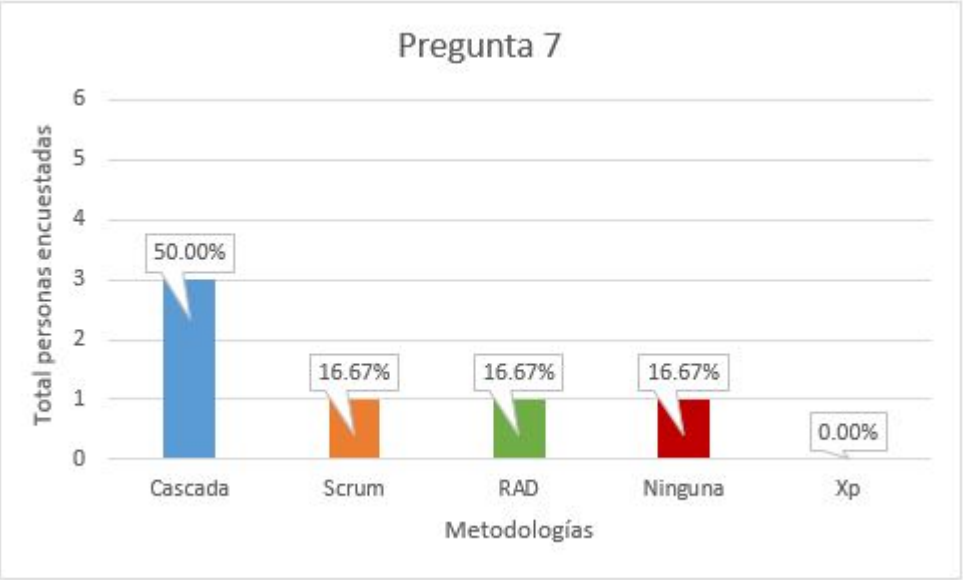


Figura 4.7: Gráfico Pregunta 7
Elaborado por: Daniel Jerez

Interpretación:

Con estos resultados se puede observar que las metodologías más utilizadas son Cascada, Scrum y RAD y que un porcentaje mínimo no utiliza ninguna metodología.

Pregunta 8:

¿Con qué frecuencia se realizan las reuniones con los actores que intervienen en un sistema a desarrollar en la DITIC, es decir son permanentes, una al inicio, a la mitad y final del proceso de desarrollo, o una al inicio y al final?

Respuesta	Analista Programador	Porcentaje
Inicio	1	16.67 %
Inicio y Final	2	33.33 %
Inicio, Mitad, Final	1	16.67 %
Permamentes	2	33.33 %

Tabla 4.6: Resultados Pregunta 8

Elaborado por: Daniel Jerez

Del total de las personas del área de desarrollo de la DITIC encuestadas, el 33.33 % respondió Permanentes, Inicio y Final, mientas que el 16.67 % respondió Inicio e Inicio, Mitad, Final, del proceso de desarrollo de un proyecto de software, A continuación, se muestra la gráfica de interpretación de esta pregunta:

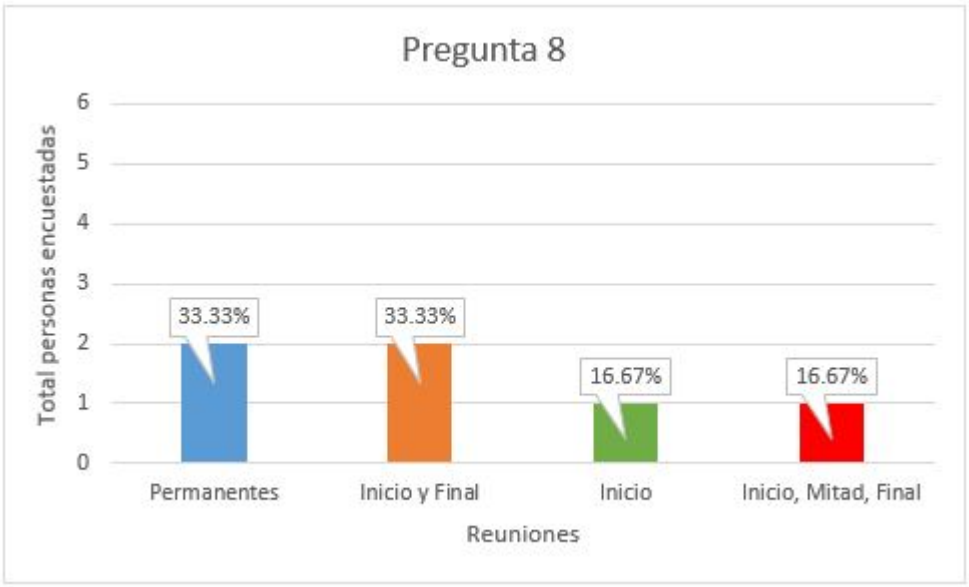


Figura 4.8: Gráfico Pregunta 8
Elaborado por: Daniel Jerez

Interpretación:

Con estos resultados podemos observar que existen varias formas de realizar las reuniones con los actores que intervienen en un sistema a desarrollarse en la DITIC lo que refleja que no existe similitud en la frecuencia con las que dichas reuniones se realizan.

Pregunta 9:

¿Cuál de las siguientes técnicas de levantamiento de la información utiliza usted para realizar un proyecto de desarrollo de software?

Respuesta	Analista Programador	Porcentaje
Reuniones de Trabajo (focus group)	6	100 %
Entrevista	0	0 %
Campo	0	0 %
Encuesta	0	0 %
Mesa de Trabajo	0	0 %

Tabla 4.7: Resultados Pregunta 9

Elaborado por: Daniel Jerez

Del total de las personas del área de desarrollo de la DITIC encuestadas, el 100 % afirmo que utiliza Reuniones de Trabajo (focus group), para el levantamiento de la

información para realizar un proyecto de desarrollo de software. A continuación, se muestra la gráfica de interpretación de esta pregunta:

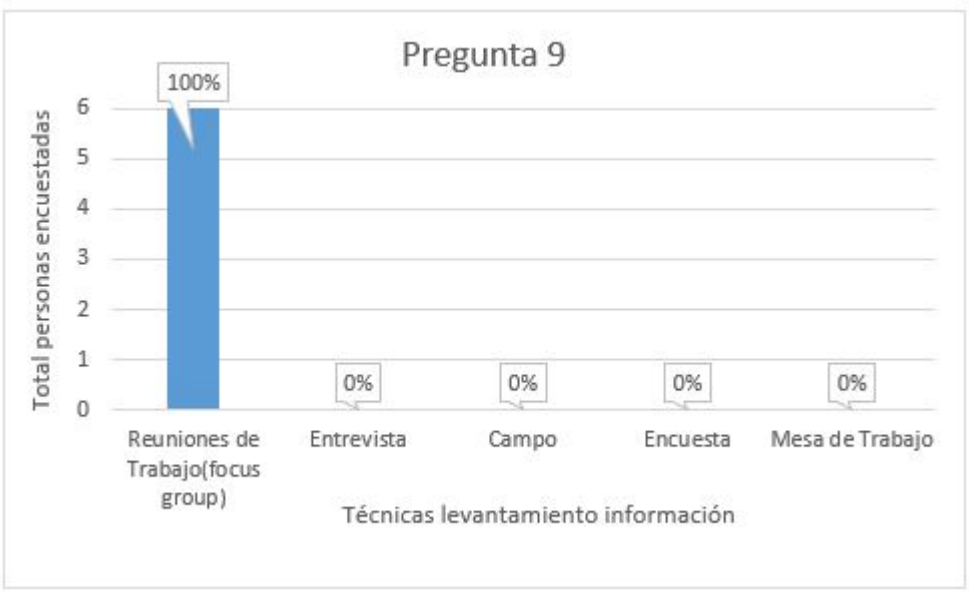


Figura 4.9: Gráfico Pregunta 9
Elaborado por: Daniel Jerez

Interpretación:

El 100 % de los encuestados utiliza focus group como técnica de levantamiento de requerimientos para realizar un proyecto de desarrollo de software pues ellos consideran que esta técnica se ajusta a sus necesidades.

Pregunta 10:

¿A su criterio, la utilización de técnicas, métodos y herramientas permiten una optimización en el tiempo planificado del proyecto?

Respuesta	Analista Programador	Porcentaje
Si	5	83.33 %
No	1	16.67 %

Tabla 4.8: Resultados Pregunta 10

Elaborado por: Daniel Jerez

Del total de las personas del área de desarrollo de la DITIC encuestadas, el 83.33 % afirmo que la utilización de técnicas, métodos y herramientas permiten una optimización en el tiempo planificado del proyecto, mientras que un 16.67 %

respondieron que no. A continuación, se muestra la gráfica de interpretación de esta pregunta:

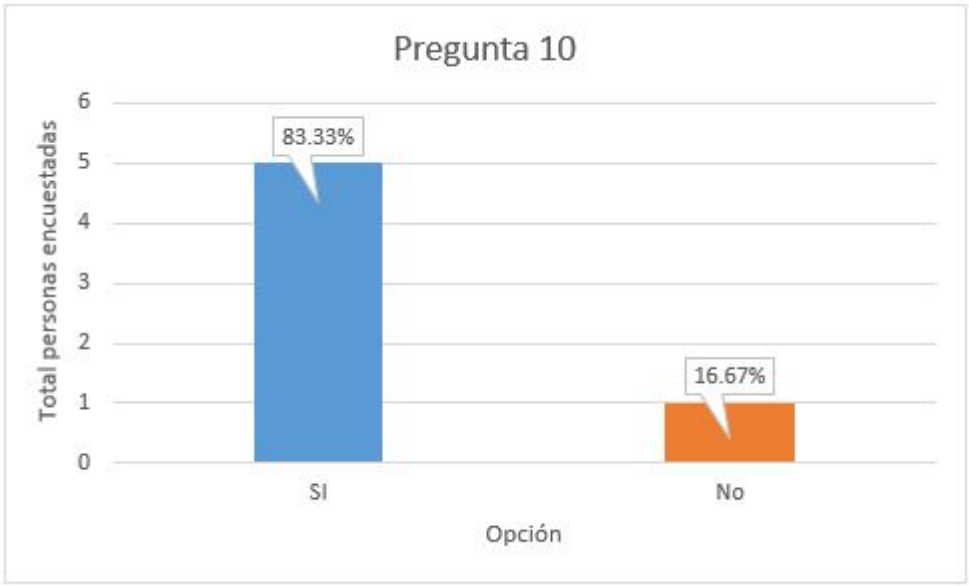


Figura 4.10: Gráfico Pregunta 10
Elaborado por: Daniel Jerez

Interpretación:

Apenas un 16.67 % de los encuestados no considera que la utilización de técnicas, métodos y herramientas permiten una optimización en el tiempo planificado de un proyecto de desarrollo de software lo que refleja que la gran mayoría está consciente de la importancia que están aportan al tiempo planificado de un proyecto.

Pregunta 11:

¿Cree usted que la utilización de técnicas, métodos y herramientas de verificación y validación de software aportan en la Calidad del producto final?

Respuesta	Analista Programador	Porcentaje
Si	5	83.33 %
No	1	16.67 %

Tabla 4.9: Resultados Pregunta 11

Elaborado por: Daniel Jerez

Del total de las personas del área de desarrollo de la DITIC encuestadas, el 83.33 % afirmo que la utilización de técnicas, métodos y herramientas de verificación

y validación de software aportan en la Calidad del producto final, mientras que un 16.67 % respondieron que no. A continuación, se muestra la gráfica de interpretación de esta pregunta:

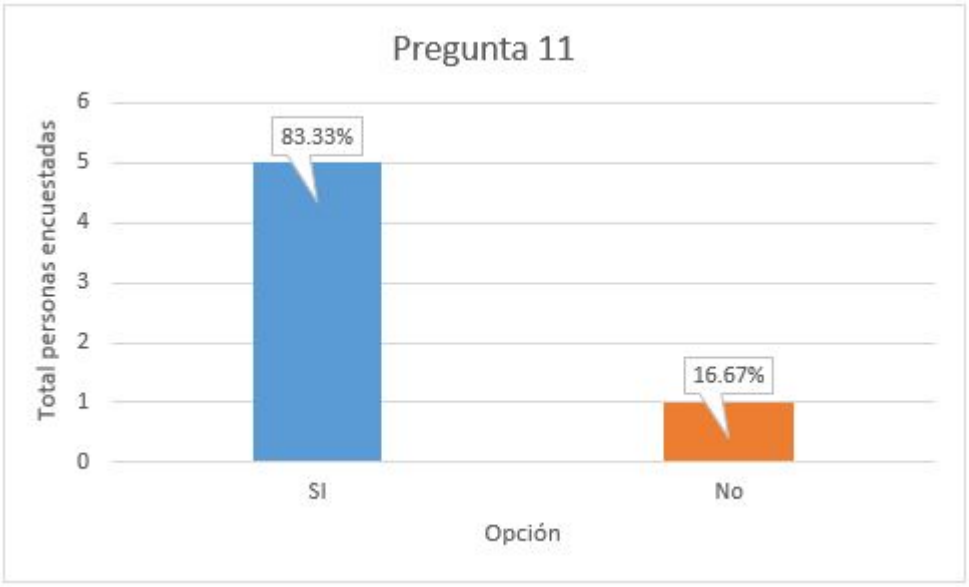


Figura 4.11: Gráfico Pregunta 11
Elaborado por: Daniel Jerez

Interpretación:

Como se puede apreciar en la gráfica la mayoría está consciente que la utilización de técnicas, métodos y herramientas de verificación y validación de software aportan en la Calidad del producto final, en contraste el 16.67 % restante considera lo contrario.

