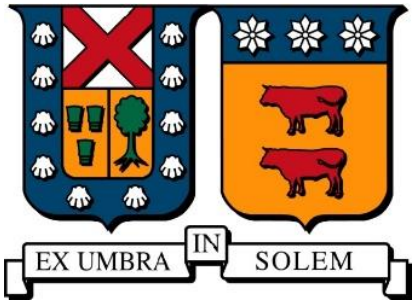
UNIVERSIDAD TECNICA FEDERICO SANTA MARIA
DEPARTAMENTO DE INFORMATICA
SANTIAGO – CHILE



ANÁLISIS DE LA CALIDAD DE UNA APLICACIÓN PARA EL DESARROLLO DE UN PLAN DE MEJORA DE ACUERDO A LAS NORMAS ISO

RAFAEL ALEJANDRO VELASCO BALAZS

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO DE EJECUCIÓN EN INFORMÁTICA.

MARCELLO VISCONTI Z. – PROFESOR GUÍA

PEDRO GODOY B. – PROFESOR CORREFERENTE

NOVIEMBRE DEL 2017

Agradecimientos

Me gustaría agradecer a varias personas, ya que sin ellas no me habría sido posible llegar tan lejos.

A los profesores de la Universidad por sus enseñanzas en estos años, y en especial a nuestro Jefe de Carrera, por su ayuda, su buena disposición y sus consejos.

A mi familia, por su apoyo incondicional en las decisiones que tomé.

Gracias a mis amigos y amigas, dentro y fuera de la universidad, de quienes aprendí a trabajar en equipo y a convivir con diferentes tipos de personas.

Resumen

En la actualidad, el software es un producto con su propio mercado, y como todo producto, está sujeto a estándares de calidad que se deben cumplir para mantenerse en la industria y satisfacer las necesidades de los clientes.

El software, una vez terminado, requiere seguir mejorando para mantener satisfechos a los clientes, para esto, los desarrolladores requieren de un plan de mejora antes de poder pensar en lanzar una nueva versión, ya que al desarrollar una nueva versión sin contar con un plan puede no garantizar la solución a todos los requerimientos del cliente, lo que resulta en una gran pérdida de tiempo y recursos.

Al desarrollar un plan de mejora los desarrolladores se aseguran de identificar todas las posibles áreas problemáticas, estas pueden surgir mediante feedback de los usuarios, auditorias, encuestas, entre otros; pero como tema para esta memoria, para la realización del plan de mejora, se realizará un análisis de calidad, de acuerdo a los estándares internacionales de la ISO 9126-1, para identificar dichas áreas.

La ISO es reconocida internacionalmente por sus estándares de calidad para las empresas y sus productos, pero es poco conocido que la ISO tiene estándares específicos para la calidad del software.

Por consiguiente, para este trabajo propuesto, además se pretende dar a conocer las reglas y los postulados de los modelos de la ISO para la calidad del software, de modo que se pueda comprender de manera general los conceptos que se pretender presentar.

Respecto al producto al que se le hará el análisis de calidad, primeramente, se pretende dar a conocer al cliente y el contexto en que fue solicitada la aplicación, describiendo los requerimientos que solicitaba el cliente y con qué recursos se cuentan, para posteriormente detallar el proceso de desarrollo y evolución de la aplicación hasta

llegar al producto en sí. Durante el proceso se pretende utilizar lenguaje técnico y metodologías especializadas para el desarrollo de software, explicando todo para la comprensión del lector.

Abstract

Currently, the software is a product with its own market, and like all products, is subject to quality standards that must be met to stay in the industry and meet the needs of customers.

The software, once completed, requires further improvement to keep customers satisfied, for this, developers require an improvement plan before you can think about launching a new version, as to develop a new version without a plan it can't guarantee the solution to all customer requirements, resulting in a great waste of time and resources.

In developing an improvement plan developers are careful to identify all possible areas of improvement, these can arise through user feedback, audits, surveys, among others; but as the theme for this memory, to carry out the improvement plan, the quality analysis according to international standards, ISO 9126-1, it will be made to identify these areas for improvement.

ISO is recognized internationally for its quality standards for companies and their products, but is little known that the ISO has specific standards for software quality.

Therefore, for this proposed work also it aims to publicize the rules and principles of the ISO models for software quality, so you can generally understand the concepts trying to present.

Regarding the product to which it will make quality analysis, first, it is intended to introduce the customer and the context in which it was requested the application, describing the requirements requested by the customer and what resources are counted, to further detail the process development and evolution of the application until the product itself was finished. During the process, it aims to use technical language and

specialized methodologies for software development, explaining everything to the reader's understanding.

Glosario

[1]. Software: Es todo programa, conjunto de programas o de rutinas que corren dentro de un computador u otro dispositivo informático, que le permite realizar tareas determinadas.

[2]. Hardware: Es el conjunto de partes físicas que componen un computador u otros dispositivos informáticos.

[3]. ISO 9001:2000: Esta Norma Internacional especifica los requisitos para un sistema de gestión de calidad donde una organización necesita demostrar su capacidad para proporcionar de forma coherente productos que satisfagan los requisitos del cliente y tiene como objetivo mejorar la satisfacción del cliente a través de la aplicación eficaz del sistema, y a su vez los procesos para la mejora continua de éste y el aseguramiento de la conformidad con los clientes. Todos los requerimientos de este estándar son genéricos y tienen la intención de ser aplicables por todo tipo de organizaciones, independiente del tipo, tamaño o producto que proveen.

[4]. IEC: La International Electrotechnical Commission, es la organización mundial líder que prepara y publica estándares internacionales para todo lo que es eléctrico, electrónico y relacionado con la tecnología.

[5]. JTC1: Es el comité donde los expertos se reúnen para desarrollar estándares de Información global y Tecnología de la Comunicación (TIC), para aplicaciones comerciales y de consumo.

[6]. SQuaRE: Siglas en inglés para “Requerimientos de Calidad y Evaluación para Sistemas y Software”.

[7]. CRM: Customer Relationship Management, esta categoría de software se identifica en estar orientada a ayudar al negocio en el manejo de los datos de sus clientes y la interacción con ellos.

[8]. Modelo de Prototipos: En el desarrollo de software, se refiere a un tipo de desarrollo evolutivo, donde el prototipo debe ser desarrollado en poco tiempo y sin usar muchos recursos, el cual es evaluado por el cliente para una retroalimentación, esto se hace periódicamente reduciendo así los riesgos y la incerteza en el desarrollo.

[9]. Beta Test: Prueba previa al lanzamiento de un producto. En el caso del Software, una aplicación se somete al uso de usuarios de testeo, los cuales utilizan la aplicación de todas las formas posibles, con el fin de comprobar el correcto funcionamiento y respuesta frente a errores que esta posee.

Índice

Introducción

En este documento se abordará el proceso de desarrollar un Plan de Mejora para una aplicación desde el punto de vista de la calidad, siendo esta el conjunto de cualidades que algo posee y que permite juzgar su valor respecto a sus similares.

Intentando a partir de esto demostrar la importancia que la calidad del Software tiene en la industria, ya que a pesar de su existencia son muy pocas las empresas nacionales de Software que creen relevante certificar la calidad de su producto. Esto ocurre porque generalmente, las personas que toman las decisiones en las empresas tienen la mentalidad de que certificar la calidad de su producto es solo un gasto adicional, mientras que no se dan cuenta de las ventajas que la certificación les puede brindar, entre todas, la más importante, es la de mantener satisfechos a los clientes con su producto.

Para lograr esto, se darán a conocer algunos de los conceptos existentes en la industria, como la “Organización internacional de Estandarización” (ISO), “Análisis de calidad” y “Plan de Mejora”; además de describir el proceso y las condiciones en las que fue creada la aplicación.

El fin de lo anteriormente mencionado, es asegurar que el lector entienda todos los conceptos abordados durante el desarrollo de este informe, sin importar si este posee o no conocimientos previos del tema.

Además de los conceptos anteriormente mencionados, se presentarán nociones de lo que es el trato con el cliente y como realizar e implementar un plan de trabajo para el desarrollo del proyecto.

En el caso particular de este trabajo, se propone realizar un análisis de calidad a una aplicación ya desarrollada, de modo que se pueda proponer un plan de mejora e intentar implementarlo, aunque sea parcialmente; para poder solucionar cualquier inconveniente que se pueda encontrar en la aplicación.

Capítulo 1:

Antecedentes.

1.1. Definición del problema

El *Software* [1], en los inicios de la computación fue menospreciado, ya que sus capacidades eran muy limitadas; teniendo mayor relevancia el *hardware* [2], porque su principal función era el procesamiento de una gran cantidad de datos. En la década de 1980 fue donde las organizaciones se dieron cuenta que al ritmo en que el hardware evolucionaba, el software existente no lograba explotar todo su potencial, por lo que el desarrollo de software comenzó a tomar importancia, y los desarrolladores vieron la oportunidad de convertir al Software en un producto más de la industria, lo que generó una gran competencia en el desarrollo y mejoramiento de las aplicaciones.

En la actualidad el Software es un producto en un mercado cada vez más competitivo, cada año surgen emprendedores que quieren introducir su Software en el mercado, teniendo esto en cuenta resulta fundamental que algo asegure que un producto puede competir con los mejores del mercado.

El cliente a la hora de decidir en qué producto invertir, se sentirá más seguro de hacerlo en un producto que pueda demostrar ser mejor que los de su competencia, y la calidad del producto es uno de los factores que más les interesa a los consumidores; además de que la calidad ayuda a la empresa a forjar su imagen en el mercado en que compite.

Se encuentran documentados los casos de las aplicaciones que salen del mercado a lo largo del tiempo, en algunos casos estos programas desaparecen por problemas públicos o que el producto de la competencia es mejor; pero son muchos los casos de productos que al lanzar una nueva versión se encuentran con que sus usuarios previos no están conformes y deciden optar por la competencia; dejando a los desarrolladores de dichas aplicaciones con dos opciones, invertir más para corregir los problemas e intentar recuperar a sus clientes o, en el peor de los casos, las pérdidas son tantas que se ven obligados a descartar la aplicación.

