Universidad de las Ciencias Informáticas

Facultad 5

Título: Desarrollo del flujo de requisitos para la Metodología Análisis de Criticidad Integral de Activos del producto SCIA

Proyecto de tesis para optar por el título de
Ingeniero en Ciencias Informáticas

Autora: Heidy Alina Nuevo León.

Tutora: Irina Elena Argota Vega.

Ciudad de la Habana, Junio, 2009.

Irina Elena Argota Vega:

Graduado de Ingeniero en Informática en el 2007, profesor de la Universidad de las Ciencias Informáticas con categoría docente de Instructor, con dos años y medios de experiencia vinculado a la docencia y a la producción de software.

Dedicatoria

A mi abuelo (papi) por ser mi ángel guardián y mi guía.

A mi mamita linda, gracias a ti se han hecho realidad todos mis sueños, te amo.

A mi abuela (mami) gracias por tu alegría y amor en toda mi vida.

A mi papá por hacerme buscar lo mejor de mí.

A mi hermana, mi chiquitica linda, eres mi tesoro más preciado. A mi hermano, mi grandulón.

A mi novio por hacer de mí la ingeniera que soy, este triunfo es tuyo también. Gracias por la felicidad que me has dado, te amo.

A Jose, Pache no saben lo feliz que soy de tenerlos cerca de mí.

A mi madrina linda por su cariño y dulzura.

Agradecimientos

Agradezco a mis padres, mis abuelos, mis primos Migue, Damarys, Marlys, Claudia mis tíos Claro, Pedro, Nory, Lucy, Jose, Yami por darme su apoyo incondicional. A mi novio por su amor y dedicación. A mis hermanitas Yami y Yisel por apoyarme y brindarme su amistad incondicional. A mi hermanito Fito por darme mi primera sobrinita linda. A mis amigos Marcel, Koty, Adrian, Kike, Yanili, Loty, Sachel, Yaima por compartir conmigo durante estos 5 años de Universidad y los que faltan. A la familia de mi novio por hacerme sentir parte de ella. A mis compañeras de trabajo Dayany y Ludisley por el conocimiento, amistad y apoyo brindado. A Yaimí y Karen por ayudarme en este trabajo. A mi tutora Irina por su apoyo y guía en el transcurso de este año. A los muchachos de drivers Dr. Trujillo, Johnson y Pedro Uriarte por aguantarme durante mi aprendizaje de QT. A todos los que de una forma u otra me han ayudado en el transcurso de la carrera y en la elaboración de este trabajo.

SÍNTESIS

Actualmente, existen múltiples aplicaciones orientadas a dar resultados en las diferentes etapas de la confiabilidad operacional sin relación entre ellas y con costos de licenciamiento muy elevados, las que presentan criterios y metodologías distintas para los diferentes negocios, lo que afecta a la Gerencia de Mantenimiento y sus diferentes unidades de Confiabilidad, Planificación, Programación y Ejecución de Mantenimiento en los diferentes negocios de Petróleos de Venezuela SA. (PDVSA), llevando consigo retrasos en los análisis de confiabilidad debido a las pérdidas de tiempo en la búsqueda de información.

La Universidad de las Ciencias Informáticas (UCI) junto a un equipo de PDVSA, asumen el desarrollo del proyecto "Sistema de Confiabilidad Integral de Activos (SCIA) para los pueblos del Alba", que se fundamenta en la integración de conocimientos, disciplinas, métodos, procedimientos y herramientas para optimizar el Impacto Total de Costos, desempeño y exposición al riesgo en la Vida del Negocio**.**

Entre las herramientas del producto SCIA se encuentra el Módulo de Herramientas Integrales de Confiabilidad Operacional, lo constituye el punto de partida en el inicio del desarrollo del proyecto.

Dentro de éste módulo se define la Metodología Análisis de Criticidad Integral de Activos (ACIA) la cual tendrá como objetivo principal establecer una jerarquía o prioridades de instalaciones, sistemas, grupos de equipos y equipos. Esta metodología está basada en un diseño propio tomando como base los mejores criterios de los métodos de Ciliberti, Mantenimiento Basado en Criticidad, las normas API 580 y API 581 para equipos estáticos, y experiencias en el desarrollo de este tipo de estudios.

PALABRAS CLAVE

Activos, Confiabilidad Operacional, Confiabilidad, Criticidad.