4.3. Interpretación de Resultados

- En la Universidad Técnica de Ambato en el departamento de Tecnología de Información y Comunicación existe un 33.32 % de las personas encuestadas que no utiliza técnicas, un 50 % que no utiliza métodos y un 100 % que no utiliza herramientas de V&V de software, esto puede verse reflejado en no asegurar que un producto software cumpla las especificaciones inicialmente planteadas, aumentar riesgos y desviaciones en tiempos establecidos, por lo que se sugiere utilizar técnicas, métodos y herramientas de V&V de software en todos proyectos que se realizan en la dirección, para mejorar la calidad de los productos.

- En el departamento encargado del desarrollo de software de la Universidad

Técnica de Ambato del grupo de personas encuestadas que utiliza técnicas de V&V de software, un 25 % utiliza el Análisis y Pruebas Funcionales, un 12.50 % utiliza Estrategias de Integración y un 6.25 % utiliza otro tipo de técnica, esto quiere decir que el software es evaluado de diferente manera, en algunos casos a los sistemas solamente se les realizan pruebas de entradas y salidas de datos, en otros solamente la técnica Estrategias de Integración; se sugiere realizar una estandarización, política en los tipos de técnicas a ser aplicadas para que se ajusten a sus necesidades.

- Por otra parte, un gran porcentaje hace uso de las metodologías existentes para realizar un proyecto de desarrollo de software, siendo las metodologías más utilizadas Cascada, Scrum y RAD. Razón por la cual se tiene problemas en aplicar uno u otro método de v&v ya que seria recomendable que se aplique una política para el uso de una sola metodología en el departamento, la cual se ajuste a sus necesidades.

- La frecuencia con la que se realizan las reuniones con los actores que intervienen en un sistema a desarrollarse en la DITIC no tienen similitud, existen varias formas de llevarlas a cabo, esto aumenta el riesgo de presentar inconvenientes al momento de la entrega final del producto, se sugiere realizar reuniones permanentes, ya que estas reuniones permiten evaluar avances y encontrar problemas que pueden ser corregidos en cada encuentro.

- La principal técnica de levantamiento de requerimientos para realizar un proyecto de desarrollo de software es focus group, pues se considera que esta técnica se ajusta a sus necesidades.

- El 83.33 % de los miembros del área de desarrollo están conscientes que la utilización de técnicas, métodos y herramientas aportan positivamente al tiempo planificado de un proyecto, de igual manera a la calidad del producto final.

4.4. Situación Actual

La Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato es la encargada de gestionar las Tecnologías de la Información y Comunicaciones de la Institución, mediante su desarrollo, administración y mantenimiento. Ver Anexo Reglamento DITIC.

4.4.1. Gestión de Desarrollo de Software en la Universidad Técnica de Ambato

De acuerdo al reglamento de la Dirección de Tecnología de Información y Comunicación aprobado el 21 de junio de 2016, la gestión del desarrollo está a cargo de un Especialista de Tecnología de Información y Comunicación, el mismo que será responsable de proponer proyectos para implementar o mejorar los sistemas de información. Además cuenta con cinco desarrolladores.

Según este mismo reglamento la gestión del desarrollo contempla las siguientes actividades:

1. Analizar, desarrollar, implementar y administrar sistemas de información según las necesidades de la Universidad;
2. Participar en el diseño de base de datos y gestión web de la Universidad;
3. Desarrollar, implementar y administrar el sistema integrado de información universitaria;
4. Coordinar la implementación de sistemas informáticos en las unidades académicas y administrativas de la Institución;
5. Realizar el mantenimiento y actualizaciones de los sistemas existentes;
6. Presentar informes que por la naturaleza de sus funciones, solicite el jefe inmediato u otras autoridades de la Universidad;
7. Monitorear y supervisar el funcionamiento del sistema integrado de la institución;
8. Realizar las pruebas necesarias del software para asegurar la calidad del mismo;
9. Desarrollar, implementar, administrar y monitorear el Sitio Web de la Universidad;
10. Coordinar la publicación de información en el portal Web;
11. Participar en la adquisición de equipos, bienes y servicios informáticos que cumplan con los requerimientos y necesidades de la Institución;
12. Documentar técnicamente el software desarrollado;
13. Generar manuales técnicos y de usuario de todos los sistemas desarrollados;

14. Elaborar políticas respecto al desarrollo de software institucional, considerando normativas y estándares internacionales, reglamentos y políticas institucionales;

15. Llevar la documentación de versiones del software desarrollado; y,

16. Las demás actividades inherentes a desarrollo de software para la Universidad Técnica de Ambato.

4.4.2. Proceso de Desarrollo de Software en la Universidad Técnica de Ambato

El proceso de Desarrollo de Software esta basado según actividades de trabajo realizadas por el área de desarrollo de software en la Dirección de Tecnología de Información y Comunicación. El área no cuenta al momento con políticas respecto al desarrollo de software institucional, considerando normativas y estándares internacionales, reglamentos o políticas institucionales. Sin embargo el área de desarrollo se basa en el proceso que se detalla a continuación:

Actividades

Las actividades que se realizan en el proceso de desarrollo de software se detallan en el siguiente flujograma:

Figura 4.12: Flujograma de Actividades Proceso de Desarrollo de Software
Elaborado por: Daniel Jerez

4.4.3. Modelos de Desarrollo de Software en la Universidad Técnica de Ambato

Las actividades que se realizan en el área de desarrollo de software en la Dirección de Tecnología de Información y Comunicación, no siguen un modelo específico, pero cumplen con las etapas establecidas para el desarrollo de software como son: Análisis, Diseño, Codificación, Pruebas y Mantenimiento; para realizar estas actividades no se utiliza una metodología específica, a pesar de que en la encuesta realizada se menciona la utilización de metodologías, no existe una política en donde se detalle la que deben utilizar.

A pesar de que no siguen un modelo específico, utilizan en el desarrollo de algunos módulos o sistemas un modelo utilizado en los inicios del desarrollo de software denominado "Codificar y Corregir", el cual contiene dos pasos: Escribir código y corregir problemas en el código; se trata de primero implementar algo de código para la realización de algún módulo y luego pensar acerca de requisitos, diseño, validación y mantenimiento.

4.4.3.1. Tipo de Desarrollo de Software

El tipo de Desarrollo de Software que se aplica en la Dirección, es: Software a la Medida ya que se realiza un desarrollo interno y personalizado partiendo desde cero o desde algún sistema ya realizado, de acuerdo a especificaciones y requerimientos definidos los cuales se alinean a los objetivos de la Universidad, por lo general estos sistemas desarrollados responden a una necesidad muy particular para la Universidad.

4.4.3.2. Tipo de Arquitectura

En el área de desarrollo de software en la Dirección de Tecnología de Información y Comunicación, no se realiza ningún estudio para definir la arquitectura a utilizarse optan por la mas adecuada que es Cliente/Servidor.

4.4.3.3. Herramientas de Desarrollo de Software

SCRIPTCASE 7	**Definición**	Es un generador de código PHP para desarrollar sistemas web completos de forma rápida y segura.
	Características	Genera formularios Web y reportes.
		Soporta HML, Ajax y JQuery.
		Validación automática de datos, control de carga de archivos.
		Crea informes detallados para análisis y toma de decisiones.
		Genera gráficos estadísticos.
		Genera Grid editable para edición de registros.
		Conexiones a múltiples bases de datos.
		Permite la creación de una lista de tareas dentro de Scriptcase para que los desarrolladores organicen su rutina y optimicen la gestión de los proyectos.
		Provee formularios de búsqueda para que el usuario disponga de varios tipos de búsqueda en informes y formularios.
		Crea formularios o informes en Maestro-Detalle.
		Desarrolla sistemas a través de una interfaz traducida a 10 idiomas.
		Soporte Google Maps y Youtube.
		Aplicaciones Blank, crea programas PHP con total libertad con el uso de funciones predefinidas en ScriptCase y la interfaz de conexión a la base de datos.

Tabla 4.10: Herramientas de Desarrollo de Software. Scriptcase 7

Elaborado por: Daniel Jerez

VISUAL STUDIO 2012-2015	**Definición**	Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles.
	Características	Entorno de desarrollo integrado completo para compilar aplicaciones web, para el escritorio de Windows y multiplataforma para iOS, Android y Windows.
		Desarrollo de aplicaciones móviles para Android utilizando C++ nativo, Apache, Cordova o Xamarin.
		Análisis de código en el acto.
		Control de versiones Git con historial detallado.
		Pruebas unitarias inteligentes para cada instrucción del código.
		Mejoras en la depuración y diagnóstico.
		Soporta más de 23 lenguajes de programación.
		Incluye las mejores herramientas del sector y cientos de componentes.
		Permite la reutilización de código.
		Aumento del rendimiento, la escalabilidad y la fiabilidad.
		Seguridad incorporada fiable.
		Permiten exponer cualquier componente como servicio XML Web utilizando el código WebMethod, y utilizar los servicios Web XML de cualquier plataforma.
		Modo gráfico de diseñar Web Forms en HTML, ASP y ASP.NET sin necesidad de modificar manualmente el código HTML.
		Acorta el ciclo de desarrollo gracias a la depuración de lenguaje cruzado, de proceso cruzado y a la depuración remota.
		Ejecuta Intellitest que es una característica que explora el método con el que escribiste y ejecutaste la unit test para cada línea del código dentro de ese método con diferentes combinaciones de entrada.
		Mejoras en los editores HTML, CSS y JSON

Tabla 4.11: Herramientas de Desarrollo de Software. Visual Studio

Elaborado por: Daniel Jerez

Microsoft SQL Server	**Definición**	Es un sistema de administración y análisis de bases de datos relacionales, para soluciones de comercio electrónico, línea de negocio y almacenamiento de datos.
	Características	Soporte de transacciones.
		Alta disponibilidad y recuperación ante desastres.
		Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información.
		Contiene un servicio núcleo del almacenamiento, procesamiento, y seguridad de los datos.
		Permite crear bases de datos relacionales para procesamiento de transacciones en línea (OLTP).
		Permite crear bases de datos para el análisis de los datos (OLAP).
		Permite crear tablas para el almacenamiento de datos, índices, vistas y procedimientos almacenados.
		Contiene componentes de extracción, transformación y carga de datos (ETL).
		Puede extraer y transformar datos de una variedad de fuentes como archivos de datos XML, archivos de formato plano y otras.
		Permite realizar replicación transaccional, combinada y estática.
		Permite el almacenamiento de reportes y controla el acceso.

Tabla 4.12: Herramientas de Administración de Bases de Datos. SQL Server

Elaborado por: Daniel Jerez

4.4.3.4. Control del Desarrollo de Software en la Universidad Técnica de Ambato

El control de los proyectos de desarrollo de software se los realiza con la herramienta informática Microsoft Project, una de las más conocidas y de fácil manejo en el mercado informático, siguiendo los siguientes pasos:

1. Creación de una agenda de trabajo.
2. Organización del proyecto (fases de un proyecto).
3. Planificación de tareas.
4. Seguimiento y Control de fases y tareas del proyecto.

4.4.4. Proceso de Pruebas

Según el artículo 11, literal h del "REGLAMENTO DE LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN –DITIC - DE LA

UNIVERSIDAD TÉCNICA DE AMBATO", el cual indica: "Realizar las pruebas necesarias de software para asegurar la calidad del mismo", las pruebas son actividades que deben ser realizadas por parte del Área de Gestión de Desarrollo, realizadas en la etapa de desarrollo/codificación del sistema.

Por otra parte según Memorando Nro. UTA-DITIC-2017-0126-M del 01 de febrero de 2017, se dispone que todo módulo desarrollado a partir de esa fecha o información a ser reportada, deberá pasar por la verificación y aprobación del Área de la Gestión de Calidad por parte del Especialista de esa área. Dicha área por ser nueva, aún no está contemplada en el "REGLAMENTO DE LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN –DITIC - DE LA UNIVERSIDAD TÉCNICA DE AMBATO", por lo que las actividades designadas al área de Gestión de Calidad se realizan mediante un oficio, memorando o petición verbal por parte del Director de la DITIC.

Dados estos antecedentes el área de Gestión de Calidad aplica métodos y técnicas de V&V de software en la fase final del desarrollo, en esta etapa se aplican métodos de caja negra, caja blanca y pruebas de interfaz, por parte del personal encargado de realizar dichas actividades.

4.5. Aspecto Legal

4.5.1. Reglamento

"REGLAMENTO DE LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN –DITIC - DE LA UNIVERSIDAD TÉCNICA DE AMBATO". Ver Anexo A.

4.6. Verificación de Software

La Verificación de Software la realizan el Área de Gestión de Calidad y el Área de Gestión de Desarrollo como una actividad establecida en el proceso de desarrollo, estas áreas actualmente centran su trabajo en la observación y comportamiento del producto final, por lo que se concluye que se está realizando una Verificación Dinámica.

