APLICACIÓN DE LAS MÉTRICAS DE CALIDAD DE SOFTWARE EN EL DISEÑO Y DESARROLLO DE UN SISTEMA PARA LA CENTRALIZACIÓN DE LOS PROCESOS DE UNA ISP BAJO AMBIENTE WEB.

Jhonn Jairo Vesga Cadena
Andrés Julián Guzmán Díaz

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
ESCUELA DE CIENCIAS NATURALES E INGENIERIA
FACULTAD DE INGENIERIA DE SISTEMAS
LINEA DE SISTEMAS DE INFORMACIÓN E INGENIERIA DEL SOFTWARE
BUCARAMANGA 2006

APLICACIÓN DE LAS MÉTRICAS DE CALIDAD DE SOFTWARE EN EL DISEÑO Y DESARROLLO DE UN SISTEMA PARA LA CENTRALIZACIÓN DE LOS PROCESOS DE UNA ISP BAJO AMBIENTE WEB.

Jhonn Jairo Vesga Cadena
Andrés Julián Guzmán Díaz

Trabajo de grado para optar el título de
Ingeniero de Sistemas

Director:
Roberto Carvajal Salamanca

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
ESCUELA DE CIENCIAS NATURALES E INGENIERIA
FACULTAD DE INGENIERIA DE SISTEMAS
LINEA DE SISTEMAS DE INFORMACIÓN E INGENIERIA DEL SOFTWARE
BUCARAMANGA 2006

Nota de Aceptación

_____

_____

_____

_____

_____

_____
Presidente del Jurado

_____
Jurado 1

_____
Jurado 2

Bucaramanga, 30 de Junio de 2006

A todos los maestros que nos orientaron a lo largo de nuestra carrera, a nuestros amigos con los que compartimos experiencias y momentos agradables y especialmente a nuestros padres que con su apoyo, palabras de aliento y motivación creyeron en nosotros para cumplir esta meta trazada en nuestras vidas.

AGRADECIMIENTOS

Quiero expresar mis agradecimientos a:

A Dios, mi guía espiritual que siempre me brindó la luz para iluminar el camino a recorrer. A mis padres, Pastor y Martha que han sido ejemplo de la mas pujante raza, quienes con sus valores y principios me enseñaron a valorar el significado del ÉXITO. A mi hermana Kelly, ejemplo de excelencia y dueña del camino de la victoria. A mi Gran Amigo Andrés Julián, transparente, fiel, sincero, único hermano adoptivo con quien me abrió las puertas de su amistad y con quien llevo tantos años de hermandad. A todos aquellos que indirectamente estuvieron pendientes de mi desempeño y creyeron en mis capacidades. “El éxito no viene solo, viene cargado de buenos amigos”

Jhonn Jairo Vesga Cadena

Quiero expresar mis agradecimientos a:

A Dios, mi padre que cada día me ilumina y me guía, mostrándome el camino del triunfo a seguir. A mis padres, Clímaco y Emperatriz, que con sus grandes valores, y un corazón de espíritu victorioso, con sus palabras, concejos y sabiduría han creído en todos mis sueños. A Carlos Mauricio, que con su fidelidad de hermano me ha apoyado en todas mis decisiones y ha dejado claro que nuestra unidad es nuestro mayor logro. A mis Tíos, por su gran apoyo incondicional han hecho posible alcanzar cada una de las metas en mi vida. A Jhonn Jairo, mi único amigo incondicional que con su carácter sincero en el desinteresado esfuerzo por lograr las metas siempre ha estado conmigo. Y en especial a Carolina Rojas, que siempre ha estado conmigo, me ha apoyado, aconsejado y acompañado en todas mis decisiones con un respeto y sincero sentimiento a lo que hoy hemos logrado formar.

Andrés Julián Guzmán Díaz

TABLA DE CONTENIDO

Pág

LISTA DE TABLAS

pág

LISTA DE FIGURAS

pág

LISTAS ANEXOS

pág

INTRODUCCIÓN

La competencia por quién debe ser el mejor a nivel empresarial se ha convertido en una lucha continua de tecnología y procesos desarrollados, teniendo en cuenta que realizar sus metas en un periodo reducido, marca la diferencia entre el éxito y el fracaso.

Sin embargo los procesos manuales siguen siendo comunes y difíciles de estandarizar ya que el cambio implica variables de adaptación que difieren de una empresa a otra, por lo tanto una investigación, recolección de requisitos, análisis de procesos, entre otras forman parte de este proceso de adecuación tecnológico.
La automatización de procesos manuales es una solución al desarrollo constante de una empresa y su permanencia en el mercado competitivo, donde la reducción de tiempo en labores comunes, entrega de informes antes de lo pedido, son algunas de las ventajas obtenidas en la aplicación tecnológica que se lleve a cabo.
Es necesario tener en cuenta que para que una empresa llegue a tal grado de evolución se necesita de un completo estudio, tanto de la parte empresarial como en la administrativa, de tal manera que el sistema que se implemente logre cumplir con los requerimientos de la empresa y el cambio que se realice sea benéfico y no perjudicial.

Las métricas de calidad de software son herramientas adecuadas para implementar este tipo de soluciones brindando corrección, estabilidad, fiabilidad, flexibilidad, eficiencia, integridad y confiabilidad, logrando software de excelente calidad que cumple con las expectativas de cada empresa en su proceso de desarrollo.

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Investigar sobre las métricas de calidad de software y su implementación en un sistema de información para la estandarización de procesos en una ISP (Internet Service Provider), a través de herramientas Web, con el fin de producir un software de calidad.

1.2 OBJETIVOS ESPECÍFICOS

- Efectuar una investigación de ingeniería del software, a fin de determinar las métricas de calidad aplicables al desarrollo de dos módulos del Sistema de Información para las ISP (Internet Service Provider).

- Determinar y aplicar el diseño que permitan el desarrollo de un prototipo robusto, flexible y reutilizables para cualquier tipo de ISP (Internet Service Provider).

- Aplicar las métricas de software seleccionadas a los prototipos a fin de obtener un producto de óptima calidad.

- Desarrollar los módulos “Administración de Usuarios” y “Gestión” del prototipo, a los cuales se les aplicará las métricas de calidad seleccionadas.

2. MARCO TEÓRICO

2.1 ADMINISTRACIÓN Y FUNCIONAMIENTO DE UNA ISP (INTERNET SERVICE PROVIDER)

Los procesos que se realizan en una ISP cumplen un factor importante y determinante cuando de rendimiento, producción y administración se refiere. La mayoría de estos procesos están basados en la recolección de información para rectificar errores que no son perceptibles por simple método de observación, por consiguiente utilizan técnicas de mercadeo, auditoria interna, control de gestión y verificación del servicio; esto con el único fin de garantizar al usuario un servicio de óptima calidad y por ende la permanencia del mismo con el proveedor de acceso a Internet.

Tres aspectos importantes para una ISP: La Calidad de Servicio, Gestión de clientes o usuarios y Divulgación en el medio.

2.1.1 Calidad del servicio. Siempre que se quiere hablar de Internet, se tiene que tener en cuenta el principal objetivo de una ISP; *La calidad de servicio*. Esto hace referencia a llenar las expectativas y las promesas que se le hicieron al usuario al momento de adquirir un contrato, de este depende que el cliente se convierta en portavoz y promotor de ventas indirecto para la compañía, un cliente satisfecho, traerá mas clientes, uno insatisfecho, acarreara muchos problemas, mala imagen y pérdidas a la empresa, de allí su importancia.

2.1.2 Gestión de Clientes. Hace referencia a la manera en la que la empresa se preocupa por el bienestar del cliente ofreciendo valores agregados, llevando históricos de acceso, estadísticas de conexión, recordando pagos para evitar

suspensión en el servicio ya que esto incomoda al cliente. De esta manera se logra tener un perfil del cliente y se puede determinar lo que se llama "Personalidad del consumidor" para lograr suplir todas las necesidades del usuario y garantizar su permanencia en la empresa.

2.1.3 Divulgación en el medio. Hace referencia a la manera en la que la empresa da a conocer los servicios que ofrece. Una buena campaña publicitaria tiene como resultado la llegada de nuevos clientes.

El negocio de Internet consiste en traer y mantener usuarios con el fin de tener la última palabra en telecomunicaciones y de esta manera ser pionera en su género. Todo este proceso se logra a través de asesores comerciales, avisos, campañas, ofertas y todo cuanto sea necesario para lograr que la gente que aun no conoce el servicio, lo adquiera.

2.2 SENTIDO DE LA CALIDAD

La calidad en cualquier área de trabajo se considera como uno de los puntos más importantes a la hora de vender algún tipo de producto. El que cumpla con requisitos impuestos por las normas existentes, como la ISO (International Standard Organization), hacen la diferencia entre el éxito o el fracaso, ya que el hecho de tener este tipo de estándar de calidad logra que cualquier producto obtenga un valor agregado, sea llamativo al mercado y se diferencie de los demás productos que se presentan.

El concepto de Calidad de Software es esencial para el propósito de esta tesis, por lo tanto las siguientes definiciones encontradas nos acercan un poco más a entender el significado de un tema tan polémico.

Según Philip Crosby: “La Calidad es una entidad alcanzable, medible y rentable que puede ser incorporada, una vez que se desee hacerlo, se entienda y se esté preparado para un arduo trabajo[1].

Según Edwin S. Shecter “La Calidad es una característica o atributo de cierta entidad; el natural o esencial carácter de algo, excelencia, superioridad o grado de excelencia”. [2]

Según la terminología de la norma ISO 8402, “Conjunto de Características de una Entidad, Producto o Servicio que le confieren su aptitud para satisfacer necesidades explicitas e implícitas” [3]

La mayoría de las empresas que existen actualmente buscan calidad a la hora de adquirir un producto. Sacrificar unos pesos de más por ahorrar presupuesto, puede ser un error que mas adelante se vea representado en productividad y desempeño, teniendo finalmente que reinvertir en un producto de calidad para lograr que dicha empresa cumpla con las metas que no se lograron con el anterior producto por falta de calidad.

2.3 CALIDAD Y GARANTÍA DE SOFTWARE

La calidad de software es una mezcla entre los estándares existentes y las exigencias de los clientes que desean calidad, muchos son los factores y varían según la necesidad de cada cliente. La garantía de calidad es un requisito necesario para todas aquellas empresas que se dedican a la producción de

---

[1] CROSBY B. Philip, La Calidad No Cuesta. El Arte de Cerciorarse de la Calidad, McGRAW HILL BOOK COMPANY, ISBN 968-26-1220-9, México, p.13
[2] SHECTER S. Edwin, Managing for World-Class Quality, Marcel Dekker (September 1, 1991, ISBN 0824777123, What is Quality, P. 29
[3] Terminología de la norma ISO 8402. Septiembre de 2004. [Citada en Mayo de 2005]. Disponible en http://dmi.uib.es/~bbuades/calidad/tsld010.htm

software, ya que sin este tipo de garantía es muy probable que sus productos no sean bien recibidos en el mercado.

2.3.1 Calidad de Software. La Calidad de software se define como la unión de los requisitos de funcionalidad y de rendimiento establecidos inicialmente, y la documentación y estudio realizado sobre el producto al cual se le está aplicando una determinada norma de calidad. La calidad de software es medible y varia según el tipo de sistema y su función para el cual fue desarrollado. En el caso de un software desarrollado para la NASA que se encarga de controlar una nave espacial o un centro de investigación, el nivel de calidad de este software debe estar reflejado en un riesgo de cero fallas ya que su principal función no le permite tomar ningún tipo de tolerancia a fallas. La calidad del software entonces depende del factor función y tiempo en funcionamiento, ya que definitivamente se tiene que definir cual es la visión del producto que se desarrolla. Éste puede ser proyectado para tener un largo ciclo de vida, entonces deberá tenerse en cuenta la flexibilidad del producto, mantenimiento, capacidad de actualización y flexibilidad para que los costos de este en el tiempo usado sean relativamente bajos.

Según Dr. Neil Hernandez Gress, Director de la Escuela de Graduados en Ingeniería y Ciencias del Instituto Tecnológico de Estudios Superiores de Monterrey, Campus Estado de México: “La Calidad del Software puede medirse después de elaborado el producto. Pero esto puede resultar muy costoso si se detectan problemas derivados de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del software.”

2.3.2 Factores que determinan la calidad del software. Los factores que afectan la calidad de software se pueden clasificar en dos grupos:

- Factores que pueden ser medidos directamente (KLDC, (Miles de líneas de código)/Unidades de tiempo)
- Factores que solo pueden ser medidos indirectamente (Facilidad de Uso y mantenimiento).

Los factores de calidad se basan en tres aspectos básicos: características operativas, su capacidad de soportar los cambios y su adaptabilidad a nuevos entornos. Estas a su vez poseen características que definen la calidad cuya explicación se detalla en el apartado 4.6:

- Corrección
- Fiabilidad
- Eficiencia
- Integridad
- Usabilidad (facilidad de manejo)
- Flexibilidad
- Facilidad de prueba
- Portabilidad
- Reusabilidad (capacidad de reutilización)
- Interoperatividad

2.3.3 Gestión de Calidad. La calidad del software es definida como la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente. [4]

[4] Pressman, Roger S. (1998) Ingeniería de software. Un enfoque práctico. Cuarta Edición, Madrid, Mc Graw-Hill Interamericana de España S.A.

La gestión de la calidad se puede entender como el conjunto de actividades y medios necesarios para definir e implementar un sistema de la calidad, por una parte y responsabilizarse de su control, aseguramiento y mejora continua, por otra. El control está dirigido al cumplimiento de requisitos, el aseguramiento a inspirar confianza en que se cumplirá el requisito pertinente y el mejoramiento al aumento de su eficiencia y eficacia.

2.3.4 Objetivo de la Calidad de Software. El objetivo que persigue la calidad en el software está orientada a:

- Incrementar la producción y rendimiento a los profesionales en determinadas áreas de trabajo e investigación.
- Mejorar la calidad del producto del software.
- Suministrar técnicas para automatizar el manejo de datos.
- Realizar una planeación eficaz en el desarrollo de software.
- Tener documentación adecuada del sistema.
- Validar y controlar formalmente la calidad del trabajo realizado.
- Cumplir con los requerimientos de la empresa en cuanto a rendimiento y productibilidad.
- Brindar seguridad y fiabilidad a la empresa o usuario final.
- Agilizar procesos y garantizar su correcto desempeño a la hora de evaluar resultados.

2.3.5 Obtención de un software de calidad. Cuando se habla de la obtención de un software de calidad, inmediatamente se debe visualizar las metodologías o procedimientos existentes para garantizar un software de calidad. Esto incluye las fases de producción de software como análisis, diseño, programación y prueba del software, con el fin de garantizar unos estándares que midan aspectos claves en un software, *Factores de Calidad*, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la Calidad del Software.

Para la descripción y obtención de un software de calidad es necesario llevar a cabo un *PLAN DEL ASEGURAMIENTO DE LA CALIDAD, el cual se aplica* en el Capitulo 3.4. Plan del aseguramiento de la calidad.

2.4 ESTIMACIÓN DE PROYECTOS

La utilización de metodologías tradicionales para la estimación de proyectos software ha resuelto correctamente la necesidad de conocer la duración de un proyecto como una variable dependiente de los recursos a emplear. Técnicas como los son los Puntos de Función y COCOMO se utilizan para establecer una estimación dependiente de un conjunto de variables consideradas en un proyecto para establecer una estimación mas precisa del mismo.

2.4.1 Puntos de Función (ALBRECHT AND HIS COLLEAGUES AT IBM)[5]. El objetivo principal del análisis de puntos por función es medir la funcionalidad de las aplicaciones, basándose en el diseño lógico y de acuerdo con la perspectiva del usuario cumplir con las siguientes características:

- Establecer una unidad estándar de medida para una aplicación
- Minimizar el gasto y el esfuerzo, mediante el aporte de las medidas.
- Ser comprensible por el personal no técnico, facilitando el entendimiento por parte de usuarios finales.
- Medir independientemente de la tecnología utilizada.

Entre los beneficios principales se encuentran:

[5] Puntos de función Albrecht. Universidad del país Vasco. Citada el 4 de junio de 2005. Disponible en http://www.sc.ehu.es/jiwdocoj/mmis/fpa.htm

- La medición de estimaciones de costo y recursos que se requieren para el desarrollo o mantenimiento de una aplicación.
- Posibilitar la implementación de un sistema de métrica.
- Apoyar la calidad y la productividad ofreciendo una visión optimizada de los procesos de desarrollo de aplicaciones.
- Servir de ayuda para determinar la compra de un paquete o el desarrollo de la aplicación en la empresa.

A continuación se describe una visión general del análisis de Puntos de Función en Desarrollo de Software*:*

- Debemos establecer el tipo de dimensión en la cual va a ser aplicada la métrica de Puntos de Función.
- Seguidamente definir las fronteras de la aplicación.
- La frontera de la aplicación separa el proyecto que está siendo estudiado de las aplicaciones externas (otros sistemas de la organización). En este punto hay que definir dónde comienza y dónde termina la aplicación o proyecto, cuáles son sus alcances (que hace, para que servirá) y sus límites (que no hace, para que no sirve).

Establecer las fronteras de la aplicación ayuda a tener una visión más clara del ambiente del producto que se está midiendo.

Como conclusión, al final de todo el proceso, se obtendrá un valor número que mide el tamaño de la aplicación en puntos por función (de igual forma que una construcción se mide en metros cuadrados). Este valor ayudará a realizar estimaciones de plazos, costos y también puede apoyar diversas técnicas para el controlar la calidad del software.

2.4.2 COCOMO. Barry Boehm, en su libro clásico sobre economía de la Ingeniería del Software, introduce una jerarquía de modelos de estimación de

Software con el nombre de **COCOMO**, por su nombre en Ingles (Constructive, Cost, Model) modelo constructivo de costos, ayuda a la planeación y estimación del calendario para un proyecto de desarrollo de software. Diversos paquetes de SW que lo soportan.

Uso de COCOMO:

- Decisiones de inversión.
- Establecer presupuesto y calendario.
- Negociación de costo / calendario / desempeño.
- Administración de riesgos.
- Decisiones de mejora.

La jerarquía de modelos de Boehm esta constituida por los siguientes:

- Modelo I. El Modelo COCOMO básico calcula el esfuerzo y el costo del desarrollo de Software en función del tamaño del programa, expresado en las líneas estimadas.

- Modelo II. El Modelo COCOMO intermedio calcula el esfuerzo del desarrollo de software en función del tamaño del programa y de un conjunto de conductores de costos que incluyen la evaluación subjetiva del producto, del hardware, del personal y de los atributos del proyecto.

- Modelo III. El modelo COCOMO avanzado incorpora todas las características de la versión intermedia y lleva a cabo una evaluación del impacto de los conductores de costos en cada caso (análisis, diseño, etc.) del proceso de ingeniería de Software.

2.5 MÉTRICAS DE CALIDAD

Las Métricas de Calidad proporcionan una indicación de cómo se va a medir para que el software cumpla con los requisitos implícitos y explícitos del cliente. Es decir cómo se va medir para que mi sistema se adapte según el análisis de los requisitos que son de prioridad para el interesado.

Primeros pasos para iniciar un programa de métricas.

Se definen diez pasos "para el éxito" que pueden ser tomados como base para la iniciación de un programa de métricas[6]. Esos pasos son los siguientes:

1. Definir los objetivos del programa: Estos deben proporcionar una indicación concreta de cómo se espera que un programa de métricas sirva a la organización para:

   a) contribuir a la mejora de la calidad de las aplicaciones producidas;
   b) contribuir a la mejora del proceso de desarrollo;
   c) contribuir a la implantación de nuevas tecnologías.

2. Asignar responsabilidades: Deben seleccionarse aquellas personas que estén convencidas de la necesidad de un programa de métricas y a las que se les dará la responsabilidad de llevar adelante el proyecto.

3. Investigar: Antes de comenzar el programa de métricas deben "hacerse los deberes". Es decir, debe obtenerse y consultarse la amplia literatura que existe sobre el tema.

[6] Grady y Caswell. Software Metrics: Establishing a Company-Wide Strategy, Prentice-Hall, 1987

4. Seleccionar las métricas iniciales: En la etapa inicial de un programa de métricas se debería tratar de que las cosas fuesen lo más simples que sea posible. Para ello es recomendable recolectar unas pocas métricas sobre las que se haya obtenido consenso, relacionadas con calidad y productividad.

5. Hacer el "marketing" del programa: Esa "venta" - en el buen sentido de la palabra - debe hacerse hacia arriba y hacia abajo. Es fundamental no sólo obtener el apoyo de los que deberán apoyar y sostener el programa (y no esperar resultados a corto plazo), sino también el apoyo - y convencimiento - del personal de desarrollo y mantenimiento y de sus líderes, que deben asumir el compromiso de obtener los datos más exactos posibles.

6. Obtener herramientas: El uso de herramientas facilita y hace consistente el proceso de captura de las métricas, y permite su registro en una forma ordenada y accesible. (No obstante, antes de adquirir herramientas debería obtenerse un cierto conocimiento en la recolección, análisis, y uso y presentación de métricas).

7. Establecer un programa de capacitación: Este paso se debería realizar en conjunto con el Numeral 5. Antes de iniciar un programa de métricas, la gente debe saber qué recolectar y para qué.

8. Alentar la participación: Esto se puede realizar en parte a través de la difusión de los éxitos y la re alimentación de la información obtenida, y a través de pedir a los participantes que hagan llegar sus ideas. Si se toma alguna acción sobre esas ideas se contribuirá a hacer que el programa se sienta como propiedad "de todos".

9. Crear una base de datos de métricas Inicialmente los datos serán simples y escasos, pero con el tiempo se podrá ir expandiendo el alcance de las mediciones.

10. Ser flexibles: Hay que tener presente que las guías y procedimientos que se establezcan al principio del programa de métricas irán evolucionando con el tiempo.[7]

[7] Métricas: Justificación - Conrado Estol. 2001

3 CALIDAD EN EL DISEÑO DE SOFTWARE

Con el paso del tiempo, la calidad del software ha adquirido diferentes conceptos, significados, percepciones; debido a que diferentes autores a nivel mundial expertos en el tema lo han afirmado cada uno desde sus diferentes puntos de vista.

Según:

- Deming, como el grado predecible de uniformidad y conformidad a un bajo costo que se ajuste a las necesidades del mercado.
- Crosby, como cumplir con los requisitos.
- Feigenbaum, como el conjunto total de las características del producto de marketing, ingeniería, fabricación y mantenimiento a través del cual el producto en uso satisfará las expectativas del cliente.
- Jurán, como la idoneidad o aptitud para el uso.

La mayoría de estas definiciones muestra el grado de satisfacción que desea el cliente para con el producto, y los requisitos explícitos que deben existir para lograrlo.

3.1 CONCEPTOS FUNDAMENTALES DEL DISEÑO

El diseño del software es un proceso iterativo a través del cual se traducen los requisitos en una representación del software. Es decir, el diseño se representa a un alto nivel de abstracción, un nivel que se puede seguir hasta requisitos específicos de datos, funcionales y de comportamiento. A medida que ocurren iteraciones del diseño, el refinamiento subsiguiente lleva a representaciones del

diseño de mucho menor nivel de abstracción. Estos todavía pueden ser seguidos hasta los requisitos, pero la conexión es mucho más sutil.

3.2 DISEÑO Y CALIDAD DEL SOFTWARE

A lo largo del proceso de diseño, se evalúa la calidad del diseño con una serie de revisiones técnicas formales. McGlaughlin [McG91] sugiere tres características que sirven de directrices para la evaluación de un buen diseño:

- El diseño debe implementar todos los requisitos explícitos contenidos en el modelo de análisis y debe acomodar todos los requisitos implícitos que desea el cliente.
- El diseño debe ser una guía que puedan leer y entender los que construyan el código y los que prueban y mantienen el software.
- El diseño debería proporcionar una completa idea de lo que es el software, enfocando los dominios de datos, funcional y de comportamiento desde la perspectiva de la implementación.

Cada una de estas características es de hecho un objetivo del proceso del diseño.

Pero, ¿cómo se alcanza cada uno de estos objetivos?

Para evaluar la calidad de una representación del diseño, se deben establecer unos criterios técnicos para un buen diseño.

1. Un diseño debería presentar una organización jerárquica que haga un uso inteligente del control entre los componentes del software.
2. El diseño debería ser modular; es decir, se debería hacer una partición lógica del software en elementos que realicen funciones y sub-funciones específicas.

3. Un diseño debería contener abstracciones de datos y procedimentales.
4. Un diseño debería producir módulos (ejemplo: subrutinas o procedimientos) que presenten características funcionales independientes.
5. Un diseño debería conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el entorno exterior.
6. Se debería producir un diseño usando un método que pudiera repetirse según la información obtenida durante el análisis de requisitos del software.

Estos criterios no se consiguen por casualidad. El proceso de diseño del software exige un buen diseño a través de la aplicación de principios fundamentales de diseño, metodología sistemática y una revisión exhaustiva.

Toda actividad intelectual se caracteriza por un conjunto de conceptos, fundamentales y de técnicas específicas. Las técnicas son la manifestación de los conceptos en su aplicación a situaciones particulares. Las técnicas vienen y van con los cambios tecnológicos, las modas intelectuales, las condiciones económicas y las preocupaciones sociales. Por definición, los principios fundamentales permanecen iguales a través del tiempo, proporcionando las bases fundamentales para el desarrollo y la evaluación de las técnicas. Los conceptos fundamentales en el diseño de la programación incluyen:

- Abstracción
- Estructura
- Guardado de información
- Modularidad
- Concurrencia
- Verificación
- Aspectos estéticos en el diseño.

3.3 ESTRUCTURA DEL GRUPO DE PROGRAMACIÓN

Todo equipo de programación debe tener una estructura interna la cual garantizará un óptimo desarrollo del mismo. Las estructuras básicas son: el grupo democrático, en el que todos los miembros participan en todas las decisiones; el grupo con jefe de programadores, en el que otros miembros del equipo apoyan y auxilian al jefe, por último, el grupo jerárquico que combina aspectos de los dos anteriores. Pueden existir variaciones de las tres estructuras mencionadas.
Un proyecto grande puede ocupar varios equipos de programadores, en cuyo caso cada uno tiene la responsabilidad de su estructura interna y de sus relaciones con los demás. Independientemente de la estructura del equipo, deben limitarse a no más de cinco a siete miembros. Esta restricción controla el número de trayectorias de comunicación en el equipo y permite una coordinación eficiente de las actividades.

Las características de cada estructura y consideraciones sobre su planeación se analizan en los párrafos siguientes:

3.3.1 Grupos democráticos. El grupo democrático ideal fue descrito primero por Weinberg como el equipo sin egoísmo (WEI71). La estructura de la administración y las trayectorias de la comunicación. Las metas y decisiones se definen por consenso, el liderazgo rota de un miembro a otro dependiendo de las tareas que se realicen y de las capacidades particulares de cada miembro. Los productos (requisitos, diseño, código fuente, manual del usuario, etc.) se discuten abiertamente y son examinados con libertad por todos los miembros.
Las ventajas de un grupo democrático incluyen la oportunidad de todos los miembros de contribuir a las decisiones, de aprender uno de otro, y la satisfacción que produce trabajar en un ambiente bien comunicado y sin presiones. Mantei sugiere que esta estructura es apropiada para proyectos de investigación y desarrollo largos y difíciles (MAN81).