Y siendo que la certificación no es obligación para las empresas, es por esta razón se les exige a los programadores implementar técnicas de trabajo que permitan disminuir al máximo los posibles problemas que puedan surgir al desarrollar una aplicación, como ejemplo, están las llamadas Metodologías Agiles, las que se basan en el desarrollo iterativo e incremental.

Para el caso de esta memoria, como la aplicación ya se encuentra desarrollada, se intentará demostrar que durante su producción se aplicaron estas metodologías; y a su resultado, se le realizará el análisis de calidad.

1.2. Objetivos

El objetivo principal de esta memoria es el desarrollar e intentar aplicar, aunque sea de manera parcial, un plan de mejora para una aplicación a partir de su análisis de calidad. Siguiendo el procedimiento formal, el que será definido más adelante; y los pasos correspondientes del proceso.

Además, se quiere demostrar la importancia de la calidad del Software dentro del mercado que actualmente está establecido.

Y finalmente, se busca demostrar la aplicación de los conocimientos adquiridos en un ámbito profesional y la utilidad que estos tienen.

Capítulo 2:

Estado de la practica en calidad

2.1 Controlar la calidad del Software

Hoy en día la calidad es importante para poder satisfacer a los clientes que piden un buen producto (software), y cada vez hay mayor competitividad en la industria de la informática, lo cual hace que cada uno de los desarrolladores busque opciones del cómo poder crear software de calidad y en ello se han constituido desde hace mucho tiempo atrás, los estándares que hoy en día rigen en torno al desarrollo de aplicaciones, cumpliendo con sus normas y parámetros en aras de conseguir la ansiada calidad.

La calidad del software se puede definir como el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. Está se puede medir y su exigencia puede variar dependiendo del sistema o entorno donde se ejecute el programa. Por ejemplo, un software diseñado para el control de maquinaria médica para operaciones debe funcionar al nivel de "cero fallas"; en cambio, un software hecho para ejecutarse una vez no requiere el mismo nivel de calidad; mientras que un software para ser usado durante un largo período de tiempo, requiere ser confiable, sostenible y flexible para disminuir los costos de mantenimiento y perfeccionamiento durante el tiempo de explotación.

La calidad de un software puede medirse después de su desarrollo, pero esto conlleva el riesgo de un costo adicional si se detectan problemas derivados de un mal diseño. Para esto es importante considerar la obtención de la calidad como su control durante todo el ciclo de vida del software.

La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

La política establecida debe estar sustentada sobre tres principios básicos: tecnológico, administrativo y ergonómico.
El principio tecnológico define las técnicas a utilizar en el proceso de desarrollo del software.
El principio administrativo contempla las funciones de planificación y control del desarrollo del software, así como la organización del ambiente o centro de ingeniería de software.
El principio ergonómico define la interfaz entre el usuario y el ambiente automatizado.

La adopción de una buena política contribuye en gran medida a lograr la calidad del software, pero no la asegura. Para el aseguramiento de la calidad es necesario su control o evaluación.

Muchos autores coinciden en que el software posee determinados índices medibles que son las bases para la calidad, el control y el perfeccionamiento de la productividad.

Para controlar la calidad del software es necesario, ante todo, definir los parámetros, indicadores o criterios de medición. Una vez seleccionados, se debe establecer el proceso de control, que requiere los siguientes pasos:

- Definir el software que va a ser controlado, clasificación por tipo de acuerdo con los estándares establecidos para el desarrollo del software.
- Seleccionar una medida que pueda ser aplicada al objeto de control.
- Crear o determinar los métodos de valoración de los indicadores, métodos manuales como cuestionarios o encuestas estándares.
- Definir las regulaciones organizativas para realizar el control, quiénes participan en el control de la calidad.

2.2 Los estándares ISO

La Organización Internacional de Normalización (ISO), se crea en 1947, después de que representantes de 25 países se reunieran para crear una organización internacional con el objetivo de “facilitar la unificación y coordinación de estándares industriales internacional” [1]. Estas normas abarcan diversos aspectos que conlleva el desarrollo industrial, por mencionar solo algunos: Gestión de la calidad, gestión energética, gestión ambiental, responsabilidad social, gestión de riesgos, códigos de lenguaje, seguridad de la información, entre otros.

En la actualidad ISO cuenta con la participación de 163 países y ha desarrollado más de 19.500 normas internacionales dirigidas al mundo empresarial. Estas normas responden a necesidades planteadas por el mercado, tanto empresas como consumidores y son desarrolladas por un comité compuesto por expertos, quienes, basándose en sus conocimientos y experiencias de éxito, componen la norma, que deberá ser votada y aprobada por los miembros de la ISO.

Las normas son de aplicación voluntaria, ya que la ISO es una entidad no gubernamental, es decir, no posee la autoridad para exigirlas; pero, en la actualidad, la importancia de las normas, sobre todo, aquellas relacionadas a la calidad de un producto, si pueden convertirse en un requisito para que una empresa se mantenga en una posición competitiva dentro del mercado, por lo tanto las normas, de una u otra manera, están presentes en el desarrollo industrial de cada país asociado a la organización, por lo que se asocia con casi cada producto desarrollado en la actualidad que sea reconocido internacionalmente.

Dentro de la organización de la ISO existen muchas normas relacionadas a la calidad de un producto, siendo la más conocida el conjunto de normas *ISO 9001:2000* [3]; pero específicamente, para el desarrollo de software, la ISO, en conjunto con la Comisión Electrotécnica Internacional (*IEC* [4]), constituyendo un Comité Técnico Conjunto (*JTC1* [5]), quienes en 1985 comenzaron a desarrollar el primer conjunto de normas para regular la calidad del software, este es el ISO 916.

2.2.1 ISO/IEC 9126

Esta norma establece un estándar internacional para la evaluación de la calidad del software, este se divide en cuatro partes: realidad, métricas internas, métricas externas y calidad en las métricas de uso y expendido. Destacando, principalmente, la primera parte de este estándar (ISO 9126-1), esta clasifica la calidad del software en un conjunto de seis características, que se resume en la figura 1 presente a continuación:

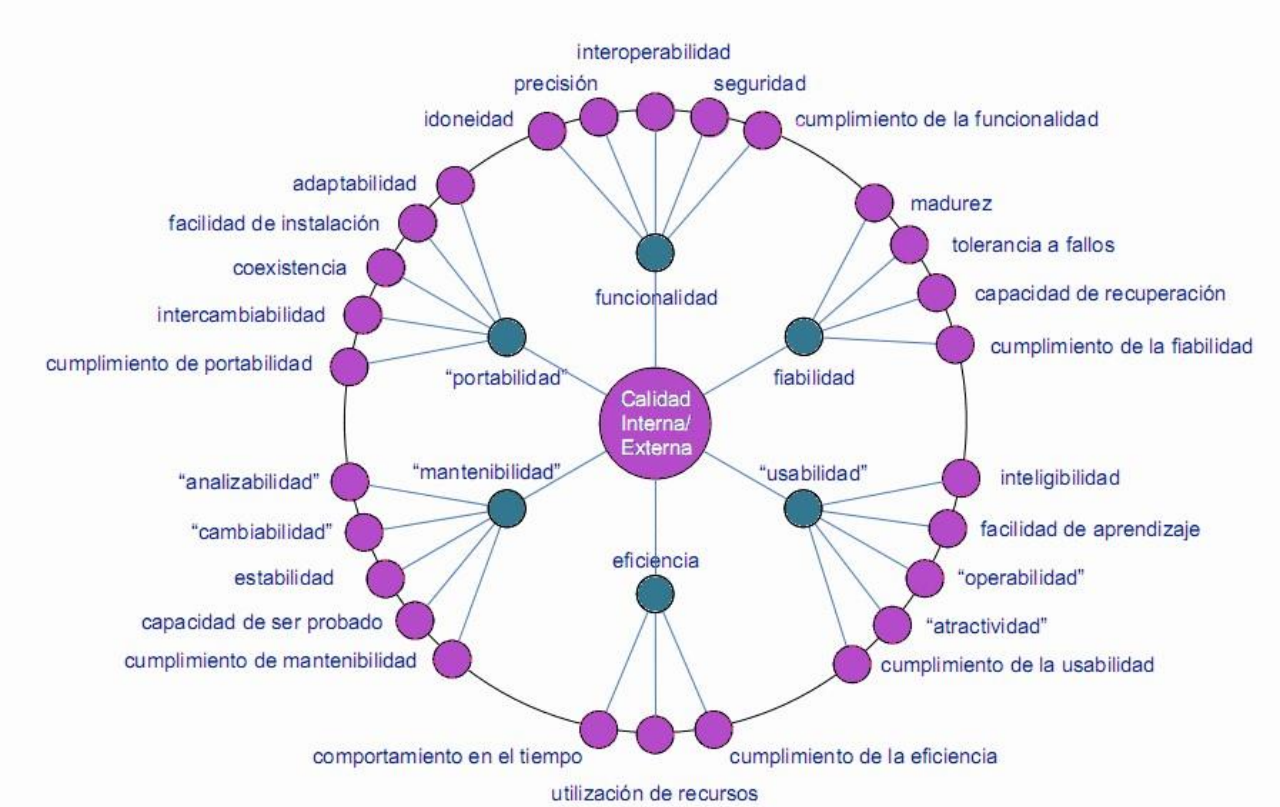


Figura 1: Este esquema muestra cómo se divide y se relacionan las partes del estándar ISO 9126-1

Funcionalidad: Un conjunto de atributos que se relacionan con la existencia de un conjunto de funciones y sus propiedades específicas. Las funciones son aquellas que satisfacen las necesidades implícitas o explícitas.

- Adecuación: Atributos del software relacionados con la presencia y aptitud de un conjunto de funciones para tareas especificadas.
- Exactitud: Atributos del software relacionados con la disposición de resultados o efectos correctos o acordados.
- Interoperabilidad: Atributos del software que se relacionan con su habilidad para la interacción con sistemas especificados.
- Seguridad: Atributos del software relacionados con su habilidad para prevenir acceso no autorizado ya sea accidental o deliberado, a programas y datos.
- Cumplimiento funcional.