4.7. Validación de Software

La Validación de Software la realizan el Área de Gestión de Calidad y el Área de Gestión de Desarrollo como una actividad establecida en el proceso de desarrollo, estas áreas actualmente centran su trabajo en la realización de inspecciones para

validar el cumplimiento de los requerimientos establecidos en el desarrollo, por lo que se concluye que se está realizando una Validación Simple.

4.8. Métodos de Verificación y Validación de Software

El Área de Gestión de Calidad utilizan los siguientes métodos:

- Caja Negra

  Estas pruebas las realizan centrándose en lo que se espera de un módulo, es decir, intentan encontrar casos en los que el módulo no realiza lo esperado.

  El encargado de realizar las pruebas de caja negra ingresa datos como entradas y analiza las salidas, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

  Con este método se intenta encontrar errores como:

  - Funciones incorrectas o ausentes
  - Errores de interfaz
  - Errores en estructuras de datos o en acceso a BD externas
  - Errores de rendimiento
  - Errores de inicialización y finalización de un programa

- Caja Blanca

  Estas pruebas las realizan revisando el diseño interno del software, haciendo énfasis en la lógica del programa, para esto se verifica que los condicionales cumplan con su función y que estén bien estructurados. Además de revisar que el código fuente se encuentre debidamente documentado.

En el Área de Gestión de Desarrollo conformada por los analistas/desarrolladores, según la encuesta realizada el 50 % utilizan los métodos mencionados, y el 50 % restante no utiliza ningún método.

4.9. Técnicas de Verificación y Validación del Software

El Área de Gestión de Desarrollo conformada por los analistas/desarrolladores, según la encuesta realizada utilizan indistintamente alguna,varias o ninguna de las técnicas mencionadas a continuación:

- Análisis y Pruebas Funcionales.

- Análisis y Pruebas Estructurales.
- Estrategias de Integración.
- Testing and Análisis.
- Error-Oriented.
- Análisis de Flujo de Transacción.
- Análisis de Algoritmos.
- Análisis de Concurrencia.
- Análisis de Falla.
- Análisis de Flujo de Datos.

El Área de Gestión de Calidad utiliza como técnicas de Verificación y Validación del Software las siguientes:

- Análisis y Pruebas Funcionales (Pruebas de aceptación, pruebas de usuario).
- Análisis de Falla.
- Inspecciones.

4.10. Herramientas para la Verificación y Validación de Software

El Área de Gestión de Desarrollo y el Área de Gestión de Calidad no utilizan herramientas de verificación y validación de software.

4.11. Análisis y presentación de métodos, técnicas y herramientas de verificación y validación de software adecuadas para el desarrollo de software en la Universidad Técnica de Ambato

El presente análisis tiene como alcance evaluar los métodos, técnicas y herramientas de V&V de software, conjuntamente con los aplicados actualmente en la DITIC, para seleccionar los que se ajusten a las necesidades en la Universidad Técnica de Ambato, y ayuden a que el software llegue al cumplimiento de las siguientes características: Funcionalidad, Confiabilidad, Usabilidad, Eficiencia, Mantenibilidad, Portabilidad, según ISO-9126.

Evaluación de métodos y técnicas

Los métodos y técnicas deberán ser evaluados según las características y subcaracterísticas establecidas en el estándar internacional para la evaluación de Software ISO-9126, primera parte FDIS 9126-1, el cual describe un modelo de dos partes para la calidad de productos de software:

- Calidad interna y externa, y
- Calidad en uso

El modelo de calidad del estándar ISO-9126 establece seis características para la calidad interna y externa, cada una de las cuales se detalla a través de un conjunto de subcaracterísticas que permiten profundizar en la evaluación de la calidad de productos de software.

Se debe crear una matriz tanto para métodos y técnicas en la cual se aplicará la ponderación de cada característica y subcaracterística definida en el estándar, respecto a las propiedades del método o técnica estudiado.

Evaluación de herramientas

Las herramientas deberán ser evaluadas según las Métricas de Calidad en Uso definidas por la Norma ISO/IEC 9126 en su parte 4 (PDTR 9126-4), que definen la aceptación del software por parte del usuario final.

Se seleccionaron las características de Eficacia y Productividad.

Se analizará los resultados de las matrices y se presentarán los métodos, técnicas y herramientas adecuados.

Con este estudio se obtendrá la mejor opción que satisfaga las necesidades actuales de la Dirección orientadas a las pruebas en los productos finales.

4.11.1. Análisis de Métodos

Definición de pesos

Los pesos son cada uno de elementos a evaluar: métodos.

Proceso de análisis

1. A cada subcaracterística se le otorga un valor porcentual, de acuerdo al cumplimiento o no de la misma y al número de las mismas, detallado en la siguiente tabla:

Número de Subcaracterísticas	Valor Porcentual	Criterio
1	100 %	Cumple
2	50 %	
3	33.33 %	
4	25 %	
5	20 %	
-	0 %	No cumple

Tabla 4.13: Criterio de Valoración de Subcaracterísticas

Elaborado por: Daniel Jerez

2. La suma de los valores dados a las subcaracterísticas define el valor de cada característica.

   $x = \sum y$

   donde, $\boldsymbol{x}$ es el valor de la característica y $\boldsymbol{y}$ el valor de la subcaracterística.

3. Se sacará el promedio de los valores de las características para cada peso.

4. Se seleccionará los tres pesos mejor puntuados.

La valoración dada a cada peso es de acuerdo a las características que tiene cada método en la detección de errores, los cuales mediante su aplicación aportan a la mejora de la calidad del software, respecto a las características que define el estándar, por lo que a continuación se citan las principales particularidades de cada método en las que está basado este análisis.

Los métodos a evaluar son los siguientes:

Peso 1: Caja Negra

1. Se ignora la estructura de control, centrándose en los requisitos funcionales del sistema.

2. Complementaria con las pruebas de caja blanca.

3. Se utiliza un conjunto de datos validos o inválidos para probar la funcionalidad, obviando los detalles internos del programa.

4. Este método ayuda a validar complejos conjuntos de acciones y condiciones de un software.

5. Este método se centra en probar la interfaz de usuario.

6. Pueden realizarse pruebas especializadas como en áreas de seguridad y de interoperabilidad.
7. Realiza pruebas negativas hasta que el sistema falle.
8. Realiza pruebas de tensión o stress cuando hay competencia por los recursos.
9. Ayuda a verificar el resultado de la interacción entre los actores y el software.
10. Ayuda a verificar el funcionamiento del software en diferentes configuraciones.
11. Con este método se puede medir en el software las siguientes características:
    a) Funcionalidad
    b) Confiabilidad
    c) Usabilidad
    d) Eficiencia
    e) Mantenibilidad
    f) Portabilidad

Peso 2: Caja Blanca

1. Se centra en los mecanismos internos de un sistema.
2. Ayuda a disminuir los errores internos en la lógica de los componentes.
3. Encuentra errores de lógica y algorítmos (Pruebas Unitarias).
4. Complementaria con las pruebas de caja negra.
5. Reconoce la falta de funcionalidad, errores de código.
6. Se trabaja con casos de prueba, aplicados en diferentes niveles.
7. Garantiza que se ejecuten por lo menos una vez todos los caminos independientes de cada módulo.
8. Ejercita todas las decisiones lógicas verdaderas y falsas.
9. Ejercitan las estructuras internas de datos para asegurar su validez.
10. Disminuye en un gran porcentaje el numero de errores existentes en los sistemas brindando mayor confiabilidad.

11. Asegura la calidad de cada unidad probada y los resultados de los mismos.
12. Aplicadas durante el proceso de desarrollo.
13. Con este método se puede medir en el software las siguientes características:
    a) Funcionalidad
    b) Confiabilidad
    c) Eficiencia
    d) Mantenibilidad

Peso 3: Método Top-Down

1. El modelo Top Down se diseña con ayuda de cajas negras.
2. Ayuda al aislamiento de los errores de salidas a interfaz.
3. Se pueden encontrar los defectos en el diseño y corregirlos tempranamente.
4. Realiza pruebas mediante simulaciones o remplazo de módulos, usados cuando el software necesita interactuar con un sistema externo.
5. Se requiere de Stubs para reemplazar los módulos inferiores aún no implementados.
6. Desarrollar Stubs que emulen a los módulos es mucho trabajo.
7. La observación de las pruebas de salida es más difícil.
8. Retrasa las pruebas del procesamiento real.
9. Este método de prueba puede ser usado en una fase temprana del sistema.
10. Con este método se puede medir en el software las siguientes características:
    a) Funcionalidad
    b) Confiabilidad
    c) Mantenibilidad

Peso 4: Método Act-like-a-customer

1. Método de prueba basado en el conocimiento de cómo un usuario usa el sistema.
2. Se enfocan en encontrar los defectos que los clientes pueden encontrar.
3. Los clientes regularmente encuentran un pequeño porcentaje de errores en los sistemas.
4. Se utiliza en pruebas de validación y en pruebas de aceptación funcional para verificar que el software cumpla con sus requerimientos.
5. Detalla pruebas basadas en el uso del cliente y su entorno.
6. Ayuda a verificar que el sistema cumpla los requisitos indicados y que los usuarios den el visto bueno.
7. Se desarrolla tipos de pruebas por parte de los clientes las cuales incluyen datos incorrectos o entradas de datos ilegales.
8. Sirve también para medir el grado de entendimiento del sistema con el cliente.
9. Pueden ser pruebas alfa y beta antes de que el software sea liberado.
10. Con este método se puede medir en el software las siguientes características:
    a) Funcionalidad
    b) Confiabilidad
    c) Usabilidad
    d) Eficiencia

Peso 5: Método ATAM

1. Sirve para evaluar la arquitectura del software.
2. Ayuda a establecer una arquitectura que satisfaga las metas de calidad
3. Revela la forma en que una arquitectura específica satisface ciertos atributos de calidad.
4. Ayuda a elegir una arquitectura adecuada para un sistema que cumpla con las necesidades de calidad.
5. Evalua aspectos como la performance o la confiabilidad en la arquitectura.

6. Encuentra un conjunto de riesgos mitigados por la arquitectura.
7. Ayuda a decidir donde hay menos riesgos al seleccionar una arquitectura.
8. Mejora la documentación de la arquitectura.
9. Es útil cuando se lo realiza temprano en el desarrollo de software, cuando el costo de cambiar las arquitecturas es mínimo.
10. Se aplica luego de que el diseño ha sido establecido, antes de la fase de codificación.
11. Con este método se puede medir en el software las siguientes características:
    a) Funcionalidad
    b) Confiabilidad
    c) Eficiencia
    d) Mantenibilidad

Peso 6: Método ADR (Active Design Reviews)

1. Asegura la calidad del producto software.
2. Ayuda a evaluar unidades de software, como módulos o componentes.
3. Mide la calidad e integridad de la documentación y la suficiencia de los servicios proporcionados por el diseño.
4. Las revisiones se las realiza como una representación de los usuarios finales.
5. Consiste en construir cuestionarios de revisión, asignar tareas a revisores y analizar los resultados para evaluar la calidad.
6. Los resultados son errores e inconsistencias.
7. Con este método se puede medir en el software las siguientes características:
    a) Funcionalidad
    b) Confiabilidad
    c) Eficiencia
    d) Mantenibilidad

Peso 7: Método ARID (Active Reviews for Intermediate Designs)

1. Es la combinación entre el método ATAM y el método ADR
2. Enfocado en el diseño conceptual y la documentación inicial.
3. Este método es adecuado para realizar la evaluación de diseños parciales en las etapas tempranas del desarrollo.
4. Permite el descubrimiento de errores a tiempo, lo que ayuda a saber si el diseño es viable.
5. Se basa en probar el diseño para ver si satisface los escenarios.
6. Se utiliza para saber si el diseño que se está realizando es sostenible.
7. Con este método se puede medir en el software las siguientes características:
   a) Funcionalidad
   b) Confiabilidad
   c) Eficiencia
   d) Mantenibilidad