Las desventajas de esta estructura son la cantidad de comunicación necesaria para tomar decisiones. El requisito de que todos los miembros trabajen juntos y la falta de autoridad y responsabilidad que puede ocurrir, lo que producirá menos iniciativa.

3.3.2 Grupos con jefe de programación. En contraste con el grupo democrático, estos grupos son muy estructurados. El jefe diseña el producto, instrumenta partes críticas de él, y toma las decisiones técnicas importantes; también asigna el trabajo de los programadores, y éstos en número de dos a cinco, escriben el código, lo depuran, documentan y prueban.

Las ventajas de esta estructura son las decisiones centralizadas y la reducción en las trayectorias de comunicación; sin embargo, su eficacia es muy sensible a las capacidades técnicas y administrativas del jefe, lo cual puede producir desmoralización en los programadores subordinados.

3.3.3 Grupos bajo jerarquía administrativa. Esta estructura ocupa un punto intermedio entre los grupos democrático y de jefe. En un grupo jerárquico, el líder del proyecto asigna tareas, asiste a revisiones y recorridos, detecta áreas de problemas, balancea las cargas de trabajo y participa en actividades técnicas.

Una desventaja importante de esta estructura es que los programadores técnicamente más competentes, tienden a ser promovidos hacia posiciones administrativas, lo cual suele desear el programador porque le proporciona prestigio y salario.

3.4 PLAN DEL ASEGURAMIENTO DE LA CALIDAD

Para el desarrollo de un *plan de Aseguramiento de la calidad* se precisan cada uno de los factores y prioridades del producto, de tal forma que se presenten de una manera clara y precisa los puntos a desarrollar para cada método propuesto.

El contenido del plan de aseguramiento incluye:

1. Propósito
2. Acrónimos y Abreviaturas
3. Referencias
4. Gestión
    - 4.1. Organización
    - 4.2. Tareas
        - 4.2.1. Planificación de Calidad
        - 4.2.2. Identificar Propiedades de Calidad
    - 4.3. Responsabilidades
5. Documentación
    - 5.1. Documentación mínima requerida
    - 5.2. Otra documentación
6. Métricas
7. Revisiones
    - 7.1. Descripción
        - 7.1.1. Revisión de Calidad de Producto
        - 7.1.2. Revisión de Ajuste al Proceso
        - 7.1.3. Revisión Técnica Formal (RTF)
    - 7.2. Requerimientos Mínimos
    - 7.3. Agenda
        - 7.3.1. Fase I – Inicial

7.3.2. Fase II – Elaboración
7.3.3. Fase III – Construcción
7.3.4. Fase IV – Transición

8. Verificación
9. Reporte de problemas y acciones correctivas
10. Herramientas, técnicas y metodologías
11. Gestión de configuración
    11.1. Métodos para la gestión de configuración
    11.2. Actividades para asegurar la calidad
12. Gestión de riesgos
    12.1. Métodos para la gestión de riesgos
    12.2. Actividades para asegurar la calidad
13. Correspondencia de este plan con el estándar IEEE 730

3.5 NORMA DE CALIDAD ISO

Un conjunto de normas internacionales genéricas que establecen sistemas de gestión de la calidad aplicados por organizaciones de cualquier tipo o tamaño que fabrican productos o componentes (hardware), fabrican software, fabrican materiales procesados, ofrecen servicios, desempeñan funciones de administración pública.

¿Qué no son las normas ISO?

- No son especificaciones de calidad de productos.
- No son obligatorias.
- No es un programa de corta duración.
- No es el punto final de la mejora continua

La Norma ISO 9001 se utiliza para establecer un sistema de gestión que proporcione confianza en la conformidad de un producto con los requisitos especificados. Ahora es la única norma de la familia ISO 9000 en base a la cual esos requisitos del sistema de la calidad pueden ser certificados por una entidad externa. La norma establece que la palabra "producto" se aplica a servicios, material procesado, hardware y software diseñado por, o pedido por, el cliente.

3.6 CONTROL DEL DISEÑO

ISO 9001:1994 Modelo para aseguramiento de calidad en el diseño, desarrollo, producción, instalación y servicio.

Este apartado es el que se encarga de controlar y verificar el diseño de los programas y aplicaciones software que se van a desarrollar para controlar que cumplan los requisitos especificados en cada proyecto.

3.6.1 Aplicación

1. Utilizar una notación adecuada.
2. Proporcionar herramientas de validación y verificación.
3. Proporcionar información suficiente sobre los distintos elementos que forman parte del diseño, así de cómo se han obtenido y su desarrollo a lo largo de todo el proyecto hasta llegar al diseño del mismo.
4. Proporcionar la documentación necesaria para comprobar que la especificación de requerimientos se satisface en el diseño.
5. Llevar un control de las configuraciones.
6. Documentar todo lo que va pasando en la fase de desarrollo.
7. Revisar todo elemento que no responda a los requerimientos del sistema, o que no sea estrictamente necesario.

3.6.2 Evaluación de Especificación de Requisito. La revisión de la especificación de requisitos del software (y/o prototipo) es llevada a cabo tanto por el desarrollador del software como por el cliente. Como la especificación forma el fundamento para el diseño y las subsiguientes actividades de ingeniería del software, se debería poner extremo cuidado al realizar la revisión.

Inicialmente la revisión se lleva acabo a nivel macroscópico. A este nivel, los revisores intentan asegurarse de que la especificación está completa, es consistente y exacta. Se estudian teniendo en cuenta los siguientes parámetros.

1. ¿Permanecen las metas y objetivos declarados para el software consistentes con las metas y objetivos del sistema?
2. ¿Se han descrito todas las interfaces importantes de todos los elementos del sistema?
3. ¿Están adecuadamente definidos el flujo y la estructura de la información para el dominio del problema?
4. ¿Son claros los diagramas?
5. ¿Son autosuficientes sin texto suplementario?
6. ¿Permanecen las funciones principales dentro del ámbito y se han descrito adecuadamente?
7. ¿Es consistente el comportamiento del software con la información que debe procesar y con las funciones que debe realizar?
8. ¿Son realistas las restricciones del diseño?
9. ¿Se han considerado los riesgos tecnológicos del desarrollo?
10. ¿Se han considerado requisitos alternativos del software?

11. ¿Se han establecido en detalle los criterios de validación?
12. ¿Son adecuados para describir un buen sistema?
13. ¿Existen inconsistencias, omisiones o redundancias?
14. ¿Se ha estado en contacto permanente con el cliente?
15. ¿Ha revisado el usuario el manual de usuario preliminar o el prototipo?
16. ¿Cómo afectan a las estimaciones de la planificación?

3.6.3 Desarrollo de la Evaluación de Especificación de Requisitos. Para desarrollar las respuestas a muchas de las cuestiones anteriores, la revisión puede enfocarse en un nivel detallado. Aquí, la preocupación está en la redacción de la especificación. Se intenta descubrir los problemas que pueden estar ocultos dentro del contenido de la especificación. Se sugieren las siguientes directrices para una revisión detallada de la especificación.

1. Tenga cuidado con los conectores persuasivos (Ejemplo: ciertamente, por tanto, claramente, obviamente, siguiendo lo anterior), y pregúntese ¿Por qué?
2. Tenga cuidado con los términos imprecisos (Ejemplo: algunos, a veces, a menudo, normalmente, ordinariamente, en la mayoría de los casos, mayoritariamente); busque una clarificación
3. Cuando se dan listas incompletas, asegúrese de que comprende todos los elementos. Las claves son: «etc., y así sucesivamente, tales como».
4. Asegúrese de que los rangos establecidos no, contienen supuestos no declarados (Ejemplo: Los códigos válidos van de 10 a 100. ¿Enteros? ¿Reales? ¿Hexadecimales?).

5. Cuidado con los verbos de significados imprecisos como «manejado, rechazado, procesado, ignorado, eliminado». Pueden interpretarse de muchas maneras.

6. Cuidado con los pronombres de significados ambiguos (Ejemplo: El módulo y/o se comunica con el módulo de validación de datos y se activa su indicador de control. ¿El indicador de control de quién?)

7. Busque frases que impliquen certidumbre (Ejemplo: siempre, todos, cada, ninguno, nunca), y exija pruebas.

8. Cuando un término es definido explícitamente en un sitio, intente sustituir la definición por otras apariciones del término.

9. Cuando se describe una estructura con palabras, dibújela para ayudar a su comprensión.

10. Cuando se especifica un cálculo, haga al menos dos ejemplos.

3.7 MÉTRICAS DE DESARROLLO

3.7.1 Recursos humanos. El planificador comienza evaluando el ámbito y seleccionando las habilidades técnicas que se requieren para llevar a cabo el desarrollo. Hay que especificar tanto la posición dentro de la organización (Ejemplo: gestor, ingeniero de software, etc.) como la especialidad (Ejemplo: telecomunicaciones, bases de datos, microprocesadores). Para proyectos relativamente pequeños (una persona-año o menos) una sola persona puede llevar a cabo todos los pasos de ingeniería del software, consultando con especialistas siempre que lo requiera. El número de personas requerida para un proyecto de software sólo puede ser determinado después de hacer una estimación del esfuerzo del desarrollo (Ejemplo: personas mes o personas año).

3.7.2 Recursos del hardware. Dentro del contexto de los recursos, el hardware también es una herramienta para el desarrollo del software. Durante la

planificación del proyecto de software se deben considerar tres categorías de hardware: el sistema de desarrollo, la máquina objetivo y los demás elementos de hardware del nuevo sistema. El sistema de desarrollo (también denominado sistema anfitrión) está compuesto por la computadora que se, utilizará durante la fase de desarrollo del software y sus periféricos asociados. Por ejemplo, se puede utilizar una computadora de 32 bits como sistema de desarrollo de un software que, eventualmente, se ejecutará en un microprocesador de 16 bits la máquina objetivo. Puede que se utilice como sistema de desarrollo porque soporte múltiples usuarios, porque pueda mantener amplios volúmenes de información a ser compartidos por los miembros del equipo de desarrollo de software o porque soporte una gran variedad de herramientas.

Dado que la mayor parte de las organizaciones de desarrollo tienen múltiples departamentos que requieren acceso al sistema de desarrollo, el planificador debe determinar cuidadosamente la ventana temporal requerida y comprobar que el recurso vaya a estar disponible. Se pueden especificar como recursos para el desarrollo de software otros elementos de hardware del sistema basado en computadora. Por ejemplo, el software de control numérico (CN) utilizado en ciertas máquinas puede requerir una máquina herramienta específica (p. ej.: un tomo de CN) para la validación en el paso de prueba; un proyecto de software de composición automática puede requerir una máquina de fotocomposición en algún momento del desarrollo. El planificador debe especificar cada elemento del hardware.

3.7.3 Recursos de software. Igual que utilizamos hardware como herramienta para construir nuevo hardware, utilizamos software como ayuda en el desarrollo de nuevo software. La primera aplicación que se le dio al software en el desarrollo de software fue lo que se denominaba reconstrucción. Se comenzó escribiendo un primitivo traductor de lenguaje ensamblador a lenguaje máquina y se usó para

desarrollar un ensamblador más sofisticado. Aumentando las posibilidades de cada versión previa, los equipos de desarrollo fueron reconstruyendo eventualmente el software hasta llegar a construir compiladores de lenguajes de alto nivel y otras herramientas que son parecidas en muchos casos a las herramientas que utilizan los ingenieros del hardware en el diseño y la ingeniería asistidos por computadora

3.7.4 Documentación de Código. La documentación interna del código fuente comienza con la elección de los nombres de los identificadores (variables y etiquetas), continúa con la localización y la composición de los comentarios y termina con la organización visual del programa.

La elección de nombres de identificadores significativos es crucial para la legibilidad. Los lenguajes que limitan la longitud de los nombres de las variables o de las etiquetas a unos pocos caracteres, implícitamente limitan la comprensión.

Considere las tres siguientes sentencias:

- Expresión en Basic: D= V* T
- Expresión en Fortran: DIST = VELHOR * TIEMPO
- Expresión en Algol: DISTANCIA = VELOCIDAD.HORIZONTAL * TIEMPO. TRANSCURRIDO. EN. SEGS.

La expresión del lenguaje BASIC es innegablemente concisa, pero el significado de D= V * T no es nada claro a no ser que el lector tenga cierta información previa. La expresión de FORTRAN proporciona más información, pero el significado de DIST y VELHOR se puede interpretar mal. La sentencia de ALGOL no deja lugar a dudas sobre el significado del cálculo. Estas sentencias ilustran la forma en que se

deben elegir los identificadores para ayudar a la documentación del código. Se puede argumentar que las expresiones llenas de palabras (como la sentencia ALGOL anterior) oscurecen el flujo lógico y hacen más difíciles las modificaciones. Obviamente hay que aplicar el sentido común al seleccionar los identificadores. Los identificadores innecesariamente largos pueden ser una fuente potencial de error (por no mencionar el dolor de espalda por estar muchas horas sentado escribiendo en un terminal de desarrollo). Sin embargo, ciertos estudios indican que, incluso para pequeños programas, la elección de identificadores significativos mejora la comprensión. En términos del modelo sintáctico/semántico, los nombres significativos "simplifican la conversión de la sintaxis del programa a la estructura semántica interna".

La posibilidad de expresar los comentarios en lenguaje natural como parte del listado del código fuente es algo que aparece en todos los lenguajes de programación de propósito general. Sin embargo, surgen ciertas inquietudes:

- ¿Cuántos comentarios son "suficientes"?
- ¿Dónde se deben situar los comentarios?
- ¿Oscurecen los comentarios la lógica del programa?
- ¿Pueden los comentarios distraer al lector?
- ¿Son los comentarios "no mantenibles", y, por tanto, no fiables?

Hay pocas respuestas definitivas a las preguntas anteriores. Pero una está clara: El software debe contener documentación interna.

Los comentarios permiten al programador comunicarse con otros lectores del código fuente y puede resultar una clara guía de comprensión durante la última fase de la ingeniería del software el mantenimiento.

Existen muchos modelos propuestos para la creación de comentarios. Los comentarios de prólogo y los comentarios descriptivos son dos categorías que requieren enfoques algo diferentes. Al principio de cada módulo debe haber un comentario de prólogo.

El formato para esos comentarios es:

1. Una sentencia de propósito que indique la función del módulo. Una descripción de la interfaz que incluya:
    - Un ejemplo de "secuencia de llamada”.
    - Una descripción de todos los argumentos
    - Una lista de todos los módulos subordinados K
2. Una explicación de los datos pertinentes, tales como las variables importantes y su uso, de las restricciones y limitaciones y de otra información importante.
3. Una historia del desarrollo que incluya:
    - El diseñador del módulo (autor)
    - El revisor (auditor) y la fecha
    - Fechas de modificación y de descripción "'.

Los comentarios descriptivos se incluyen en el cuerpo del código fuente y se usan para describir las funciones de procesamiento. YanTassel [Y AN78] ex presa una importante recomendación para crear esos comentarios al decir: "los comentarios

deben proporcionar algún extra, no sólo parafrasear el código " Además los comentarios descriptivos deben:

Describir los bloques de código en lugar de comentar cada línea. Usar líneas en blanco o tabulaciones de forma que sean fácilmente distinguibles del Código. Que sean correctos; un comentario incorrecto o que se pueda interpretar mal es peor que no ponerlo. Con unos recursos nemotécnicos apropiados para los identificadores: unos buenos comentarios, se asegura una documentación interna adecuada.

Cuando se representa un diseño de procedimientos detallado mediante un lenguaje de diseño de programas, se puede incluir directamente la documentación del diseño en el listado fuente como sentencias de cometario. Esta técnica es particularmente útil cuando la implementación se lleva a cabo en lenguaje ensamblador y ayuda a asegurar que, tanto el código como el diseño, podrán fácilmente mantenerse al hacer cambios sobre ellos. La forma en que el código fuente aparece en el listado es una importante contribución a la legibilidad. El sangrado del código fuente realza las construcciones lógicas y los bloques de código, tabulando desde el margen izquierdo de forma que se, vean deslazados esos atributos. Al igual que los comentarios, no está clara cuál es la mejor forma de sangrar. El sangrado manual puede: llegar a ser complicado al tener que modificar el código y algunos experimentos indican que sólo una mejora marginal en la inteligibilidad. Probablemente, la mejor aproximación sea usar una herramienta de formato automático de código que sangre adecuadamente el código fuente. Al eliminar el exceso de sangrado del código, el codificador, el formulario puede ser mejorado con relativamente poco esfuerzo.

3.7.5 Recursos. La segunda tarea de la planificación del desarrollo de software es la estimación de los recursos requeridos para acometer el esfuerzo de desarrollo de software. En la base se encuentran las herramientas existentes hardware y software para dar soporte al esfuerzo de desarrollo. En el nivel más alto se encuentra el recurso primario que se requiere siempre la gente. Cada recurso queda especificado mediante cuatro características: descripción del recurso, informe de disponibilidad, fecha cronológica en la que se requiere el recurso, tiempo durante el que será aplicado el recurso. Las dos últimas características pueden verse como una ventana temporal. La disponibilidad del recurso para una ventana específica tiene que establecerse lo más pronto posible.

3.7.6 Herramientas para el Desarrollo. Existen las herramientas de tipo front-end y las back-end. Las herramientas de "tipo front-end automatizan las primeras actividades del proceso de desarrollo de sistemas. Entre los muchos aspectos que se toman en cuenta al desarrollar herramientas para esta fase, se hallan las técnicas de soporte para ayudar al analista a preparar especificaciones formales que carezcan de ambigüedades, a validar las descripciones del sistema con el objeto de determinar su consistencia y completes, ya seguir la evolución de los requerimientos de la aplicación en características que formen parte del sistema que finalmente será implantado. Hasta donde sea posible, esta ayuda debe ser automatizada (por ejemplo, la computadora valida automáticamente las descripciones del sistema). A menudo, las herramientas de tipo front-end proporcionan soporte para el desarrollo de modelos gráficos de sistemas y procesos. Los diagramas de flujo de datos son representativos de este tipo de herramienta. Los diagramas de flujo de datos representan en forma gráfica (más que por escrito) los procesos y flujos de datos del sistema.

Las herramientas de tipo back-end tienen como finalidad ayudar al analista a formular la lógica del programa, los algo ritmos de procesamiento y la descripción

física de datos, también ayudan ala interacción con los dispositivos (para entrada y salida), etc. Estas actividades convierten los desafíos lógicos del software en un código de programación que es el que finalmente da existencia a la aplicación.

Dado que su empleo esta destinado al desarrollo de software, este tipo de herramienta también se conoce como herramientas para programación asistida por computadora.

3.7.7 Herramientas asistidas por computadora para la ingeniería de sistemas CASE). Las siglas CASE se emplean con bastante frecuencia en la comunidad de sistemas de información para denotar la ingeniería de sistemas asistida por computadora o la ingeniería de software asistida por computadora. Aunque el uso de este último término está más diseminado, el primero es más exacto ya que el objetivo a largo plazo de las herramientas CASE es automatizar los aspectos clave de todo el proceso de desarrollo, desde el principio hasta el final. Para aquellos que emplean el término ingeniería de software asistida por computadora, hacemos mención de que el desarrollo de una aplicación comienza con la especificación de requerimientos, no con la codificación del software. Es así como las extensiones de CASE hacen referencia al mismo proceso.

4 CALIDAD EN EL PRODUCTO

El objetivo de aplicar las métricas de Calidad no es necesariamente alcanzar una calidad perfecta, sino la necesaria y suficiente para cada contexto de uso a la hora de la entrega y del uso por parte de los usuarios.

Por lo tanto es necesario comprender las necesidades reales de los usuarios con tanto detalle como sea posible (requisitos).

4.1 DIFERENTES ASPECTOS DE LA CALIDAD

Asegurar la calidad en la elaboración de un software implica llevar procedimientos y metodologías que encaminan a cumplir cada una de las especificaciones y prioridades (requisitos) que se determinaron sobre el producto.

Dentro de las pautas se incluyen 3 aspectos fundamentales:

- Interna: Medible a partir de las características intrínsecas, como el código fuente.
- Externa: Medible en el comportamiento del producto, como en una prueba
- En uso: durante la utilización efectiva por parte del usuario

La calidad en el producto (obtención de un Software de Calidad) según las prioridades de cada cliente (propiedades de calidad), se forma desde el momento en que se define un *Proceso de calidad* a seguir. Por consiguiente, un proceso depende de lo realizado en el paso anterior para alcanzar el objetivo o requisito esperado. (Ver Figura 1)

4.2 CALIDAD EN EL CICLO DE VIDA DEL SOFTWARE

El desarrollo de un Software esta relacionado a un *Ciclo de vida*, comprendido por una serie de fases. Cada fase es un aspecto de calidad del software aplicado donde la fase numero 2 depende de la fase numero uno en términos de calidad. (Ver Figura 2).

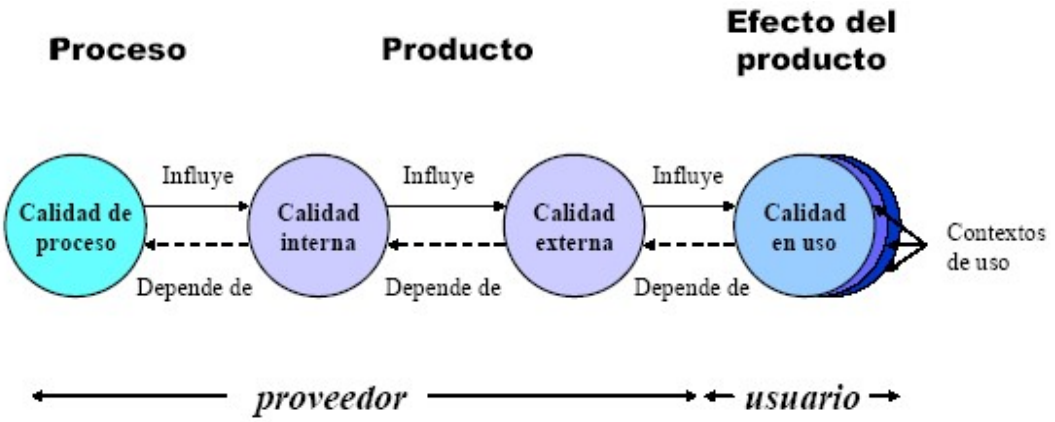


Figura 1. Diferentes Aspectos de la Calidad del Software[8].

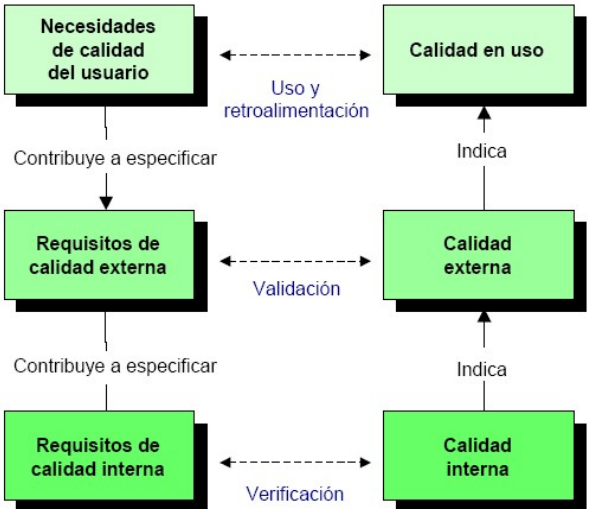


Figura 2. Calidad en el Ciclo de Vida del Software[9]

[8] Grupo de investigación Kybele. Ingeniería del Software y Base de datos. Calidad del Producto. Citado el 4 de julio de 2005. Disponible en: http://kybele.escet.urjc.es/documentos/GCSW/T4-CalidadProducto.pdf

4.3 CARACTERÍSTICAS, SUBCARACTERÍSTICAS Y ATRIBUTOS DE CALIDAD

Los atributos de calidad son las pautas que debe tener un software para satisfacer los requerimientos del usuario. Atributos internos y externos utilizados en la implementación de cada proceso o subproceso involucrado durante el desarrollo del a aplicación. (Ver Figura 3)

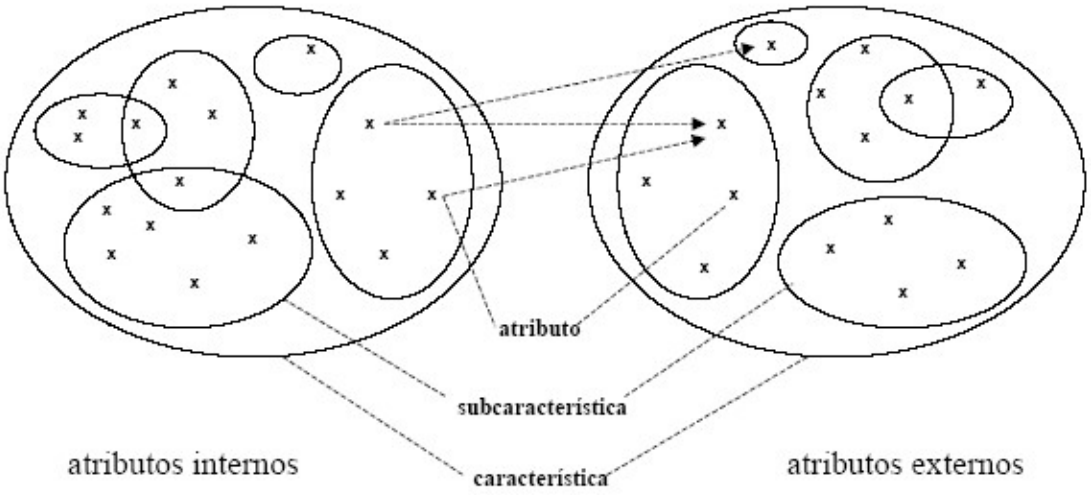


Figura 3. Características, Subcaracterísticas y Atributos de Calidad[10]

[9] Grupo de investigación Kybele. Ingeniería del Software y Base de datos. Calidad del Producto. Citado el 4 de julio de 2005. Disponible en: http://kybele.escet.urjc.es/documentos/GCSW/T4-CalidadProducto.pdf

[10] Grupo de investigación Kybele. Ingeniería del Software y Base de datos. Calidad del Producto. Citado el 4 de julio de 2005. Disponible en: http://kybele.escet.urjc.es/documentos/GCSW/T4-CalidadProducto.pdf

4.4 MODELO DE MCCALL ET AL. (1977)

El modelo de McCall organiza los factores en tres ejes o puntos de vista desde los cuales el usuario puede contemplar la calidad de un producto, basándose en once factores de calidad organizados en torno a los tres ejes y a su vez cada factor se desglosa en otros criterios. (Ver Figura 4)

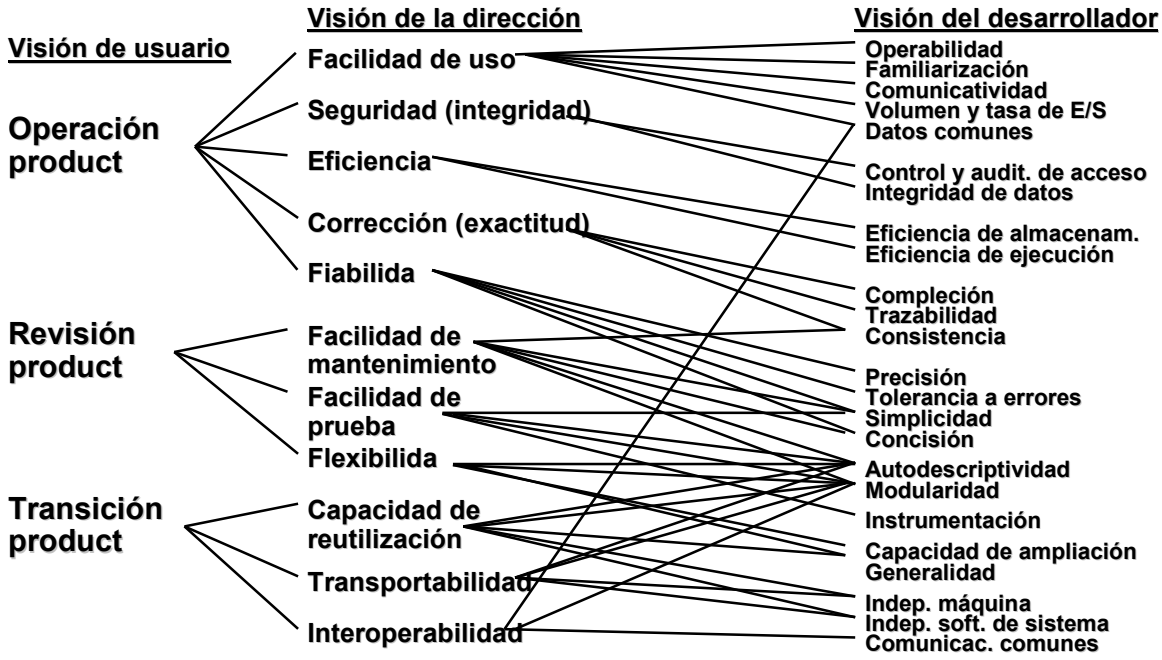


Figura 4. Modelo de McCall et al. (1977)[11]

[11] Grupo de investigación Kybele. Ingeniería del Software y Base de datos. Calidad del Producto. Citado el 4 de julio de 2005. Disponible en: http://kybele.escet.urjc.es/documentos/GCSW/T4-CalidadProducto.pdf

4.5 ISO/IEC 9126: TECNOLOGÍAS DE LA INFORMACIÓN CALIDAD DE LOS PRODUCTOS SOFTWARE.

Parte 1: Modelo de Calidad

Parte 2: Métricas Externas

Parte 3: Métricas Internas

Parte 4: Métricas de Calidad en Uso

Ejemplos de Calidad en Uso

- Validar la completitud de una definición de requisitos;
- Identificar objetivos para el diseño software;
- Identificar requisitos para las pruebas del software;
- Identificar requisitos para el aseguramiento de la calidad;
- Identificar criterios de aceptación para un producto software completado.