Tabla de Contenido

Cada Día las exigencias de la producción y del mercado llevan a la realización de ajustes en los presupuestos y estos deben estar alineados con las políticas de mantenimiento existentes. Para poder realizar las actividades sin afectar los procesos, es necesario establecer patrones y políticas derivadas de herramientas confiables que puedan justificar las inversiones y gastos en un momento determinado dentro de los negocios de Petróleos de Venezuela SA. (PDVSA): Exploración, Producción, Refinación, Comercio.

A través de estudios realizados por especialistas de la institución se han identificado algunas inconsistencias que limitan el correcto y eficiente desempeño de las áreas asociadas a la confiabilidad operacional, resaltando:

- Uso de diversas herramientas privativas con altos costos de licenciamiento, sin derechos de análisis o extensión de sus funcionalidades.
- Desarrollo de soluciones particulares sobre hojas de cálculo del paquete de oficina de Microsoft Windows, orientadas a dar resultados a sus procesos de modo independiente y sin relacionarse con otros sistemas externos.
- No se posee una fuente de información unificada que permita realizar encuestas, análisis, estudios de los históricos de las fallas y los resultados obtenidos.
- No existe una homologación y validación de los criterios y metodologías utilizadas en las distintas locaciones de la empresa.
- No se aplican procedimientos para que las personas vinculadas directamente a la producción aporten sus experiencias a los departamentos de mantenimiento.

Producto a las relaciones Cuba-Venezuela, y a los proyectos existentes en la UCI y a los precedentes proyectos realizados bajo este convenio se asume el proyecto SCIA para los pueblos del Alba", conformando un equipo multidisciplinario de estudiantes, profesores de la UCI, los cuales trabajarán en conjunto con especialistas e Ingenieros de Confiabilidad del hermano país venezolano.

El Proyecto permitirá homologar metodologías de las diferentes etapas de la confiabilidad operacional, mejorar el proceso de gestión de mantenimiento, realizar análisis probabilístico de riesgo de exploración, perforación y yacimientos petrolíferos mejorando la eficiencia en las operaciones de los procesos realizados en PDVSA.

A diferencia de los sistemas utilizados actualmente no poseen un ambiente amigable para el usuario, tampoco existe una integración de las tres etapas de la confiabilidad operacional, no poseen un Repositorio de Información Unificada (RIU) que faciliten los análisis y tienen costos de licenciamientos muy elevados. El SCIA estará basado en tecnologías y herramientas que permitan mejorar los procesos en la confiabilidad operacional y gestión de mantenimiento de forma escalable y distribuida con un enfoque a la soberanía tecnológica bajo la coordinación de PDVSA.

Dentro de las herramientas del modelo conceptual definidas para la realización del proyecto mostradas en la figura 1 se encuentra Metodología Análisis de Cuidado Integral de Activos (ACIA), sin embargo no se cuenta con las especificaciones de requisitos de este módulo, lo que imposibilita el avance en el desarrollo del módulo de "Herramientas Integrales de Confiabilidad Operacional"; tarea que le corresponde al analista de software dentro de la Fase Inicial del proyecto.