Interpretación de resultados

CARACTERISTICAS	P1	P2	P3	P4	P5	P6	P7	SUBCARACTERISTICAS	P1	P2	P3	P4	P5	P6	P7
Funcionalidad	100%	100%	60%	100%	80%	80%	80%	Adecuación	20%	20%	20%	20%	20%	20%	20%
								Exactitud	20%	20%	20%	20%	20%	20%	20%
								Interoperabilidad	20%	20%	20%	20%	20%	20%	20%
								Conformidad	20%	20%	0%	20%	0%	0%	0%
								Seguridad	20%	20%	0%	20%	20%	20%	20%
Confiabilidad	100%	100%	100%	100%	100%	100%	100%	Nivel de Madurez	25%	25%	25%	25%	25%	25%	25%
								Tolerancia a fallos	25%	25%	25%	25%	25%	25%	25%
								Capacidad de recuperación	25%	25%	25%	25%	25%	25%	25%
								Capacidad de fiabilidad	25%	25%	25%	25%	25%	25%	25%
Usabilidad	100%	0%	0%	100%	0%	0%	0%	Capacidad de ser entendido	20%	0%	0%	20%	0%	0%	0%
								Capacidad de ser aprendido	20%	0%	0%	20%	0%	0%	0%
								Capacidad de ser operado	20%	0%	0%	20%	0%	0%	0%
								Capacidad de atracción	20%	0%	0%	20%	0%	0%	0%
								Cumplimiento de usabilidad	20%	0%	0%	20%	0%	0%	0%
Eficiencia	100%	100%	0%	100%	100%	100%	100%	Comportamiento del tiempo	33.33%	33.33%	0%	33.33%	33.33%	33.33%	33.33%
								Utilización de recursos	33.33%	33.33%	0%	33.33%	33.33%	33.33%	33.33%
								Cumplimiento de eficiencia	33.33%	33.33%	0%	33.33%	33.33%	33.33%	33.33%
Mantenibilidad	100%	100%	100%	0%	100%	100%	100%	Capacidad de ser analizado	20%	20%	20%	0%	20%	20%	20%
								Capacidad de ser cambiado	20%	20%	20%	0%	20%	20%	20%
								Estabilidad	20%	20%	20%	0%	20%	20%	20%
								Capacidad de ser probado	20%	20%	20%	0%	20%	20%	20%
								Cumplimiento de mantenibilidad	20%	20%	20%	0%	20%	20%	20%
Portabilidad	40%	0%	0%	0%	0%	0%	0%	Capacidad de adaptación	0%	0%	0%	0%	0%	0%	0%
								Capacidad de instalación	0%	0%	0%	0%	0%	0%	0%
								Coexistencia	20%	0%	0%	0%	0%	0%	0%
								Capacidad de ser reemplazado	0%	0%	0%	0%	0%	0%	0%
								Cumplimiento de portabilidad	20%	0%	0%	0%	0%	0%	0%
	90%	67%	43%	67%	63%	63%	63%								

Figura 4.13: Análisis de Métodos para la Calidad Interna y Externa
Elaborado por: Daniel Jerez

P1: Caja Negra

P2: Caja Blanca

P3: Top-Down

P4: Act-like-a-customer

P5: ATAM

P6: ADR

P7: ARID

Según el análisis realizado los métodos Caja Negra, Caja Blanca y Act-like-a-customer son los tres mejor puntuados, los cuales mediante la detección de errores aportan a que un sistema pueda alcanzar las características detalladas en el modelo de calidad del estándar ISO-9126.

4.11.2. Análisis de Técnicas

Definición de pesos

Los pesos son cada uno de elementos a evaluar: técnicas.

Proceso de análisis

1. A cada subcaracterística se le otorga un valor porcentual, de acuerdo al cumplimiento o no de la misma y al número de las mismas, detallado en la siguiente tabla:

Número de Subcaracterísticas	Valor Porcentual	Criterio
1	100 %	Cumple
2	50 %	
3	33.33 %	
4	25 %	
5	20 %	
-	0 %	No cumple

Tabla 4.14: Criterio de Valoración de Subcaracterísticas

Elaborado por: Daniel Jerez

2. La suma de los valores dados a las subcaracterísticas define el valor de cada característica.

   $x = \sum y$

   donde, $\boldsymbol{x}$ es el valor de la característica y $\boldsymbol{y}$ el valor de la subcaracterística.

3. Se sacará el promedio de los valores de las características para cada peso.

4. Se seleccionará los tres pesos mejor puntuados.

La valoración dada a cada peso es de acuerdo a las características que tiene cada técnica en la detección de errores, las cuales mediante su aplicación aportan a la mejora de la calidad del software, respecto a las características que define el estándar, por lo que a continuación se citan las principales particularidades de cada técnica en las que está basado este análisis.

Los técnicas a evaluar son las siguientes:

Peso 1: Análisis y Pruebas Funcionales

1. Ayuda a comprobar que los sistemas desarrollados funcionan acorde a las especificaciones funcionales y requisitos del cliente.

2. Su objetivo principal es analizar el producto terminado.
3. Toma el punto de vista del usuario final.
4. Se ocupan en encontrar defectos, enlaces rotos,enlaces ortográficos.
5. Pueden ser realizadas por analistas o usuarios finales para medir el comportamiento del sistema.
6. Complementaria con las pruebas de caja negra ya que enfoca su atención a cómo se genera las respuestas.
7. Sus objetivos son aumentar la satisfacción del cliente y mejorar la productividad.
8. No considera la estructura interna del sistema.
9. Se pueden realizar en todos los niveles de pruebas.
10. Con esta técnica se puede medir en el software las siguientes características:
    a) Funcionalidad
    b) Confiabilidad
    c) Usabilidad
    d) Eficiencia
    e) Mantenibilidad
    f) Portabilidad

Peso 2: Análisis y Pruebas Estructurales

1. Realiza pruebas basadas en el código.
2. Se escriben casos de prueba suficientes para que cada sentencia en el programa se ejecute por lo menos una vez.
3. Se escriben casos de prueba suficientes para que cada decisión en el programa se ejecute por lo menos una vez.
4. Se escriben casos de prueba suficientes para que cada condición en el programa se ejecute por lo menos una vez.
5. Se pueden realizar en todos los niveles de pruebas.

6. Basadas en el flujo de control, ya que cumple todas las sentencias de un programa.

7. Basadas en el flujo de datos, ya que requiere que para cada variable, cada segmento sea ejecutado por los menos una vez.

8. Con esta técnica se puede medir en el software las siguientes características:
   a) Funcionalidad
   b) Confiabilidad
   c) Eficiencia
   d) Mantenibilidad

Peso 3: Estrategias de Integración

1. Realiza pruebas del software ensamblando todos sus módulos.

2. Su objetivo es encontrar fallas al integrar varios componentes.

3. Comprueba que las conexiones y comunicaciones entre los diferentes módulos funcionen.

4. Prueba de manera conjunta los módulos que ya funcionan correctamente por separado, para analizar su comportamiento como un todo.

5. Pueden ser incrementales o no incrementales.

6. Estas pruebas se llevan a cabo cada vez que se integra un módulo.

7. Al probar todo el programa se puede producir un gran conjunto de fallos, cuyo origen es difícil de identificar.

8. Se realizan al final de un ciclo de construcción.

9. Con esta técnica se puede medir en el software las siguientes características:
   a) Funcionalidad
   b) Confiabilidad
   c) Usabilidad
   d) Eficiencia
   e) Mantenibilidad
   f) Portabilidad

Peso 4: Testing and Analysis

1. Las actividades de pruebas y análisis ocurren a lo largo del desarrollo y evolución de los sistemas de software.
2. Se las realiza desde la entrega de requerimientos hasta la posterior evolución.
3. Evaluá la calidad depende de cada parte del proceso de software.
4. La prueba y el análisis de software están completamente integrados.
5. Con esta técnica se puede medir en el software las siguientes características:
   a) Funcionalidad
   b) Confiabilidad
   c) Usabilidad
   d) Eficiencia
   e) Mantenibilidad
   f) Portabilidad

Peso 5: Error-Oriented

1. Son técnicas que se enfocan en evaluar la presencia o ausencia de errores en el proceso de programación.
2. Son aplicadas en la etapa de desarrollo.
3. Las pruebas basadas en el error encuentran errores en el proceso de programación. Estas pruebas pueden ser conducidas por historias de errores del programador y por las medidas de complejidad del software.
4. Las pruebas basadas en los fallos tienen como objetivo demostrar que ciertas fallas no necesariamente están en el código.
5. Las pruebas de probable corrección ayudan a disminuir la probabilidad de que existan fallos en el programa. Se ejecutan pruebas en métodos y funciones con el objetivo de determinar que las mismas están o no implementadas de forma correcta.
6. Con esta técnica se puede medir en el software las siguientes características:
   a) Funcionalidad
   b) Confiabilidad

c) Eficiencia

d) Mantenibilidad

Peso 6: Análisis de Flujo de Transacción

1. El flujo de transacción se refiere al movimiento de datos a través de un camino de llegada que convierte la información del mundo exterior en una transacción.
2. Se evalúa la transacción y de acuerdo a su valor, el flujo sigue por uno de los caminos de acción.
3. Realiza pruebas sobre el diagrama de flujo de transacciones.
4. Realiza pruebas en las entradas y en las salidas del flujo de transacción.
5. Son más utilizadas en las bases de datos.
6. Con esta técnica se puede medir en el software las siguientes características:
   a) Funcionalidad
   b) Confiabilidad
   c) Mantenibilidad

Peso 7: Stress Testing (Análisis de Estrés)

1. Este tipo de prueba es usado para determinar la estabilidad de un sistema.
2. Pone énfasis en la robustez, disponibilidad y manejo de errores bajo condiciones desfavorables o una carga pesada.
3. Su objetivo es asegurar que el software no se bloquee en caso de existir recursos insuficientes como memoria o espacio en disco.
4. Son pruebas no funcionales, ya que miden el rendimiento, carga y estrés.
5. Identifica cuellos de botella en los sistemas.
6. Permite conocer los límites que soporta el sistema.
7. Permite tomar decisiones sobre configuraciones de hardware.
8. Con esta técnica se puede medir en el software las siguientes características:
   a) Funcionalidad
   b) Confiabilidad

c) Eficiencia

d) Mantenibilidad

Peso 8: Análisis de Algoritmos

1. El Análisis de Algoritmos es considerado un método automatizado muy útil en la creación de pruebas que son mucho más potentes en términos de cobertura de los requisitos y la cobertura de datos de prueba

2. Es más rápido de crear.

3. Este tipo de prueba resuelve el problema de los cambios frecuentes en los requisitos y la lenta creación de casos de prueba manual.

4. Con esta técnica se puede medir en el software las siguientes características:

   a) Funcionalidad

   b) Confiabilidad

   c) Eficiencia

   d) Mantenibilidad

Peso 9: Análisis de Concurrencia

1. Se lo conoce también como multi-user testing.

2. Este tipo de prueba busca detectar defectos en el sistema cuando múltiples usuarios realizan la misma acción y al mismo tiempo en una aplicación, módulo o base de datos.

3. Verifica el número de operaciones por intervalo de tiempo.

4. Permite analizar el pico máximo de operaciones realizadas en un momento dado.

5. Permite medir el tiempo de respuesta de un sistema cuando existe concurrencia en un momento determinado.

6. Ayuda a verificar el número de usuarios que en un instante están conectados a la aplicación.

7. Ayuda a medir la demanda de uso de una aplicación.

8. Con esta técnica se puede también medir el rendimiento del sistema.

9. Con esta técnica se puede medir en el software las siguientes características:
    a) Confiabilidad
    b) Eficiencia
    c) Mantenibilidad

Peso 10: Análisis de Falla

1. Las pruebas basadas en fallas utilizan un modelo de fallas directamente para hipotetizar fallas potenciales en un programa bajo prueba.
2. Se utilizan para crear o evaluar conjuntos de pruebas basados en la eficacia en la detección de fallas hipotéticas.
3. Se usan comúnmente en pruebas funcionales y estructurales.
4. Con esta técnica se puede medir en el software las siguientes características:
    a) Funcionalidad
    b) Confiabilidad
    c) Eficiencia
    d) Mantenibilidad

Peso 11: Análisis de Flujo de Datos

1. Son las pruebas realizadas en los diagramas de flujo de datos o DFD.
2. Realiza pruebas sobre las representaciones del diagrama de flujo.
3. Analiza seleccionando caminos basados en cómo un elemento sintáctico puede afectar el cálculo de otro.
4. Con esta técnica se puede medir en el software las siguientes características:
    a) Funcionalidad
    b) Confiabilidad
    c) Eficiencia
    d) Mantenibilidad

Peso 12: Análisis de Cobertura

1. Las pruebas de cobertura miden la cantidad de pruebas realizadas por un conjunto de casos de prueba.
2. Crea casos de prueba adicionales para aumentar la cobertura.
3. Ayuda a encontrar áreas de un programa que no han sido cubiertas por un conjunto de casos de prueba.
4. Ayuda a determinar una medida cuantitativa de la cobertura de código, que indirectamente mide la calidad de la aplicación o producto.
5. Se utiliza comúnmente en las pruebas de unidad.
6. Se realizan los casos de unidad necesarios para lograr un 100 % de cobertura.
7. Verifican la funcionalidad y estructura de cada componente individualmente.
8. Con esta técnica se puede medir en el software las siguientes características:
   a) Funcionalidad
   b) Confiabilidad
   c) Eficiencia
   d) Mantenibilidad

Peso 13: Inspecciones

1. Las inspecciones de software son revisiones manuales y colaborativas que se pueden aplicar a cualquier artefacto de software desde documentos de requisitos hasta código fuente.
2. La inspección complementa las pruebas ayudando a controlar muchas propiedades que son difíciles o imposibles de verificar dinámicamente.
3. Su flexibilidad hace que la inspección sea particularmente valiosa cuando otros análisis más automatizados no son aplicables.
4. Aplicable en todas las etapas del ciclo del desarrollo de software.
5. No se requiere de la utilización de herramientas sofisticadas.
6. Se utilizan listas de chequeo.
7. Ayuda a prevenir el mal funcionamiento de los sistemas.