4.6 MODELO DE CALIDAD PARA CALIDAD INTERNA Y EXTERNA

Figura 5. Modelo para la Calidad Interna y Externa[12]

4.6.1 Funcionalidad. Existen diferentes conceptos:

- Adecuación: Capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados.
- Exactitud: Capacidad del producto software para proporcionar los resultados o efectos correctos o acordados, con el grado necesario de precisión.
- Interoperabilidad: Capacidad del producto software para interactuar con uno o más sistemas especificados.
- Seguridad de acceso: Capacidad del producto software para proteger información y datos de manera que las personas o sistemas no autorizados no puedan leerlos o modificarlos, al tiempo que no se deniega el acceso a las personas o sistemas autorizados
- Cumplimiento funcional: Capacidad del producto software para adherirse a normas, convenciones o regulaciones en leyes y prescripciones similares relacionadas con funcionalidad.

4.6.2 Fiabilidad. Se relacionan los siguientes conceptos:

- Madurez: Capacidad del producto software para evitar fallar como resultado de fallos en el software.
- Tolerancia a fallos: Capacidad del software para mantener un nivel especificado de prestaciones en caso de fallos software o de infringir sus interfaces especificados.

[12] Grupo de investigación Kybele. Ingeniería del Software y Base de datos. Calidad del Producto. Citado el 4 de julio de 2005. Disponible en: http://kybele.escet.urjc.es/documentos/GCSW/T4-CalidadProducto.pdf

- Capacidad de recuperación: Capacidad del producto software para reestablecer un nivel de prestaciones especificado y de recuperar los datos directamente afectados en caso de fallo.
- Cumplimiento de la fiabilidad: Capacidad del producto software para adherirse a normas, convenciones o regulaciones relacionadas con al fiabilidad.

4.6.3 Usabilidad. Se relacionan los siguientes conceptos:

- Capacidad para ser entendido: Capacidad del producto software que permite al usuario entender si el software es adecuado y cómo puede ser usado para unas tareas o condiciones de uso particulares.
- Capacidad para ser aprendido: Capacidad del producto software que permite al usuario aprender sobre su aplicación.
- Capacidad para ser operado: Capacidad del producto software que permite al usuario operarlo y controlarlo.
- Capacidad de atracción: Capacidad del producto software para ser atractivo al usuario.
- Cumplimiento de la usabilidad: Capacidad del producto software para adherirse a normas, convenciones, guías de estilo o regulaciones relacionadas con la usabilidad.

4.6.4 Eficiencia. Se relacionan los siguientes conceptos:

- Comportamiento temporal: Capacidad del producto software para proporcionar tiempos de respuesta, tiempos de proceso y potencia apropiados, bajo condiciones determinadas.

- Utilización de recursos: Capacidad del producto software para usar las cantidades y tipos de recursos adecuados cuando el software lleva a cabo su función bajo condiciones determinadas.
- Cumplimiento de la eficiencia: Capacidad del producto software para adherirse a normas o convenciones relacionadas con la eficiencia.

4.6.5 Mantenibilidad. Se relacionan los siguientes conceptos:

- Capacidad para ser analizado: Es la capacidad del producto software para serle diagnosticadas deficiencias o causas de los fallos en el software, o para identificar las partes que han de ser modificadas.
- Capacidad para ser cambiado: Capacidad del producto software que permite que una determinada modificación sea implementada.
- Estabilidad: Capacidad del producto software para evitar efectos inesperados debidos a modificaciones del software.
- Capacidad para ser probado: Capacidad del producto software que permite que el software modificado sea validado.
- Cumplimiento de la mantenibilidad: Capacidad del producto software para adherirse a normas o convenciones relacionadas con la mantenibilidad.

4.6.6 Portabilidad. Se relacionan los siguientes conceptos:

- Adaptabilidad: Capacidad del producto software para ser adaptado a diferentes entornos especificados, sin aplicar acciones o mecanismos distintos de aquellos proporcionados para este propósito por el propio software considerado.
- Instalabilidad: Capacidad del producto software para ser instalado en un entorno especificado.

- Coexistencia: Capacidad del producto software para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes.
- Capacidad para reemplazar: Capacidad del producto software para ser usado en lugar de otro producto software, para el mismo propósito, en el mismo entorno.
- Cumplimiento de la portabilidad: Capacidad del producto software para adherirse a normas o convenciones relacionadas con la portabilidad.

4.7 MODELO PARA LA CALIDAD INTERNA Y EXTERNA SEGÚN MC CALL

4.7.1 Corrección. Hasta donde satisface un programa su especificación y logra los objetivos propuestos por el cliente.

4.7.2 Fiabilidad. Hasta donde el Software cumple con exactitud su función para la cual fue requerida.

4.7.3 Eficiencia. Es la cantidad de recursos informáticos y código necesarios para que un programa realice su función.

4.7.4 Usabilidad. Es la facilidad de manejo del Software

4.7.5 Portabilidad. El esfuerzo necesario para transferir el programa de un entorno de hardware o Software a otro diferente.

4.8 MODELO DE CALIDAD PARA LA CALIDAD EN USO

La Calidad en uso, es la capacidad del producto software para permitir que usuarios específicos logren realizar tareas específicas con productividad, efectividad, seguridad y satisfacción[13]. (Ver Figura 6)

[13] Julia González Rodríguez y Luis Olsina. Hacia la Medición de Calidad en Uso Web. Universidad de Alicante. España. Citado el 14 de Octubre de 2005. Disponible en Internet en: http://www.dlsi.ua.es/webe01/articulos/s222.pdf

Figura 6. Modelo de Calidad para la Calidad en uso[14]

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

4.8.2 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.

4.8.3 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.

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

[14] Grupo de investigación Kybele. Ingeniería del Software y Base de datos. Calidad del Producto. Citado el 4 de julio de 2005. Disponible en: http://kybele.escet.urjc.es/documentos/GCSW/T4-CalidadProducto.pdf

4.9 EVALUACIÓN DEL PRODUCTO SOFTWARE: ISO 14598

Los estándares de calidad definen la Norma ISO 14598 en una métrica que permite dar una estimación objetiva sobre productos de Software.

4.9.1 Proceso de la Evaluación. El proceso de evaluación entrega como resultado una valoración acerca de la calidad interna y externa definida en el análisis del sistema, el objetivo de evaluar el software y específicamente las propiedades de calidad del mismo es cumplir a cabalidad con los requisitos que el cliente desea alcanzar con el producto final.

- **Establecer el propósito de la Evaluación** Productos intermedios:
  a) decidir sobre la aceptación de un producto intermedio de un subcontratista;
  b) decidir cuando un proceso está completo y cuando remitir los productos al siguiente proceso;
  c) predecir o estimar la calidad del producto final;
  d) recoger información con objeto de controlar y gestionar el proceso.

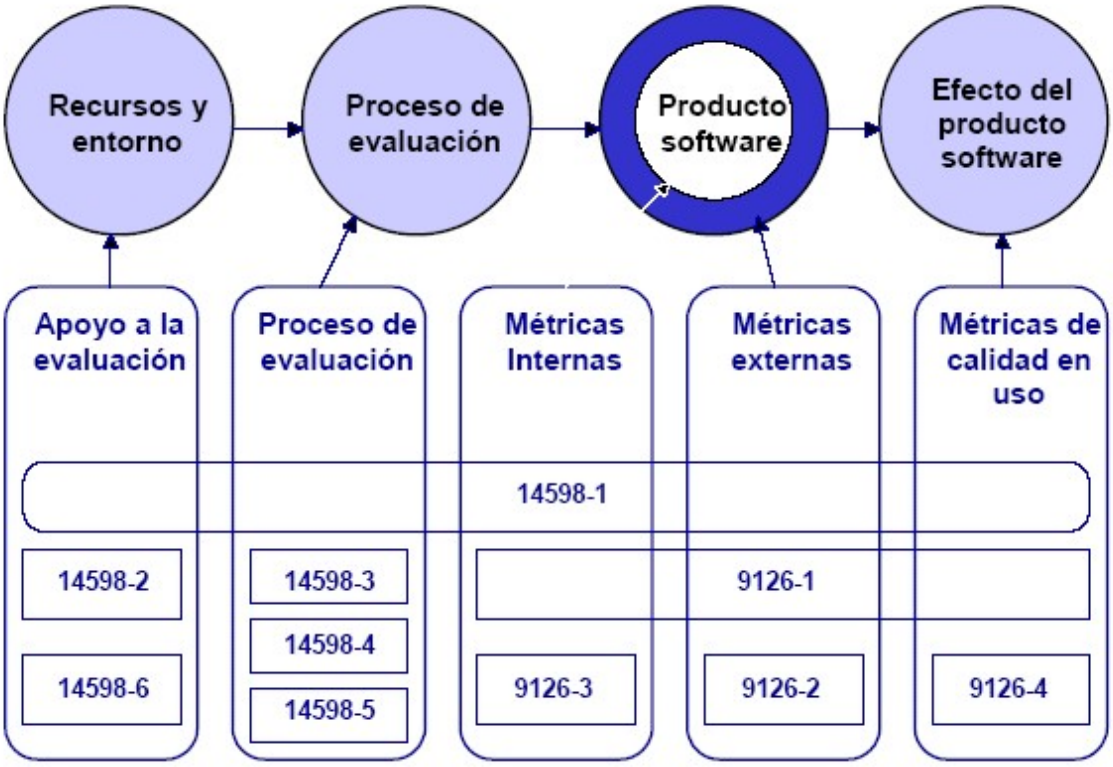


Figura 7. Evaluación del Producto Software[15]

Producto final:

a) decidir sobre la aceptación del producto;

b) decidir cuando publicar el producto;

c) comparar el producto con otros productos competitivos;

d) seleccionar un producto entre productos alternativos;

e) valorar tanto el aspecto positivo como negativo cuando está en uso;

f) decidir cuando mejorar o reemplazar un producto.

[15] Grupo de investigación Kybele. Ingeniería del Software y Base de datos. Calidad del Producto. Citado el 4 de julio de 2005. Disponible en: http://kybele.escet.urjc.es/documentos/GCSW/T4-CalidadProducto.pdf

- **Identificar los tipos de Productos**

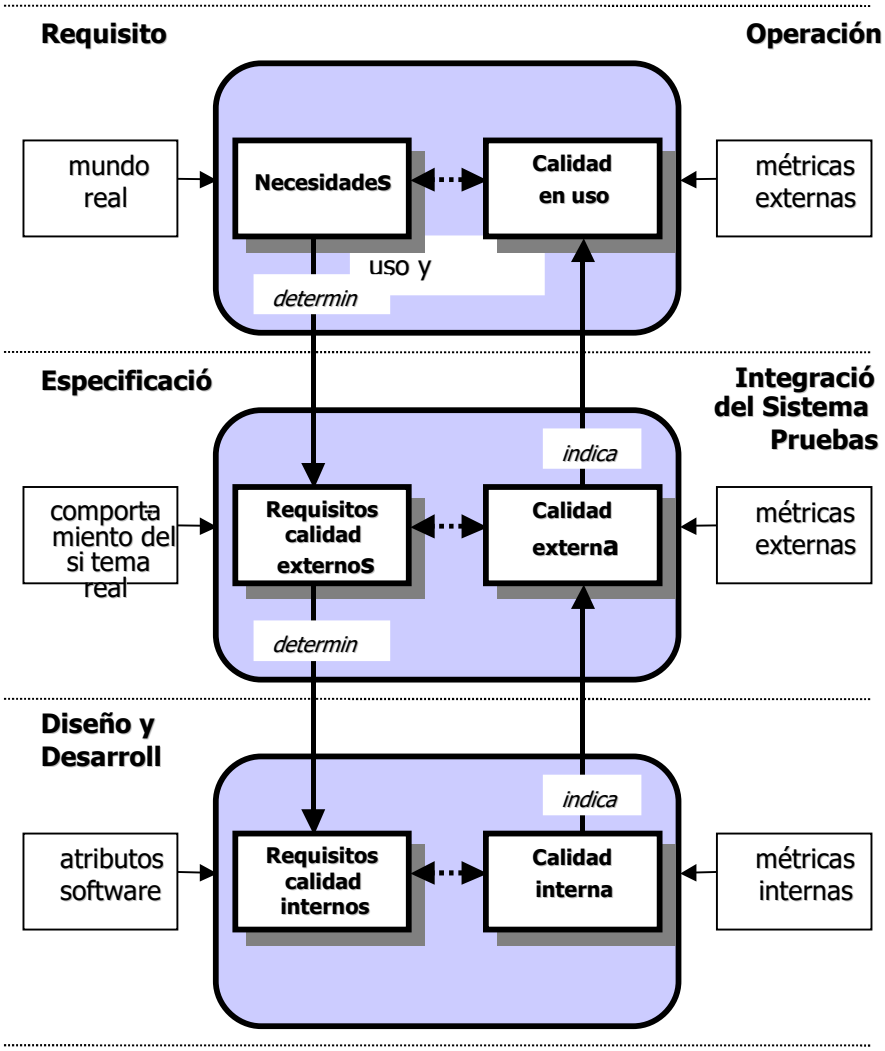


Figura 8. Identificar los tipos de producto[16]

[16] Grupo de investigación Kybele. Ingeniería del Software y Base de datos. Calidad del Producto. Citado el 4 de julio de 2005. Disponible en: http://kybele.escet.urjc.es/documentos/GCSW/T4-CalidadProducto.pdf

- **Especificar el Modelo de Calidad**

  ISO 9126 - 1

- **Seleccionar Métricas**

  ISO 9126 – 2 y 3

- **Establecer Niveles para las Métricas**

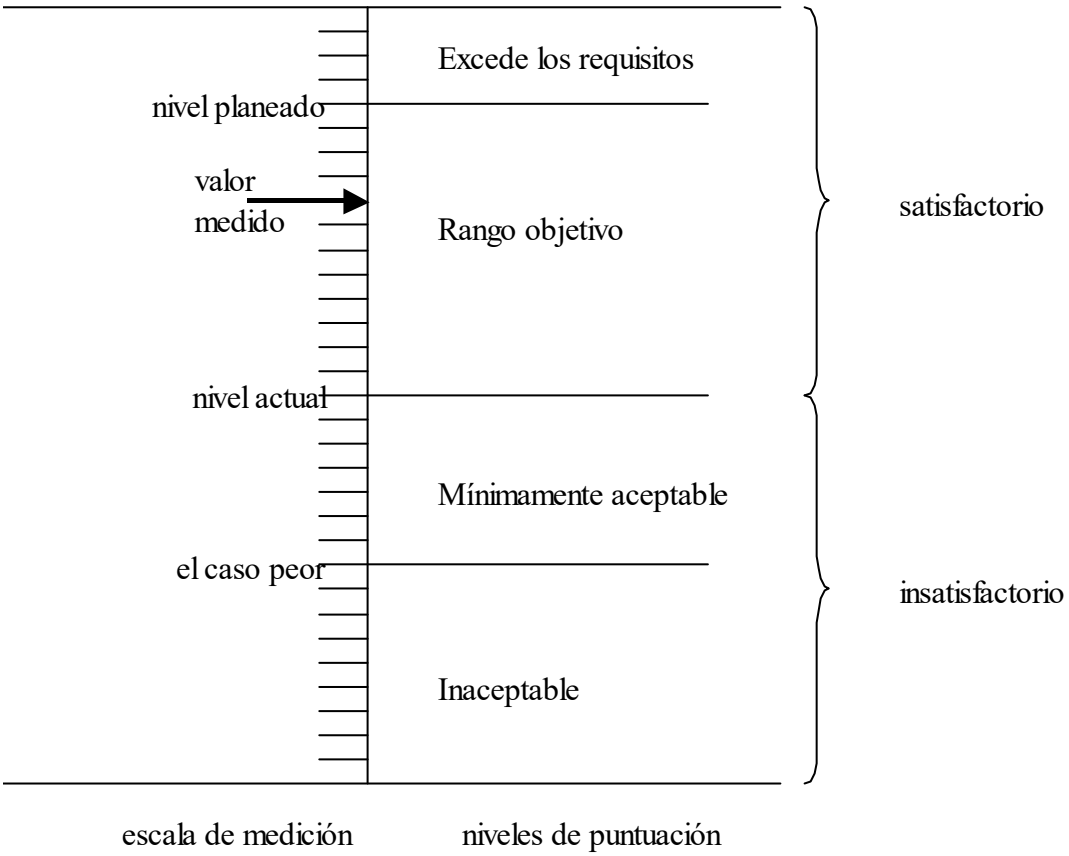


Figura 9. Niveles de Puntuación de las Métricas[17]

[17] Grupo de investigación Kybele. Ingeniería del Software y Base de datos. Calidad del Producto. Citado el 4 de julio de 2005. Disponible en: http://kybele.escet.urjc.es/documentos/GCSW/T4-CalidadProducto.pdf

- **Establecer criterios de Valoración** El evaluador debe preparar un procedimiento para esto, con criterios distintos para diferentes características de calidad, cada uno pudiendo estar expresado en términos de subcaracterísticas individuales, o una combinación ponderada de subcaracterísticas. El procedimiento habitualmente incluirá otros aspectos como el tiempo y el coste que contribuyen a la estimación de la calidad de un producto software en un entorno concreto.

- **Producir plan de Evaluación** El plan de evaluación describe los métodos de evaluación y el programa de acciones del evaluador (UNE 71048-3, UNE 71048-4 o UNE 71048-5). Debe ser consistente con el plan de mediciones (UNE 71048-2).

- **Tomar Medidas** Para la medición, las métricas seleccionadas se aplican al producto software. Los resultados son valores expresados en las escalas de las métricas.

- **Comparar con Criterios** En el paso de puntuación, el valor medido se compara con los criterios predeterminados

- **Valorar Resultados** La valoración, que resume un conjunto de niveles calificados, es el paso final del proceso de evaluación del

software.

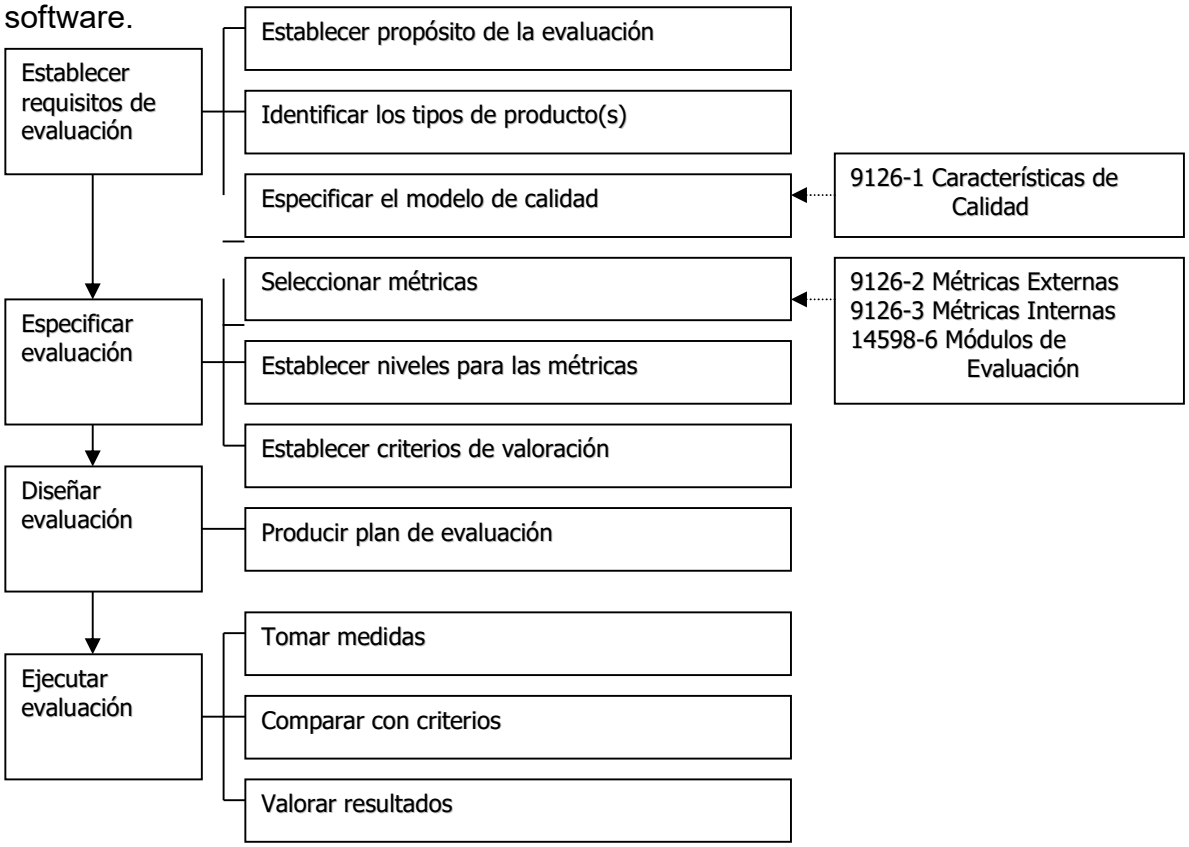


Figura 10. Evaluación del Software[18]

[18] Grupo de investigación Kybele. Ingeniería del Software y Base de datos. Calidad del Producto. Citado el 4 de julio de 2005. Disponible en: http://kybele.escet.urjc.es/documentos/GCSW/T4-CalidadProducto.pdf

5 APLICACIÓN DE LAS MÉTRICAS DE CALIDAD EN EL SISTEMA CIC

5.1 PLAN DEL ASEGURAMIENTO DE LA CALIDAD

5.1.1 Propósito. El propósito de este plan es especificar las actividades que se realizarán para asegurar la calidad del software a construir. En él se detallan los módulos que se van a revisar y los estándares, normas o métodos a aplicar; los métodos y procedimientos que se utilizarán para revisar que la elaboración de los productos se realice como los establece el modelo de ciclo de vida del proyecto; y procedimientos para informar a los responsables de los productos los defectos encontrados y realizar un seguimiento de dichos defectos hasta su corrección.

5.1.2 Acrónimos y Abreviaturas.

SQA: Aseguramiento de la Calidad del Software.

AMC: Aplicación de las Métricas de Calidad.

5.1.3 Gestión. En las sub-secciones siguientes se especifican los elementos de la organización que tienen influencia sobre la calidad del software, como está conformada la línea de gestión de calidad, de quien es la autoridad y responsabilidad por la calidad del software. El encargado del área de gestión de calidad en el proyecto es el responsable de realizar la gestión que asegura que el proceso establecido sea realmente implementado y que los productos de ese proceso cumplan con los criterios de calidad establecidos en este plan. La gestión de calidad es una disciplina de gestión.

Las disciplinas de gestión brindan soporte a las disciplinas básicas (Requerimientos del Sistema, Análisis, Diseño, Implementación y Pruebas) las cuales se realizan en forma paralela a ellas. (Ver Tabla 1).

Tabla 1. AMC, Organización

Definir, Estructura del Grupo Desarrollador	Grupo Desarrollador
Selección de Atributos de Calidad de Software.	Director del Proyecto
Análisis del Sistema de Información	
Documento Análisis del Sistema	Grupo Desarrollador / Director del Proyecto.
Evaluación Documento Análisis del Sistema	Grupo Desarrollador / Director del Proyecto.
Diseño del Sistema de Información	
Documento Descripción de Casos de Uso	Grupo Desarrollador / Director del Proyecto.
Evaluación Descripción de Casos de Uso	Grupo Desarrollador / Director del Proyecto
Diagramación UML	Grupo Desarrollador / Director del Proyecto
Evaluación Diagramación UML	Grupo Desarrollador / Director del Proyecto
Desarrollo del Sistema de Información	
Selección de Recursos de Software (Lenguaje de Programación, Bases de Datos, Servidor Web)	Grupo Desarrollador
Herramientas para el Desarrollo	Grupo Desarrollador

Implementación del Sistema de Información CIC	
Evaluación de Propiedades de Calidad	Grupo Desarrollador / Usuarios del Sistema

- **Tareas** En esta sección se describen las tareas de calidad a realizar, indicando para cada una: el entregable asociado y la influencia de la tarea en la calidad del producto.

Tabla 2. AMC, Tareas

Actividad	Entregable Asociado
Planificación de Calidad	Plan de Calidad
Identificar Propiedades de Calidad	Propiedades de Calidad
Revisión de Calidad de Producto	Evaluaciones de Revisión de SQA
Revisión de Entregas de Módulos	Evaluaciones de Entrega de Módulos del SQA

- **Planificación de Calidad** La Planificación de la Calidad son actividades que establecen los objetivos y los requisitos para la calidad, así como los requisitos para la aplicación de los elementos del sistema de la calidad. (**ISO 8402**).

El Plan de Calidad es la fuente guía donde se expone los métodos a seguir durante cada uno de los procesos que involucran el Producto de Software.