Fiabilidad: Un conjunto de atributos relacionados con la capacidad del software de mantener su nivel de prestación bajo condiciones establecidas durante un período establecido.

- Madurez: Atributos del software que se relacionan con la frecuencia de falla por fallas en el software.
- Recuperabilidad: Atributos del software que se relacionan con la capacidad para restablecer su nivel de desempeño y recuperar los datos directamente afectos en caso de falla y en el tiempo y esfuerzo relacionado para ello.
- Tolerancia a fallos: Atributos del software que se relacionan con su habilidad para mantener un nivel especificado de desempeño en casos de fallas de software o de una infracción a su interfaz especificada.
- Cumplimiento de Fiabilidad: La capacidad del producto software para adherirse a normas, convenciones o legislación relacionadas con la fiabilidad.

Usabilidad: Un conjunto de atributos relacionados con el esfuerzo necesario para su uso, y en la valoración individual de tal uso, por un establecido o implicado conjunto de usuarios.

- Aprendizaje: Atributos del software que se relacionan al esfuerzo de los usuarios para reconocer el concepto lógico y sus aplicaciones.
- Comprensión: Atributos del software que se relacionan al esfuerzo de los usuarios para reconocer el concepto lógico y sus aplicaciones.
- Operatividad: Atributos del software que se relacionan con el esfuerzo de los usuarios para la operación y control del software.
- Atractividad.

Eficiencia: Conjunto de atributos relacionados con la relación entre el nivel de desempeño del software y la cantidad de recursos necesitados bajo condiciones establecidas.

- Comportamiento en el tiempo: Atributos del software que se relacionan con los tiempos de respuesta y procesamiento y en las tasas de rendimientos en desempeñar su función.
- Comportamiento de recursos: Usar las cantidades y tipos de recursos adecuados cuando el software lleva a cabo su función bajo condiciones determinadas.

Mantenibilidad: Conjunto de atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software.

- Estabilidad: Atributos del software relacionados con el riesgo de efectos inesperados por modificaciones.
- Facilidad de análisis: Atributos del software relacionados con el esfuerzo necesario para el diagnóstico de deficiencias o causas de fallos, o identificaciones de partes a modificar.
- Facilidad de cambio: Atributos del software relacionados con el esfuerzo necesario para la modificación, corrección de falla, o cambio de ambiente.
- Facilidad de pruebas: Atributos del software relacionados con el esfuerzo necesario para validar el software modificado.

Portabilidad: Conjunto de atributos relacionados con la capacidad de un sistema software para ser transferido desde una plataforma a otra.

- Capacidad de instalación: Atributos del software relacionados con el esfuerzo necesario para instalar el software en un ambiente especificado.
- Capacidad de reemplazamiento: Atributos del software relacionados con la oportunidad y esfuerzo de usar el software en lugar de otro software especificado en el ambiente de dicho software especificado.
- Adaptabilidad: Atributos del software relacionados con la oportunidad para su adaptación a diferentes ambientes especificados sin aplicar otras acciones o medios que los proporcionados para este propósito por el software considerado.
- Co-Existencia: Coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes.

Posteriormente, la ISO 9126 se ve reemplazada por una nueva propuesta, la que se plantea en 1999 y se aprueba en el año 2000; se le denomina *SQuaRE* [6], y queda bajo la nomenclatura ISO/IEC 25000

2.2.2 ISO/IEC 25000

Este conjunto de normas tiene como objetivo la creación de un marco común para evaluar la calidad del producto software. El conjunto se divide en cinco secciones principales, como se muestra a continuación en la figura 2:

Figura 2: Este esquema muestra las secciones del estándar ISO/IEC 2500, indicando a que se especifica cada una.

ISO/IEC 2500n: División de Gestión de Calidad

Las normas que forman este apartado definen todos los modelos, términos y definiciones comunes referenciados por todas las otras normas de la familia 25000. Actualmente esta división se encuentra formada por:

- ISO/IEC 25000 - Guide to SQuaRE: Contiene el modelo de la arquitectura de SQuaRE, la terminología de la familia, un resumen de las partes, los usuarios previstos y las partes asociadas, así como los modelos de referencia.
- ISO/IEC 25001 - Planning and Management: Establece los requisitos y orientaciones para gestionar la evaluación y especificación de los requisitos del producto software.

ISO/IEC 2501n: División de Modelo de Calidad

- Las normas de este apartado presentan modelos de calidad detallados incluyendo características para calidad interna, externa y en uso del producto software. Actualmente esta división se encuentra formada por:
- ISO/IEC 25010 - System and software qualitymodels: Describe el modelo de calidad para el producto software y para la calidad en uso. Esta Norma presenta las características y subcaracterísticas de calidad frente a las cuales evaluar el producto software.
- ISO/IEC 25012 - Data Qualitymodel: Define un modelo general para la calidad de los datos, aplicable a aquellos datos que se encuentran almacenados de manera estructurada y forman parte de un Sistema de Información.

ISO/IEC 2502n: División de Medición de Calidad

- Estas normas incluyen un modelo de referencia de la medición de la calidad del producto, definiciones de medidas de calidad (interna, externa y en uso) y guías prácticas para su aplicación. Actualmente esta división se encuentra formada por:
- ISO/IEC 25020 - Measurementreferencemodel and guide: Presenta una explicación introductoria y un modelo de referencia común a los elementos de medición de la calidad. También proporciona una guía para que los usuarios seleccionen o desarrollen y apliquen medidas propuestas por normas ISO.
- ISO/IEC 25021 - Qualitymeasureelements: Define y especifica un conjunto recomendado de métricas base y derivadas que puedan ser usadas a lo largo de todo el ciclo de vida del desarrollo software.
- ISO/IEC 25022 - Measurement of quality in use: Define específicamente las métricas para realizar la medición de la calidad en uso del producto.
- ISO/IEC 25023 - Measurement of system and software productquality: Define específicamente las métricas para realizar la medición de la calidad de productos y sistemas software.
- ISO/IEC 25024 - Measurement of data quality: Define específicamente las métricas para realizar la medición de la calidad de datos.

ISO/IEC 2503n: División de Requisitos de Calidad

- Las normas que forman este apartado ayudan a especificar requisitos de calidad que pueden ser utilizados en el proceso de licitación de requisitos de calidad del producto software a desarrollar o como entrada del proceso de evaluación. Para ello, este apartado se compone de:
- ISO/IEC 25030 – Quality requirements: Provee de un conjunto de recomendaciones para realizar la especificación de los requisitos de calidad del producto software.

ISO/IEC 2504n: División de Evaluación de Calidad

- Este apartado incluye normas que proporcionan requisitos, recomendaciones y guías para llevar a cabo el proceso de evaluación del producto software. Esta división se encuentra formada por:
- ISO/IEC 25040 - Evaluationreferencemodel and guide: Propone un modelo de referencia general para la evaluación, que considera las entradas al proceso de evaluación, las restricciones y los recursos necesarios para obtener las correspondientes salidas.
- ISO/IEC 25041 – Evaluation guide for developers, acquirers and independentevaluators: Describe los requisitos y recomendaciones para la implementación práctica de la evaluación del producto software desde el punto de vista de los desarrolladores, de los adquirentes y de los evaluadores independientes.
- ISO/IEC 25042 - Evaluation modules: Define lo que la Norma considera un módulo de evaluación y la documentación, estructura y contenido que se debe utilizar a la hora de definir uno de estos módulos.
- ISO/IEC 25045 - Evaluation module forrecoverability: Define un módulo para la evaluación de la subcaracterística“Recuperabilidad” (Recoverability).

La división de extensión ISO/IEC 25050 a ISO/IEC 25099 se reserva para normas o informes técnicos que aborden dominios de aplicación específicos o que puedan ser utilizados para complementar otras normas de la familia SQuaRE.

Capítulo 3:

Aplicación de los estándares en el desarrollo de Software

3.1 Desarrollo de la aplicación

Ya que la regulación del modelo de la ISO no son algo obligatorio en las empresas, nada regula que los desarrolladores de Software deban seguir la metodología que estipulan las normas; pero de forma opuesta, si no se tienen en consideración, en una empresa puede llegar el momento en que se quiera solicitar la certificación de la ISO y si el Software no cumple las normas no se podrá obtener la certificación y el producto deberá ser desarrollado nuevamente, lo que será una gran pérdida de tiempo y de recursos para la empresa.

A continuación, se describirá el proceso de desarrollo de una aplicación para una empresa, la que fue desarrollada por un estudiante; y se le aplicará un análisis de calidad de acuerdo con uno de los modelos de la ISO. Con el fin de medir la calidad de un Software desarrollado por alguien que no es un profesional y medir los conocimientos que este adquirió.

En el periodo entre Enero y Febrero del 2015, el área de la Banca Hipotecaria de la empresa “Banco de Chile” encomienda el desarrollo de una aplicación *CRM* [7] con el fin de mejorar la atención de sus clientes que solicitan Créditos Hipotecarios.
De aquí en adelante**,** se considerará como **Cliente** a la gerencia del Área de la Banca Hipotecaria. Similarmente, se considerará como **Programador** al encargado de desarrollar la aplicación.
Y en las interacciones entre ambos se usarán asignaciones con viñetas, de la siguiente manera:

- Para referirse al **Cliente**.
  - Para referirse al **Programador**.

En la primera reunión con el **Cliente**, éste dio a conocer los requerimientos de la aplicación que desean:

- Una aplicación de tipo CRM con plataforma web y programada en lenguaje PHP o similar.
- Que está les permita a los ejecutivos ver las solicitudes de crédito hipotecario.
- Que facilite la comunicación con sus clientes de la organización.
- Que se pueda asignar nuevas solicitudes a cada ejecutivo.

Estos requerimientos, primeramente, se deben clasificar; teniendo como requerimientos funcionales los siguientes:

- Que permita asignar solicitudes a los usuarios.
- Que permita a los usuarios ver el detalle de las solicitudes.
- Que permita a los usuarios comunicarse con los clientes.