8. Contribuyen al mejoramiento de la calidad desde etapas tempranas del desarrollo.

9. Con esta técnica se puede medir en el software las siguientes características:
   a) Funcionalidad
   b) Confiabilidad
   c) Usabilidad
   d) Eficiencia
   e) Mantenibilidad
   f) Portabilidad

Peso 14: Model Checking (Verificación de Modelos)

1. Usa modelos de comportamiento esperado para producir especificaciones de casos de prueba que pueden revelar discrepancias entre el comportamiento real del programa .

2. Prueba las propiedades del programa.

3. Encuentra violaciones en las especificaciones del sistema.

4. Explora exhaustivamente todos los estados del programa hasta no exista ningún defecto.

5. Con esta técnica se puede medir en el software las siguientes características:
   a) Funcionalidad
   b) Confiabilidad
   c) Usabilidad
   d) Eficiencia
   e) Mantenibilidad
   f) Portabilidad

Análisis de resultados

P1: Análisis y Pruebas Funcionales

P2: Análisis y Pruebas Estructurales

P3: Estrategias de Integración

CARACTERISTICAS	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	SUBCARACTERISTICAS	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14
Funcionalidad	100%	100%	100%	100%	100%	60%	100%	100%	0%	100%	100%	100%	100%	100%	Adecuación	20%	20%	20%	20%	20%	20%	20%	20%	0%	20%	20%	20%	20%	20%
															Exactitud	20%	20%	20%	20%	20%	20%	20%	20%	0%	20%	20%	20%	20%	20%
															Interoperabilidad	20%	20%	20%	20%	20%	20%	20%	20%	0%	20%	20%	20%	20%	20%
															Conformidad	20%	20%	20%	20%	20%	0%	20%	20%	0%	20%	20%	20%	20%	20%
															Seguridad	20%	20%	20%	20%	20%	0%	20%	20%	0%	20%	20%	20%	20%	20%
Confiabilidad	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	Nivel de Madurez	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%
															Tolerancia a fallos	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%
															Capacidad de recuperación	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%
															Capacidad de fiabilidad	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%	25%
Usabilidad	100%	0%	100%	100%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	Capacidad de ser entendido	20%	0%	20%	20%	0%	0%	0%	0%	0%	0%	0%	0%	20%	20%
															Capacidad de ser aprendido	20%	0%	20%	20%	0%	0%	0%	0%	0%	0%	0%	0%	20%	20%
															Capacidad de ser operado	20%	0%	20%	20%	0%	0%	0%	0%	0%	0%	0%	0%	20%	20%
															Capacidad de atracción	20%	0%	20%	20%	0%	0%	0%	0%	0%	0%	0%	0%	20%	20%
															Cumplimiento de usabilidad	20%	0%	20%	20%	0%	0%	0%	0%	0%	0%	0%	0%	20%	20%
Eficiencia	100%	100%	100%	100%	100%	0%	100%	100%	100%	100%	100%	100%	100%	100%	Comportamiento del tiempo	33.33%	33.33%	33.33%	33.33%	33.33%	0%	33.33%	33.33%	33.33%	33.33%	33.33%	33.33%	33.33%	33.33%
															Utilización de recursos	33.33%	33.33%	33.33%	33.33%	33.33%	0%	33.33%	33.33%	33.33%	33.33%	33.33%	33.33%	33.33%	33.33%
															Cumplimiento de eficiencia	33.33%	33.33%	33.33%	33.33%	33.33%	0%	33.33%	33.33%	33.33%	33.33%	33.33%	33.33%	33.33%	33.33%
Mantenibilidad	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	Capacidad de ser analizado	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%
															Capacidad de ser cambiado	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%
															Estabilidad	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%
															Capacidad de ser probado	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%
															Cumplimiento de mantenibilidad	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%
Portabilidad	40%	0%	100%	20%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	Capacidad de adaptación	0%	0%	20%	0%	0%	0%	0%	0%	0%	0%	0%	0%	20%	0%
															Capacidad de instalación	0%	0%	20%	0%	0%	0%	0%	0%	0%	0%	0%	0%	20%	0%
															Coexistencia	20%	0%	20%	0%	0%	0%	0%	0%	0%	0%	0%	0%	20%	0%
															Capacidad de ser reemplazado	0%	0%	20%	0%	0%	0%	0%	0%	0%	0%	0%	0%	20%	0%
															Cumplimiento de portabilidad	20%	0%	20%	20%	0%	0%	0%	0%	0%	0%	0%	0%	20%	0%
	90%	67%	100%	87%	67%	43%	67%	67%	50%	67%	67%	67%	100%	83%															

Figura 4.14: Análisis de Técnicas para la Calidad Interna y Externa
Elaborado por: Daniel Jerez

P4: Testing and Analysis

P5: Error-Oriented

P6: Análisis de Flujo de Transacción

P7: Stress Testing (Análisis de Estrés)

P8: Análisis de Algoritmos

P9: Análisis de Concurrencia

P10: Análisis de Falla

P11: Análisis de Flujo de Datos

P12: Análisis de Cobertura

P13: Inspecciones

P14: Model Checking (Verificación de Modelos)

Las técnicas mejor puntuadas son Estrategias de Integración, Inspecciones, y Análisis y Pruebas Funcionales. Estas técnicas se complementan y pueden trabajar en conjunto con los métodos ya seleccionados.

4.11.3. Análisis de Herramientas

Para analizar las herramientas se utilizaron las Métricas de Calidad en Uso definidas por la Norma ISO/IEC 9126 en su parte 4 (PDTR 9126-4) que definen la aceptación del software por parte del usuario final, ya que el análisis se lo realizará sobre herramientas informáticas que permiten la verificación y validación de software. Ver anexo B Tabla de Métricas Calidad en Uso.

Se seleccionaron las características de Eficacia y Productividad.

Para la Eficacia se seleccionó la métrica *Terminación de la tarea*, que mide la proporción de las tareas que se han terminado.

Para la Productividad se seleccionó la métrica *Tiempo de tarea*, que mide el tiempo que toma en completar una tarea.

Proceso de análisis

Para el análisis de las herramientas mediante las métricas de Calidad en Uso se sigue el procedimiento utilizado por Erick Ortega en su Proyecto de Graduación[3], ESPOL detallado a continuación:

Característica	**Nombre de la métrica**	**Propósito**	**Fórmula**	**Descripción**	**Interpretación**
Eficacia	*Terminación de la tarea*	¿Qué proporción de las tareas ha sido teminado?	$X = \frac{A}{B}$	A =Número de tareas completadas B =Número total de tareas esperadas	0<=X<=1 Cuanto más cerca de 1.0 mejor.
Productividad	*Tiempo de tarea*	¿Cuánto tiempo se demora en completar una tarea?	$X = T_a$	T_a =Tiempo de tarea	0<=X El menor, el mejor.

Tabla 4.15: Métricas de Calidad en Uso seleccionadas para el análisis de herramientas
Adaptado de: ISO/IEC TR 9126-4:2004

1. Revisión de funcionalidad de los productos de software.
2. Pruebas de usuario.
3. Revisión de las mediciones realizadas.

Revisión de funcionalidad de los productos de software

Para la selección de las herramientas se ha investigado cada una de ellas, analizando sus funciones, características, principales ventajas y desventajas, y se han seleccionado las siguientes herramientas Open Source, cumpliendo así con el Decreto Ejecutivo No. 1014 emitido el 10 de Abril de 2008, que dispone el uso de Software Libre en los sistemas y equipamientos informáticos de la Administración Pública de Ecuador:

1. Bugzilla Testopia
2. QABook Desktop
3. Selenium IDE
4. Sahi

Pruebas de usuario

Se ha revisado la funcionalidad de cada software y en función de los requerimientos que tiene la DITIC en cuanto a una herramienta de validación y verificación de software, se han seleccionado varias tareas que se probarán en los herramientas mencionados:

1. Ingresar Requerimientos
2. Ingresar Casos de Prueba
3. Ejecutar Pruebas
4. Ingresar Resultados
5. Ver Errores
6. Generar/Ver Reportes

Terminación de la tarea						
Software	Nº Tarea	Nombre Tarea	Cumple	Número de tareas completadas A	Número de tareas esperadas B	**Terminación de la tarea X=A/B**
Bugzilla Testopia	1	Ingresar Requerimientos	Si	5	6	**0,83**
	2	Ingresar Casos de Prueba	No			
	3	Ejecutar Pruebas	Si			
	4	Ingresar Resultados	Si			
	5	Ver Errores	Si			
	6	Generar/Ver Reportes	Si			
QABook Desktop	1	Ingresar Requerimientos	Si	6	6	**1**
	2	Ingresar Casos de Prueba	Si			
	3	Ejecutar Pruebas	Si			
	4	Ingresar Resultados	Si			
	5	Ver Errores	Si			
	6	Generar/Ver Reportes	Si			
Selenium IDE	1	Ingresar Requerimientos	Si	5	6	**0,83**
	2	Ingresar Casos de Prueba	Si			
	3	Ejecutar Pruebas	Si			
	4	Ingresar Resultados	No			
	5	Ver Errores	Si			
	6	Generar/Ver Reportes	Si			
Sahi	1	Ingresar Requerimientos	Si	6	6	**1**
	2	Ingresar Casos de Prueba	Si			
	3	Ejecutar Pruebas	Si			
	4	Ingresar Resultados	Si			
	5	Ver Errores	Si			
	6	Generar/Ver Reportes	Si			

Figura 4.15: Análisis de Herramientas según la Métrica Terminación de la Tarea
Elaborado por: Daniel Jerez

Tiempo de tarea			
Software	Nº Tarea	Nombre Tarea	**Tiempo de tarea Ta [mm:ss] X=Ta**
Bugzilla Testopia	1	Ingresar Requerimientos	**01:45,6**
	2	Ingresar Casos de Prueba	**01:40,3**
	3	Crear Pruebas	**02:17,8**
QABook Desktop	1	Ingresar Requerimientos	**00:33,7**
	2	Ingresar Casos de Prueba	**01:19,8**
	3	Crear Pruebas	**01:28,1**
Selenium	1	Ingresar Requerimientos	**00:48,3**
	2	Ingresar Casos de Prueba	**00:53,4**
	3	Crear Pruebas	**00:30,5**
Sahi	1	Ingresar Requerimientos	**00:30,4**
	2	Ingresar Casos de Prueba	**00:48,9**
	3	Crear Pruebas	**00:28,9**

Figura 4.16: Análisis de Herramientas según la Métrica Tiempo de Tarea
Elaborado por: Daniel Jerez

Revisión de las mediciones realizadas

Se realizaron las siguientes mediciones en las herramientas analizadas: el número de tareas completadas y el tiempo de minutos utilizados para la realización de cada tarea.

Estas mediciones de tiempo se realizaron tomando en cuenta la facilidad de ser operado, facilidad de ser entendido e intuición-atracción de la herramienta.

Según la métrica Terminación de Tarea las herramientas que cumplieron con la todas las tareas evaluadas son: QABook Desktop y Sahi como se aprecia en la Tabla 4.15.

De acuerdo a la métrica Tiempo de Tarea las mismas herramientas han alcanzado el menor tiempo al cumplir con dichas tareas lo que se detalla en la Tabla 4.16.

Por lo expuesto anteriormente se seleccionaron las dos mejores opciones que son: QABook Desktop y Sahi.

4.11.4. Métodos, técnicas y herramientas de verificación y validación de software adecuadas para el desarrollo de software en la Universidad Técnica de Ambato

Una vez realizado el estudio y análisis de los métodos, técnicas y herramientas para V&V de software, se establecen los siguientes:

Métodos:

- Caja Negra
- Caja Blanca
- Act-like-a-customer

Los métodos Caja Negra y Caja Blanca se utilizan actualmente para realizar verificación y validación de software en la Dirección, la cual centra su atención sobre productos finales. El método Act-like-a-customer se enfoca netamente en los usuarios, lo cual es importante a la hora de verificar que el software cumpla con sus requerimientos.

Técnicas:

- Estrategias de Integración
- Inspecciones
- Análisis y Pruebas Funcionales

La técnica Estrategias de Integración se ajusta ya que actualmente los módulos desarrollados en la DITIC forman parte de un sistema integrado "SIUTA", mediante esta técnica se puede detectar fallos al integrar varios componentes como es el caso del sistema mencionado y comprueba que la comunicación entre los diferentes módulos funcione, la técnica Inspecciones complementa a las técnicas seleccionadas, ya que estas se pueden aplicar a cualquier artefacto de software desde la revisión de requisitos hasta la revisión de código fuente y finalmente la técnica Análisis y Pruebas Funcionales apoya a los métodos previamente seleccionados ya que centra su trabajo en los productos finales.

Herramientas:

- QABook Desktop

- Sahi

Estas herramientas pueden apoyar a los métodos y técnicas seleccionados y a la verificación y validación de software que actualmente se aplica en la DITIC.

Diagrama sugerido para la aplicación de métodos, técnicas y herramientas para V&V de software según flujograma de las actividades que se realizan en el proceso de desarrollo de software en la Dirección de Tecnología de la Universidad Ténica de Ambato.