El Plan de Calidad es el Documento que enuncia las prácticas, los medios y la secuencia de las actividades ligadas a la calidad, ya sean específicas de un producto, proyecto o contrato particular. (**ISO 8402**)

La influencia de la *Planificación de Calidad,* en la calidad del producto es la planeación y seguimiento continuo que le se le da a cada uno de los procesos y

actividades que involucran el análisis, diseño, desarrollo, implementación y pruebas del producto.

- **Identificar Propiedades de Calidad** Las Propiedades de Calidad son aquellas que permiten evaluar la calidad en cuanto a las características de Operación, facilidad de mantenimiento y adaptabilidad a nuevo entornos. Estas propiedades de Calidad son tomadas en cuentas y definidas de acuerdo al enfoque y las prioridades que el producto de Software debe tener según las exigencias del cliente. Algunas de estas propiedades son, funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad.

- **Revisión de Calidad del Producto** Para que un producto software sea de Calidad según las exigencias y necesidades del Cliente, se llevan a cabo revisiones técnicas formales en cada etapa análisis, diseño, desarrollo, implementación y pruebas. El resultado de la evaluación nos ayuda a llegar al punto de calidad de la perspectiva que el cliente junto con el grupo desarrollador estableció.

- **Revisión de Entregas de Módulos** La Revisión de Entrega de Módulos en la fase de implementación del sistema, es el ciclo en el cual tanto los usuarios del sistema, director del proyecto y grupo desarrollador evalúan el módulo entregado de acuerdo a los atributos de calidad. Los resultados de esta Revisión, complementa lo planteado, analizado, diseñado y desarrollado puesto en marcha.

- **Documentación** El objetivo de esta sección es especificar los documentos que dirigen el desarrollo del proyecto y que deberán ser revisados como parte de las actividades de aseguramiento de la calidad. Para cada documento se indica el objetivo del documento, la plantilla, norma y/o estándar que se usa para elaborar el documento y el contenido mínimo que debe tener dicho documento.

Documentación Mínima Requerida

- Análisis del Sistema
- Evaluación Análisis del Sistema
- Descripción de Casos de Uso
- Evaluación Especificación de Casos de Uso
- Desarrollo del Sistema (Diagramas UML)
- Evaluación Diagramas UML
- Revisión Entrega de Módulos

- **Análisis del Sistema** El Sistema Control Interno de Clientes (CIC), es un Sistema de Información basado en Web y compuesto por dos (2) Módulos: el de Gestión y el de Administración de Clientes. Con el desarrollo de estos dos módulos se automatizan procesos manuales mejorando la efectividad y productividad de la empresa.

El *Módulo de Gestión,* hace referencia a la manera en la que la empresa se preocupa por el bienestar del cliente ofreciendo valores agregados, llevando históricos de acceso, estadísticas de conexión, recordando pagos para evitar suspensión en el servicio. De esta manera se logra tener un perfil del cliente y se puede determinar lo que se llama “Personalidad del consumidor” para lograr suplir todas las necesidades del usuario y garantizar su permanencia en la empresa. Las actividades que realiza es el *Reporte de Gestión de Usuarios.*

El Módulo de Administración de Clientes, hace referencia al control que va a tener cada uno de los vendedores de la empresa para con sus clientes y sus comisiones, llevando así un control total desde que un cliente potencial empieza a realizar consultas sobre los posibles servicios que pueda adquirir por parte de la empresa, hasta el punto en el que se realiza el contrato. Las actividades del Módulo son *Ingreso de Clientes, Legalización de Clientes y Consultar Comisiones.*

Definiciones de Módulos:

- *Descripción de Usuario:* Parte del análisis donde se particulariza cada usuario del módulo y las actividades que va a realizar dentro del sistema.
- *Ambiente de Usuario:* Parte del análisis donde se precisa el contexto en el que cada usuario del módulo descrito interactuará con el Sistema.
- *Perfil de Usuario:* Parte del análisis donde se da un resumen del usuario, se especifica que tipo de usuario representa en el sistema, permisos del usuario de accesos al sistema, responsabilidades a cargo y las actividades a realizar.

- **Métricas para el Análisis del Sistema**

Para desarrollar la evaluación del Análisis del Sistema, la revisión puede enfocarse en un nivel detallado. Aquí, la preocupación está en la redacción de la especificación. Se intenta descubrir los problemas que pueden estar ocultos dentro del contenido de la especificación. Se sugieren las siguientes directrices para una revisión detallada de la especificación.

1. Tenga cuidado con los conectores persuasivos (p. Ej.: ciertamente, por tanto, claramente, obviamente, siguiendo lo anterior), y pregúntese « ¿Por qué?».
2. Tenga cuidado con los términos imprecisos (Ejemplo: algunos, a veces, a menudo, normalmente, ordinariamente, en la mayoría de los casos, mayoritariamente); busque una clarificación
3. Cuando se dan listas incompletas, asegúrese de que comprende todos los elementos. Las claves son: «etc., y así sucesivamente, tales como».

4. Asegúrese de que los rangos establecidos no, contienen supuestos no declarados (Ejemplo: Los códigos válidos van de 10 a 100. ¿Enteros? ¿Reales? ¿Hexadecimales?).

5. Cuidado con los verbos de significados imprecisos como «manejado, rechazado, procesado, ignorado, eliminado». Pueden interpretarse de muchas maneras.

6. Cuidado con los pronombres de significados ambiguos (Ejemplo: El módulo y/o se comunica con el módulo de validación de datos y se activa su indicador de control. ¿El indicador de control de quién?)

7. Busque frases que impliquen certidumbre (Ejemplo: siempre, todos, cada, ninguno, nunca), y exija pruebas.

8. Cuando un término es definido explícitamente en un sitio, intente sustituir la definición por otras apariciones del término.

9. Cuando se describe una estructura con palabras, dibújela para ayudar a su comprensión.

10. Cuando se especifica un cálculo, haga al menos dos ejemplos.

(Ver Anexo A. Análisis del Sistema Control Interno de Clientes - CIC)

- **Evaluación Análisis del Sistema** La revisión de la especificación de requisitos del Sistema para la Centralización de Servicios de una ISP es llevada a cabo tanto por el grupo desarrollador del software como por el cliente. Como la especificación forma el fundamento para el diseño y las subsiguientes actividades de ingeniería del software, se debería poner extremo cuidado al realizar la revisión.

Inicialmente la revisión se lleva acabo a nivel macroscópico. A este nivel, los revisores intentan asegurarse de que la especificación está completa, es consistente y exacta. (Ver Anexo C. Evaluación Final Análisis del sistema).

- **Descripción de Casos de Uso** Cualquier sujeto que se comunica (interacciona) con el sistema y que es externo al sistema mismo. Representan ROLES que interpretan personas o periféricos cuando el sistema opera. No necesariamente coincide con USUARIOS. Un usuario puede interpretar distintos roles. Cada uno de ellos será un actor.

TIPOS DE ACTORES:

1. Primarios: interaccionan con el sistema para explotar su funcionalidad. Trabajan directa y frecuentemente con el Software.
2. Secundarios: soporte del sistema para que los primarios puedan trabajar.
3. Iniciadores: no interactúan con el sistema pero desencadenan el trabajo de otro actor.

(Ver Anexo B. Especificación de Casos de Uso)

- **Evaluación Especificación de Casos de Uso**

(Ver Anexo E. Evaluación Final Especificación de Casos de Uso)

- **Desarrollo del Sistema (Diagramas UML)**

Diagrama: una representación gráfica de una colección de elementos de modelado, a menudo dibujada como un grafo con vértices conectados por arcos

UML define una notación que se expresa como **diagramas** que sirven para representar modelos/subsistemas o partes de ellos *El 80 por ciento de la mayoría*

de los problemas pueden modelarse usando alrededor del 20 por ciento de UML-- Grady Booch.

a) Diagrama de Casos de Uso

Casos de Uso es una técnica para capturar información de cómo un sistema o negocio trabaja, o de cómo se desea que trabaje.

No pertenece estrictamente al enfoque orientado a objetos, es una técnica para captura de requisitos.

b) Diagrama de Secuencia

Un diagrama de Secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo. El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo. Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.

c) Diagrama de Colaboración

Un Diagrama de Colaboración muestra una interacción organizada basándose en los objetos que toman parte en la interacción y los enlaces entre los mismos (en cuanto a la interacción se refiere). A diferencia de los Diagramas de Secuencia, los Diagramas de Colaboración muestran las relaciones entre los roles de los objetos. La secuencia de los mensajes y los flujos de ejecución concurrentes deben determinarse explícitamente mediante números de secuencia.

En cuanto a la representación, un Diagrama de Colaboración muestra a una serie de objetos con los enlaces entre los mismos, y con los mensajes que se intercambian dichos objetos. Los mensajes son flechas que van junto al enlace por el que “circulan”, y con el nombre del mensaje y los parámetros (si los tiene) entre paréntesis. Cada mensaje lleva un número de secuencia que denota cuál es el mensaje que le precede, excepto el mensaje que inicia el diagrama, que no lleva número de secuencia.

d) Diagrama de Clases

El Diagrama de Clases es el diagrama principal para el análisis y diseño.

Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones. El modelo de casos de uso aporta información para establecer las clases, objetos, atributos y operaciones.

(Ver Anexo D. Diagramas UML)

- **Evaluación Diagramas UML**

(Ver Anexo F. Evaluación Final Diagramas UML)

- **Revisión Entrega de Módulos**

a. El objetivo es descubrir defectos en el sistema e inconsistencias
b. Cualquier documento producido en el proceso puede ser revisado
c. El equipo de revisión deberá ser relativamente pequeño y las revisiones deberán ser relativamente cortas
d. La revisión deberá ser grabada y almacenada
e. Los comentario hechos durante la revisión deberán ser clasificados
f. Sin acciones. No se requiere cambiar el software o la documentación

g. Enviadas a reparación. El diseñador o programador deberá corregir una el fallo identificado
h. Reconsideración total del diseño. El problema identificado en la revisión impacta sobre otras partes del diseño. Algunos juicios verificaran si se ha resuelto los problemas de la forma mas efectiva
i. Los errores en los requerimientos y especificaciones podrían enviarse a el cliente

(Ver Anexo G. Evaluación Final Entrega de Módulos)

5.1.4 **Métricas** En Métricas se hace referencia al documento Propiedades de Calidad Identificadas para el proyecto. En este documento se describe para cada propiedad de calidad identificada, las métricas que se utilizarán para medir dicha propiedad de calidad.

El Objetivo del Plan de Calidad, es desarrollar un Sistema de Información que satisfaga todas las necesidades del cliente. Para este desarrollo precisamos cada una de las *Propiedades de Calidad* del producto, de tal forma que tengamos de una manera clara y precisa los puntos a desarrollar, evaluar y enfocarnos a medida que llevamos a cabo el desarrollo y entrega del producto.

- **Propiedades de Calidad**

Atributos de Calidad para el Sistema Control Interno de Clientes (CIC)

a. Corrección. La métrica corresponde a documentos como análisis del sistema, descripción de casos de uso y diagramas UML.
b. Fiabilidad. La métrica corresponde al análisis de los resultados lógico (Sistema CIC), contra los físicos (Documentación de la empresa). Por consiguiente se realizará una comparación de los valores de los contratos

Legalizados por cada vendedor y sus comisiones contra el reporte de legalizados y comisiones que el sistema muestra.

c. Eficiencia. La métrica corresponde a la especificación de recursos de software y herramientas para el desarrollo.
d. Usabilidad. La métrica corresponde al análisis comparativo de las funciones que se realizaban manualmente con las automatizadas y presentes en el Sistema Control Interno de Clientes.
e. Portabilidad. La métrica corresponde a la investigación y análisis de los recursos de software utilizados por el sistema.

- **Estructura del Grupo Desarrollador** Los siguientes factores deben ser considerados cuando se selecciona la estructura del equipo del proyecto de software:

  - Dificultad del problemas a ser resuelto
  - El tamaño de las líneas de código de los programas resultantes o puntos función
  - El tiempo en que el equipo estará junto (tiempo de vida del equipo)
  - El grado de modularidad del problema
  - La calidad y confiabilidad requeridas del sistema a ser construido
  - La rigidez de la fecha de entrega
  - El grado de sociabilidad (comunicación) requerida para el proyecto

El grupo de Programación a el cual está orientado es el Democrático, por lo tanto cada uno de los integrantes participamos en todas las decisiones que definan, encaminen, desarrollen y apliquen el proyecto. Según la definición del Grupo de Programación democrático, “la participación de todos los integrantes de contribuir en la toma de decisiones, el aprendizaje continuo por parte de los dos miembros, la facilidad de comunicación y sin presiones que se llevarán a cabo” nos lleva a elegir esta clasificación de grupo.

- **Descripción del Equipo Desarrollador**

  - **Analista de Sistemas**
    - Responsable del conjunto de requisitos modelados en los casos de uso
    - Requisitos funcionales y no funcionales
    - Delimitar el sistema
    - Encontrar actores y casos de uso
    - Asegurar modelos de casos de uso completos y consistentes
    - Elaborar un glosario para mantener la consistencia semántica
    - Dirigir el modelado
    - Coordinar la captura de requisitos
  - **Analista y Diseñador de Casos de Uso**
    - Responsable de la descripción detallada de un caso de uso
    - Establecer una comunicación estrecha y eficaz con los usuarios (directos)
  - **Diseñador de Interfaces**
    - Diseñar las interfaces de usuario en su aspecto visual
    - Desarrollar prototipos de interfaces de usuario para algunos casos de uso
  - **Arquitecto**
  - Requerimientos de la arquitectura y prioridades del modelo de casos de uso
  - Análisis que garantice la integridad del modelo de análisis y que éste sea consistente y legible
  - Diseño que garantice la integridad de los modelos de diseño y despliegue
  - Implantación que garantice la integridad del modelo de implantación y la asignación de componentes a los diferentes nodos.
  - **Ingeniero de Casos de Uso**
    - Análisis con el fin de mantener la integridad de las realizaciones de casos de uso

- Diseño con el fin de detallar las realizaciones de casos de uso y verificar la correspondencia entre análisis y diseño

- **Integrador de Sistemas**
  - Planificar la secuencia de construcciones necesarias en cada iteración
  - Integrar cada construcción a partir de sus partes implementadas
- **Diseñador de Pruebas**
  - Garantizar la integridad del modelo de pruebas y hacer la planeación correspondiente.
  - Establecer objetivos de prueba apropiados
  - Seleccionar y describir casos de prueba y los procedimientos de prueba
- **Ingeniero de Pruebas de Integración**
  - Ejecutar las pruebas de integración
  - Verificar el correcto funcionamiento de componentes
  - Documentar los defectos
- **Ingeniero de Pruebas del Sistema**
  - Ejecutar las pruebas del sistema para cada iteración completa
  - Verificar los resultados en conjunto con los usuarios finales
  - Documentar los defectos
  - Facilidad para tener familiaridad con el comportamiento observable del sistema.

Según Mantei “sugiere que esta estructura es apropiada para proyectos de investigación y desarrollo largos y difíciles”, esta definición ha sido tomada en lo que concierne con el proyecto que hemos venido llevando a cabo de *investigación.*

Características del Grupo Desarrollado Democrático del Proyecto CIC

Las desventajas de Grupo de Programación Democrático presenta por Mantei son:

- La cantidad de comunicación necesaria para tomar decisiones.

- El requisito de que todos los miembros trabajen juntos y la falta de autoridad y responsabilidad que puede ocurrir, lo que producirá menos iniciativa.

En el proyecto CIC el grupo se encuentra integrado por 2 personas por lo tanto el flujo de información que debe pasar por los integrantes debe ser de forma clara y precisa definiendo el desarrollo del proyecto.

- **Métricas para el Desarrollo**

- **Especificación de Requisitos (Front-End).** A través de de la especificación de casos de uso
- **Recursos Humanos.** El Capital Humano está conformado, por dos integrantes los cuales están en la capacidad de analizar, diseñar, desarrollar y aplicar el Sistema de Control Interno de clientes modelado para las ISP.
- **Recursos de Software**
  - Apache. Software por excelencia, compite contra servidores Microsoft, brinda seguridad estabilidad y confiabilidad. La utilización de Apache como servidor Web se basa en razones de peso. Apache es un producto con muchos años de experiencia, muy probado a todos los niveles, tanto en lo que respecta a rendimiento como seguridad. Apache proporciona utilidades muy necesarias como el soporte SSL o el VirtualHosting de páginas. Además, la ventaja principal es que desde un servidor Web vamos a poder servir todo tipo de contenido: desde las páginas estáticas de toda la vida, pasando por cgi's y scripts en PHP.
  - MySQL. Es un sistema de gestión de bases de datos el cual tiene como principales características su velocidad en consultas y robustez, al igual que seguridad lo que garantiza confiabilidad. Mysql, base de datos potente en la Web, tiene como principio rapidez y efectividad a la hora de buscar información.

- PHP. Se describe como un lenguaje de programación que se introduce dentro de las paginas HTML, la mayoría de su sintaxis esta basada en C, Java y Perl. Es compatible con bases de datos Oracle, MySQL, PostgresSQL, Sybase, Adabas entre otros. PHP - Sofwate libre de programación, se ejecuta en el servidor, hacen la pareja perfecta con MySql, aunque sencillo de aprender es bastante poderoso. PHP y MySQL se han convertido en el estándar para el desarrollo de aplicaciones Web. PHP como un poderoso lenguaje de programación que se ejecuta del lado del servidor y que ha sido creado específicamente para el desarrollo de aplicaciones Web, y MySQL como un poderoso motor de bases de datos relaciones que proporciona el repositorio perfecto para aplicaciones que requieren acceso rápido y confiable a la información. A estas ventajas, se añade el importante hecho de que tanto PHP como MySQL son Software Libre, lo que permite iniciar el desarrollo de aplicaciones y distribuirlas sin necesidad de pagar licencias por ello.

- **Herramientas para el Desarrollo (Back-End).**

RATIONAL ROSE. Desarrollo basado en modelos con UML. Rational Rose es la herramienta más premiada de desarrollo de software, basada en modelado, que es parte de la solución integrada de Rational Software totalmente diseñada para cumplir los actuales desafíos del desarrollo de software. Rational Rose esta probado para construir mejor software, más rápidamente.

5.1.5 Revisiones. En esta sección se define la Revisión de Calidad de Producto, Revisión de Ajuste al proceso, sus objetivos y mecanismos.

- **Revisión de Calidad de Producto.** El objetivo es revisar los productos que se definieron como claves para asegurar la calidad. Detectar desviaciones en los objetivos de calidad definidos e informar a los responsables para que sean corregidas. Para ello se revisan los productos para verificar que

cumplan con los estándares (sección 6) y con los objetivos de calidad utilizando las CheckLists definidas para el producto.

Se debe verificar que no queden correcciones sin resolver en los informes de revisión previos, si se encuentra alguna no resuelta, debe ser incluida en la siguiente revisión. Se debe identificar, documentar y seguir la pista a las desviaciones encontradas y verificar que se hayan realizado las correcciones.

Como salida se obtiene el Informe de revisión de SQA, que contiene todas las desviaciones o defectos encontrados durante la revisión. Este informe debe ser distribuido a los responsables del producto y se debe asegurar que ellos son conscientes de las desviaciones o discrepancias encontradas y de las acciones correctivas que deben realizar.

- **Revisión de Ajuste al Proceso.** El objetivo es revisar si los productos se obtuvieron realizando las actividades que se indican en el Modelo de Proceso. Para ello, se revisan los productos que se definen como claves para verificar el cumplimiento de las actividades definidas en el proceso, durante todo el ciclo de vida del software.

  Se debe recoger la información necesaria de cada producto, buscando hacia atrás los productos previos que deberían haberse generado y son entrada para el producto objeto de revisión, para poder establecer los criterios de revisión y evaluar si el producto cumple con las especificaciones.

  Esta información se obtiene de los siguientes documentos:

- Plan del Proyecto
- Plan de la Desarrollo
- Plan de Verificación y Validación

Se debe verificar si todos los pasos del proceso de desarrollo son seguidos apropiadamente.

Antes de comenzar, se debe verificar en los informes de revisión previos que todas las desviaciones fueron corregidas, si no es así, las faltantes se incluyen para ser evaluadas.

Como salida se obtiene el Informe de revisión de SQA correspondiente a la revisión de ajuste al proceso, que contiene todas las desviaciones o defectos encontrados durante la revisión. Este informe debe ser distribuido a los responsables de las actividades y se debe asegurar que ellos son conscientes de las desviaciones o discrepancias encontradas y de las acciones correctivas que deben realizar.

6 EXPERIENCIA EN EL DESARROLLO DEL SISTEMA DE INFORMACIÓN CIC

El sistema de información CIC (Control Interno de Clientes) fue concebido luego de analizar deficiencias en algunos de los procesos internos de la empresa y los resultados obtenidos en cada uno de ellos. La inconsistencia de la información, el tiempo requerido y la disminución en las ventas se convirtieron en objetivo primario a mejorar. Nació entonces la necesidad de crear un sistema de información que se encargara de estos procesos y eliminara los procedimientos manuales, reduciendo el tiempo el cual podría ser usado exclusivamente a realizar gestión, y solucionando el déficit en las ventas.

La solución a estos problemas debía ser atendida en menos de 2 meses, tiempo único disponible para la entrega de los módulos. Esto hacia pensar a los desarrolladores en “trabajar sobre la marcha” es decir, teniendo en cuenta los conocimientos de los procesos internos por parte de uno de los desarrolladores sumado con los requerimientos de la empresa manifestados a través del gerente, debía ser suficiente para el desarrollo de cada uno de los módulos, dejando claro que no había tiempo para realizar una documentación del cómo se estaba trabajando.

El primer problema a solucionar se enfocaba en la gestión de clientes realizado por cada uno de los vendedores. Este proceso consistía en el registro de los clientes interesados en adquirir servicios en una planilla que posteriormente era usada por ellos para realizar las llamadas y tratar de concretar negocios. Estas plantillas constantemente eran desechadas una vez se realizaba el proceso y solo se tenían presentes aquellos clientes que habían adquirido algún servicio con la empresa.

Este módulo se desarrolló con los requerimientos iniciales, y cumplía con su objetivo, pero posteriormente se requería tener un histórico de aquellos clientes para realizar estadísticas, además se debía tener registro del rendimiento de cada uno de los vendedores con respecto a los clientes que tenia en su base de datos. Nuevamente el equipo de desarrollo tuvo que modificar el software para adaptarle los nuevos requerimientos, dando como resultado un módulo más eficiente y con nuevas características que daban gran valor agregado a las soluciones de los problemas existentes. Este modulo con sus posteriores cambios fue entregado en 1 mes.

El segundo problema se enfocaba en el pago de comisiones de cada uno de los vendedores, este proceso realizado manualmente por cantidad de contratos y planes tomaba alrededor de 1 semana, demorando el pago de las comisiones a los vendedores y ocupando tiempo necesario para otros procesos de la empresa.

La primera opción que se planteó por parte del gerente consistía en generar un listado en pantalla el cual mostraba cada uno de los vendedores con la cantidad de contratos vendidos por plan y la respectiva comisión. Esto solucionó parcialmente el problema ya que aunque se obtenía el total de contratos y el valor a pagar de las comisiones de cada vendedor, no se tenía un detalle de cada plan y cuánto valor se asignaba del plan a comisión. Como solución definitiva se planteó generar todo un reporte en pantalla con posibilidad de ser impreso donde se listaba lo mencionado anteriormente. Este módulo redujo a minutos la generación de reportes de comisiones de vendedores, proceso que como se había mencionado tomaba 1 semana.

Este módulo se terminó 10 días antes de lo acordado, el tiempo restante fue utilizado para realizar pruebas y hacer correcciones si fuere necesario.

Posteriormente se creo un módulo de acceso, para evitar que entre vendedores se realizara gestión ajena, es decir que un vendedor accediera a la lista de clientes de otro vendedor e intentara atraerlos para si mismo.

El sistema fue implementado en Bucaramanga y entró en periodo de prueba durante 1 mes. Los resultados fueron mejor de los esperados. Los objetivos principales al pensar en la creación de estos módulos fueron satisfactorios. Esto ocasionó que las sucursales de Bogota, Medellín y Manizales solicitaran a Bucaramanga la implementación de estos módulos. Los resultados obtenidos fueron casos de éxito en cada una de las ciudades.

Teniendo en cuenta los aspectos de la calidad mencionados en el capítulo 4, la descripción presentada anteriormente refleja la calidad en uso, debido a la trascendencia que el sistema obtuvo con su aplicación a otras sucursales.

Ahora analizando la calidad externa, ésta no se cumpliría ya que las pruebas ejecutadas para los módulos entregados no fueron planeadas y organizadas y se centraron únicamente funcionalidad. Para el proceso de pruebas se seleccionaron cinco vendedores para el módulo de gestión quienes aplicaron las pruebas que consideraron convenientes obteniendo como resultado una aceptación general del producto entregado más no una aproximación específica de los requerimientos de cada módulo.

Para cumplir con la calidad externa, se debió planear el desarrollo y medición mediante pruebas de los siguientes aspectos con sus respectivas características:

- Integridad de datos
    - Verificar el acceso a la base de datos del sistema
    - Verificar lectura simultánea de registros
    - Verificar el bloqueo de registros en los procesos de actualización

    - Verificar la correcta recuperación y almacenamiento de información.
- Funcionalidad
- Interfaz de usuario
    - Verificar la facilidad de navegación del sistema
    - Verificar la pertinencia de los mensajes o pantallas presentadas durante la interacción del usuario
    - Verificar que el sistema funcione en diferentes navegadores y versiones
- Rendimiento que verifica los tiempos de respuesta del sistema
- Carga y estrés que verifica la respuesta del sistema con el ingreso simultáneo de usuarios
- Control de acceso y seguridad
    - Verificar el acceso al sistema con los diferentes usuarios
    - Verificar la disponibilidad de opciones según el perfil de cada usuario

Una vez definidos los tipos de pruebas a realizar hubiese sido conveniente definir la estrategia a seguir para cumplir con cada una y los documentos de soporte de los resultados obtenidos. Las pruebas funcionales debieron estar guiadas por un documento que permitiera a los usuarios seguir las indicaciones de un caso de prueba con datos reales y registrar el comportamiento del sistema.

Por otro lado, la calidad interna está altamente ligada con el proceso de desarrollo del producto, el cual debe seguir una metodología que aplique las mejores prácticas de esta disciplina que son:

- Desarrollo iterativo
- Gestión de requerimientos
- Arquitectura de componentes
- Modelo visual

- Continua verificación de calidad
- Gestión de cambios

Para el desarrollo del CIC se siguió la metodología en cascada, en la cual todas las actividades involucradas en el desarrollo siguen un patrón lineal, por lo que no permite el desarrollo iterativo.

La gestión de requerimientos es un acercamiento sistemático para encontrar, documentar, organizar y seguir los requerimientos que cambian en un sistema. El proyecto no soportó esta característica debido a que no se documentaron los requerimientos iniciales y el desarrollo realizado para cada módulo no estuvo centrado en ellos para su seguimiento en caso de cambios posteriores. Teniendo en cuenta la metodología RUP los documentos adecuados para haber cumplido con esta característica serían: Documento Visión, Especificación de requerimientos de software y Especificación de casos de uso.

Una arquitectura de componentes es una arquitectura basada en el reemplazo de componentes, y debido a la independencia de los mismos, esto ayuda a administrar la complejidad y promueve la reutilización. El CIC no cumplió con esta característica ya que es un sistema desarrollado de manera procedimental y su código aplicaría solo para si mismo y no para otros sistemas de información, lo cual no lo hace reutilizable.