Y de la misma forma, se tienen como requerimientos no funcionales los siguientes:

- Deber ser una aplicación de tipo CRM.
- Debe ser una aplicación web.
- Debe ser programada en lenguaje PHP o similar.

Teniendo en cuenta estos requerimientos y el tiempo disponible, el **Programador** opta por desarrollar la aplicación siguiendo un *Modelo de Prototipos* [8], el que se puede observar en la figura 3; para minimizar las discrepancias entre sus ideas y las del **Cliente**.

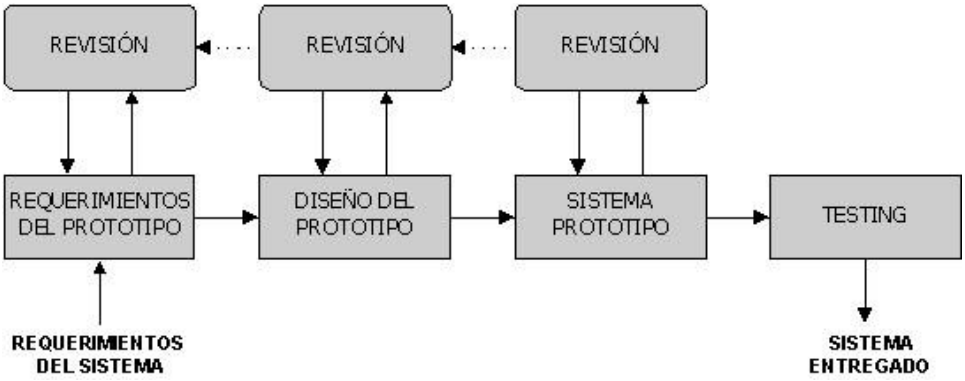


Figura 3: Este esquema muestra como es el proceso del Modelo de Prototipos.

.

Los equipos computacionales del **Cliente** funcionan en un entorno que utiliza como base el sistema operativo Windows, además ellos se encuentran protegidos a intrusiones externas físicas y virtuales, por lo que para programar la aplicación y probarla se utilizó un equipo servidor del **Cliente** que contaba con el programa XAMPP, una suite opensource que incluye un servidor Apache, las librerías de PHP, una base de datos MySQL, entre otros servicios; y se usó una versión portable del programa Notepad ++ como editor de código.

En un comienzo, se realizó un bosquejo en papel para darle una idea inicial al **Cliente** de lo captado por el **Programador**, el que se puede observar en la figura 4; en él se presentan dos ventanas de un navegador web, una para el inicio de sesión y otra ,siendo la ventana principal de la aplicación, que se encuentra dividida en dos partes, en la parte izquierda se muestra un menú y en el sector derecho, se muestra la información del usuario; además es posible ver un diagrama de secuencia del sistema del proceso de inicio de sesión.

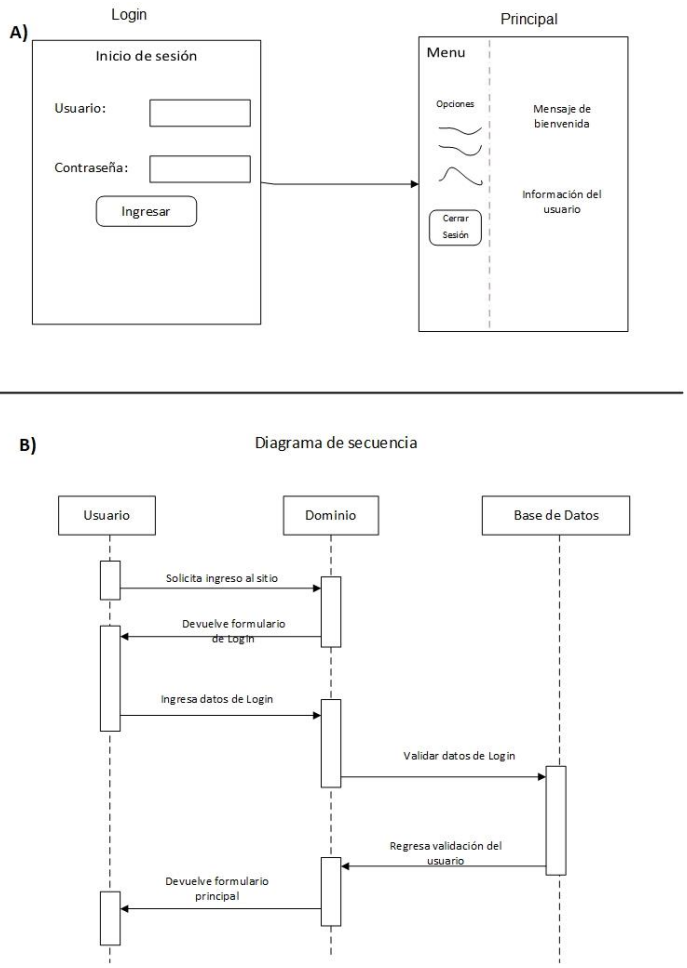


Figura 4:

A) Bosquejo básico de la interfaz de la aplicación para el proceso de inicio de sesión

B) Diagrama de secuencias que muestra la relación entre las partes para el inicio de sesión.

Al **Cliente** le parece bien la idea, pero además solicita que en la ventana principal esté el logo de la empresa, teniendo en cuenta esto, se decide dividir la ventana principal en 3 partes, agregando un sector superior donde estará ubicado el logo de la empresa.

En los días siguientes, luego de haber programado lo presentado en el bosquejo, el **Programador** le pide al **Cliente** más detalles respecto a los requerimientos de la aplicación, funcionales y no funcionales. Las que a continuación se detallarán junto a la respuesta por parte del **Cliente**:

- ¿Quiénes serían los usuarios de la aplicación?

- La aplicación será usada por los ejecutivos de la empresa, los que estarán divididos en tres grupos de usuarios:
    - Ejecutivos de sucursal: Son los usuarios que interactúan directamente con los clientes de la empresa, deben ser capaces de poder manipular las solicitudes de crédito que les son asignadas.
    - Jefes de Zona: Son usuarios responsables por los usuarios “Ejecutivos de sucursal”, deben ser capaces de poder manipular las solicitudes de crédito que llegan a la sucursal y de gestionar el trabajo realizado por los ejecutivos.
    - Administrador: Es un usuario único, el que es capaz de gestionar tanto a las solicitudes de crédito como a los demás tipos de usuarios.

- ¿Qué funciones necesitan que cumpla la aplicación?
- La aplicación debe permitir principalmente facilitar la gestión de las solicitudes de crédito que recibe la empresa, para esto es requerido que el usuario administrador pueda agregar los nuevos registros a la aplicación y más específicamente, debe poder realizar la asignación de éstos a los usuarios ejecutivos y organizarlas según el estado que estas adquieren al comunicarse los usuarios con los clientes de la organización; y además se deben poder ver los detalles de estas, por lo que se le pide al **Cliente** una plantilla con la información necesaria.

- ¿En qué sistemas tienen pensado implementar la aplicación?
- Ya que se trata de una aplicación web, se debe poder acceder a ella desde cualquier equipo de la empresa, y para esto, ser compatible con el navegador por defecto del sistema operativo Windows, Internet Explorer, que es el que utilizan los empleados de la empresa, más específicamente, la versión 7.0.

- ¿Cómo desean ingresar los nuevos registros a la base de datos?
- Pensando en que los registros son las nuevas solicitudes de crédito que recibe la organización, estas son registradas en otra aplicación con la que ya contaba la empresa, y diariamente son extraídas en un archivo de Microsoft Excel, por lo que se piensa que para llenar la base de datos de esta aplicación se empleara este archivo como base, pensando en la creación de una función que permita la carga del archivo Excel y traspasarlo a una base de datos.

Teniendo más detalles de los requerimientos, se continúa desarrollando la aplicación para que sea compatible con Internet Explorer 7, siendo esta la versión usada por el **Cliente**, finalmente se decide programar combinando los lenguajes PHP, HTML, CSS y Javascript, con una base de datos MySQL.

Para empezar, se desarrolla la aplicación pensando en las funciones para el usuario "Administrador"; al que se le muestra en la ventana principal un resumen con la cantidad total de solicitudes en las bases de datos y cuantas de estas son nuevas. Además, se desarrolla el método que le permita cargar un archivo de Microsoft Excel con las solicitudes nuevas del día desde el equipo a la base de datos; en la parte izquierda de la ventana aparece la información del usuario, su nombre y el tipo de usuario al que corresponde, además de un botón con que puede cerrar la sesión; y se crea un menú con tres opciones:

- Asignar solicitudes: Las nuevas solicitudes que ingresan a la base de datos deben ser asignadas a los diferentes usuarios "ejecutivos", para esto se usa el dato de "Región" en las solicitudes y se agrupan en cinco zonas: Norte, Centro norte, Centro, Centro Sur y Sur; una vez seleccionada la zona aparecen los usuarios "ejecutivos" de está, donde el "administrador" puede seleccionar a que usuarios asignarle solicitudes y cuantas asignarle.

- Solicitudes: Esta opción abre un submenú con cuatro secciones: Pendientes, Aceptadas, Rechazadas y Terminadas; estas representan el estado de las solicitudes en el sistema; al seleccionar una opción se carga una lista con las solicitudes de crédito con dicho estado, mostrando algunos de los datos fundamentales junto a una opción para mostrar la planilla con todos los datos de la solicitud, en esta planilla hay opciones para cambiar el estado de la solicitud.

- Configuración: Esta opción se creó para modificar los datos personales del usuario, en un comienzo solo se habilito la opción de modificar la contraseña del usuario.