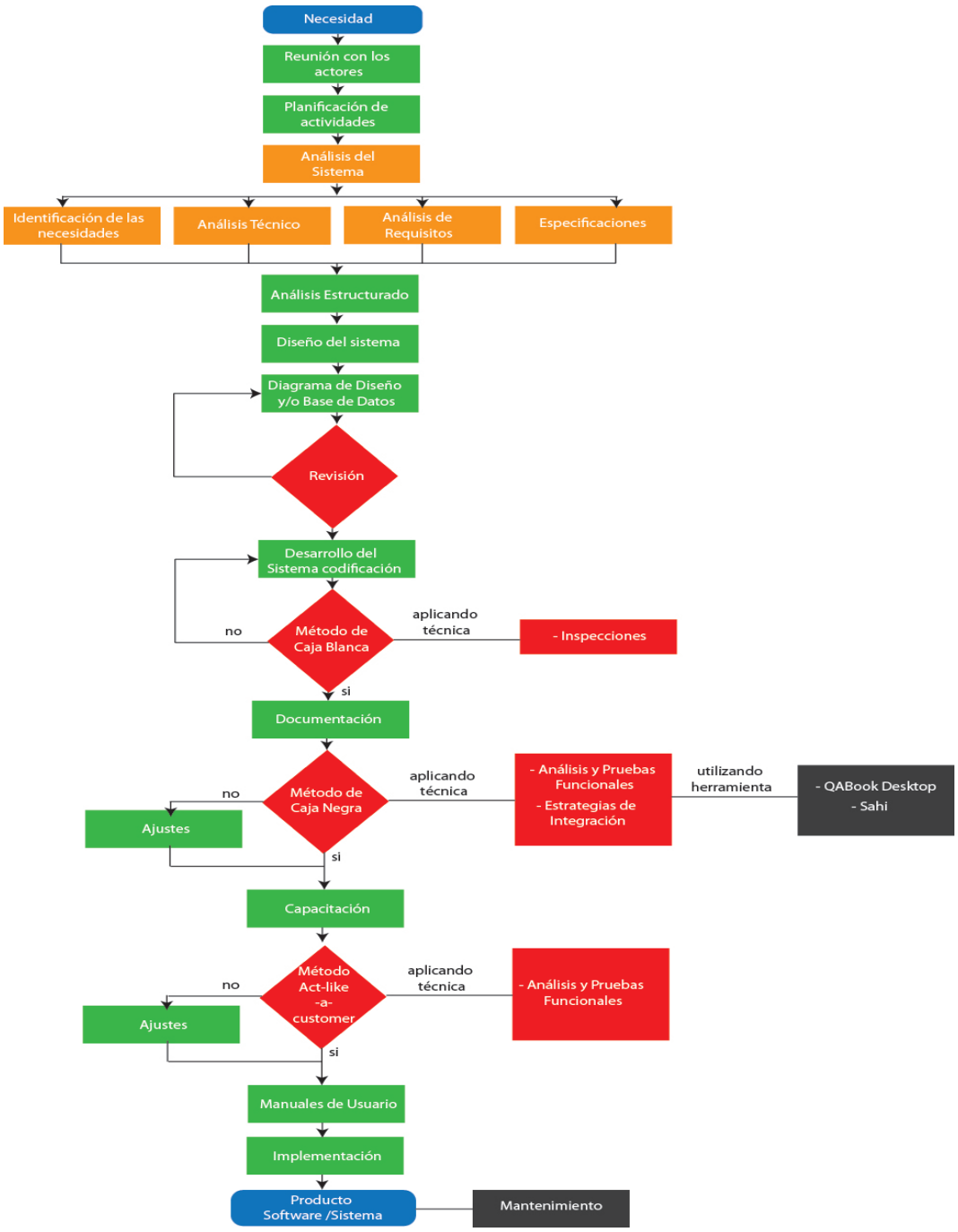


Figura 4.17: Flujograma sugerido para la aplicación de métodos, técnicas y herramientas para V&V de software en la DITIC
Elaborado por: Daniel Jerez

CAPÍTULO 5

Conclusiones y Recomendaciones

5.1. CONCLUSIONES

- En la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato se realizan actividades tales como Reunión con los actores (Definición de requisitos), Planificación de actividades, Análisis, Diseño, Desarrollo(Codificación), Capacitación, Pruebas y Mantenimiento las cuales se asemejan a un modelo en cascada, además de que utilizan en el desarrollo de algunos módulos un modelo de los inicios de la programación Codificar y Corregir, el cual genera pérdida de recursos como dinero, esfuerzo y tiempo.
- En la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato no existe una metodología estándar para el apoyo en el proceso el desarrollo de software.
- En la Dirección de Tecnología de Información y Comunicación se aplica Verificación Dinámica ya que centra su trabajo en la observación y comportamiento del producto final, y Verificación Simple ya que su trabajo se enfoca en la realización de inspecciones para validar el cumplimiento de los requisitos establecidos en el desarrollo.
- En base al Modelo de Calidad de la norma ISO/IEC 9126 se puede crear un criterio de evaluación que permita elegir métodos y técnicas los cuales mediante su aplicación en la detección de errores aportan al cumplimiento de las características y subcaracterísticas de dicho modelo, las cuales son claves al momento de brindar calidad a un software.
- Para la selección de una herramienta de software se puede utilizar las métricas de Eficacia “Terminación de la tarea” y Productividad “Tiempo de la Tarea” establecidas en el estándar ISO/IEC 9126.

5.2. RECOMENDACIONES

- Se recomienda que el método de caja blanca sea realizado por el área de desarrollo, ya que las personas adecuadas para la realización de pruebas de este tipo son los desarrolladores, puesto que ellos son los que mejor conocen los componentes internos del sistema, lo que permitirá que se optimice el tiempo de desarrollo del proyecto.

- Se recomienda elaborar políticas respecto al desarrollo de software institucional, considerando normativas y estándares internacionales, reglamentos y políticas institucionales, para así cumplir con el Reglamento de la DITIC, y con el numeral 2 del subcomponente 410 TECNOLOGÍA DE LA INFORMACIÓN, 410-07 Desarrollo y adquisición de software aplicativo de las Normas de Control Interno de la Contraloría General del Estado, el cual menciona; "Adopción, mantenimiento y aplicación de políticas públicas y estándares internacionales para: codificación de software, nomenclaturas, interfaz de usuario, interoperabilidad, eficiencia de desempeño de sistemas, escalabilidad, validación contra requerimientos, planes de pruebas unitarias y de integración." .

- Se sugiere realizar un estudio para seleccionar un modelo del proceso de desarrollo de software, el cual utilice verificación y validación en todas sus etapas, y una metodología ágil que apoye en el proceso de desarrollo de software, y así lograr construir un software que pueda alcanzar lo que el usuario espera, además de cumplir con lo que se detalla en las Normas de Control Interno de la Contraloría General del Estado, en su subcomponente 410 TECNOLOGÍA DE LA INFORMACIÓN, 410-07 Desarrollo y adquisición de software aplicativo, numeral 5, el cual menciona que en el proceso de desarrollo, mantenimiento o adquisición de software deben considerarse estándares de desarrollo de documentación y calidad.

Bibliografía

[1] P. Granda, "Análisis de las metodologías ágiles y su incidencia en la creación del portafolio de servicio para la unidad de extensión universitaria de la universidad técnica del norte de la ciudad de ibarra," Master's thesis, Universidad Técnica del Norte de la Ciudad de Ibarra, 2016.

[2] F. Tituana, "Análisis de métodos, técnicas y herramientas deverificación y validación de software usados por empresas ecuatorianas desarrolladoras de software," Master's thesis, Escuela Superior Politécnica del Litoral, 2009.

[3] E. Ortega, "Estudio de aplicabilidad y comparativo de un modelo de calidad a productos de software con la norma iso/iec 9126." 2010.

[4] J. Zamora, "Análisis de los procesos de verificación y validación en las organizaciones software," Master's thesis, Universidad Carlos III de Madrid, 2011.

[5] M. Deutsch, *Verification and Validation.* Prentice Hall: Jensen y C. Tonies, 1979.

[6] I. Jacaboson, G. Booch, and J. Rumbaugh, *El Proceso Unificado de Desarrollo de Software.* Addison Wesley, 2000.

[7] I. Sommerville, *Ingeniería de Software.* Pearson Educación, 2005.

[8] R. Pressman, *Ingeniería del Software: Un enfoque práctico.* McGraw Hill, 1997.

[9] P. Letelier, "Proceso de desarrollo de software," Master's thesis, Departamento de Sistemas Informáticos y Computación Universidad Politécnica de Valencia, 2003.

[10] B. Boehm, "A spiral model of software develpment and enhancement," *IEEE Computer*, 1988.

[11] W. Royce, “Managing the developmento of large software systems: concepts and technique,” *IEEE Westcon*, 1970.

[12] R. Balzer, *A 15 Year Perspective on Automatic Programming.* IEEE Transactions on Software Engineering, 1985.

[13] H. Mills and D. ONeil, *The Management of Software Engineering.* IBM Systems, 1980.

[14] “Ieee standard 610. compilation of ieee standard computer glossaries.”

[15] “Iso 9126-1 software quality characteristics and metrics part 1 - quality characteristics and sub-characteristics. (1997).”

[16] S. Rakitin, *Software Verification and Validation for Practitioners and Managers.* Artech House, 2001.

[17] M. Pezzé and M. Young, *Software Testing and Analysis: Process, Principles, and Techniques.* Wiley India, 2008.

Anexos y Apéndices

Anexo A

REGLAMENTO DE LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN –DITIC - DE LA UNIVERSIDAD TÉCNICA DE AMBATO

Universidad Técnica de Ambato
Consejo Universitario
Av. Colombia 02-11 y Chile (Cdla. Ingahurco) - Teléfonos: 593 (03) 2521-081 / 2822960 - Fax: 2521-084
Ambato – Ecuador

RESOLUCIÓN: 1254-CU-P-2016

El Honorable Consejo Universitario de la Universidad Técnica de Ambato, en sesión extraordinaria efectuada el martes 21 de junio de 2016, visto el oficio UTA-P-1416-2015, del 9 de noviembre de 2015, suscrito por el Doctor Ángel Polibio Chaves, Procurador de la Institución, mediante el cual remite los informes particulares, para fines de aprobación en segunda y definitiva de cinco reglamentos: entre ellos el **"REGLAMENTO DE LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN – DITIC- DE LA UNIVERSIDAD TÉCNICA DE AMBATO"**, el mismo que fue aprobado en primera instancia mediante Resolución 2470-CU-P-2013, del 23 de diciembre de 2013; en uso de sus atribuciones contempladas en el literal g) del artículo 21 del Estatuto Universitario, y, demás normativa legal aplicable para el efecto.

RESUELVE:

Aprobar en segunda y definitiva el adjunto **"REGLAMENTO DE LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN – DITIC- DE LA UNIVERSIDAD TÉCNICA DE AMBATO"**.

Ambato junio 21, 2016

Dr. MSc. Galo Naranjo López
PRESIDENTE DEL H. CONSEJO
UNIVERSIDAD TÉCNICA DE AMBATO

Ab. MSc. José Romo Santana
SECRETARIO GENERAL

copias: Rectorado
VAC
VAD
DECANOS FACULTADES:
DIFIN DTH
DITIC
PROCURADURÍA
SECRETARÍA GENERAL
Auditoria Interna

GN/JR/AV

EL HONORABLE CONSEJO UNIVERSITARIO DE LA UNIVERSIDAD TÉCNICA DE AMBATO

CONSIDERANDO:

Que, el Artículo 226 de la Constitución de la República del Ecuador establece el principio de legalidad, mediante el cual las instituciones del Estado, sus organismos, dependencias, las servidoras o servidores públicos y las personas que actúen en virtud de una potestad estatal ejercerán solamente las competencias y facultades que les sean atribuidas en la Constitución y la ley;

Que, el Artículo 355 de la propia Ley Suprema, en concordancia con los artículos 17 y 18 de la Ley Orgánica de Educación Superior, determinan que el Estado reconocerá a las universidades y escuelas politécnicas autonomía académica, administrativa, financiera y orgánica, acorde con los objetivos del régimen de desarrollo y los principios establecidos en la Constitución recalcando que uno de los mecanismos para ejercer esta autonomía es la gestión de los procesos internos;

Que, el artículo 1 de la Ley Orgánica de Transparencia y Acceso a la Información Pública determina el principio de publicidad de la información pública y en la disposición transitoria segunda Ibídem determina que es obligación de las instituciones del Estado crear portales en internet para garantizar dicho principio e interactuar con la ciudadanía, por lo tanto, se requiere determinar claramente las funciones de la Unidad que tendrá esta responsabilidad;

Que, el artículo 4 y demás pertinentes de la Ley del Sistema Nacional de Registro de Datos Públicos, determina que es responsabilidad de las instituciones del Sector Público que administren bases o registros de datos públicos la integridad, protección y control de dicha información, por lo tanto se requiere determinar las acciones con las cuales la Universidad Técnica de Ambato. cumpla con lo señalado;

Que, el Estatuto Universitario en los artículos 129, 130 y 131, determina la misión de la Dirección de Tecnología de Información, sus atribuciones y responsabilidades y su estructura interna;

Que, las tecnologías de la Información y la comunicación se constituyen en herramientas importantes para el desarrollo administrativo y académico de la Universidad Técnica de Ambato, y son un aporte sustancial para la adecuada prestación del servicio público educación superior a la colectividad;

Que, en la actual sociedad de la información se requiere una adecuada administración de los bienes que conforman el patrimonio tecnológico de la Universidad Técnica de Ambato;

UNIVERSIDAD TÉCNICA DE AMBATO * CONSEJO UNIVERSITARIO *

Que, es necesario tener un cuerpo legal que permitirá definir y establecer los parámetros sobre los cuales se efectuará el trabajo en la Dirección de Tecnología de Información y Comunicación, en base a la adecuada definición de procesos, asignación y distribución de acciones al personal que labora en dicha Unidad;

En uso de sus atribuciones, constantes en el artículo 21 literal g) del Estatuto Universitario:

RESUELVE

APROBAR Y EXPEDIR EL REGLAMENTO DE LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN –DITIC- DE LA UNIVERSIDAD TECNICA DE AMBATO

CAPITULO I
DEL ÁMBITO, FINES Y OBJETIVOS

Artículo 1. *Finalidad.*- La Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato con sus siglas DITIC es la encargada de gestionar las Tecnologías de la Información y Comunicaciones de la Institución, mediante su desarrollo, administración y mantenimiento; a fin de contribuir al direccionamiento estratégico, al desarrollo de las capacidades educativas, la seguridad integral de los espacios académicos y el apoyo al desarrollo Institucional.