El modelo visual es el uso de una semántica de gran valor, que utiliza notaciones de diseño gráfico y texto para capturar el diseño del software. Una notación como UML permite un mayor nivel de abstracción, manteniendo rigurosamente sintaxis y semántica. La documentación de UML debe ser estrictamente realizada antes de iniciar la fase de desarrollo del proyecto, ejecutar este proceso posteriormente, reflejaría el sistema de información completo incluyendo todas sus falencias. El CIC realizó esta documentación después de la fase de desarrollo y utilizó solo los

diagramas de casos de uso, de secuencia y de colaboración, dejando a un lado el diagrama de clases que también es aplicable a modelos de programación no orientados a objetos ya que permite visualizar la estructura del sistema.

Continua verificación de la calidad: Es importante que la calidad de todos los artefactos sea evaluada en varios puntos del ciclo de vida a medida que madura el proyecto. Particularmente cuando un software ejecutable se produce, este debe estar sujeto a demostraciones y pruebas en importantes escenarios de cada iteración, para que se obtenga un mayor entendimiento tangible de diseño y una eliminación a tiempo de los defectos arquitectónicos. Debido a que el CIC utilizó la metodología en cascada, se determinó la realización de las pruebas después de cada fase de construcción de los diferentes módulos.

Gestión de cambios: Un desafío que existe cuando se desarrollan sistemas de software, es que se encuentran diversidad de desarrolladores, organizados en diferentes equipos, posiblemente en diferentes lugares, trabajando en equipo en múltiples iteraciones, actualizaciones, productos y plataformas. Cuando existe una ausencia de disciplina de control, es muy probable que el proceso de desarrollo entre en caos. Disciplina de configuración y gestión de cambios para resolver este reto. Para el desarrollo de los diferentes requerimientos del CIC, se dividieron tareas puntuales de un mismo requerimiento. Finalizando la tarea los desarrolladores se reunían para acoplar las labores realizadas y ponerlas en funcionamiento. Esta práctica tiene muchos inconvenientes debido a la ausencia de estándares a la hora de desarrollar cada una de las labores, como por ejemplo el uso de variables, la creación de procedimientos y falta de documentación en el código.

Ahora bien si se lee en detalle lo descrito anteriormente, se podría concluir que el CIC no aplicó estrictamente las normas para aseguramiento de calidad debido a

la ausencia de varios factores encontrados en la calidad interna y externa, sin embargo, el efecto de producto fue positiva lo que garantiza la calidad de uso.
Por lo tanto, ¿de qué depende la decisión de optar por una metodología de desarrollo u otra, o de tomar en cuenta todos los indicadores de medición de calidad en un producto de desarrollo de software?

Como se menciona Sun en el curso OO-226 (2003), existen diversas metodologías y no todas son aplicables a todos los proyectos. Para el CIC, el tiempo era un factor decisivo que no podía ser sustituido por documentación, por el contrario debía tener como único resultado el producto final y además garantizar la calidad de uso. Es de tener en cuenta que esto fue posible también ya que se tenía un conocimiento previo de los procesos internos y las consecuencias de la alteración en uno de ellos.

La metodología en cascada exige alta documentación y es aplicable a aquellos proyectos cuyos requerimientos son conocidos plenamente al inicio del mismo y tienden ser estables. Sun además recomienda seleccionar esta metodología cuando el proyecto tiene pocos riesgos. Este criterio fue aplicado en el CIC, sin embargo, la documentación llevada fue débil.

Por otro lado, las técnicas de desarrollo rápido deben ser usadas para pequeños desarrollos que exigen su puesta en marcha en un lapso de tiempo reducido, teniendo siempre presente que el producto final cumplirá con los requerimientos iniciales, y que una vez concluida su fase de desarrollo, la documentación creada tendrá como objetivo la explicación de cada uno de las labores realizadas y los factores que se tuvieron en cuenta para su realización, más no la explicación de código o definición de artefactos como diagrama de clases y diagramas de secuencia. Pero si se podrá obtener un documento de especificación de requerimientos o un documento visión que orientaran al lector a tener una visión del sistema desarrollado.

Por último se concluye que la documentación en un proyecto es un camino para lograr un software de calidad, más no manifiesta que su ausencia genere software de mala calidad de uso. Existen proyectos exitosos en cada una de las metodologías de desarrollo de software, sin embargo, la decisión de cuál es aplicable para un proyecto depende tanto del alcance del mismo y del equipo de desarrollo con que se cuente. Pero no hay que olvidar que independiente de la metodología seleccionada ésta debe aplicarse de manera estricta pues de alguna manera, han sido producto de investigaciones y de la experiencia de las personas involucradas en el área de Ingeniería del Software.

Por lo tanto, para este tipo de proyectos después del análisis realizado se recomiendan las siguientes actividades y plantillas a seguir. Es importante mencionar que la base de esta propuesta está dada por los flujos de trabajo de la metodología RUP (Captura de requerimientos, Análisis, Arquitectura, Diseño e Implementación), sin embargo, se excluyen algunas actividades debido a las condiciones especiales de este tipo de desarrollos rápidos y a que este proyecto no se basó en tecnologías orientadas a objetos. Además se incluye la actividad de Administración del Proyecto, en donde se definen los lineamientos para el aseguramiento de la calidad de este tipo de desarrollos. Finalmente, las plantillas presentadas al final del capítulo son una abstracción de los puntos más representativos del RUP orientados a la documentación de un proyecto de desarrollo rápido.

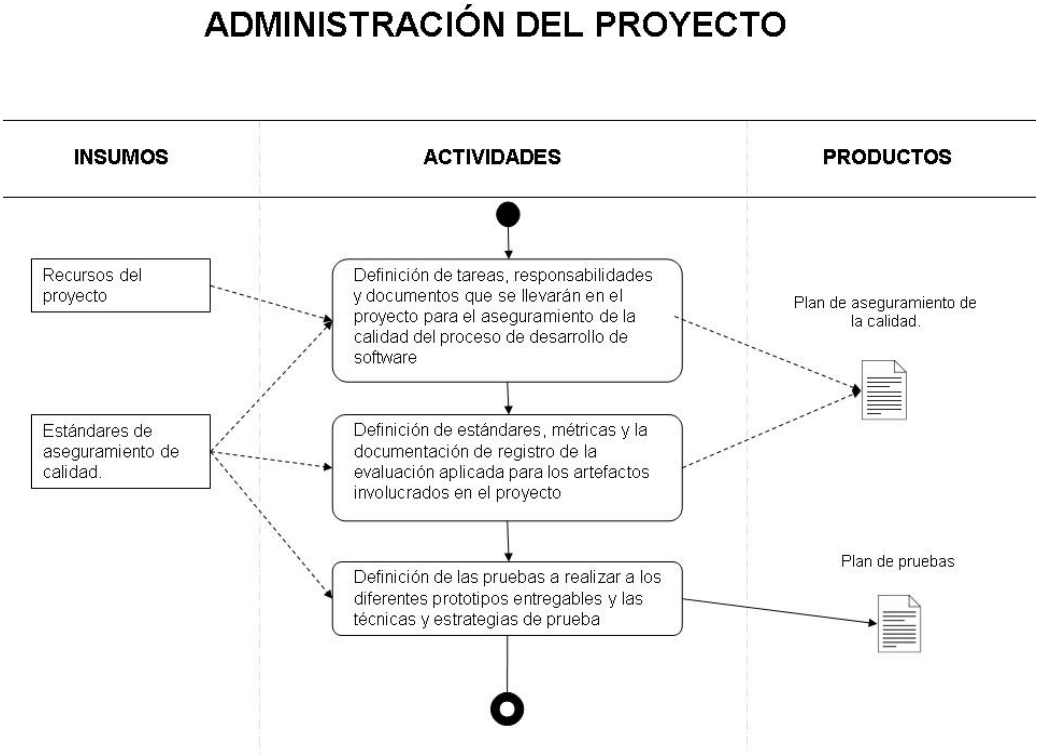


Figura 11. Actividades propuestas para administración del proyecto

La administración del proyecto para desarrollos rápidos de software, debe ser una actividad puntual desarrollada al inicio del proyecto por la persona de mayor experiencia en el tema, con el fin de dar los lineamientos de ejecución de todos los flujos de trabajo que se desarrollarán. Es por ello que los productos corresponden a planes que pretender orientar el desarrollo de software, velando por la calidad del producto y teniendo en cuenta los límites de recursos y tiempos generalmente involucrados en este tipo de proyectos. (Revisar al final del documento las plantillas del plan de aseguramiento de calidad y el plan de pruebas).

Finalmente, las aprobaciones de artefactos se propone que sean realizadas por el dueño del negocio y los usuarios funcionales, debido a que generalmente se cuenta con una o máximo dos personas en el proyecto y ellas deberán asumir todos los roles involucrados en el proceso de desarrollo de software. Los

artefactos técnicos como diagramas y código, se propone en caso que se cuente con más de una persona, recibir retroalimentación mutua de manera informal.

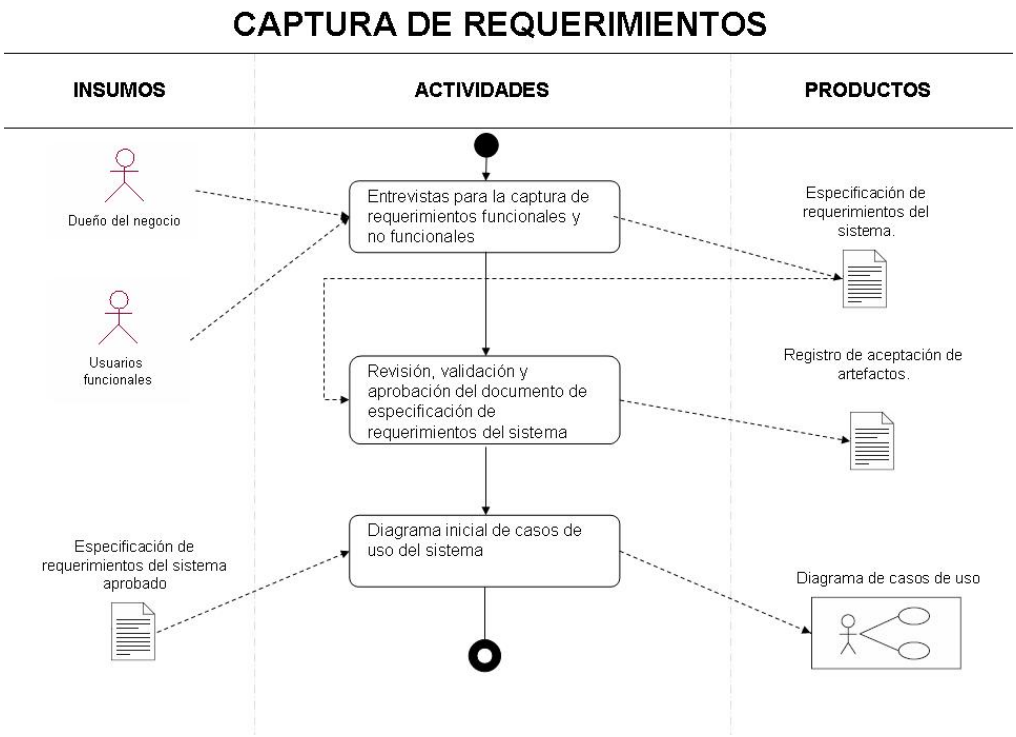


Figura 12. Actividades propuestas para captura de requerimientos

En caso que el contexto del problema sea suficientemente conocido por los desarrolladores, es posible eliminar el documento visión que se encarga de definir el alcance del proyecto a desarrollar y se debería enfocar los esfuerzos en la captura detallada de requerimientos funcionales y no funcionales documentándolos a través de la plantilla de especificación de requerimientos del software. Este documento deberá ser validado y aprobado por el dueño del negocio y los usuarios funcionales. Finalmente después de aprobado, la especificación de requerimientos del sistema será el insumo para la elaboración del un diagrama inicial de casos de uso.

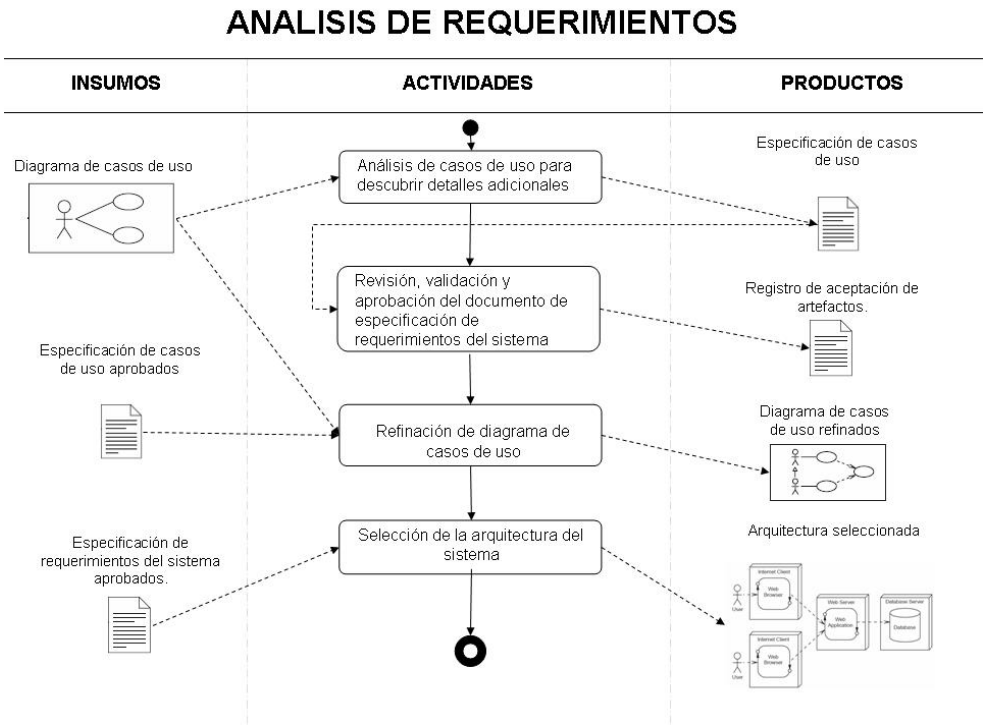


Figura 13. Actividades propuestas para análisis de requerimientos

Debido a que los casos de uso serán los que conducen todo el proceso de desarrollo, se recomienda encontrar más detalles y especificarlos en un documento formal, el cual una vez aprobado por los usuarios funcionales, llevará a la refinación del modelo inicial de casos de uso. Dentro de este flujo de trabajo propuesto no se obtiene un diagrama de clases debido a que el sistema no utiliza tecnologías orientados a objetos.

Además, en la presente propuesta no se tuvo en cuenta un flujo de trabajo separado para la arquitectura, sino que se integró la actividad de la selección de la arquitectura del sistema dentro del flujo de trabajo de análisis de requerimientos. Esto debido a que el documento de especificación de requerimientos del sistema documenta también los requerimientos no funcionales los cuales para el caso de sistemas Web desarrollados con PHP no requieren un análisis exhaustivo de arquitectura a diferencia de sistemas distribuidos o de sistemas empresariales

multinivel que incluyeran servidores de aplicaciones y marcos de trabajo para cada capa.

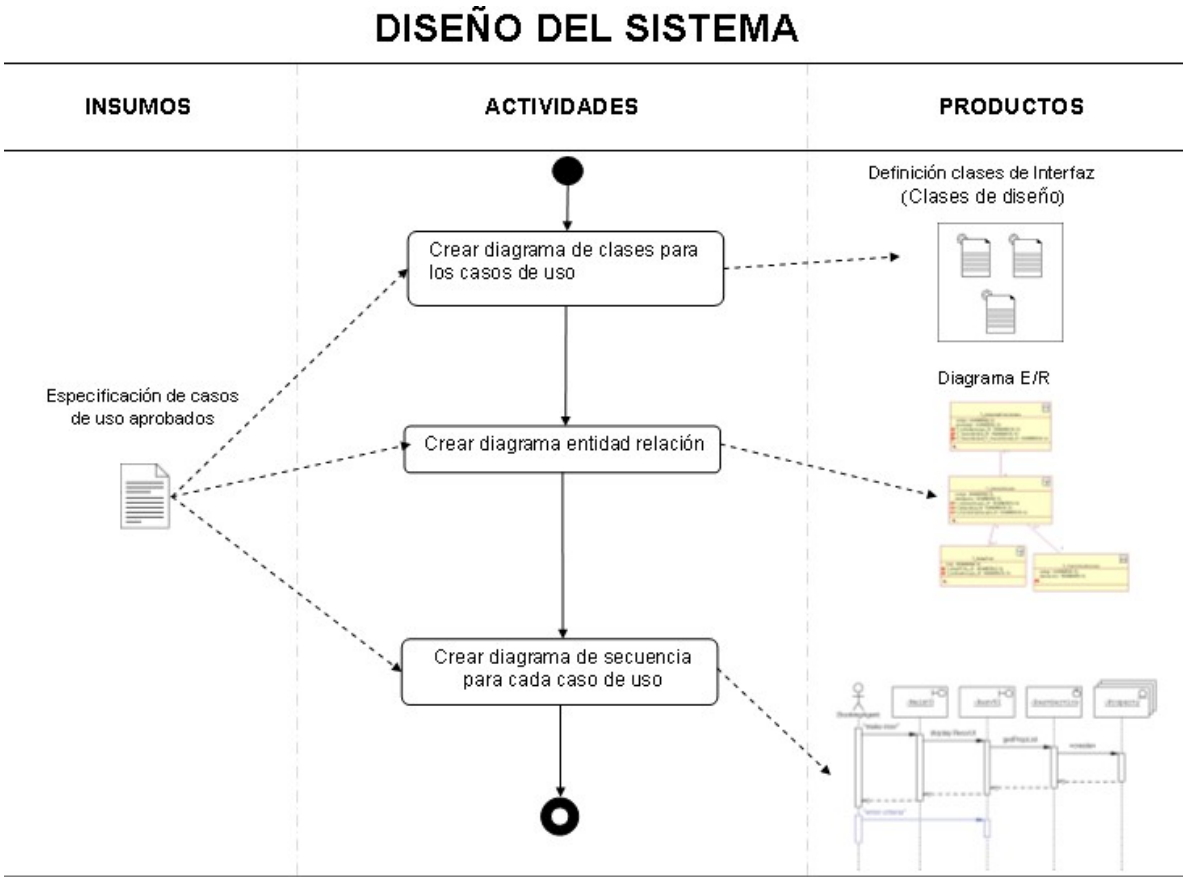


Figura 14. Actividades propuestas para diseño del sistema

Para los sistemas Web que no utilizan tecnologías orientadas a objetos, el único diagrama de clases que se debe trabajar es el diagrama de clases de diseño en donde se especifiq0uen las interfaces del cliente (html y javascripts), y las de control (PHP). Además, se incluyen las actividades de creación del diagrama entidad – relación y el diagrama de secuencia que presenta la forma como interactúan a través del tiempo las diferentes páginas html y php para obtener el resultado esperado de cada caso de uso.

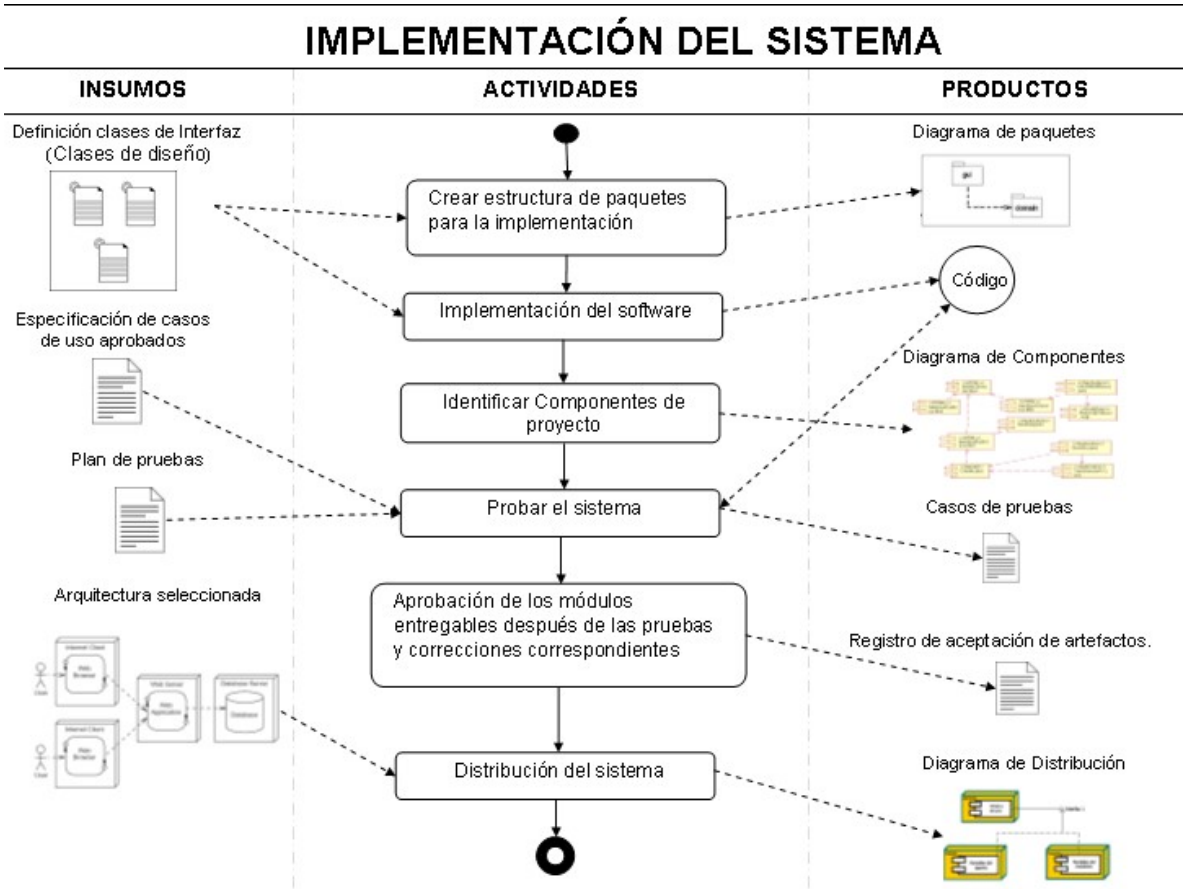


Figura 15. Actividades propuestas para implementación del sistema

La primera actividad propuesta es la elaboración de un diagrama de paquetes para organizar la estructura de directorios donde se alojan las páginas html y php planeadas para el desarrollo. Luego de ello, las páginas se implementan y se convierten en código fuente el cual será probado. Estas pruebas descritas basadas en el plan de pruebas y la especificación de casos de uso aprobados y son ejecutadas por los usuarios funcionales. La forma de captura de los resultados de las pruebas con los usuarios finales se recogerá en la plantilla de casos de prueba y al realizar los ajustes correspondientes después de realizadas las pruebas, los usuarios finales deberán aprobar el módulo o casos de uso entregados en la plantilla de registro de aceptación de artefactos.

Finalmente, una vez culminadas las actividades de prueba y aprobados los entregables se pasa a la distribución del sistema en ambiente de producción el cual toma como insumo la arquitectura definida desde el flujo de trabajo análisis de requerimientos. A continuación se presentan las plantillas que se proponen utilizar durante los diferentes flujos de trabajo presentados en las figuras 10-15.

6.1 PLANTILLA DE PLAN DE ASEGURAMIENTO DE LA CALIDAD

6.1.1 Introducción. Esta Introducción del documento de aseguramiento de la calidad, presenta una visión general del documento. Contiene el propósito, alcance, definiciones y demás descripciones que acercan al lector a entender este documento.

6.1.2 Definiciones, acrónimos y abreviaciones Esta sección define todo lo referente al glosario, acrónimos, sinónimos y abreviaturas necesarias para entender este documento.

6.1.3 Referencias Esta subdivisión proporciona una lista completa de todos los documentos referidos en este documento. Identifique cada documento por título, señale el número si es aplicable, la fecha, y la organización que lo publica. Especifique las fuentes de las cuales las referencias pueden ser obtenidas. Esta información se puede proporcionar referenciada a un apéndice o a otro documento.

6.1.4 Objetivos de Calidad Este punto referencia a través de qué estrategias se va a garantizar la calidad de los requerimientos del sistema mencionados en el documento de especificación de requerimientos de software.

6.1.5 Documentación y responsabilidades Lista la documentación que debe ser producida durante el proyecto para asegurar que el producto que se está desarrollando satisface los requerimientos. Los documentos sugeridos para el desarrollo rápido de aplicaciones son:

- Especificación de requerimientos de software (SRS), realizada por el desarrollador del sistema y aprobada por el dueño del negocio y los usuarios funcionales.
- Especificación de casos de uso, realizada por el desarrollador del sistema y aprobada por los usuarios funcionales.
- Registro de aceptación de artefactos realizada por el dueño del negocio y/o los usuarios funcionales.
- Plan de pruebas, realizada por el desarrollador del sistema.
- Casos de prueba, realizada por los usuarios funcionales y aprobados los cambios por los mismos.

6.1.6 Estándares y lineamientos. Esta sección referencia los estándares y lineamientos que se utilizarán en el proyecto. Para este tipo de proyectos de desarrollo rápido se sugiere seguir los siguientes diagramas de UML que son aplicables a proyectos orientados y no orientados a objetos y referenciados durante los diferentes flujos de trabajo:

- Diagrama de casos de uso
- Diagramas de clases de diseño
- Diagrama entidad – relación
- Diagramas de paquetes
- Diagramas de secuencia
- Diagramas de componentes
- Diagrama de distribución]

6.1.7 Métricas. Esta sección describe las métricas usadas para el proyecto, encargadas de monitorear los cambios, encausados siempre a las medidas tomadas para el aseguramiento de la calidad. Se sugieren las siguientes medidas:

Tabla 3: Medidas del progreso del proyecto

Categoría	Concepto medido	Medida
Progreso del proyecto	Progreso del trabajo	Estado de los casos de uso Estado de los casos de prueba
Calidad del proyecto	Correcciones funcionales	Severidad de los defectos Duración de los defectos

Tabla 4: Estado de los casos de uso

Necesidad de información	• Determinar el progreso de la definición de los casos de uso requerimientos.
Categoría	• Progreso del proyecto
Concepto medido	• Progreso del trabajo
Meta	• El objetivo de esta medida es asegurar que la definición de los casos de uso está progresando paulatinamente a medida que avanza el proyecto.
Medida base	Número de casos de uso por estado
Atributos	Estado del caso de uso Propuesto Aprobado Incorporado al sistema Validado por un caso de prueba exitoso

Tabla 5: Estado de los casos de prueba

Necesidad de información	• Determinar el progreso de la ejecución de los casos de prueba
Categoría	• Progreso del proyecto
Concepto medido	• Progreso del trabajo
Meta	• El objetivo de esta medida es asegurar que los casos de prueba planeados y ejecutados están progresado a medida que avanza el proyecto.
Medida base	Número de casos de prueba planeados, ejecutados, exitosos y fallidos.
Atributos	Estado del caso de prueba Planeado Ejecutado Exitoso Fallido

Tabla 6: Severidad de los defectos

Necesidad de información	• Evaluar la calidad del producto y los entregables de acuerdo al número de defectos abiertos.
Categoría	• Calidad del producto
Concepto medible	• Correcciones funcionales
Meta	• El objetivo de esta medida es hacer seguimiento de la calidad del software entregado.
Medida de base	Defectos abiertos
Atributos	Severidad del defecto La severidad puede ser clasificada de la siguiente manera: Severidad 1: Crítica. El defecto hace que el producto se pueda utilizar. Severidad 2: Alta. La utilización del producto puede continuar

	pero la falla presentada no es deseable. Severidad 3: Media. La utilización del producto puede continuar pero presenta algunas fallas manejables. Severidad 4: Baja. Corresponde solo a correcciones de forma.