Teniendo un primer prototipo operacional se le muestra al **Cliente** los avances. A partir de esta reunión surgen una serie de dudas, por lo que es necesario analizar y desarrollar cada una:

- Al asignarle las solicitudes a los “ejecutivos”, ¿Tiene alguna prioridad el modo de asignación?
  - Inicialmente, no se tenía ninguna consideración al asignar las solicitudes, solamente que se repartiera la misma cantidad entre los ejecutivos disponibles, por lo que al revisar el resultado de este proceso se encontraron casos de ejecutivos de la zona central con solicitudes pertenecientes a la zona sur, por ejemplo; para solucionar esto se mejora la consulta SQL que busca las nuevas solicitudes para asignar, implementando el mismo principio que con los ejecutivos, al seleccionar una zona la aplicación busca entre todas las solicitudes las correspondientes a dicha zona, además se arregla la consulta SQL que asigna las solicitudes a los ejecutivos de modo que le dé prioridad al factor “Región”, tratando de asegurar que las solicitudes se le asignen a un ejecutivo de la misma región.

- Al ver la planilla con la información completa de la solicitud, ¿Es posible realizar una comunicación directa con él cliente?
  - La plantilla en un comienzo se diseñó para mostrar la información completa de la solicitud de crédito, sin considerar una comunicación directa con el cliente; para añadir esta opción se modificó la aplicación para invocar al programa Outlook de Microsoft (ya que todos los empleados de la empresa lo utilizan), desde la plantilla, de modo que se le permite al ejecutivo enviarle directamente un correo electrónico al cliente asociado a la solicitud, dicho correo puede ser configurado para contener un mensaje predeterminado, por lo que se agrega la opción al menú de “Configuración”.

- Pensando a futuro, cuando se tenga un número considerable de solicitudes, ¿Es posible implementar un medio para buscar una solicitud específica?
  - Considerando la cantidad de solicitudes que deberá procesar el sistema, implementar un sistema de búsqueda resulta fundamental, al modificar la aplicación se decide colocar la interfaz del sistema de búsqueda en el sector izquierdo de la ventana principal, bajo la información del usuario, y se codifica para que funcione de manera diferente para cada tipo de usuario; el usuario “Administrador” puede buscar cualquier solicitud, el usuario “Jefes de Zona” puede buscar cualquier solicitud perteneciente a su Zona encargada, y el usuario “Ejecutivo” solo puede buscar solicitudes que le hayan sido asignadas. El sistema de búsqueda permite ingresar como campo de búsqueda el RUT o el nombre (o parte de este), de la persona asociada a la solicitud.

- ¿Qué pasaría si un ejecutivo se enferma o toma vacaciones y tiene solicitudes pendientes?
  - Pensando en la posibilidad que un ejecutivo no pueda trabajar y que las solicitudes que tiene asignadas no queden pendientes indefinidamente se decide otorgarle al usuario “Administrador” la facultad de gestionar las solicitudes, para esto se agrega al menú de la aplicación la opción de “Gestión”, donde el “Administrador” puede seleccionar a uno o más “Ejecutivos” y liberar las solicitudes pendientes que tengan asignadas, de este modo estas pueden ser asignadas a otros “Ejecutivos”.

- ¿Cómo se agregan nuevos usuarios?
  - Hasta este punto, el único modo de agregar usuarios es añadirlos directamente a la base de datos, para simplificar esto, se agregan las opciones en el menú de “Configuración”, por lo que este se divide en un submenú con 3 opciones: “Contraseña y Contacto”, donde cada usuario puede modificar la contraseña de su cuenta y configurar un mensaje predeterminado al contactar a su cliente, “Agregar usuarios”, opción exclusiva del “Administrador” que le permite ingresar los datos y crear nuevos usuarios, y paralelamente se piensa en añadir la opción de “Eliminar usuarios”, facultad del “Administrador” para eliminar a los usuarios que dejen la empresa.

- ¿Qué tipo de base de datos tiene asociada la aplicación?
  - Al consultar sobre la base de datos, se le informa al **Cliente** la decisión tomada previamente y el motivo de esta, además se le informa cómo funciona la interfaz; el **Cliente** solicita, de ser posible, cambiar el tipo de base de datos asociada a la aplicación, de MySQL a una Microsoft Access; ya que es el tipo de base de datos que usan mayormente las demás aplicaciones de la organización; se realiza el cambio como lo solicitó el **Cliente** antes de efectuar los cambios descritos anteriormente, pero al probar la aplicación resulta evidente que una de las consecuencias producto del cambio en la

base de datos es que el tiempo de procesamiento aumenta, y en las funciones complejas de la aplicación resulta muy notorio.

Analizando lo conversado con el Cliente, se extraen nuevos requerimientos, y del mismo modo, se deben clasificar; teniendo como requerimientos funcionales los siguientes:

* Que la asignación de las solicitudes sea más precisa, con relación a la ubicación.
* Que permita a los usuarios comunicarse con los clientes (requerimiento que fue olvidado en un principio).
* Que se puedan realizar búsquedas de solicitudes.
* Que permita reasignar solicitudes.
* Que permita agregar nuevos usuarios al sistema.

Y de la misma forma, se tienen como requerimiento no funcional el siguiente:

* Deber contar con una base de datos de tipo Microsoft Access.

Realizados estos cambios se tiene un segundo prototipo operacional, al informarle al **Cliente**, este decide comenzar con la *beta test* [9] de la aplicación, proceso en el que se incluyen al usuario “Administrador” y un conjunto de cinco usuarios “Ejecutivos”; para esto primeramente se crea un Manual del Usuario para la aplicación, donde se detallan en profundidad las funciones que puede realizar la aplicación, como utilizarla y las capacidades con las que cuenta cada tipo de usuario.

Días después, al recibir el feedback de los usuarios, el **Cliente** solicita una nueva reunión con el **Programador**, de la cual surgen nuevos requerimientos, los que serán presentados, analizados y desarrollados a continuación:

- Solicitan cambiar el texto del menú, además de agregar una opción para volver a la página de inicio de la aplicación.
  - Este punto resulta el más sencillo, ya que el principal cambio es implementar un botón que lleve al inicio de la aplicación, el cual se coloca al principio del menú, el resto de los cambios consisten solo en modificar texto:
    - En el menú se cambia “Solicitudes” por “Estados”, y el submenú se modifica “Aceptadas” por “Acepta Oferta”, “Rechazadas” por “Rechaza Oferta” y “Terminadas” por “Muestra Interés”.
    - En el menú de “Configuración “, en su submenú, se cambia “Eliminar usuarios” por “Bloquear usuarios”, esto ya que no quieren eliminar la información de la base de datos, por lo que, al bloquear a un usuario, la cuenta queda restringida de acceso a la aplicación.

- Requieren que el usuario “Administrador” pueda ver y modificar la información de otros usuarios.
  - Para que el usuario “Administrador” pueda ver y modificar la información de otros usuarios se agrega la opción al menú de “Configuración”, añadiendo el submenú “Perfiles de usuarios”, donde se selecciona al usuario requerido según su zona de clasificación.

- Al trabajar con las solicitudes de crédito, algunos de sus clientes muestran interés y piden comunicarse con ellos en otro momento, pero a los “Ejecutivos” se les olvida, por lo tanto, solicitan un método que permita recordarles a los usuarios que vuelvan a contactar a sus clientes.
  - Para implementar una función de recordatorio, a la plantilla con la información de la solicitud de crédito, se le agrega un paso antes de clasificarla como “Muestra Interés”, solicitándole al usuario ingresar una fecha para que le recuerde de la solicitud, de este modo la aplicación compara la fecha ingresada con la fecha actual y cambia el estado de la solicitud a “Pendiente” para que el usuario “Ejecutivo” la vea otra vez.

- Solicitan la creación de un resumen para identificar el estado de avance del procesamiento de las solicitudes en cada zona.
  - Para cumplir con este requerimiento se crea una función que busca en la base de datos la cantidad de solicitudes de crédito correspondientes a cada “Estado”, repitiendo esto para cada Zona; mientras que la aplicación ordena las cifras en un cuadro resumen “Solicitudes / Zona”, el cual se ubica en el menú de “Gestión”, quedando de este modo un submenú con dos opciones: “Gestión de Ejecutivos” y “Resumen Solicitudes”.

- Solicitan que se implemente la capacidad de imprimir la información que muestra la aplicación.
  - Para lograr esto, se crea una función que captura el contenido de la ventana principal y permite imprimirla, la función se implementa dentro de toda la aplicación, por lo que es posible imprimir el contenido de casi cualquier ventana, excepto la ventana de inicio.

Una vez aplicados estos nuevos requerimientos, se le muestra al **Cliente** la aplicación, éste se muestra conforme y da su aprobación para finalizar el proceso de desarrollo y la beta test; concluyendo con la versión 1.0 de la aplicación. Posteriormente se actualiza el Manual del Usuario con los últimos cambios ejecutados y se distribuye entre los usuarios de la aplicación, concretando en su totalidad el proceso de desarrollo de esta.

3.2 Análisis de la calidad según la ISO/IEC 9126-1

Para realizar el proceso de control de la calidad, como se mencionó anteriormente; primeramente, se debe clasificar la aplicación; y en este caso, se tiene un Software empresarial de tipo CRM web. Después se debe escoger un índice medible como objeto de control, en este caso, se decide escoger el factor de tiempo. Posteriormente se debe establecer el o los métodos de valorización para el índice escogido, el cual ya fue escogido desde un comienzo, se realizará la valorización en base al estándar de calidad de la ISO. Y finalmente, se debe establecer quienes participan del proceso de control, en este caso, es el **Programador** quien realizará el procese de control.

El modelo ISO/IEC 9126-1, como se mencionó anteriormente, destaca seis características, a continuación, se procederá a analizar la aplicación descrita en base a cada una de éstas.

- **Funcionalidad**

  Analizando todos los requerimientos que surgieron de las reuniones con el **Cliente** y de la aprobación de la aplicación desarrollada por parte de este, se puede decir que esta cuenta con funciones que logran satisfacer todos los requerimientos y en general, en un tienen un tiempo de ejecución aceptable, no más de 10 segundos, salvo por la función que genera el cuadro resumen “Solicitudes / Zona”, ya que realiza muchas lecturas y filtros a la tabla más grande de la base de datos, donde en las últimas instancias de desarrollo esta función tuvo un tiempo de ejecución de casi 2 minutos..