Artículo 2.*Ámbito de Trabajo.*- La Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato es la dependencia responsable de administrar los recursos de Tecnologías de Información y Comunicación y sistemas de la Institución y de las entidades con las que ésta comparte recursos informáticos, además establecer normas, estándares, planes y metodologías para la gestión de la tecnología de la información, comunicaciones e infraestructura de redes, sistemas, equipos, bases de datos, comunicaciones informáticas y de telefonía IP.

Artículo 3. *Visión.*- La visión de la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato es constituirse en una Dirección líder en el diseño, desarrollo y optimización de tecnologías de la información y comunicación, con calidad y con espíritu de servicio, consolidada como modelo a seguir para otras Direcciones y Unidades de la Universidad Técnica de Ambato.

Artículo 4. *De la Misión.*- La misión de la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato es investigar, asesorar, dirigir, planificar,

administrar bienes y servicios, y ejecutar planes, programas y proyectos con el fin de proveer nuevas tecnologías de información y comunicaciones (TICs) que permitan optimizar la gestión institucional, atención al cliente y toma de decisiones, con calidad, productividad y mejoramiento continuo; garantizando la disponibilidad, integridad y confiabilidad de la información, software, hardware, datos y comunicaciones institucionales; estableciendo normas y políticas internas para su gestión.

Artículo 5. *Valores.-* Los valores humanos conductores de la acción de la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato y que deben ser practicados por todos sus integrantes serán los tipificados en el Estatuto Universitario, Plan Estratégico y Código de Ética de la institución y demás normativa legal aplicable para el efecto.

Artículo 6. *Políticas.-* Las políticas de la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato en relación con: la calidad, confidencialidad, controles internos, seguridad, propiedad intelectual, legalidad del software, comunicación, divulgación de la información, deberán guardar conformidad con las definidas en el Plan Estratégico de Desarrollo Institucional y aprobadas por el H. Consejo Universitario.

Artículo 7. *Objetivo General.-* Esta Unidad tendrá como objetivo general el coordinar y asegurar el cumplimiento de las actividades de la gestión de las Tecnologías de Información y Comunicación, que permitan el alineamiento con los objetivos estratégicos de la Universidad Técnica de Ambato y sus funciones operativas.

Artículo 8. *Objetivos Específicos.-* Los objetivos específicos de la Dirección de Tecnología de Información y Comunicación de la Universidad Técnica de Ambato son:

a) Estandarizar las normas de usabilidad, accesibilidad y buenas prácticas de las Tecnologías de Información y Comunicación en la Universidad Técnica de Ambato en función a políticas y prácticas de nivel mundial;

b) Garantizar la disponibilidad y correcto funcionamiento de los servicios de Tecnologías de Información en todas las unidades organizativas de la comunidad universitaria;

c) Planificar, desarrollar, monitorear y controlar los recursos relacionados con las Redes de comunicación, mantenimiento y soporte técnico, desarrollo de aplicaciones y sistemas informáticos, bases de datos y seguridad informática;

d) Establecer mecanismos de seguridad de la información que permita resguardar y proteger la misma, garantizando la confidencialidad y disponibilidad.

UNIVERSIDAD TECNICA DE AMBATO * CONSEJO UNIVERSITARIO *

CAPITULO II
ESTRUCTURA ORGANIZACIONAL Y FUNCIONAL

Sección Primera
Estructura Organizacional

Artículo 9. *De la Estructura Interna.*- La Dirección de Tecnología de Información y Comunicación estará posicionada dentro de la estructura organizacional de la Universidad, en un nivel que le permita realizar actividades de asesoría y apoyo a toda la comunidad universitaria; participar en la toma de decisiones de la Universidad y generar procesos de mejora en el área de tecnología de información y comunicación, y de acuerdo al Estatuto Universitario.

Sección Segunda
Estructura Funcional

Artículo 10. *De la Gestión de Recursos de Producción, Redes y Mantenimiento.*- La gestión de recursos de producción, redes y mantenimiento estará a cargo de un Especialista de Tecnología de Información y Comunicación el mismo que será responsable de proponer proyectos para implementar o mejorar la conectividad de la Universidad Técnica de Ambato, y coordinar las actividades relacionadas a:

a) Brindar soporte técnico de correo electrónico institucional;
b) Asistir con soporte técnico a los equipos de la red- Internet;
c) Proporcionar soporte técnico de la infraestructura de las redes de comunicación;
d) Proponer y administrar mecanismos de acceso y control de la red de comunicaciones;
e) Supervisar el mantenimiento de la red de comunicaciones;
f) Coordinar la implementación y actualización de la red de comunicaciones;
g) Administrar las redes de comunicación, correo electrónico e Internet;
h) Realizar la instalación de cableado estructurado y fibra óptica de la red de comunicaciones;
i) Presentar informes técnicos de entrega recepción de equipos informáticos;
j) Elaborar planes de mantenimiento para aplicación en todas las Facultades y Dependencias de la Universidad Técnica de Ambato;
k) Proporcionar mantenimiento técnico de equipo informático;
l) Realizar configuración de equipos activos de la red de la Universidad;
m) Participar en la adquisición de equipos, bienes y servicios informáticos que cumplan con los requerimientos y necesidades de la Institución;
n) Elaborar políticas respecto al acceso y uso de Internet; y adquisición de equipos informáticos, considerando reglamentos y políticas institucionales;
o) Mantener versiones de la estructura de red; y,
p) Las demás actividades inherentes a mantenimiento de equipo informático, Redes e

Internet.

Artículo 11. *De la Gestión de Desarrollo.-* La gestión de desarrollo estará a cargo de un Especialista de Tecnología de Información y Comunicación el mismo que será responsable de proponer proyectos para implementar o mejorar los sistemas de información de la Institución, y coordinar las actividades relacionadas a:

a. Analizar, desarrollar, implementar y administrar sistemas de información según las necesidades de la Universidad;
b. Participar en el diseño de bases de datos y gestión web de la Universidad;
c. Desarrollar, implementar y administrar el sistema integrado de información universitaria;
d. Coordinar la implementación de sistemas informáticos en las unidades académicas y administrativas de la Institución;
e. Realizar el mantenimiento y actualizaciones de los sistemas existentes;
f. Presentar informes que por la naturaleza de sus funciones, solicite el jefe inmediato u otras autoridades de la Universidad;
g. Monitorear y supervisar el funcionamiento del sistema integrado de la institución;
h. Realizar las pruebas necesarias de software para asegurar la calidad del mismo;
i. Desarrollar, implementar, administrar y monitorear el Sitio Web de la Universidad;
j. Coordinar la publicación de información en el portal Web;
k. Participar en la adquisición de equipos, bienes y servicios informáticos que cumplan con los requerimientos y necesidades de la Institución;
l. Documentar técnicamente el software desarrollado;
m. Generar manuales técnicos y de usuario de todos los sistemas desarrollados;
n. Elaborar políticas respecto al desarrollo de software institucional, considerando normativas y estándares internacionales, reglamentos y políticas institucionales;
o. Llevar la documentación de versiones del software desarrollado; y,
p. Las demás actividades inherentes a desarrollo de software para la Universidad Técnica de Ambato.

Artículo 12. *De la Gestión de Seguridades.-* La gestión de seguridades estará a cargo de un Especialista de Tecnología de Información y Comunicación el mismo que será responsable de proponer proyectos para implementar o mejorar la seguridad de las plataformas tecnológicas de la Universidad Técnica de Ambato, y coordinar las actividades relacionadas a:

a. Analizar, diseñar e implementar los esquemas de seguridad de las plataformas tecnológicas de la Universidad;
b. Registrar y controlar las incidencias de seguridad presentadas, gestión de indicadores de eficiencia;
c. Evaluación técnica de nuevos productos y tecnologías, análisis de mercado inherentes a la seguridad;
d. Elaboración de propuestas y políticas de mejora de Seguridad Informática.

UNIVERSIDAD TÉCNICA DE AMBATO * CONSEJO UNIVERSITARIO *

considerando normativas y estándares internacionales, reglamentos y políticas institucionales;

e. Administrar y gestionar el funcionamiento de las plataformas virtuales, que maneje la Dirección;
f. Realizar pruebas y procedimientos de seguridad informática;
g. Crear usuarios y roles; para subdominios y bases de datos, y llevar la documentación correspondiente, garantizando la seguridad respectiva;
h. Crear y actualizar claves a docentes, administrativos, trabajadores y estudiantes;
i. Presentar informes que por la naturaleza de sus funciones, solicite el jefe inmediato u otras autoridades de la Universidad;
j. Elaborar y ejecutar planes o procedimientos de respaldo y recuperación de las bases de datos institucionales;
k. Monitorear frecuentemente el equipamiento (hardware, software y aplicaciones) que contienen las bases de datos institucionales con el fin de evitar posibles pérdidas de la información;
l. Coordinar con el Área de Redes para la administración, configuración y monitoreo del Firewall Institucional;
m. Participar en la adquisición de equipos, bienes y servicios informáticos que cumplan con los requerimientos y necesidades de la Institución;
n. Llevar un registro de versiones de documentos relacionados a la seguridad informática; y,
o. Demás actividades inherentes a la seguridad de la información de la Universidad Técnica de Ambato.

Artículo 13. *Funciones de la Gestión de Base de Datos.-* La gestión de base de datos estará a cargo de un Especialista de Tecnología de Información y Comunicación el mismo que será responsable de proponer proyectos para implementar o mejorar las bases de datos de la Universidad Técnica de Ambato, y coordinar las actividades relacionadas a:

a. Analizar, desarrollar, implementar y administrar bases de datos según las necesidades de la Universidad;
b. Participar en el análisis, diseño e implementación de sistemas de información y gestión Web de la Universidad;
c. Administrar y actualizar el sistema de replicación de las bases de datos de la institución;
d. Colaborar en la ejecución de los planes y procedimientos de respaldo y recuperación de las bases de datos institucionales;
e. Crear y modificar los planes de mantenimiento en los servidores que gestionen bases de datos;
f. Realizar el mantenimiento y las actualizaciones de bases de datos existentes;
g. Monitorear y supervisar permanentemente el adecuado funcionamiento de los servidores de base de datos de la institución;

UNIVERSIDAD TECNICA DE AMBATO * CONSEJO UNIVERSITARIO *

h. Realizar las pruebas de integridad, seguridad y rendimiento de las bases de datos;
i. Presentar informes que por la naturaleza de sus funciones, solicite el jefe inmediato u otras autoridades de la Universidad;
j. Documentar técnicamente las bases de datos existentes;
k. Apoyar con la capacitación o entrenamiento de los usuarios de los sistemas;
l. Participar en la adquisición de equipos, bienes y servicios informáticos que cumplan con los requerimientos y necesidades de la Institución;
m. Llevar la documentación adecuada de versiones o cambios realizados en las bases de datos; y,
n. Las demás actividades inherentes al análisis, diseño, implementación y administración de las bases de datos de la Universidad Técnica de Ambato.

CAPITULO III
DE LOS PROCESOS

Artículo 14. *De los Procesos de Gestión de Desarrollo.-* Los procesos de Gestión de Desarrollo estarán enmarcados siguiendo técnicas y modelos actuales de ingeniería y arquitectura del software, los mismos que cubran actividades de análisis, diseño, desarrollo, implementación, pruebas y mantenimiento del software.

Artículo 15. *De los Procesos Gestión de Recursos de Producción, Redes y Mantenimiento.-* Los procesos de Gestión de Recursos de Producción, Redes y Mantenimiento seguirán estándares de calidad y modelos de comunicación necesarios para garantizar el acceso a los servicios TICs que brinda la institución, así como en el plan de mantenimiento de la Universidad para asegurar condiciones operativas en los equipos de cómputo.

Artículo 16. *De los Procesos de Gestión de Seguridades.-* Los procesos de Gestión de Seguridades se enmarcaran en estándares o normas internacionales de seguridad de acuerdo a las necesidades de la Institución.