Tabla 7: Duración de los defectos

Necesidad de información	• Evaluar la calidad del producto haciendo seguimiento de la duración de todos los defectos abiertos.
Categoría	• Calidad del producto
Concepto medible	• Correcciones funcionales
Meta	• El objetivo de esta medida es hacer seguimiento de la calidad del software entregado determinando el tiempo que duran los defectos abiertos. Un gran número de defectos abiertos por una considerable cantidad de tiempo, significa que la calidad del software es baja y se está incrementando el mantenimiento y soporte del mismo.
Medida base	Número de días de los defectos abiertos
Atributos	Fecha de envío del defecto Estado del defecto (enviado, asignado, abierto, resuelto)

6.1.8 Registros de Calidad. Descripción de registros de calidad que han sido aprobados. En esta sección se deben relacionar todos los documentos de registro de aceptación de artefactos llevados durante el proyecto.

6.2 PLANTILLA DE PLAN DE PRUEBAS

6.2.1 Introducción. Descripción del plan para probar los módulos del sistema en evaluación y presenta los siguientes objetivos:

- Identificar información del proyecto y componentes de software que deben

ser probados.

- Listar los requerimientos de prueba
- Recomendar y describir las estrategias de prueba a utilizar.
- Identificar los recursos requeridos y proveer un estimado del esfuerzo necesario para invertir en las pruebas.
- Listar los elementos entregables de las actividades de prueba.

6.2.2 Referencias. Lista de documentos de soporte al plan de pruebas como los casos de uso y los casos de prueba.

6.2.3 Requerimientos de prueba. Lista de los ítems a probar con sus correspondientes objetivos, como funcionalidad, integridad de datos, cierres de periodos o ciclos, interfaz de usuario, rendimiento, carga y estrés y control de acceso y seguridad.

6.2.4 Estrategias de prueba. Se presenta las técnicas a utilizar para realizar las pruebas identificadas en los requerimientos de pruebas. Para ello se describe el objetivo de la prueba, la técnica a utilizar, criterios de finalización de la prueba y consideraciones especiales, tal como se presenta a continuación el ejemplo de la documentación de la estrategia de prueba de integridad de datos:

Tabla 8: Ejemplo de estrategia de prueba de integridad de datos

Objetivo de la prueba:	Asegurar que los métodos y procesos de acceso a la base de datos funcionan adecuadamente sin producir corrupción de datos.
Técnica:	• Invocar cada método y proceso de base de datos buscando datos válidos e inválidos cada vez. • Revisar que los datos han sido almacenados satisfactoriamente y que todos los eventos ocurrieron como debían.
Criterio de finalización:	Todos los métodos de acceso funcionaron sin ninguna corrupción de información.
Consideraciones especiales:	• Las pruebas pueden requerir un ambiente de desarrollo para la base de datos. • Los procesos deben ser invocados manualmente. • Las bases de datos pequeñas o con limitados números de registros deben incrementarse para analizar el comportamiento en un ambiente más real.

6.2.5 Herramientas. Lista de aplicaciones de software utilizadas para aplicar las técnicas descritas en las estrategias de prueba.

6.3 PLANTILLA DE ESPECIFICACIÓN DE REQUERIMIENTOS DEL SOFTWARE

6.3.1 Introducción. La introducción del documento de especificación de requerimientos de software debe dar una visión general de todas las necesidades del negocio. Debe contener objetivo, alcance, definiciones, acrónimos, abreviaturas y referencias.

El documento de especificación de requerimientos captura las necesidades del sistema mediante entrevistas, utilizando lenguaje natural.

6.3.2 Objetivo. Especifica el propósito de este documento, describe el comportamiento externo de la aplicación y subsistemas identificados, también describe requerimientos no funcionales, restricciones y otros factores necesarios para tener un completo marco de los requerimientos del sistema.

6.3.3 Referencias. Esta subdivisión proporciona una lista completa de todos los documentos referidos en este documento. Identifique cada documento por título, señale el número si es aplicable, la fecha, y la organización que lo publica. Especifique las fuentes de las cuales las referencias pueden ser obtenidas. Esta información se puede proporcionar referenciado a un apéndice o a otro documento.

6.3.4 Descripción General. Describe los factores generales que afectan el producto y sus requerimientos. En esta sección no se describen requerimientos específicos, por el contrario brinda una contextualización de los requerimientos en general.

6.3.5 Requerimientos específicos. Esta sección contiene toda la descripción de los requerimientos de software a un nivel detallado, entendible para diseñadores,

suficiente para dar solución a los requerimientos del sistema y posteriormente realizar las pruebas necesarias.

6.3.6 Funcionalidad. Esta sección describe los requerimientos funcionales del sistema los cuales están expresados en lenguaje natural. Para varias aplicaciones esto constituye la parte principal de la especificación de requerimientos de software. Generalmente está organizada por características, por departamentos o por módulos.

<Requerimiento funcional uno>
Descripción del requerimiento

<Requerimiento funcional dos>
Descripción del requerimiento

<Requerimiento funcional tres>
Descripción del requerimiento

6.3.7 Usabilidad. Esta sección incluye todos los requerimientos que afectan la usabilidad, por ejemplo, especificar el tiempo de entrenamiento para usuarios normales y usuarios administradores para optimizar la utilización del sistema; especificar requerimientos para acomodar el sistema a estándares de usabilidad como IBM o interfaces de Microsoft.

<Requerimiento de usabilidad uno>
Descripción del requerimiento
<Requerimiento usabilidad dos>
Descripción del requerimiento
<Requerimiento usabilidad tres>
Descripción del requerimiento

6.3.8 Confiabilidad. Esta sección incluye todos los requerimientos que afectan la confiabilidad del sistema, por ejemplo, definir el porcentaje de tiempo que debe estar disponible la aplicación, tiempos estimados para la recuperación de fallos, estándares para la precisión de datos.

<Requerimiento de confiabilidad uno>
Descripción del requerimiento

<Requerimiento confiabilidad dos>
Descripción del requerimiento

<Requerimiento confiabilidad tres>
Descripción del requerimiento

6.3.9 Rendimiento. Esta sección incluye todos los requerimientos que afectan el rendimiento del sistema, e incluye la definición de tiempos de respuesta de la aplicación con la interacción del usuario, como por ejemplo, tiempos de respuesta por transacción, número de usuarios simultáneos, utilización de recursos como memoria, disco y dispositivos de comunicación.

<Requerimiento de rendimiento uno>
Descripción del requerimiento
<Requerimiento rendimiento dos>
Descripción del requerimiento

<Requerimiento rendimiento tres>
Descripción del requerimiento

6.3.10 Restricciones de diseño. Esta sección indica las restricciones de diseño que se deben tener en cuenta para construir el sistema como por ejemplo

lenguajes de desarrollo específicos, herramientas de desarrollo a utilizar, restricciones de arquitectura, de componentes y librerías entre otras.

<Restricción de diseño uno>
Descripción de la restricción

<Restricción de diseño dos>
Descripción de la restricción

<Restricción de diseño tres>
Descripción de la restricción

6.4 PLANTILLA DE REGISTRO ACEPTACIÓN DE ARTEFACTOS

Tabla 9: Identificación del artefacto

Nombre		Fecha	
Descripción			
Número de documento (si aplica)		Número de versión:	
Ubicación (nombre de archivo y ruta)			

Tabla 10: Identificación del dueño del artefacto

Nombre del dueño		Cargo	

Tabla 11: Identificación de revisores

Nombre	Cargo	Firma	Fecha	Observaciones

Tabla 12: Identificación de aprobadores

Nombre	Cargo	Firma	Fecha	Observaciones

6.5 PLANTILLA DE ESPECIFICACIÓN DE CASOS DE USO

Especificación del Caso de Uso
<Caso de Uso>

6.5.1 Introducción. La introducción de la especificación de casos de uso da una visión general de todo el documento. Incluye el propósito, el alcance, las definiciones, las siglas, las abreviaturas, las referencias, y la descripción de este documento.

6.5.2 Referencias. Esta subdivisión proporciona una lista completa de todos los documentos referidos en este documento. Identifique cada documento por título, señale el número si es aplicable, la fecha, y la organización que lo publica. Especifique las fuentes de las cuales las referencias pueden ser obtenidas. Esta información se puede proporcionar referenciado a un apéndice o a otro documento. Por favor hacer referencia al documento que originó el requerimiento y al diagrama de caso de uso.

6.5.3 Estado del caso de uso. Describe el estado del caso de uso, sea Propuesto, Aprobado, Incorporado al sistema o validado por un caso de prueba exitoso.

6.5.4 Flujo de Eventos. Una descripción textual de como el caso de uso es realizado en términos de objetos de colaboración. Su propósito principal es de

resumir los diagramas conectados al caso de uso y explicar como ellos son relacionados.

6.5.5 Descripción breve

Descripción del flujo normal. Se debe enumerar cada paso con un número encerrado entre corchetes. La descripción debe incluir un iniciador, la acción que ejecuta, el participante y el servicio que provee.

Descripción del flujo alterno. Se debe enumerar cada paso alterno con la secuencia del paso normal, encerrado entre corchetes.

6.5.6 Requerimientos especiales. Una descripción textual que recoge todos los requerimientos, como requerimientos no funcionales, sobre las realizaciones de casos de uso no consideradas en el modelo de diseño, pero estas necesidades serán tenidas en cuenta durante la puesta en marcha.

6.5.7 Precondiciones. Es el estado del sistema que debe estar presente antes que un caso de uso sea realizado. Ejemplo: que el cajero este activo, que el actor exista en el sistema, entre otros.

6.5.8 Poscondiciones. Es el estado del sistema que debe estar presente después que un caso de uso sea realizado. Ejemplo: Correr aplicaciones para actualizar datos, ejecutar procedimientos específicos.

6.5.9 Observaciones. Aquí se registran las solicitudes realizadas al equipo de trabajo técnico y algunas consideraciones que se requieran para el caso de uso.

Tabla 13: Tabla de registro

Ingeniero	Solicitud	Fecha de Entrega

6.6 PLANTILLA DE CASOS DE PRUEBAS

Tabla14: Plantilla de caso de pruebas

Caso de Prueba – No.			
Sistema	*Nombre del sistema que se está probando*	**Módulo**	*Nombre del módulo que se está probando*
Ejecutor	*Nombre de la persona que ejecuta la prueba*	**Fecha**	*Fecha de ejecución de la prueba*
Cargo	*Cargo que desempeña la persona que ejecuta la prueba*	**Tipo prueba**	*Definición del tipo de prueba que se está aplicando, ya sea funcional, de interfaz, de cierre de ciclos o de seguridad y control de acceso.*

Propósito:	*Mencionar el objetivo de la prueba y qué caso de uso se está probando*
Prerequisitos:	*Mencionar qué otros casos de uso o procesos son necesarios para realizar la prueba*
Datos de Prueba:	*Lista de algunos datos reales para la ejecución de la prueba.*
Pasos:	*Descripción detallada de todos los pasos para la ejecución de la prueba*
Resultados esperados	*Descripción de los resultados que debe presentar el sistema después de la ejecución de los pasos de la prueba.*
Observaciones (ESTA ES LA ÚNICA CASILLA QUE EL USUARIO DEBE LLENAR, LAS DEMÁS SON	*Relación de los resultados verdaderos obtenidos en la prueba, el ejecutor debe mencionar si la prueba fue satisfactoria porque se obtuvieron los resultados esperados. En caso contrario, se describe qué paso no se cumplió y cómo se comportó el sistema y en caso de presentar un error, se transcribe el mensaje correspondiente)*

INSTRUCCIONES A SEGUIR O A TENER EN CUENTA)	*Si el ejecutor desea relacionar otro tipo de observación que ayude al mejoramiento del caso que se está probando, se debe incluir dentro de esta celda.*
Estado	*Planeado, Ejecutado, Exitoso o Fallido*
Severidad de los defectos (Esto aplica solo si el caso de prueba tiene estado fallido)	*Severidad 1* *Severidad 2* *Severidad 3* *Severidad 4*
Fecha de envío del defecto (Esto aplica solo si el caso de prueba tiene estado fallido)	Fecha
Estado del defecto (Esto aplica solo si el caso de prueba tiene estado fallido)	*Enviado* *Asignado* *Abierto* *Resuelto*

7 RESULTADOS OBTENIDOS

- El Incremento de la producción y rendimiento a los profesionales en determinadas áreas de trabajo e investigación.
- La aplicación de técnicas para automatizar el manejo de datos.
- La Realización una planeación eficaz en el desarrollo de software.
- Documentación adecuada del sistema.
- Cumplimiento de los requerimientos de la empresa en cuanto a rendimiento y productibilidad.
- Seguridad y fiabilidad a la empresa y usuario final.
- Eficiencia en procesos garantizando su correcto desempeño en la hora de evaluación de resultados.

8 CONCLUSIONES

Los Factores de Calidad que hacen del Software un desarrollo diferente, cumpliendo los Objetivos que persigue la calidad de Software son:

- **Corrección**: (Hasta donde satisface un programa su especificación y logra los objetivos propuestos por el cliente.)

Justificación: Se realizaron los diagramas de UML para describir las características que componen el software. Descritas en la especificación de casos de uso.

- **Fiabilidad:** (Hasta donde el Software cumple con exactitud su función para la cual fue requerida.)

Justificación: La forma de evaluar si el Software realizado es fiable o no, es en los reporte de comisiones para vendedores, donde la evaluación que se realiza es tomando en cuenta el numero de contratos legalizados por vendedor debe ser igual a reporte que muestra el Sistema CIC, al igual que en el módulo de Gestión de Clientes donde la administración realizada a cada uno de los clientes potenciales manual y en el sistema, siendo cotejada esta información los resultados fueron los mismos.

- **Eficiencia:** Es la cantidad de recursos informáticos y código necesarios para que un programa realice su función.

Justificación: Se especificaron los recursos de Software y herramientas para el desarrollo que se deben utilizar para que se ejecute el software.

- **Usabilidad:** Es la facilidad de manejo del Software

Justificación: A cada asesor, se le explico que las tareas que comúnmente venia manejando como proceso manual, estaba implementada en el sistema que debía poner a prueba en dos semanas, teniendo como éxito la migración total de su trabajo al Sistema CIC.

- **Portabilidad**: El esfuerzo necesario para transferir el programa de un entorno de hardware o Software a otro diferente.

Justificación: De acuerdo a la eficiencia del programa los recursos validados para la realización del software nos permiten que este factor de calidad se cumpla. De tal forma que si queremos instalar el software en un servidor tanto Linux como Windows funcione igual.

- Según la investigación realizada, podemos argumentar que el uso de metodologías desarrollo de software son mas que una opción a la hora de pensar en cualquier tipo de proyecto, son la llave para que este sea de alto rendimiento y cumpla con las expectativas iniciales.

- Se puede caer en un error grave, si no se realiza la investigación apropiada de las diferentes metodologías para obtener un software de calidad, ya que se estaría aplicando una métrica equivocada y no se obtendría lo que en verdad se esta buscando.

- Fue muy difícil tratar de mostrarle al usuario que para obtener el software solicitado, era indispensable realizar un estudio a fondo, para tener precisión sobre que herramientas y ayudas de desarrollo de software podrían aplicarse para obtener software de calidad.

- La aplicación de métricas de calidad es tan complejo como su estudio mismo, por ende si no se realiza una completa investigación acerca de las métricas disponibles, se puede errar en su elección y perder tiempo que no se va a recuperar y retrazará el proceso.

- Estamos concientes que el estudio realizado sobre las métricas para obtener calidad de software, aun queda abierto y su campo de aplicación es bastante extenso, y que incluso el estudio realizado esta cerrado a lo que se necesito para el proyecto, pero se pudo evadir algunos puntos que pudieron llegar a ser significantes y haber logrado un mejor desarrollo.

- Queda abierta la investigación en las métricas de calidad de software para un proyecto aún mayor y el cual permita incursionar mas en estas metodologías.

9 BIBLIOGRAFÍA

[1] GARMUS, David y HERRON David. Measuring the Software Process: A Practical Guide to Functional Measurements. Prentice Hall. 1996.

[2] Breves notas sobre la Medición de los Atributos Externos del Software. [En línea]. Universidad del País Vasco. España [Citado 25 enero de 2004]. Disponible en Internet en: http://www.sc.ehu.es/jiwdocoj/mmis/externas.htm

[3] PRESSMAN, Roger S. Ingeniería del *Software*: un enfoque práctico. Segunda Edición. Madrid, España: Mc Graw Hill. 1993. p. 16 ISBN 968-422-674-8.

[4] COCOMO [En línea]. CSE Center for Software Engineering. Actualizado el 23 de septiembre de 2004. [Citado 3 de marzo de 2004]. Disponible en Internet en: http://sunset.usc.edu/research/COCOMOII/index.html

[5] Repositorio de documentos del Portal de Ingeniería del Software [En línea]. Fábrica de Software. [Citado 6 de septiembre de 2004]. Disponible en Internet en: http://www.fabricadesoftware.cl/fabrica_documentos.php

[6] Método de Análisis Puntos de Función MkII [En línea]. Universidad del País Vasco Campus de GIPUZKOA. España 2004. [Citado 6 de noviembre de 2004]. Disponible en Internet en: http://www.sc.ehu.es/jiwdocoj/mmis/fpmkii.htm

[6] BASILI, Víctor R. The TAME Project: Towards Improvement-Oriented Software Environments: IEEE Transaction on Software Engineering. Computer science technical report series. Enero de 1998.

[6] Manual de programación en PHP [En línea]. Desarrollo Web: Manejo de Sesiones. [Citado 10 de diciembre de 2004]. Disponibles en Internet en: http://www.desarrolloweb.com/articulos/320.php?manual=12 y http://www.desarrolloweb.com/articulos/321.php?manual=1

[6] Portal de Programación en castellano. [En línea]. PHP. 2004. [Citado 10 de diciembre de 2004]. Disponibles en Internet en: http://www.programacion.com/php/

[9] Sitio web de Apache Web Server. [En línea] Actualizado en Agosto de 2006. [Citado el 4 de febrero de 2005] Disponible en: http://www.apache.org

[10] Curso Sun Microsystem OO-226. Object-Oriented Analysis and Design Using UML. Revisión C. 2003.

[11] Sitio Web Metodología Rational Unified Process. Actualizado en enero de 2003. [citado el 6 de abril de 2005] Disponible en http://172.16.20.7/rup/index.html.

[12] Desarrollando un plan de aseguramiento de calidad [En. línea]. Washington State Office of Laboratory Quality Assurance. Abril de 2000. [Citado el 6 de mayo de 2005]. Disponible en: http://www.doh.wa.gov/hsqa/FSL/Documents/LQA_Docs/QAPlan.pdf.

[13] Ejemplo de plantillas de Product Acceptance Plan. [En línea]. HomeSafenet Project. 13 de marzo de 2004. [Citado en junio de 2006]. Disponible en: http://www5.myflorida.com/cf_web/myflorida2/healthhuman/business/toolsandprocs/product_acceptanceplan.doc.

ANEXO A

Análisis del Sistema

1. MODULO DE GESTIÒN

1.1 Descripción del usuario

En nuestro proyecto *CIC Modulo Gestión de Clientes* trabajamos con 2 clases de usuarios.

- Administrador: Tiene un perfil de acceso a la Información donde podrá de una lista de registro seleccionar un vendedor y mirar en un rango de fechas, la gestión a cada uno de los clientes que ha venido llevando cada vendedor.
- Vendedor: Gestión de clientes conociendo el registro de llamadas que se ha venido llevando con los clientes y su histórico.

La diferencia entre el Administrador y el Vendedor se basa en que el Administrador tiene un ingreso a la Página de Gestión con un perfil especificado, donde podrá revisar la Gestión que cada Vendedor ha venido realizando con cada uno de los clientes. Mientras el vendedor entrará a realizar Gestión solo con los clientes propios.

1.1.1 Ambiente del Usuario. CIC (Modulo de Gestión) es un sistema que se desenvuelve en la Web, por tal motivo el principal entorno grafico que se manejara será cualquier navegador de cualquier sistema operativo. Los Usuarios tendrán acceso al sistema internamente, desde la empresa, o globalmente, desde

cualquier sitio donde se tenga acceso a Internet y en ambos casos se obtendrán los mismos resultados, garantizando seguridad y confiabilidad a la hora del envío de Información

La filosofía de "Interfaz CIC" proporciona al usuario la manera más agradable y sencilla de obtener la información necesaria en cualquier momento con tan solo dar un click.

1.1.2 Perfil Del Usuario

1.1.2.1 <Administrador>

Representativo	• Persona con perfil de Administrador
Descripción	• Persona autentificada en el Sistema encargada de la revisión y control de la gestión que lleva a cabo cada vendedor.
Tipo	• Administrador
Responsabilidades	• Mantener el control de la Gestión que cada Vendedor tenga con sus propios clientes, también puede realizar la Gestión y dejar reportada la llamada dentro del cliente seleccionado.
Criterio Satisfactorio	• El control de clientes y vendedores del sistema está representado en la facilidad, rapidez y comodidad de realizar la Gestión a Clientes de una manera Confiable y segura.
Salidas	• Búsquedas de Vendedores y reportes de llamadas basados en rangos de fechas.
Comentario / Problemas	• El no tener un conocimiento total de la gestión que ejecuta cada Vendedor. Es importante que la información que se encuentre en el sistema de Gestión este actualizada por el vendedor.

1.2.1.2 <Vendedor>

Representativo	• Persona encargada de realizar la Gestión de los Clientes
Descripción	Persona autentificada en el Sistema que lleva el registro de Gestión (Llamadas realizadas) realizado a cada uno de los clientes.
Tipo	• Usuario del Sistema
Criterio Satisfactorio	• El control de clientes y vendedores del sistema está representado en la facilidad, rapidez y comodidad de realizar la Gestión a Clientes de una manera Confiable y segura.
Salidas	• Registro de clientes, Gestión del cliente en un rango de fecha determinada.
Comentarios / Problemas	• Cada llamada debe ser inmediatamente registrada en el sistema y actualizar la información del CIC. • Los datos de observación deben ser lo suficientemente claros y entendibles.

2. MODULO DE ADMINISTRACION DE CLIENTES

2.1 Descripción del Usuario

En nuestro proyecto *CIC Modulo de Administración de Clientes* trabajamos con 2 clases de usuarios.

- Administrador: Tiene un perfil de acceso a la Información donde podrá de una lista de registro seleccionar un vendedor y mirar en un rango de fechas, los clientes que ha legalizado, obteniendo información sobre cada uno de los atributos del contrato y las comisiones del vendedor seleccionado.
- Vendedor: Usuario del Sistema que sigue el siguiente proceso con los clientes.

- registra un cliente potencial en la base de datos, adquiriendo este vendedor prioridad sobre la persona que registra.
- Legalización de un cliente ya anteriormente registrado en la base de datos.

Una vez un vendedor haya registrado un cliente en el Sistema, el vendedor tiene prioridad sobre el interesado, por lo tanto en el momento de que el mismo vendedor o cualquier otro realice una legalización, el cliente automáticamente pasa a ser del registro del vendedor que realizó su registro.

2.1.1 Ambiente del Usuario CIC. (Modulo de Administración de Clientes) es un sistema que se desenvuelve en la Web, por tal motivo el principal entorno grafico que se manejara será cualquier navegador de cualquier sistema operativo.

Representativo	• Persona con perfil de Administrador
Descripción	• Persona autentificada en el Sistema encargada de la revisión y control del sobre cada vendedor y sus planes de venta con cada cliente registrados durante un rango de fecha determinada.
Tipo	• Administrador
Responsabilidades	• Mantener el control de las Comisiones que cada Vendedor tenga con las legalizaciones realizadas a los clientes.
Criterio Satisfactorio	•El control de las comisiones del vendedor sobre los clientes legalizados este actualizada y el acceso sea confiable y seguro.
Salidas	• Información de los registros de clientes de un vendedor en un rango de fechas seleccionado, los atributos que mostrara el sistema son (No, No Contrato, Cliente, Producto, Tarifa, Porcentaje, Comisión, No Factura, Vigencia # de meses, Recibo de caja, Firma Teso, Fecha).
Comentario / Problemas	• Es importante que la información que se encuentre en el Sistema de Registro de Clientes este actualizada por el vendedor.

Los Usuarios tendrán acceso al sistema internamente, desde la empresa, o globalmente, desde cualquier sitio donde se tenga acceso a Internet y en ambos casos se obtendrán los mismos resultados, garantizando seguridad y confiabilidad a la hora del envío de Información

La filosofía de "Interfaz CIC" proporciona al usuario la manera más agradable y sencilla de obtener la información necesaria en cualquier momento con tan solo dar un clic.

2.1.2 Perfil Del Usuario

2.1.2.1 <Administrador>

Representativo	• Persona con perfil de Administrador
Descripción	• Persona autentificada en el Sistema encargada de la revisión y control del sobre cada vendedor y sus planes de venta con cada cliente registrados durante un rango de fecha determinada.
Tipo	• Administrador
Responsabilidades	• Mantener el control de las Comisiones que cada Vendedor tenga con las legalizaciones realizadas a los clientes.
Criterio Satisfactorio	•El control de las comisiones del vendedor sobre los clientes legalizados este actualizada y el acceso sea confiable y seguro.
Salidas	• Información de los registros de clientes de un vendedor en un rango de fechas seleccionado, los atributos que mostrara el sistema son (No, No Contrato, Cliente, Producto, Tarifa, Porcentaje, Comisión, No Factura, Vigencia # de meses, Recibo de caja, Firma Teso, Fecha).
Comentario / Problemas	• Es importante que la información que se encuentre en el Sistema de Registro de Clientes este actualizada por el vendedor.

2.1.2.2<Vendedor>

Representativo	• Persona encargada de realizar el registro y legalización de los Clientes
Descripción	Persona autentificada en el Sistema que lleva el registro de Gestión (Llamadas realizadas) realizado a cada uno de los clientes.
Tipo	• Usuario del Sistema
Criterio Satisfactorio	• El control de clientes del sistema está representado en la facilidad, rapidez y comodidad de realizar el proceso de legalización a Clientes de una manera Confiable y segura.
Salidas	• Registro de nuevos clientes.
Comentarios / Problemas	• El ingreso de los Datos de Cada cliente y su legalización es el proceso que se lleva a cabo para la generación de las comisiones de cada vendedor. Es importante tener los datos íntegros del Cliente actualizados.

ANEXO B

Evaluación Final Análisis del Sistema

Fecha: 5 de Enero del 2003.

- **MÓDULO DE GESTIÓN**
  Descripción del Usuario

1. ¿Se ha estado en contacto permanente con el cliente?

SI_**X**_ NO___

Observaciones: El cliente ha sido vital en el proceso, ya que los aportes brindados hacen que el sistema cumpla con las expectativas iniciales.