- **Fiabilidad**

Referente a la presencia de errores en la aplicación durante su uso; al momento de desarrollarla, se realizaban constantes pruebas al código, con lo que fue posible solucionar los errores a medida que iban surgiendo, consiguiendo así prevenir la aparición de nuevos errores para el momento en que se liberó la aplicación para su uso dentro de la organización, esto se probó arduamente durante las últimas semanas de desarrollo, durante la beta test, tratando de ingresar datos de todas las formas erróneas posibles y de forzar el ingreso a funciones no correspondientes para el tipo de usuario correspondiente; de modo que lo único que podría afectar su uso es una interrupción en la red de la organización. Respecto al factor tiempo, el equipo servidor nunca se apaga, por lo que permite afirmar que la aplicación es fiable a largos periodos de tiempo.

- **Usabilidad**

Referente al uso, a pesar de que algunos usuarios tenían problemas en sus primeros intentos de usar la aplicación, está es relativamente sencilla de utilizar, ya que, al desarrollar la aplicación se optó por implementar una interfaz minimalista, sin adornos ni mucha variedad de colores, de modo que los usuarios no se distrajeran; además, que después del desarrolló del manual de usuario, el que tiene instrucciones que detalla cómo utilizar adecuadamente la aplicación, que hace cada función y que puede hacer cada tipo de usuario, enfatizando los privilegios que posee cada uno; el tiempo en que un usuario promedio aprende a dominar el uso de la aplicación es relativamente corto, en promedio, entre 4 y 5 horas.

Mientras que, para el usuario administrador, ya que durante su desarrollo se optó por una base de datos Microsoft Access, no debería tener mayores problemas para usarla, dada su similitud de uso con el programa Microsoft Excel.

- **Eficiencia**

Respecto a su eficiencia, dado que la aplicación es de tipo web, no necesita de muchos recursos para su óptimo funcionamiento, su principal requerimiento es contar con una conexión estable que les permita a los usuarios poder acceder a ella desde cualquier lugar cubierto por la red de la organización, y cualquier problema que se presente con esto, es externo al control del **Cliente** o del **Programador**. Y el tiempo de carga de la aplicación es rápido, no es mayor a 30 segundos desde las sucursales más remotas de la organización.

Pero, si se llegó a detectar un problema con la eficiencia de la aplicación, desde el punto de vista del **Programador**, ya que en un comienzo la aplicación se desarrolló con una base de datos MySQL, pero por petición del **Cliente**, como se mencionó antes, se cambió a una base de datos Microsoft Access; y por este cambio, fue notorio que a una mayor cantidad de datos se produce un incremento notable del tiempo que la aplicación necesita para procesarlos.

Además, respecto a cómo se ingresan los registros en la base de datos, al observar el proceso de extraer los registros desde otra aplicación para posteriormente ingresarlos en la base de esta aplicación, se ve que el proceso toma mucho tiempo, consumiendo entre una hora y una hora y media de tiempo de trabajo a diario; lo que se piensa, no resulta eficiente.

- **Mantenibilidad**

En cuanto a su mantención, dado que el desarrollo de la aplicación fue realizado bajo un contrato de duración limitada, se pensó en que la tarea de mantener o modificar el código de la aplicación quedaría a cargo de otra persona, más específicamente, a cargo del informático del área; de modo que dentro del código se dejó comentado qué hace cada función y se utilizaron nombres largos para las variables, para que sea fácil de entender a qué corresponde cada una.

Para realizar cambios en la aplicación se mantiene una copia del código en el servidor para poder realizar modificaciones y probarlas sin interrumpir o perjudicar a los usuarios en su uso paralelo, para que una vez validados y aprobados los cambios, sea cosa de copiar la nueva versión de la aplicación al enlace principal del servidor. En este caso, el factor tiempo resulta variable, ya que depende de que modificación se quiera realizar a la aplicación el tiempo que se tardara en hacerlo.

Pero, además, por parte de la organización, se solicitó que otra persona quedara a cargo de la base de datos, el usuario "Administrador", y es por esto el motivo del cambio en el tipo de la base de datos, ya que una base Microsoft Access es similar a Microsoft Excel en cuanto a estructura, lo que le permitiría al usuario "Administrador" mantener la base de datos sin necesidad de ser un experto DBA.

- **Portabilidad**

  Considerando el tipo de aplicación desarrollada, la portabilidad resulta una característica poco relevante, ya que, al ser una aplicación web, no requiere de un instalador para agregarla en cada equipo en que se utilizará; ni requiere ser accedida desde otro tipo de dispositivos que no sean los equipos de la organización. Por esto, el factor tiempo para la portabilidad es de 0.

  Y si se quiere reemplazar la aplicación por otra, las opciones serían contratar a otro desarrollador para crear una nueva aplicación que cumpla los mismos requerimientos o contratar el servicio de un software ya existente, lo que se reflejaría en un gasto extra de recursos, tiempo y dinero principalmente, para la organización y no resultaría conveniente.

Como observación adicional, a estas alturas está por finalizar el periodo de práctica, lo que da por terminado el contrato entre el **Programador** y el **Cliente**; por lo que, se solicita una última reunión con el cliente para proponer el desarrollo de un plan de mejora, argumentando su necesidad a partir del análisis de calidad anteriormente realizado. Al **Cliente** le interesa la propuesta y accede a que el **Programador** presente un plan de mejora.

Capítulo 4:

Desarrollo del plan de mejora

Como se mencionó al inicio, el software como producto requiere de constante mejora para mantenerse en el mercado, para esto, una de las herramientas existentes es el “plan de mejora”, el que permite desarrollar un proceso continuo para la actualización del producto, en la industria del Software esto es fundamental, ya que les permite a los desarrolladores realizar nuevas versiones de una aplicación de forma periódica y que esta solucione los requerimientos de sus usuarios.

A niveles organizacionales, el plan de mejora le permite a una empresa descubrir sus fortalezas y debilidades, y de esta forma aprovecharlas al máximo.

El plan de mejora abarca los siguientes pasos, de acuerdo al esquema presentado en la figura 5.

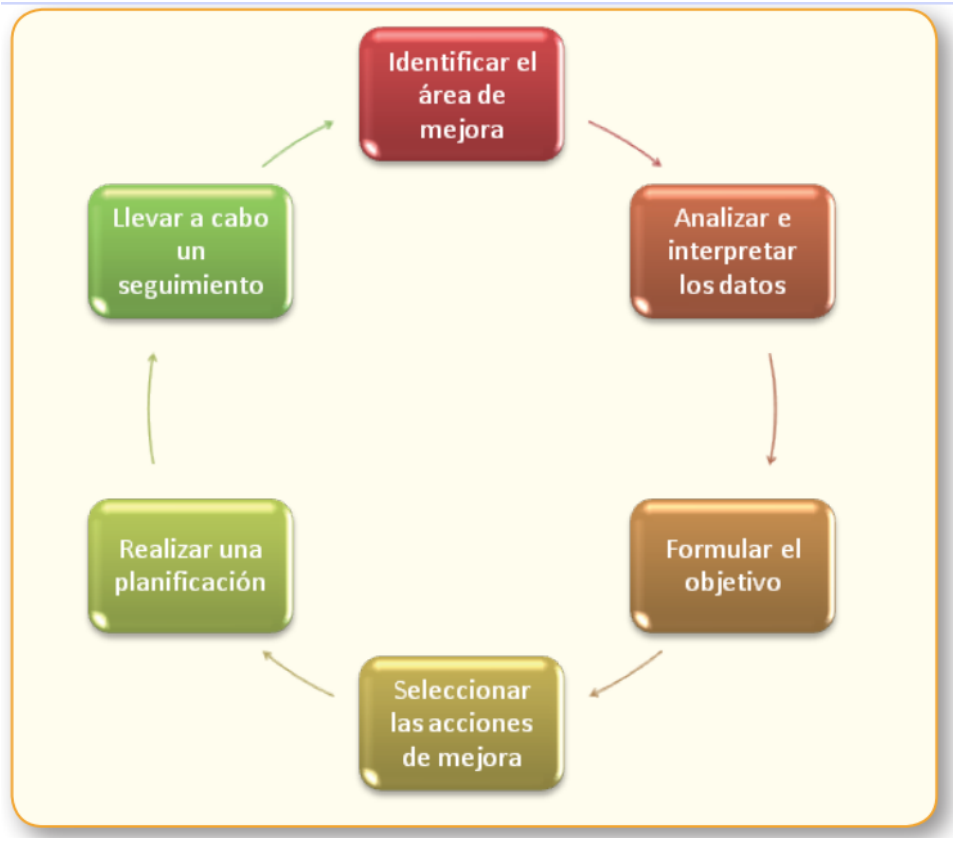


Figura 5: Esquema que muestra el ciclo para realizar un plan de mejora.

4.1 Identificar áreas de mejora.

Lógicamente el punto de partida del plan de mejora es recopilar información de diversos métodos, algunos de estos son investigación del mercado, feedback de los usuarios, encuestas, auditorias y/o, en este caso particular, el análisis de la calidad; con el fin de encontrar las posibles áreas de mejora de la aplicación.

Las áreas de mejora son las partes del producto donde se identifica un problema y/o posible falencia; y que, a partir del análisis de calidad a la aplicación, se han podido identificado algunas áreas claves que serán el foco central en el desarrollo del plan de mejora, estas son con respecto a la funcionalidad y la eficiencia de la aplicación, y que se especifican a continuación:

- El tiempo de ejecución alto de la función que genera el cuadro resumen “Solicitudes / Zona”.
- El aumento en el tiempo para procesar los registros por la nueva base de datos.
- El tiempo que se pierde en traspasar los registros desde otra aplicación a esta.