Artículo 17. *De los Procesos de Gestión de Base de Datos.-* Los procesos de Gestión de Base de Datos deberán seguir estándares y normas de calidad para creación de base de datos, modelado de datos y mantenimiento de los mismos.

Artículo 18. *De los Responsable de la coordinación de los procesos y de la gestión.-* Los procesos de gestión y las funciones de cada uno de ellos serán de responsabilidad del Director de la Unidad, quien deberá asignarlas a los funcionarios correspondientes y además tomar todas las medidas administrativas incluso las correctivas, que sean del caso para garantizar su seguimiento y cumplimiento.

DISPOSIONES GENERALES

PRIMERA.- De conformidad con el artículo 20 del Código Civil, el uso del genérico en las

UNIVERSIDAD TECNICA DE AMBATO * CONSEJO UNIVERSITARIO *

diferentes disposiciones de este Reglamento. comprende tanto al género masculino como al género femenino.

SEGUNDA.- Con la expedición del presente Reglamento, quedan derogado el REGLAMENTO PARA EL FUNCIONAMIENTO DEL CENTRO DE INFORMÁTICA DE LA UTA, aprobado en segunda y definitiva mediante resolución 617-86-CU-P, del 14 de octubre de 1986, y demás cuerpos normativos internos, resoluciones y disposiciones que le contrapongan.

DISPOSICIÓN TRANSITORIA.-

PRIMERA.- El Director de la Dirección de Tecnologías de la Información y Comunicación tomará las acciones requeridas para que los procesos que se encuentran ejecutando en la referida Unidad se ajusten a los lineamientos determinados en este Reglamento, en el menor tiempo posible.

SEGUNDA.- El Director de la Dirección de Tecnologías de la Información y Comunicación, remitirá al Máximo Organismo Universitario, los instrumentos legales y políticas que sean necesarias para la adecuada aplicación del presente Reglamento en el menor tiempo posible.

DISPOSICIÓN FINAL.- El presente Reglamento entrará en vigencia una vez que el Honorable Consejo Universitario lo apruebe en Segunda y definitiva.

Dado en la Sala de Sesiones del H. Consejo Universitario de la Universidad Técnica de Ambato, a los veintiún días del mes de junio de dos mil dieciséis.

Dr. MSc. Galo Naranjo López
Presidente del H. Consejo Universitario
UNIVERSIDAD TÉCNICA DE AMBATO

Ab. MSc. José Romo Santana
Secretario del H. Consejo Universitario
UNIVERSIDAD TÉCNICA DE AMBATO

Certifico.- Que la aprobación y expedición del "**REGLAMENTO DE LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN- DITIC- DE LA UNIVERSIDAD TÉCNICA DE AMBATO**"; fue discutido y aprobado en primera en sesión ordinaria del veintitrés de diciembre de dos mil trece, mediante Resolución 2470-CU-P-2013, y en segunda y definitiva en sesión ordinaria del veintiuno de junio de dos mil dieciséis, con Resolución 1254-CU-P-2016.

Ab. MSc. José Romo Santana
Secretario del H. Consejo Universitario
UNIVERSIDAD TÉCNICA DE AMBATO

Universidad Técnica de Ambato
Consejo Universitario

Av. Colombia 02-11 y Chile (Cdla. Ingahurco) - Teléfonos: 593 (03) 2521-081 / 2822960 - Fax: 2521-084
Ambato – Ecuador

Por reunir los requisitos de Ley, ejecútese y publíquese el "**REGLAMENTO DE LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN –DITIC- DE LA UNIVERSIDAD TÉCNICA DE AMBATO**", en el sitio web de la Institución; Ambato, veintiuno de junio de dos mil dieciséis.

Dr. MSc. Galo Naranjo López
Presidente del H. Consejo Universitario
UNIVERSIDAD TÉCNICA DE AMBATO

UNIVERSIDAD TÉCNICA DE AMBATO
PRESIDENCIA
CONSEJO UNIVERSITARIO

En cumplimiento a la orden impartida por el Doctor MSc. Galo Naranjo López, Presidente del Honorable Consejo Universitario de la Universidad Técnica de Ambato, se publicó en el sitio web de la Institución el "**REGLAMENTO DE LA DIRECCIÓN DE TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN –DITIC- DE LA UNIVERSIDAD TÉCNICA DE AMBATO**"; Ambato, cuatro de julio de dos mil dieciséis.

Ab. MSc. José Romo Santana
Secretario del H. Consejo Universitario
UNIVERSIDAD TÉCNICA DE AMBATO

UNIVERSIDAD TÉCNICA DE AMBATO
CONSEJO UNIVERSITARIO

Anexo B

MÉTRICAS DE CALIDAD EN USO DEFINIDA POR LA NORMA ISO/IEC 9126 (PDTR 9126-4)

8.1 Effectiveness metrics

Effectiveness metrics assess whether the tasks performed by users achieve specified goals with accuracy and completeness in a specified context of use. They do not take account of how the goals were achieved, only the extent to which they were achieved (see E.2.1.2).

Table 8.1 Effectiveness metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	12207 reference	Target audience
Task effectiveness	What proportion of the goals of the task is achieved correctly?	User test	$M1 = [1-\Sigma A_i]$ A_i= proportional value of each missing or incorrect component in the task output	0<= M1 <=1 The closer to 1.0 the better.	—	A= proportion	Operation (test) report User monitoring record	6.5 Validation 5.3 Qualification testing 5.4 Operation	User Human interface designer
NOTE Each potential missing or incomplete component is given a weight A_i based on the extent to which it detracts from the value of the output to the business or user. (If the sum of the weights exceed 1, the metric is normally set to 0, although this may indicate negative outcomes and potential safety issues.) (See for example G.3.1.1.) The scoring scheme is refined iteratively by applying it to a series of task outputs and adjusting the weights until the measures obtained are repeatable, reproducible and meaningful.									
Task completion	What proportion of the tasks is completed?	User test	X = A/B A = number of tasks completed B = total number of tasks attempted	0<= X <=1 The closer to 1.0 the better.	Ratio	A = Count B = Count X = Count/Count	Operation (test) report User monitoring record	6.5 Validation 5.3 Qualification testing 5.4 Operation	User Human interface designer
NOTE This metric can be measured for one user or a group of users. If tasks can be partially completed the Task effectiveness metric should be used..									
Error frequency	What is the frequency of errors?	User test	X = A/T A = number of errors made by the user T= time or number of tasks	0<= X The closer to 0 the better.	Absolute	A = Count	Operation (test) report User monitoring record	6.5 Validation 5.3 Qualification testing 5.4 Operation	User Human interface designer
NOTE This metric is only appropriate for making comparisons if errors have equal importance, or are weighted.									

8.2 Productivity metrics

Productivity metrics assess the resources that users consume in relation to the effectiveness achieved in a specified context of use. The most common resource is time to complete the task, although other relevant resources could include the user's effort, materials or the financial cost of usage.

Table 8.2 Productivity metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	12207 reference	Target audience
Task time	How long does it take to complete a task?	User test	X = Ta Ta = task time	0<= X The smaller the better.	Interval	T= Time	Operation (test) report User monitoring record	6.5 Validation 5.3 Qualification testing 5.4 Operation	User Human interface designer
Task efficiency	How efficient are the users?	User test	X = M1 / T M1 = task effectiveness T = task time	0<= X The larger the better.	—	T= Time X= proportion/ time	Operation (test) report User monitoring record	6.5 Validation 5.3 Qualification testing 5.4 Operation	User Human interface designer

NOTE 1 Task efficiency measures the proportion of the goal achieved for every unit of time. Efficiency increases with increasing effectiveness and reducing task time. It enables comparisons to be made, for example between fast error-prone interfaces and slow easy interfaces (see for example F.2.4.4)..

NOTE 2 If Task completion has been measured, task efficiency can be measured as Task completion/task time. This measures the proportion of users who were successful for every unit of time. A high value indicates a high proportion of successful users in a small amount of time.

Economic productivity	How cost-effective is the user?	User test	X = M1 / C M1 = task effectiveness C = total cost of the task	0<= X The larger the better.	—	C=value X= proportion/ value	Operation (test) report User monitoring record	6.5 Validation 5.3 Qualification testing 5.4 Operation	User Human interface designer

NOTE Costs could for example include the user's time, the time of others giving assistance, and the cost of computing resources, telephone calls, and materials

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	12207 reference	Target audience
Productive proportion	What proportion of the time is the user performing productive actions?	User test	X = Ta / Tb Ta = productive time = task time - help time - error time - search time Tb = task time	0<= X <=1 The closer to 1.0 the better.	Absolute	Ta=Time Tb=Time X= Time/ Time	Operation (test) report User monitoring record	6.5 Validation 5.3 Qualification testing 5.4 Operation	User Human interface designer
NOTE This metric requires detailed analysis of a videotape of the interaction (see Macleod M, Bowden R, Bevan N and Curson I (1997) The MUSiC Performance Measurement method, Behaviour and Information Technology, 16, 279-293.)									
Relative user efficiency	How efficient is a user compared to an expert?	User test	Relative user efficiency X = A / B A = ordinary user's task efficiency B = expert user's task efficiency	0<= X <=1 The closer to 1.0 the better.	Absolute	X = proportion/ proportion	Operation (test) report User monitoring record	6.5 Validation 5.3 Qualification testing 5.4 Operation	User Human interface designer

NOTE The user and expert carry out the same task. If the expert was 100 % productive, and the user and expert had the same task effectiveness, this metric would give a similar value to the Productive proportion.

8.3 Safety metrics

Safety metrics assess the level of risk of harm to people, business, software, property or the environment in a specified context of use. It includes the health and safety of the both the user and those affected by use, as well as unintended physical or economic consequences.

Table 8.3 Safety metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	12207 reference	Target audience
User health and safety	What is the incidence of health problems among users of the product?	Usage statistics	X = 1-A / B A = number of users reporting RSI B = total number of users	0<= X <=1 The closer to 1 the better.	Absolute	A = count B = count X = count/ count	Usage monitoring record	5.4 Operation	User Human interface designer
NOTE Health problems can include Repetitive Strain Injury, fatigue, headaches, etc.									
Safety of people affected by use of the system	What is the incidence of hazard to people affected by use of the system?	Usage statistics	X = 1-A / B A = number of people put at hazard B = total number of people potentially affected by the system	0<= X <=1 The closer to 1 the better.	Absolute	A = count B = count X = count/ count	Usage monitoring record	5.3 Qualification testing 5.4 Operation	User Human interface designer Developer
NOTE An example of this metric is Patient Safety, where A = number of patients with incorrectly prescribed treatment and B = total number of patients									
Economic damage	What is the incidence of economic damage?	Usage statistics	X = 1-A/ B A = number of occurrences of economic damage B = total number of usage situations	0<= X <=1 The closer to 1 the better.	Absolute	A = count B = count X = count/ count	Usage monitoring record	5.4 Operation	User Human interface designer Developer

NOTE This can also be measured based on the number of occurrences of situations where there was a risk of economic damage

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	12207 reference	Target audience
Software damage	What is the incidence of software corruption?	Usage statistics	X = 1-A / B A = number of occurrences of software corruption B = total number of usage situations	0<= X <=1 The closer to 1 the better.	Absolute	A = count B = count X = count/ count	Usage monitoring record	5.4 Operation	User Human interface designer Developer

NOTE 1 This can also be measured based on the number of occurrences of situations where there was a risk of software damage

NOTE 2 Can also be measured as X = cumulative cost of software corruption/usage time.

8.4 Satisfaction metrics

Satisfaction metrics assess the user's attitudes towards the use of the product in a specified context of use.

NOTE: Satisfaction is influenced by the user's perception of properties of the software product (such as those measured by external metrics) and by the user's perception of the efficiency, productivity and safety in use.

Table 8.4 Satisfaction metrics

Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Interpretation of measured value	Metric scale type	Measure type	Input to measurement	12207 reference	Target audience
Satisfaction scale	How satisfied is the user?	User test	X = A/B A = questionnaire producing psychometric scales B = population average	0<X the larger the better	Ratio.	A= Count X= Count	Operation (test) report User monitoring record	6.5 Validation 5.3 Qualification testing 5.4 Operation	User Human interface designer Developer
NOTE Examples of psychometric questionnaires can be found in F.3.									
Satisfaction questionnaire	How satisfied is the user with specific software features?	User test	$X' = \Sigma(A_i)/n$ A_i = response to a question n = number of responses	Compare with previous values, or with population average	Ord.	A= Count X= Count	Operation (test) report User monitoring record	6.5 Validation 5.3 Qualification testing 5.4 Operation	User Human interface designer Developer
NOTE If the questionnaire items are combined to give an overall score, they should be weighted, as different questions may have different importance.									
Discretionary usage	What proportion of potential users choose to use the system?	Observation of usage	X = A/B A= number of times that specific software functions/applications/systems are used B = number of times they are intended to be used	0<=X<=1 The closer to 1 the better	Ratio	A = Count B = Count X = Count/Count	Operation (test) report User monitoring record	6.5 Validation 5.3 Qualification testing 5.4 Operation	User Human interface designer
NOTE This metric is appropriate when usage is discretionary.									