2. ¿Se ha descrito de manera específica y adecuada las características de cada uno de los usuarios del Sistema. ?

SI_**X**_ NO___

Observaciones: Los usuarios fueron especificados con sus respectivas funciones dentro del sistema.

3. ¿Existen inconsistencias, omisiones o redundancias en las descripciones de cada uno de los usuarios del modulo?

SI__ NO__**X**_

Observaciones: No se presentan este tipo de problemas.

4. ¿La definición del contexto en que los usuarios van a trabajar esta descrita de una forma correcta y precisa?

SI_**X**_ NO___

Observaciones: Los documentos son el respaldo en el cual se puede mirar cualquier inquietud presentada durante el proceso.

4. ¿Se han considerado en detalle el contorno en que el usuario se va a desempeñar?

SI_**X**_ NO___

Observaciones: Las funciones y procedimientos son claramente definidas.

- **MÓDULO DE ADMINISTRACION DE CLIEN TES**
  Descripción del Usuario

1. ¿Se ha estado en contacto permanente con el cliente?

SI_**X**_ NO___

Observaciones: El usuario ha manifestado sus dudas e inquietudes con respecto al manejo y desarrollo del sistema.

2. ¿Ha revisado el usuario el manual de usuario preliminar o el prototipo?

SI_**X**_ NO___

Observaciones: El manual ha sido diseñado entre usuario y producción.

3. ¿Se ha descrito de manera específica y adecuada las características de cada uno de los usuarios del Sistema. ?

SI_**X**_ NO___

Observaciones: Las funciones generales y específicas son bien definidas.

4. ¿Existen inconsistencias, omisiones o redundancias en las descripciones de cada uno de los usuarios del modulo?

SI__ NO_**X**_

Observaciones: **Ninguna**

5. ¿La definición del contexto en que los usuarios van a trabajar esta descrita de una forma correcta y precisa?

SI_**X**_ NO___

Observaciones: No hay confusión en cuanto al contexto.

6. ¿Se han considerado en detalle el contorno en que el usuario se va a desempeñar?

SI_**X**_ NO___

Observaciones: Todos los puntos han sido tomados en cuenta para el contorno del usuario.

Yo SERGIO CADENA, firmo de conformidad en que todas las observaciones y comentarios realizados al Análisis del Sistema, Control Interno de Clientes, en las evaluaciones previas, han sido incluidos dentro del documento "Anexo1. Análisis del Sistema" realizado por Jhonn Jairo Vesga Cadena y Andrés Julián Guzmán Díaz, teniendo como precedente y cotejándolo con el listado anterior, Evaluación Análisis del Sistema".

_____

SERGIO CADENA
FIRMO DE CONFORMIDAD
Bucaramanga, Colombia, A 5 de Enero del 2003

ANEXO C

Descripción de casos de uso

Fecha	Versión	Descripción	Autor
15/Enero/04	1.0	Primera versión del Proyecto	Jhonn Jairo Vesga Cadena Andrés Julián Guzmán Díaz.

TABLA DE CONTENIDO ESPECIFICACIÓN DE CASOS DE USO

CONTENIDO

CASOS DE USO

1. LOGIN Y PASSWORD

1.1 BREVE DESCRIPCION

Interfaz ubicada en la página principal, donde el usuario (Administrador y Vendedor) se autentifica siendo verificado por el Sistema, determinando el perfil autorizado para usar los módulos señalados.

Se inicia con el llenado de los campos login y password y termina cuando se permite el acceso al Sistema.

1.2. FLUJO DE EVENTOS

El Usuario (Administrador y Vendedor) hace solicitud en el navegador de ingreso al Sistema. El la pagina de Inicio de Sesión la primera vista del cliente será una ventana en donde se mostraran los campos de login y password, el usuario ingresara en los campos los datos requeridos para acceder al modulo.

1.3. PRECONDICIÓN

- Ser una persona con un perfil creado en el Sistema

1.4. POSCONDICIÓN

- Se realizará la conexión y se permitirá el acceso al Modulo.

2. MODULO DE GESTIÓN

2.1 REPORTE DE GESTION DE USUARIOS

2.1.1 Breve Descripción

Servicio del Sistema que permite llevar el control de la llamadas realizadas a cada cliente.

Modulo de Gestión de Usuarios Registrados que se inicia con la selección de un rango (fecha), se selecciona un cliente del reporte y finaliza con el ingreso de la observación realizada al cliente.

2.1.2. Flujo De Eventos

- El vendedor hace solicitud al sistema del Modulo de Gestión.
- El Sistema mostrará una página “Gestión de Usuarios Registrados” en donde el vendedor debe ingresar el rango deseado (fecha) para la generación del *Reporte de Gestión*; una lista de clientes del vendedor.
- El vendedor seleccionara el cliente y dando clik en “Hacer Gestión”, ingresara a la página del cliente donde guardará la información pertinente del cometido realizado, siendo esta especificada con un “Nombre de Contacto y Observación”. Del mismo modo va a encontrar registros anteriores de llamadas.

2.1.3. PRECONDICIÓN

- Ser un Vendedor registrado en el Sistema.
- Tener por lo menos un Cliente a realizar Gestión

2.1.4. POSCONDICIÓN

- Se mostraran los registros de las llamadas realizadas al cliente cada registro con su
- Nombre de contacto y observación de la llamada.
- Se agregara un nuevo registro de la nueva Gestión Realizada al cliente seleccionado.

3. MODULO DE ADMINISTRACIÓN DE CLIENTES

3.1. INGRESO DE CLIENTES

3.1.1 Breve Descripción

Servicio del Sistema que permite ingresar en la base de datos un cliente potencial para el vendedor.

Modulo de Administración de Clientes que se inicia con el ingreso de datos de un cliente potencial y finaliza con el registro para el vendedor en el sistema.

3.1.2. Flujo De Eventos

El vendedor hace solicitud al sistema del Ingreso de Clientes ya sea como "Persona Natural, o Jurídica".

El Sistema mostrará una página "Registro de Clientes Persona Natural/Jurídica" en donde el vendedor debe ingresar los datos (Nombre, Apellido, Teléfono, Medio) del cliente.

El vendedor enviará los Datos para agregar el registro a su Base de Datos.

3.1.3. Precondición

- Ser un Vendedor registrado en el Sistema.
- Tener los Datos del Cliente.

3.1.4. Poscondición

Se agregará el registro del cliente Potencial.

3.2. LEGALIZACIÓN DE CLIENTES

3.2.1 Breve Descripción

Servicio del Sistema que permite Legalizar un Cliente.
Legalizar: Dejar registro del contrato cuando el cliente ya tomo el servicio.

Modulo de Administración de Clientes Legalizados que se inicia con el ingreso de datos de un cliente y finaliza con el contrato (tipo de plan y vigencia) y el registro para el vendedor en el Sistema.

3.2.2. Flujo De Eventos

El vendedor hace solicitud al sistema de Legalización de Cliente.
El Sistema mostrará una página "*Legalización de Contratos*" en donde el vendedor debe ingresar los datos (No Contrato, Cliente, Teléfono, Login, Vigencia, Plan, Forma de pago) del contrato.
El vendedor enviará los Datos para agregar el registro a la tabla del vendedor y a la tabla de clientes.

3.2.3. Precondición

- Ser un Vendedor registrado en el Sistema.
- Tener los Datos del Cliente.
- Tener los Datos del Contrato.

3.2.4. Poscondición

Se agregará el registro del cliente y su contrato.

3.3. CONSULTAR COMISIONES

3.3.1 Breve Descripción

Servicio del Sistema que permite consultar las comisiones de un Vendedor en un

rango de fecha establecida.

Modulo del Administrador se inicia seleccionando a un Vendedor y estableciendo un Rango de Fecha y finaliza con el reporte de clientes (Datos de cliente y contrato), asi como la comisión obtenida y el total de la misma.

3.3.2. Flujo De Eventos

El Administración hace solicitud al sistema de Reporte de comisiones.
El Sistema mostrará una página "*Reporte Ventas Legalizadas por Vendedores*" en donde el Administrador escoge un Vendedor y el rango de fechas a ser reportado.
El Sistema mostrará el reporte de los clientes legalizados en ese periodo, las especificaciones del contrato y las comisiones del vendedor.

3.3.3. Precondición

- Tener Perfil de Administrador.

3.3.4. Poscondición

- Se mostrará el reporte seguro y Actualizado.

ANEXO D

Evaluación Final Descripción de Casos de Uso

Fecha: Febrero 1 del 2004.

1. ¿Se ha estado en contacto permanente con el cliente?

SI_**X**_ NO___

Observaciones: **Ninguna**

CASO DE USO: Login y Password

1. ¿Se han realizado las descripciones referentes al caso de uso *Login y Password* correctamente?

SI_**X**_ NO___

Observaciones: **Ninguna**

2. ¿Esta claramente definido el Flujo de Eventos según las necesidades del software para el caso de uso *Login y Password*?

SI_**X**_ NO___

Observaciones: **Ninguna**

3. ¿Tanto la Precondición como la Poscondición, son claros?

SI_**X**_ NO___

Observaciones: **Ninguna**

4. ¿Existen inconsistencias, omisiones o redundancias en la Descripción del Caso de Uso *Login y Password*?

SI_**X**_ NO___

Observaciones: **Ninguna**

MODULO DE GESTION

CASO DE USO: Reporte de Gestión de Usuarios

1. ¿Se han realizado las descripciones referentes al caso de uso *Reporte de Gestión de Usuarios* correctamente?

SI_**X**_ NO___

Observaciones: **Ninguna**

2. ¿Esta claramente definido el Flujo de Eventos según las necesidades del software para el caso de uso *Reporte de Gestión de Usuarios*?

SI_**X**_ NO___

Observaciones: **Ninguna**

3. ¿Tanto la Precondición como la Poscondición, son claros?

SI_**X**_ NO___

Observaciones: **Ninguna**

4. ¿Existen inconsistencias, omisiones o redundancias en la Descripción del Caso de Uso *Reporte de Gestión de Usuarios*?

SI_**X**_ NO___

Observaciones: **Ninguna**

MODULO DE ADMINISTRACIÓN DE CLIENTES

CASO DE USO: Ingreso de Clientes

1. ¿Se han realizado las descripciones referentes al caso de uso *Ingreso de Clientes* correctamente?

SI_**X**_ NO___

Observaciones: **Ninguna**

2. ¿Esta claramente definido el Flujo de Eventos según las necesidades del software para el caso de uso *Ingreso de Clientes*?

SI_**X**_ NO___

Observaciones: **Ninguna**

3. ¿Tanto la Precondición como la Poscondición, son claros?

SI_**X**_ NO___

Observaciones: **Ninguna**

4. ¿Existen inconsistencias, omisiones o redundancias en la Descripción del Caso de Uso *Ingreso de Clientes*?

SI_**X**_ NO___

Observaciones: **Ninguna**

CASO DE USO: Legalización de Clientes

1. ¿Se han realizado las descripciones referentes al caso de uso *Legalización de Clientes* correctamente?

SI_**X**_ NO___

Observaciones: **Ninguna**

2. ¿Esta claramente definido el Flujo de Eventos según las necesidades del software para el caso de uso *Legalización de Clientes*?

SI_**X**_ NO___

Observaciones: **Ninguna**

3. ¿Tanto la Precondición como la Poscondición, son claros?

SI_**X**_ NO___

Observaciones: **Ninguna**

4. ¿Existen inconsistencias, omisiones o redundancias en la Descripción del Caso de Uso *Legalización de Clientes*?

SI_**X**_ NO___

Observaciones: **Ninguna**

CASO DE USO: Consultar Comisiones

1. ¿Se han realizado las descripciones referentes al caso de uso *Consultar Comisiones* correctamente?

SI_**X**_ NO___

Observaciones: **Ninguna**

2. ¿Esta claramente definido el Flujo de Eventos según las necesidades del software para el caso de uso *Consultar Comisiones*?

SI_**X**_ NO___

Observaciones: **Ninguna**

3. ¿Tanto la Precondición como la Poscondición, son claros?

SI_**X**_ NO___

Observaciones: **Ninguna**

4. ¿Existen inconsistencias, omisiones o redundancias en la Descripción del Caso de Uso *Consultar Comisiones*?

SI_**X**_ NO___

Observaciones: **Ninguna**

Yo SERGIO CADENA, firmo de conformidad en que todas las observaciones y comentarios realizados en la Descripción de Casos de Uso del Sistema, Control Interno de Clientes, en las evaluaciones previas, han sido incluidos dentro del documento "Anexo2. Descripción de Casos de Uso" realizado por Jhonn Jairo Vesga Cadena y Andrés Julián Guzmán Díaz, teniendo como precedente y cotejándolo con el listado anterior, Evaluación Descripción de Casos de Uso".