4.2 Analizar e interpretar los datos.

Del punto anterior, el principal problema detectado con respecto a la eficiencia de la aplicación es el tiempo que tarda en procesar todos los datos de la base, ya que este aumenta proporcionalmente al volumen de datos, en los últimos momentos del desarrollo de la aplicación, cuando se comenzó a llenar la base de datos; diariamente se ingresaban cerca de mil registros nuevos, lo que a futuro supone una cantidad de datos considerable y un tiempo de espera elevado. Este problema se relaciona directamente al problema de la funcionalidad, ya que el alto tiempo de procesamiento de los datos es directamente proporcional al tiempo de ejecución de la función que genera el cuadro resumen “Solicitudes / Zona”.

Con respecto al otro punto identificado, al cómo se ingresan los registros a la base de datos, se piensa que es un modo muy ineficiente, ya que, considerando que fue necesario cambiarle el tipo de base de datos a la aplicación a una de tipo Microsoft Access, porque es el tipo de base de datos que usan el resto de las aplicaciones de la organización, se piensa que debe existir un método de ingresar los registros directamente en la base de datos de la aplicación sin la necesidad de realizar el proceso de extracción mencionado anteriormente.

Los problemas analizados anteriormente, serán clasificados como:

- Problema principal, a aumento en el tiempo para procesar los registros por la nueva base de datos, y en criterio a que el tiempo aumentara constantemente con el paso del tiempo.
- Problema secundario, al tiempo que se pierde en traspasar los registros desde otra aplicación a esta, y en criterio a que el tiempo empleado es relativamente constante.

Lo anterior con el fin de abreviar futuras referencias a estos problemas.

4.3 Formular el objetivo.

Considerando el problema secundario detectado, se piensa que la solución más lógica es la de editar la otra aplicación que registra las solicitudes de crédito hipotecario para que se ingresen directamente en la base de datos de la nueva aplicación, basándose en el hecho en que ambas ocupan la misma base de datos.

Respecto al problema principal detectado, se piensa en dos posibles soluciones:

- Cambiar el tipo de base de datos a una que requiera de menos recursos para su funcionamiento.

  Esta alternativa parece ser la más óptima, considerando el tiempo y complejidad que requiere, ya que a estas alturas el plazo para el desarrollo de la aplicación y para el vencimiento del contrato entre el **Programador** y la organización se aproxima. Pero su implementación depende exclusivamente del **Cliente**, ya que implementar una nueva base de datos podría llegar a significar un costo adicional, pero además al considerar esto contradice a la solución pensada anteriormente para el problema secundario.

- Modificar las funciones de la aplicación para reducir su complejidad y optimizar los tiempos de procesamiento y de ejecución.

  Esta última alternativa debería ser la más lógica de implementar, porque consiste en reevaluar el código de la aplicación para hacerlo más eficiente, pero el proceso de reducir la complejidad de una función es una tarea difícil, que requiere de constante pruebas y evaluaciones, y que en su aplicación es poco viable, porque como se mencionó anteriormente, por el tiempo que está quedando no hay certeza en que se logre concretar a tiempo.

4.4 Seleccionar acciones de mejora.

Luego de analizar las áreas de mejora y de pensar en objetivos, queda ver que se puede realizar, para ello es necesario proponer estas ideas al **Cliente** para que apruebe y autorice el plan.

Después de proponerle las ideas anteriormente descritas, el **Cliente** se muestra interesado ante la idea de mejorar la aplicación, pero inmediatamente se niega a la idea de cambiar el tipo de base de datos, con lo que esa idea queda descartada; por lo tanto, con la segunda idea de reducir la complejidad de las funciones, se le informa al **Cliente** que es algo que se estima que tomara mucho tiempo, por lo que no queda del todo convencido si aplicarla o no. Finalmente por el tipo de contrato del **Programador**, para el problema secundario, el modificar otras aplicaciones ya existentes queda fuera de sus privilegios, por lo que esa idea queda descartada, pero el **Cliente** admite que el proceso de traspaso de los registros es tedioso, por lo que se considera el hecho de que los usuarios “Ejecutivos” tratan directamente con las personas en las sucursales, por lo que en conjunto con el **Cliente** se llega a la idea de realizar una función para que estos usuarios puedan agregar directamente las solicitudes de crédito que las personas realicen en las sucursales de la organización, lo que reduciría la cantidad de registros ingresada en la otra aplicación y reduciría el tiempo necesario para realizar el traspaso.

Finalmente, la principal acción para mejorar la aplicación, es la de implementar un medio para que los usuarios “Ejecutivos” puedan agregar los registros directamente en la base de datos. Mientras que la otra alternativa de modificar las funciones para reducir su complejidad se considera, pero con menor prioridad por parte del **Cliente**.

4.5 Realizar una planificación.

Una vez seleccionadas las acciones de mejora, se llega a un acuerdo con el **Cliente** de realizar el trabajo de forma independiente, teniendo como plazo para concretarlo un mes.

Por lo que antes de iniciar el desarrollo de las mejoras en las áreas identificadas, es necesario una planificación, por lo tanto, primeramente, hay que considerar que pasos se deben realizar para implementar la nueva funcionalidad, esto porque se le dará mayor prioridad como objetivo. Estos son:

Tabla N°1: Planificación de la implementación de la nueva función.

Tareas a realizar	**Tiempo estimado**
Programar la nueva función.	2 días.
Agregar la nueva función a la aplicación.	1 día.
Hacer pruebas con la base de datos.	1 día.
Corregir cualquier error detectado	1 – 2 días.

.

De manera similar, para realizar la operación de reducir la complejidad de las funciones de la aplicación; si bien tiene una menor prioridad y al Cliente no le importa tanto su realización, el **Programador** quiere intentar realizarlo, se debe pensar en los pasos:

Tabla N°2: Planificación para reducir la complejidad del código.

Tareas a realizar	**Tiempo estimado**
Definir con exactitud la complejidad de las funciones existentes.	4 - 5 días.
Desarrollar funciones paralelas con menor complejidad.	3 – 4 días.
Implementar nuevas funciones en la aplicación.	3 – 4 días.
Testear el funcionamiento en busca de errores.	1 semana.

Teniendo lo anterior en cuenta, se estima que, para implementar la nueva funcionalidad, considerando los pasos mencionados al inicio de este punto, el tiempo máximo es de una semana, considerando el peor escenario, donde se presentó algún error en la implementación y requiere corrección.

Con lo anterior ya definido, implica que se usara el resto del tiempo, tres semanas aproximadamente, en intentar optimizar las funciones de la aplicación.

4.6 Llevar a cabo un seguimiento.

Durante el periodo que dure la implementación del plan de mejora, se hace el compromiso con el cliente de mantenerlo informado de los avances mediante mensajes semanales, los que serán enviados por medio de correo electrónico o por mensajería a través del Smartphone, de modo que sepa constantemente el estado de la actualización.

Capítulo 5:

Implementación del plan de mejora.

Para implementar el plan de mejora se debe administrar adecuadamente el tiempo del que se dispone, por lo que se impone un horario de trabajo similar al que se tenía trabajando en la organización del **Cliente**, de este modo, en la primera semana se logra desarrollar la nueva funcionalidad para la aplicación, usando el diseño principal de la aplicación no se requirió de mucho para adaptarlo y con los programas utilizados, Notepad++ en conjunto con la distribución de XAMPP 5.2; se crear una función con el formulario adecuado con el cual se pudiera agregar datos en las columnas correspondientes de la tabla en la base de datos, sin que esta genere errores durante su ejecución, datos como: Nombre, Apellidos, Teléfono, Dirección, Comuna, Región, Empleo, entre otros.

Para reducir la complejidad de las funciones ya existentes, primeramente, como se mencionó, se debe analizar su complejidad, refiriéndose al orden de complejidad de ejecución de las funciones, esto es el tiempo que tarda cada función en ejecutarse; para esto se observa el código desarrollado y se buscan las funciones más complejas.

En la figura 6 del anexo, se dejan dos seudocódigos como ejemplo de las funciones más complejas que se crearon durante el desarrollo de la aplicación.

A)

```
//Asignar solicitudes

if(condicional para tipo de usuario 1){
    if(condicional de boton 1){
    accion "seleccionar zona";
        if(condicional para zona vacia){
            if(condicional si es zona 1){
                accion "leectura y selección de datos de acuerdo a criterio de zona 1";
            } else if(condicional si es zona 2){
                accion "leectura y selección de datos de acuerdo a criterio de zona 2";
            } else if(condicional si es zona 3){
                accion "leectura y selección de datos de acuerdo a criterio de zona 3";
            } else if(condicional si es zona 4){
                accion "leectura y selección de datos de acuerdo a criterio de zona 4";
            } else if(condicional si es zona 5) {
                accion "leectura y selección de datos de acuerdo a criterio de zona 5";
            }
            accion "se procesan las solicitudes del condicional anterior";
            while(condición numerica){
                accion "leectura y selección de datos de acuerdo al criterio previo";
            }
        } else {
            accion "mensaje de error";
        }

    }
    if(condicional de boton 2){
        accion "procesar datos previos";
        if (condicional de numeros){
            if(condicional de usuarios seleccionados){
                accion "determinar array";
                foreach(condicion largo array){
                    acción "seleccion de datos";
                    while(condicion de array 1){
                        accion "determinar datos";
                    }
                }
                for(condicion de repeticion){
                    accion "asignacion de datos";
                    while(condicion de array 2){
                        if(condicional de validacion de datos){
                            accion "actualizar datos de tabla";
                        }
                    }
                    if(condicional de cambios en la base){
                        accion "determinar largo array";
                        while(condicion de array 3){
                            if(condicional de valor){
                                accion "actualizar datos de tabla";
                            }
                        }
                    }
                }
                    accion "mensaje de accion realizada";
                } else {
                    accion "mensaje de error";
                }
        } else {
            accion "mensaje de error";
        }
    }
}else if(condicional para tipo de usuario 2){
```

B)

```
1  //Resumen de Solicitudes
2
3  if("primera condicional"){
4      realiza accion;
5  }else if("segunda condicional"){
6      Lectura y cuenta de datos de la tabla
7
8      Primera lectura de tabla para zona 1 con condicional 1
9      Segunda lectura de tabla para zona 1 con condicional 2
10     Tercera lectura de tabla para zona 1 con condicional 3
11     Cuarta lectura de tabla para zona 1 con condicional 4
12
13     Primera lectura de tabla para zona 2 con condicional 1
14     Segunda lectura de tabla para zona 2 con condicional 2
15     Tercera lectura de tabla para zona 2 con condicional 3
16     Cuarta lectura de tabla para zona 2 con condicional 4
17
18     Primera lectura de tabla para zona 3 con condicional 1
19     Segunda lectura de tabla para zona 3 con condicional 2
20     Tercera lectura de tabla para zona 3 con condicional 3
21     Cuarta lectura de tabla para zona 3 con condicional 4
22
23     Primera lectura de tabla para zona 4 con condicional 1
24     Segunda lectura de tabla para zona 4 con condicional 2
25     Tercera lectura de tabla para zona 4 con condicional 3
26     Cuarta lectura de tabla para zona 4 con condicional 4
27
28     Primera lectura de tabla para zona 5 con condicional 1
29     Segunda lectura de tabla para zona 5 con condicional 2
30     Tercera lectura de tabla para zona 5 con condicional 3
31     Cuarta lectura de tabla para zona 5 con condicional 4
32 }
```

Figura 6:

A) Ejemplo de seudocódigo de la función para asignar solicitudes, dada su longitud completa solo se adjunta la mitad del seudocódigo.

B) Ejemplo de seudocódigo de la función que genera el cuadro resumen “Solicitudes / Zona”.

Después de haber revisado los ejemplos dejados anteriormente, se descubre que, la primera de estas tiene un orden de complejidad cuasi-lineal: $\boldsymbol{\theta(n \log n)}$, donde se observa una función de iteración dentro de una función de bucle; y la otra, posee una complejidad lineal: $\boldsymbol{\theta(n)}$, donde se observa que se lee la tabla varias veces, pero sin ninguna función adicional que lo haga más complejo.

De lo anterior, analizándolo matemáticamente, hay que considerar que, dependiendo de la función, se leen y/o se escriben en la base de datos "**n**" número de elementos que, para realizar esto, la aplicación tarda un tiempo "**t**", el cual varia si el proceso es de lectura o escritura, además de que varía de acuerdo a la capacidad de procesamiento que entrega el Hardware del servidor donde se encuentra la base de datos. Considerando que el número "**n**" aumenta con el paso del tiempo, de acuerdo al orden de complejidad de las funciones analizadas, el valor de "**t**" puede aumentar drásticamente, y se comprobó durante el desarrollo de la aplicación que diariamente, se agregaban aproximadamente mil elementos a la base de datos.

Ahora, hablando específicamente del caso de la función que crea el cuadro resumen "Solicitudes / Zona" que pidió el **Cliente**, esta función interactúa con la tabla de las solicitudes, la que agrupa todas las solicitudes ingresadas por los clientes de la organización, y dentro de esta tabla se tiene un número considerable de columnas; esta función primeramente lee esta tabla cuatro veces, para contar el número de solicitudes en los distintos estados en que los clasifica el **Cliente**; y esto se repite cinco veces, una por cada zona en que el **Cliente** agrupa las distintas regiones, esto en total da veinte lecturas de la tabla más grande de la base de datos. Considerando la cantidad de datos que se tienen en la base, y al ritmo en que aumentan, no es de sorprender que esta función requiera de tanto tiempo para ejecutarse.

Con lo anterior en mente, se intenta modificar el código para reemplazar alguna iteración o bucle que genera la demora en procesar la información en algunas de las funciones, pero luego de algunos intentos, se llega a la conclusión que lo único que se puede hacer es reemplazar una iteración por otra o un bucle por otro, ya que otras

modificaciones no funcionan correctamente o terminan generando un orden de complejidad aún mayor, como en el caso de la función de asignar solicitudes, expuesta en el ejemplo, al reemplazar una iteración por un bucle, se obtuvo un orden de complejidad cuadrático ($\boldsymbol{\theta}(\boldsymbol{n^2})$); con el tiempo avanzando se busca alguna otra forma de reducir los tiempos de procesamiento de datos, pensando que además del código fuente de la aplicación, lo otro que influye son las consultas a la base de datos, las que se analizan descubriendo por primera vez lo complejas que resultaron en realidad, teniendo en una misma función subconsultas dentro de una consulta principal y sentencias “JOIN” lo que resulta en un proceso de muchas consultas a la base de datos. Habiendo deducido esto se realizan algunas pruebas modificando las consultas e intentando simplificarlas, pero esto solo resulta en errores durante su ejecución.

A estas alturas del desarrollo, ya se acabó el tiempo acordado con el **Cliente**, por lo que se llega a un acuerdo de ir a la organización e implementar la nueva función solicitada, para que los usuarios puedan agregar directamente solicitudes de sus clientes a la base de datos, además de enseñársela al administrador de la aplicación y entregar el manual actualizado para el resto de los usuarios; posteriormente se habla con el **Cliente** y se le informan los detalles sobre el desarrollo realizado y las mejoras que se podrían continuar implementando, como continuar realizando pruebas para reducir el orden de complejidad de algunas funciones o adaptar la aplicación para que sea compatible con otros navegadores, considerando que en algún momento se dejará de utilizar Internet Explorer; pero el Cliente no muestra interés en continuar mejorando la aplicación y decide concluir el desarrollo de esta y continuar usándola como se encuentra.

En el periodo de tiempo desde que concluyó el desarrollo de la aplicación hasta el momento en que se escriben estas palabras, se vuelve a contactar al **Cliente** para consultarle sobre la aplicación, donde este comunica que en la actualidad, ya no están usando la aplicación dentro de la organización; al consultar el motivo de esto, esté responde que después de un año de haber utilizado la aplicación, está comenzó a volverse obsoleta, se tenían almacenados muchos datos y se requerían nuevas

funcionalidades solicitadas por la organización para poder trabajar adecuadamente con ellos, y que por cambios en esta, no habían conservado a nadie calificado en el área del **Cliente** que pudiera hacerle mantenimiento o actualizar la aplicación para cubrir las nuevas funcionalidades solicitadas, por lo que la organización decidió comprar el programa de otra empresa que le permitía abarcar varias problemáticas de manera simultánea, entre ellas, aquella por la que se solicitó el desarrollo de la aplicación en un comienzo.

Conclusiones

A modo de conclusión, durante el desarrollo del trabajo se cumplió el objetivo principal, desarrollar un plan de mejora a partir del análisis de la calidad de un software, concluyendo, de acuerdo a lo investigado, que este logra cumplir con los requisitos para ser un análisis efectivo; y que el plan que se logró desarrollar a partir de este cumple su objetivo de descubrir las debilidades de la aplicación, pudiendo proponer varias alternativas de mejora de acuerdo a las necesidades del cliente.

Además, del hecho de crear la aplicación, se puede concluir lo importante que es contar con un plan o estrategia para el desarrollo de un software, el escoger un modelo de programación y el hecho de mantener una comunicación constante con el cliente; sin esto, el desarrollo de la aplicación podría haber derivado en una serie de problemas, lo que complicaría la relación con el cliente y perjudicaría al desarrollador.

Lo anterior apoya los postulados que a pesar de ser “informáticos” se requieren más que habilidades técnicas, se hacen necesarias las habilidades blandas para interactuar con el cliente y los compañeros de trabajo para poder establecer una comunicación adecuada entre todas las partes involucradas en un proyecto.

Además, del hecho de que la aplicación dejará de usarse en la organización por falta de mantención, apunta al hecho de que, dentro de una organización, o en el caso de las grandes organizaciones, dentro de cada área, haya algún especialista encargado de mantener y/o actualizar las herramientas que se utilizan dentro de esta.

Finalmente, respecto a la estimación del tiempo de aplicación para el plan de mejora, se puede decir que hubo diferencias, ya que para la primera tarea se estimó que tardaría cerca de una semana, y en la realizar tomo 5 días; mientras que para la segunda tarea se estimó un tiempo de 3 semanas y en la realidad faltó tiempo para poder completarlo; esto remarca que cada programador debe tener en cuenta su propia habilidad además de la dificultad individual de cada tarea a la hora de querer cumplir en los tiempos de entrega de un proyecto.

Referencias

[1]. CALIDAD DE PRODUCTO Y PROCESO SOFTWARE. CAPITULO 10: CALIDAD DE PRODUCTO Y PROCESO SOFTWARE. Coral Calero Muñoz, Mario Piattini Velthuis, María Ángeles Moraga de la Rubia.

[2]. Software Engineering. A Practitioner's Approach. Seventh Edition. Roger S. Pressman.

[3]. Universidad Complutense Madrid. Departamento de Sistemas Informáticos y Programación. Ingeniería del Software de Gestión 1. El Producto: Software. Juan Pavón Mestras.

[4]. Pontificia Universidad Católica del Perú. Facultad de Ciencias e Ingeniería. Ingeniería Informática. Normas de la Calidad del Producto Software, versión 1.0. Karin Meléndez, Abraham Dávila.

[5]. Universidad Miguel Hernández de Elche. Planes de Mejora. Procedimiento para el diseño y seguimiento de los planes de mejora.

[6]. Universidad Técnica Federico Santa María. Departamento de Informática. Gestión de Calidad en Proyectos TI. Marcello Visconti Z.

[7]. Universidad Técnica Federico Santa María. Departamento de Informática. Requisitos de Calidad Importantes; Norma ISO/IEC 9126 Parte 1. Marcello Visconti Z.

[8]. http://dle.rae.es/?w=diccionario

[9]. http://www.iso.org/iso/home/about.htm

[10]. http://iso25000.com/index.php/normas-iso-25000

[11]. http://www.normas9000.com/que-es-iso-9000.html

[12]. http://www.inn.cl/node/22

[13].https://www.isotools.org/2015/04/15/tipos-de-certificaciones-laborales-que-puede-obtener-tu-empresa-2/

[14].http://qbit.com.mx/blog/2012/02/03/certificaciones-de-calidad-en-desarrollo-de-software/

[15]. http://www.investopedia.com/

[16]. http://www.corfo.cl/sobre-corfo