_____

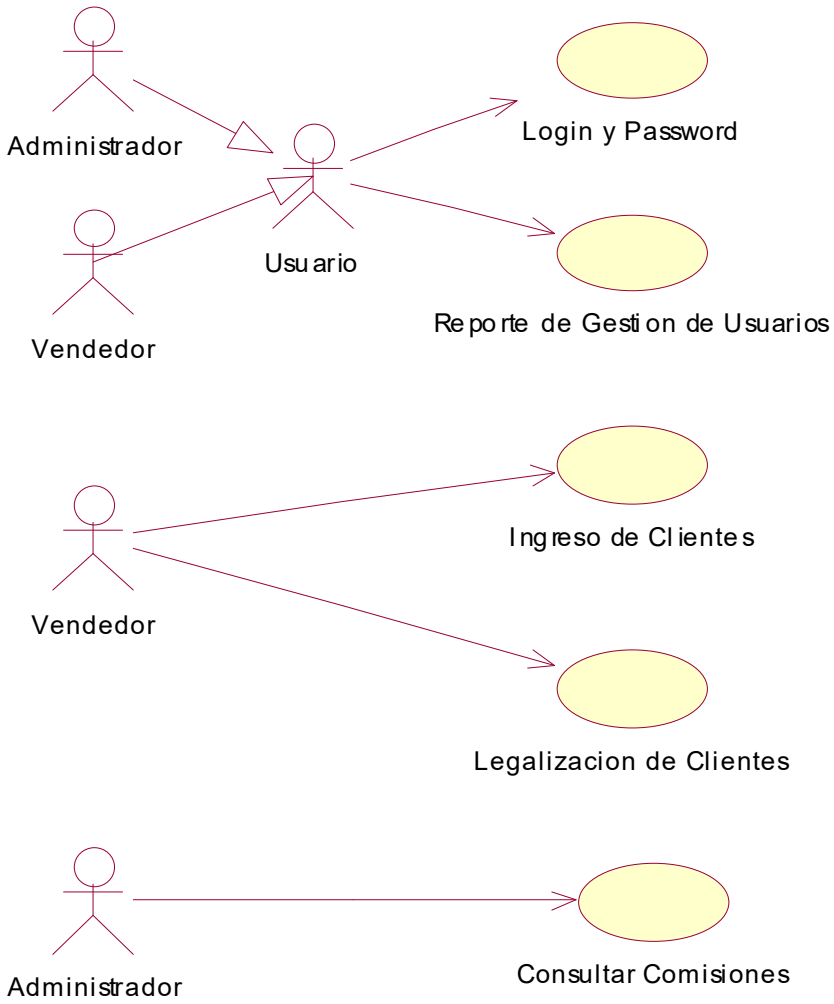
SERGIO CADENA
FIRMO DE CONFORMIDAD

Bucaramanga, Colombia, A 1 de Febrero del 2004

ANEXO E

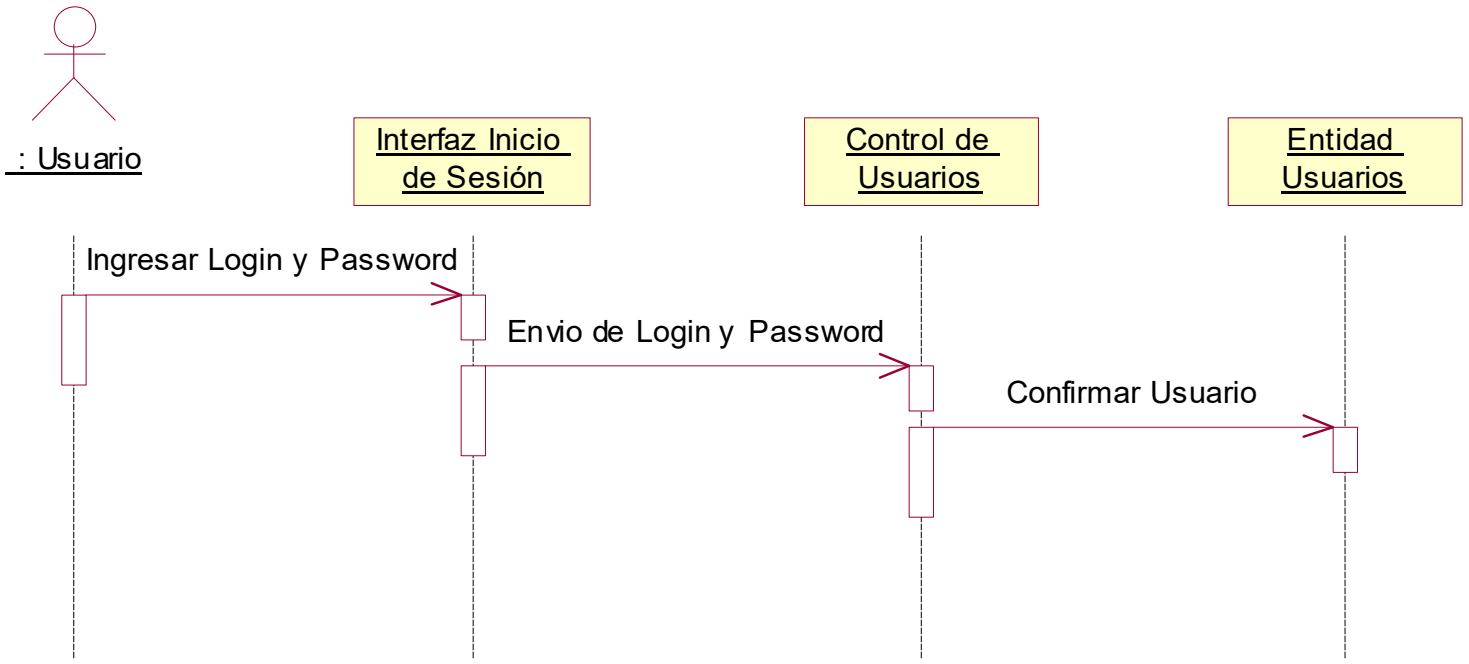
Diagramas UML

DIAGRAMA DE CASOS DE USO



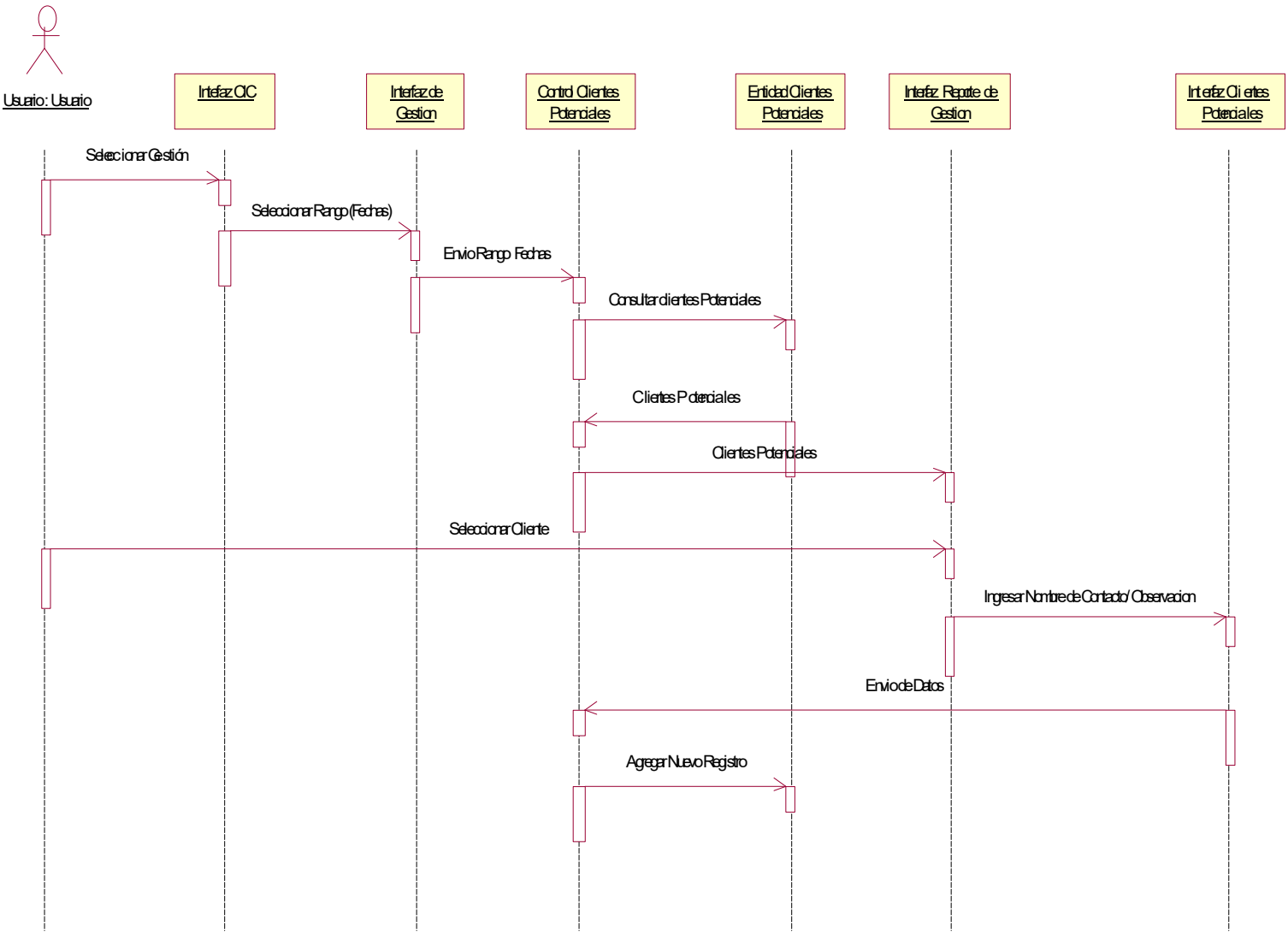
DIAGRAMAS DE SECUENCIA

LOGIN Y PASSWORD



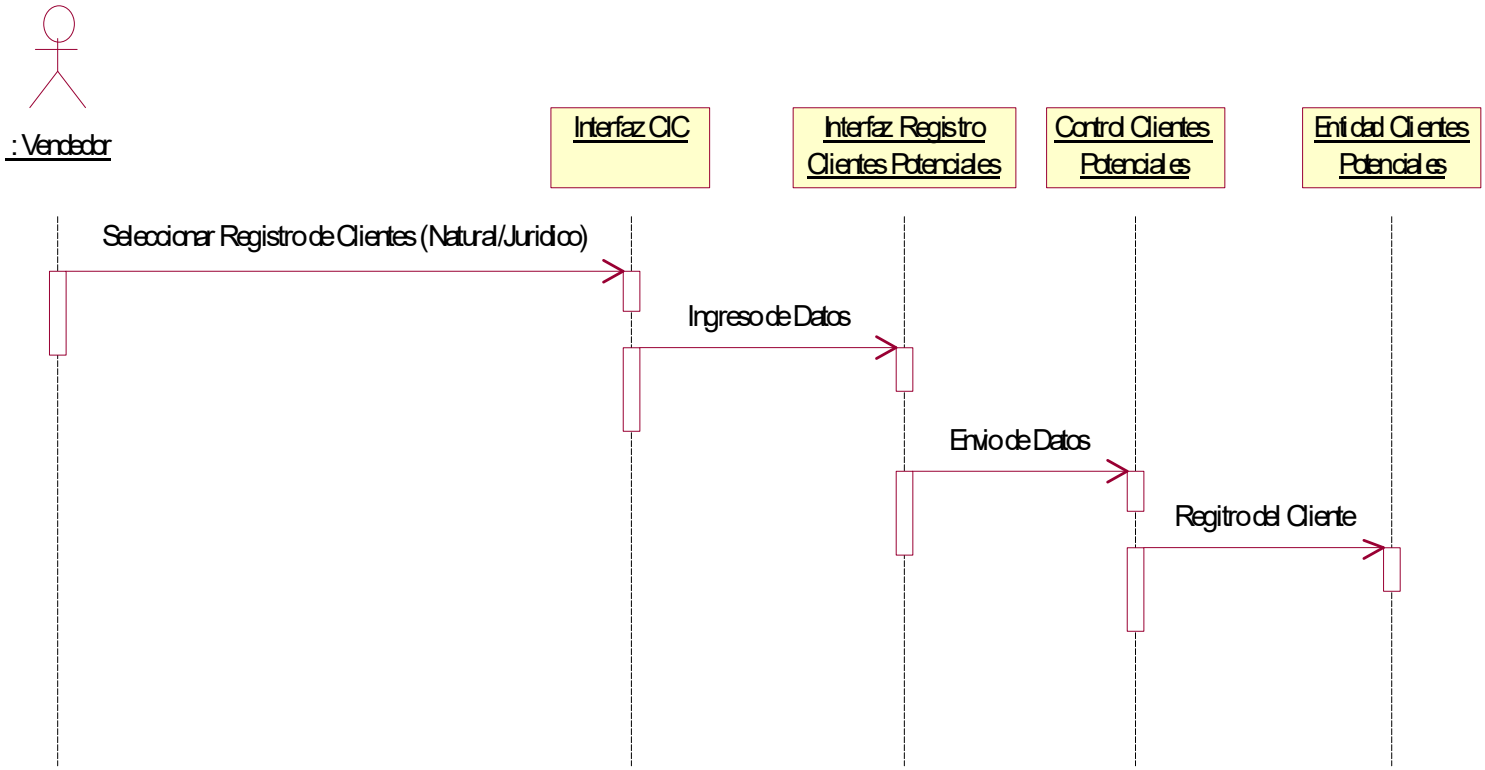
MODULO DE GESTION

REPORTE DE GESTION DE USUARIOS

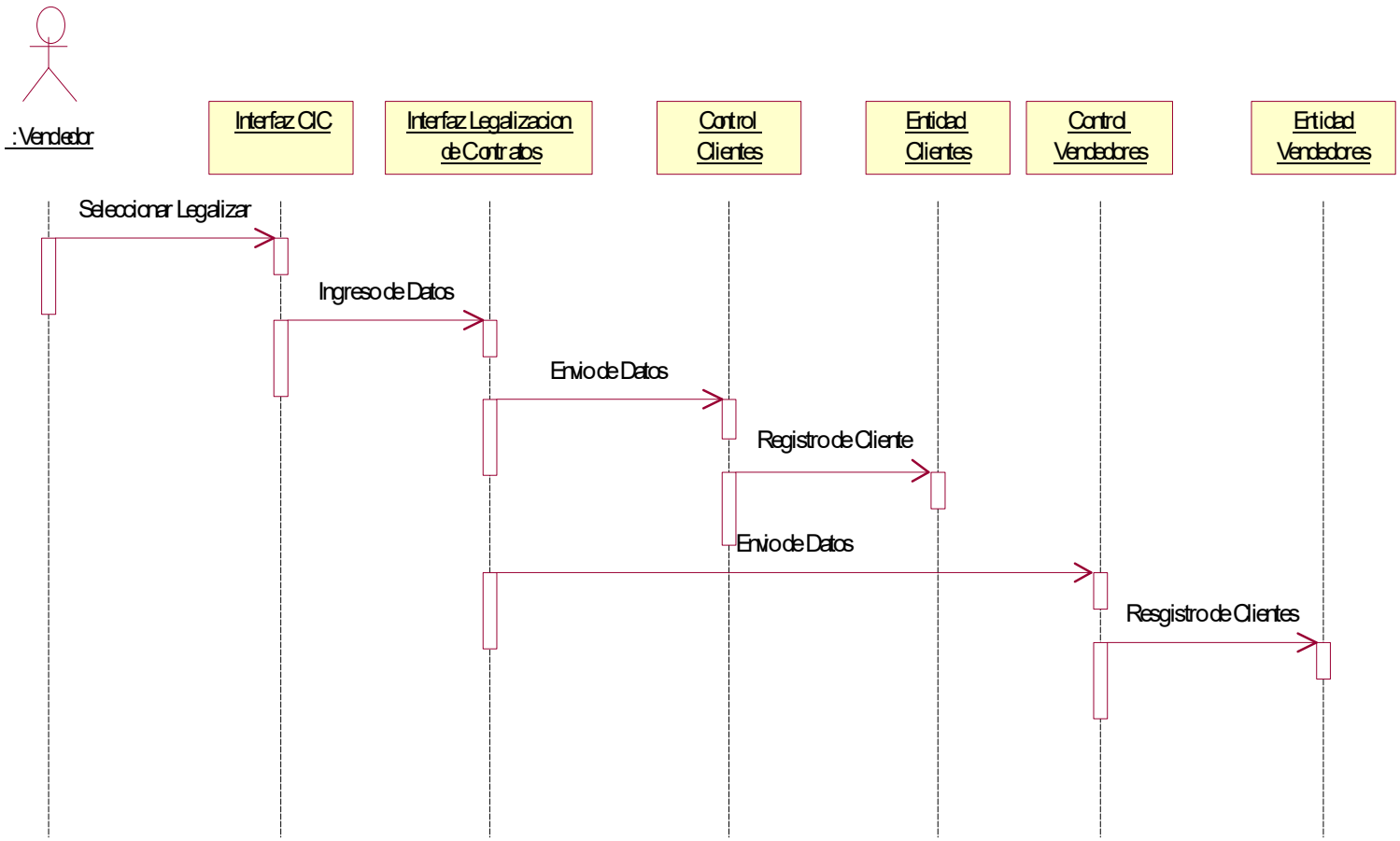


MODULO DE ADMINISTRACION DE CLIENTES

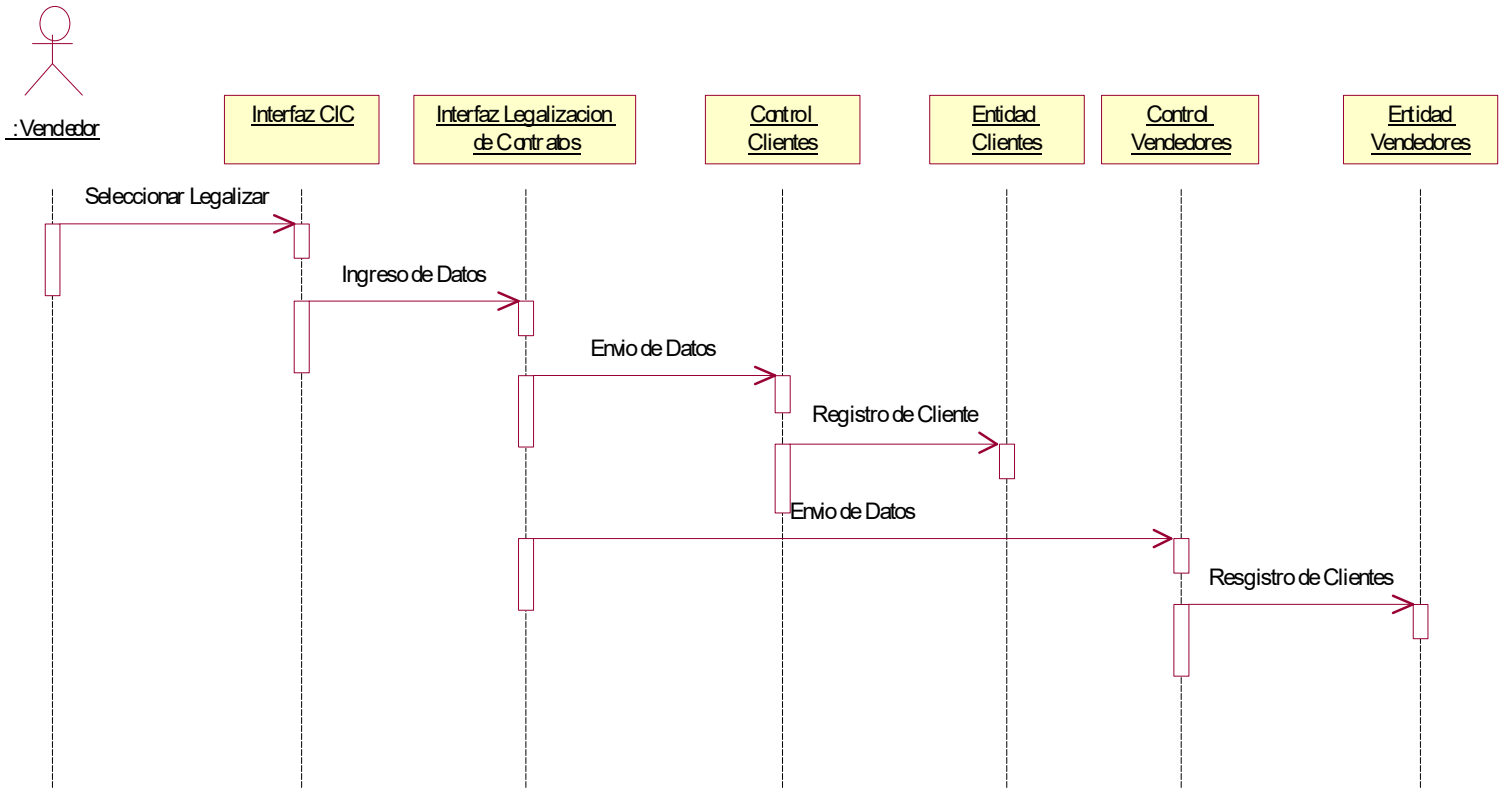
INGRESO DE CLIENTES



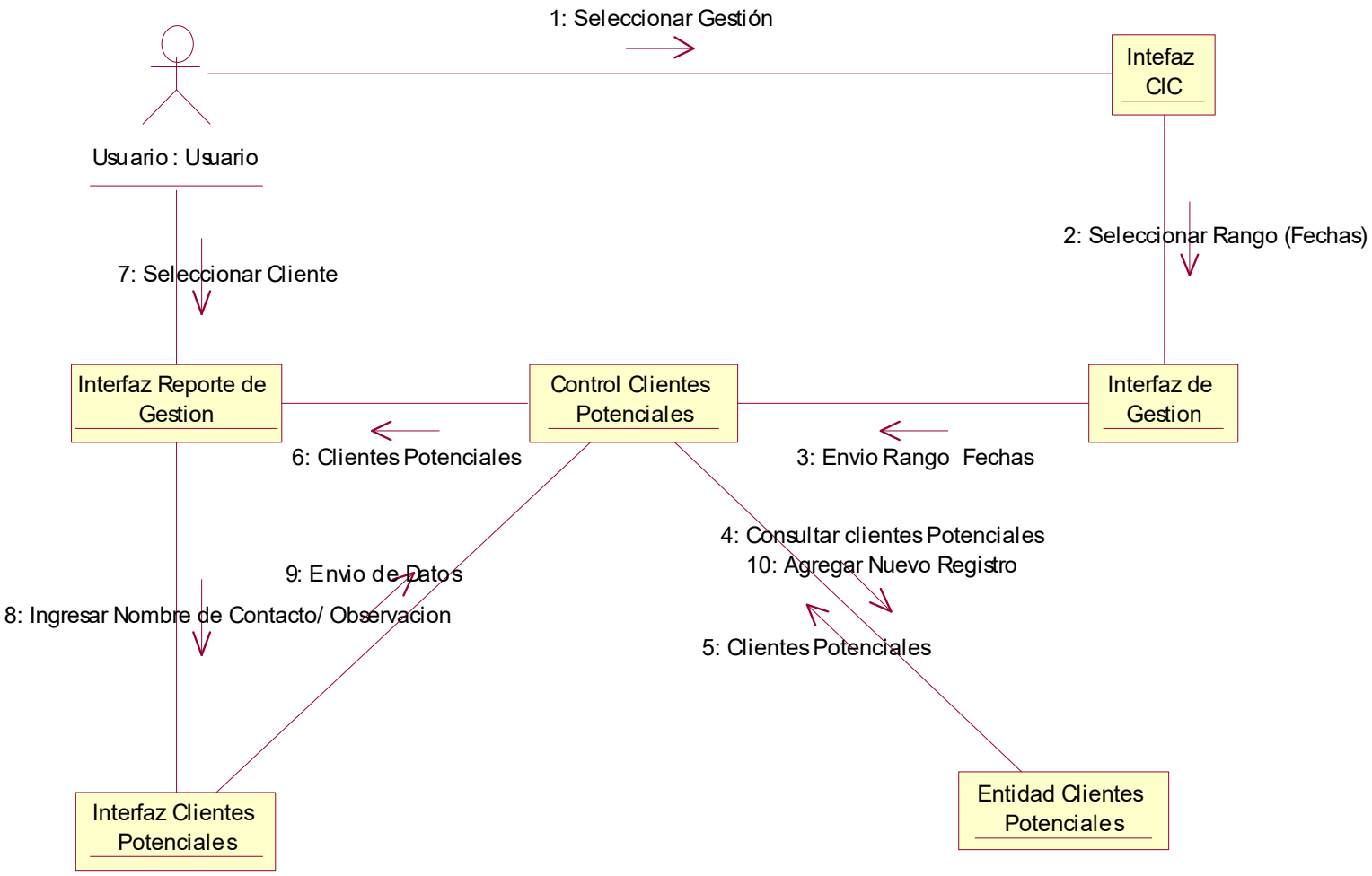
LEGALIZACION DE CLIENTES



CONSULTAR COMISIONES

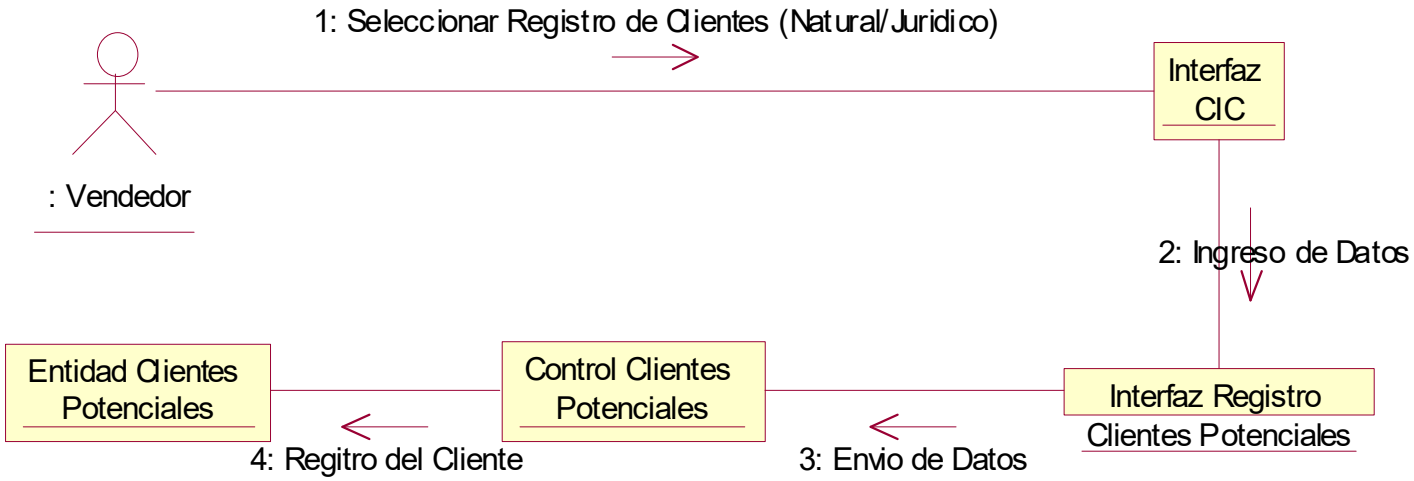


DIAGRAMAS DE COLABORACIÓN
REPORTE DE GESTION DE USUARIOS



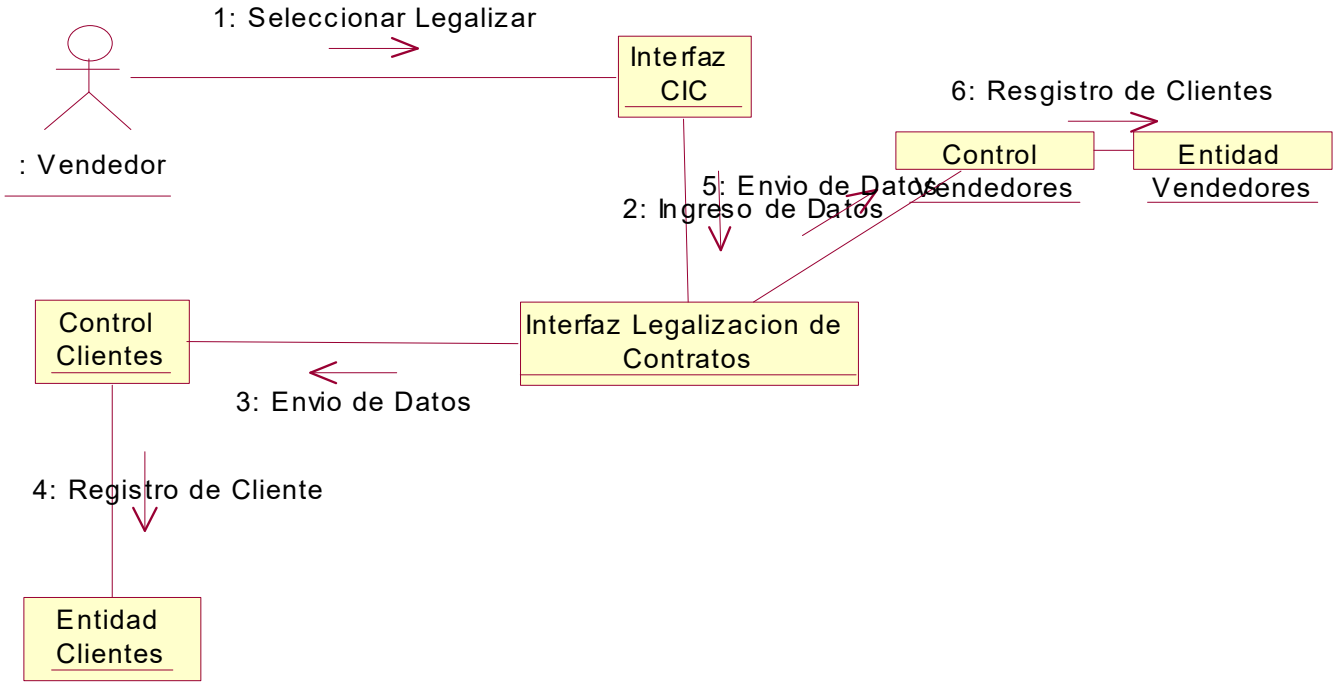
MODULO DE ADMINISTRACION DE CLIENTES

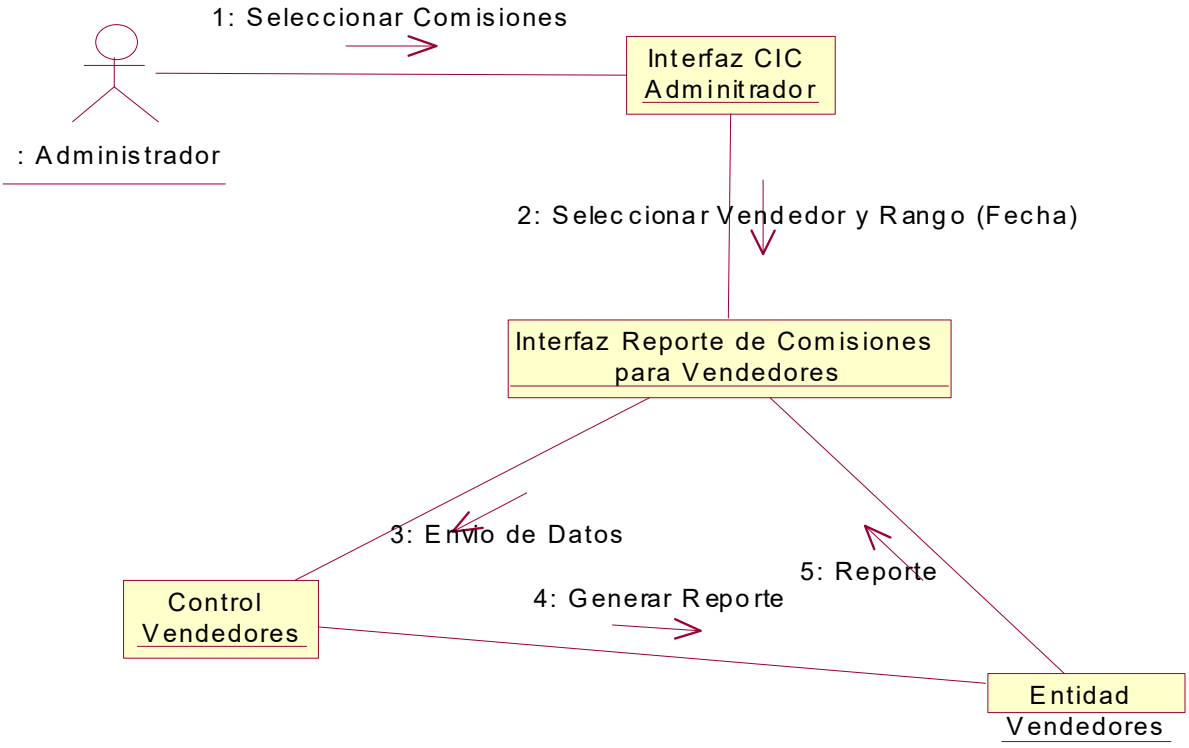
INGRESO DE CLIENTES



LEGALIZACION DE CLIENTES

CONSULTAR COMISIONES




1: Seleccionar Comisiones
Interfaz CIC
Adminitrador
: Administrador
2: Seleccionar Vendedor y Rango (Fecha)
Interfaz Reporte de Comisiones
para Vendedores
3: Envio de Datos
5: Reporte
4: Generar Reporte
Control
Vendedores
Entidad
Vendedores

ANEXO F

Evaluación Final Diagramas UML

Fecha: Febrero 25 del 2004.

1. ¿Se ha estado en contacto permanente con el cliente?

SI_**X**_ NO___

Observaciones: **Ninguna**

DIAGRAMA DE CASOS DE USO

2. ¿Se ha definido cada uno de los actores del sistema?

SI_**X**_ NO___

Observaciones: **Ninguna**

3. ¿La relación que existe entre el Actor y el caso de uso esta relacionada y representa lo mismo que en el documento Anexo2.Descripción de Casos de Uso?

SI_**X**_ NO___

Observaciones: **Ninguna**

4. ¿Son Claros los Diagramas?

SI_**X**_ NO___

Observaciones: **Ninguna**

DIAGRAMAS DE SECUENCIA Y COLABORACIÓN

5. ¿Se encuentra representado el comportamiento del software con la información que debe procesar y con las funciones que debe realizar?

SI_**X**_ NO___

Observaciones: **Ninguna**

6. ¿Existen inconsistencias, omisiones o redundancias en los diagramas?

SI_**X**_ NO___

Observaciones: **Ninguna**

7. ¿Se han descrito todas las interfaces importantes de todos los elementos del sistema?

SI_**X**_ NO___

Observaciones: **Ninguna**

8. ¿Se encuentra representada la interacción entre los objetos y los mensajes que intercambian entre sí junto con el orden temporal de los mismos?

SI_**X**_ NO___

Observaciones: **Ninguna**

9. ¿Se han considerado todas las especificaciones funcionales del sistema?

SI_**X**_ NO___

Observaciones: **Ninguna**

Yo SERGIO CADENA, firmo de conformidad en que todas las observaciones y comentarios realizados en la Descripción de Casos de Uso del Sistema, Control Interno de Clientes, en las evaluaciones previas, han sido incluidos dentro del documento "Anexo2. Descripción de Casos de Uso" realizado por Jhonn Jairo Vesga Cadena y Andrés Julián Guzmán Díaz, teniendo como precedente y cotejándolo con el listado anterior, Evaluación Diagramas UML

_____

SERGIO CADENA
FIRMO DE CONFORMIDAD
Bucaramanga, Colombia, A 25 de Febrero del 2004

ANEXO G

Evaluación Final Entrega de Módulos

Fecha: 12 de Abril del 2004.

- **MÓDULO DE GESTIÓN**

Según la descripción del Módulo de Gestión realizada en el Anexo1. Análisis del Sistema, tanto los *usuarios finales* (vendedores) como el *Administrador* trabajaron y colocaron a prueba desde el día 29 de Marzo del 2004 al 12 de abril del mismo año, realizando observaciones y aclaraciones corregidas en ese periodo para su total entrega y satisfacción por parte del cliente.

ADMINISTRADOR

1. ¿Cotejando la información obtenida por medio del sistema, con la gestión realizada por cada uno de los vendedores a los clientes potenciales tanto en el sistema como manualmente pudieron comprobar la *fiabilidad* del sistema?

SI_**X**_ NO___

Observaciones: Se realizaron talleres de pruebas en los cuales se verificó la información entregada por el sistema junto con los reportes físicos (contratos), teniendo una respuesta del 100% de concordancia entre ambos resultados.

2. ¿Supliendo los procesos manuales ahora en el Sistema CIC, certifica que las interfaces están totalmente claras asistiendo a su fácil administración ?

SI_**X**_ NO___

Observaciones: En los talleres realizados inicialmente se mostraba la interfaz sin dar indicaciones previas de cómo se debía usar, obligando a utilizarlo de manera intuitiva, logrando en un principio un resultado de 80% aciertos, una vez dada la orientación respecto a la funcionalidad de la interfaz, todo fue aclarado y corregido a un 100%.

3. ¿La autentificación como administrador es correcta, de tal forma que los privilegios obtenidos son los mismos definidos en el anexo1. Análisis del Sistema?

SI_**X**_ NO___

Observaciones: Las diferentes pruebas realizadas mostraban de manera efectiva los diferentes menús correspondientes, evitando así el acceso a ciertas áreas de trabajo, determinando así claramente las funciones disponibles para cada usuario, evitando en un 100% el acceso a áreas no permitidas o de acceso limitado a administradores.

4. ¿El control de clientes y vendedores del sistema está representado en la facilidad, rapidez y comodidad de realizar la Gestión a los vendedores de una manera Confiable y segura?

SI_**X**_ NO___

Observaciones: Los vendedores se mostraron satisfechos al usar su módulo correspondiente, ya que elimino totalmente el uso de papeles, agendas y

recordatorios en donde anotar aquellas labores que el módulo a tomado como propias.

VENDEDOR

4. ¿Cotejando la información obtenida por medio del sistema, con la gestión realizada por cada vendedor a los clientes potenciales tanto en el sistema como manualmente pudieron comprobar la *fiabilidad* del sistema?

SI_**X**_ NO___

Observaciones: La gestión de los clientes se agilizó en un 80%, ya que podían tener acceso inmediato a las características principales del sistema, disminuyendo el tiempo de ejecución en cada tarea y teniendo un completo control en cada una de sus gestiones, con comentarios hora y día realizados.

5. ¿Supliendo los procesos manuales ahora en el Sistema CIC, certifica que las interfaces están totalmente claras asistiendo a su fácil administración ?

SI_**X**_ NO___

Observaciones: Se revisaron una a una las interfaces con sus respectivas opciones, se encontraron errores ortográficos los cuales fueron posteriormente corregidos.

6. ¿La autentificación como vendedor es correcta, de tal forma que respeta la sesión de cada uno, mostrando la información correcta de los clientes potenciales sin intervenir con la información presente de los demás vendedores?

SI_**X**_ NO___

Observaciones: Se realizaron pruebas de mezcla de sesiones y no hubo inconvenientes, se corrigió un problema de tiempos de sesión, el cual era muy corto y hacia que se cerrara el sistema en varias ocasiones.

7. ¿El control de clientes potenciales está representado en la facilidad, rapidez y comodidad de realizar la Gestión a cada uno de ellos de manera Confiable y segura?

SI_**X**_ NO___

Observaciones: En esta prueba hubo cierto descontento por una regla interna de la empresa que antes no podía ser controlada por el gerente, pero si por el sistema, aunque esto se salía de nuestras manos, mostrando al cliente que las reglas planteadas se cumplían correctamente.

MODULO DE ADMINISTRACIÓN DE CLIENTES

ADMINISTRADOR

8. ¿Cotejando la información obtenida por medio del sistema, con la legalización de contratos especificando el tipo de contrato realizado por cada vendedor y efectuando su reporte de comisión tanto en el sistema como manualmente pudieron comprobar la *fiabilidad* del sistema?

SI_**X**_ NO___

Observaciones: Completamente, el sistema arroja datos 100% fiables, los cuales son usados en el departamento de contabilidad.

9. ¿Considera que el sistema es Fácil de usar, rápido y cómodo para de realizar las consultas, confiable al sacar las comisiones, y segura por las autentificaciones y privilegios de los usuarios del sistema?

SI_**X**_ NO___

Observaciones: En las pruebas los diferentes departamentos vinculados con esta labor quedaron satisfechos con los resultados arrojados por el sistema.

VENDEDOR

10. ¿Cotejando la información obtenida por medio del sistema, con la legalización de contratos especificando el tipo de contrato realizado y efectuando su reporte de comisión tanto en el sistema como manualmente pudieron comprobar la *fiabilidad* del sistema?

SI_**X**_ NO___

Observaciones: Los vendedores se sintieron satisfechos con las comisiones entregadas y no hubo problemas de inconsistencias en los dineros despachados.

11. ¿Considera que el sistema es Fácil de usar, rápido y cómodo para de realizar los registros y legalización de cliente y segura por las autentificaciones de cada vendedor en el Sistema?

SI_**X**_ NO___

Observaciones: Sin duda, los vendedores encuentran el módulo bastante intuitivo, no hay problemas de confusión después de la capacitación brindada.

Yo SERGIO CADENA, firmo de conformidad en que todas las observaciones y comentarios realizados en periodo de Pruebas de la entrega de módulos del Sistema, Control Interno de Clientes, han sido revisadas y satisfactoriamente corregidas por Jhonn Jairo Vesga Cadena y Andrés Julián Guzmán Díaz, teniendo como precedente y cotejándolo con el listado anterior, Evaluación Entrega de Módulos".

_____

SERGIO CADENA
FIRMO DE CONFORMIDAD
Bucaramanga, Colombia, A 12 de Abril del 2004