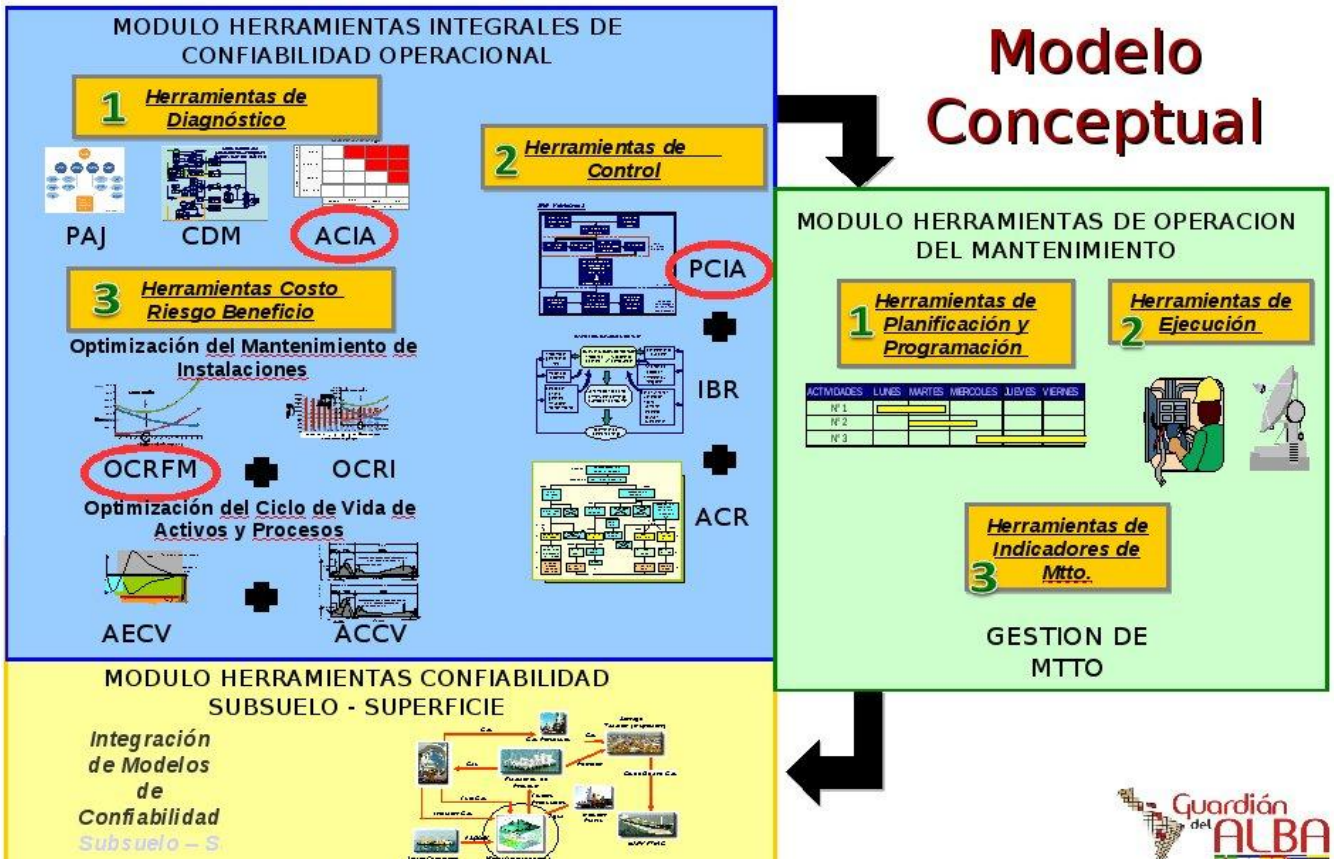


Figura 1 Modelo Conceptual

Para dar solución a la situación problemática existente se tiene que responder la presente interrogante del **Problema Científico:** ¿Cómo lograr una correcta gestión de requisitos en el desarrollo de la Metodología ACIA del producto SCIA?

El **Objeto de Estudio** de la investigación es el Proceso de Captura de Requisitos, y el **campo de acción**, el desarrollo de la Ingeniería de requisitos para la Metodología ACIA.

De acuerdo con el problema científico planteado la **Idea a Defender** definida es la siguiente:

Si se desarrolla eficientemente el flujo de requisitos de la Metodología ACIA para el proyecto SCIA entonces se dispondrá de una guía para el proceso de desarrollo de software.

El **Objetivo General** que se desea alcanzar para darle cumplimiento a este trabajo es:

Desarrollar el flujo de requisitos para la Metodología Análisis de Criticidad Integral de Activos del producto SCIA.

Definiéndose las siguientes **Tareas investigativas** para lograr un mayor nivel de detalles en la investigación:

- Estudio sobre el proyecto SCIA y la Metodología ACIA para la comprensión del problema.
- Realización del levantamiento de requisitos del software para el desarrollo de la metodología.
- Análisis de las técnicas de ingeniería de software para la captura de requisitos.
- Identificación de los casos de uso para la comprensión de la metodología.
- Descripción detallada de los casos de uso arquitectónicamente significativos para su mejor entendimiento.
- Realización del modelo de casos de uso para esquematizar el problema.
- Desarrollo del prototipo no funcional de interfaz de usuario del SCIA con la metodología hacia para su validación.

Para el desarrollo de las Tareas investigativas se combinan diferentes **Métodos y Técnicas en la búsqueda y procesamiento de la información**, los fundamentales son:

A nivel teórico:

- *Métodos de análisis-síntesis e inducción-deducción:* Para el estudio de las concepciones y conceptos empleados en el campo.
- *Análisis histórico-lógico:* Para conocer, con mayor profundidad, los antecedentes y las tendencias actuales referidas a las herramientas de confiabilidad existentes.

A nivel empírico

- *Entrevistas:* En el diagnóstico de necesidades, fueron entrevistados no sólo un grupo de profesionales, ingenieros y especialistas que desarrollan un determinado

rol dentro del proyecto y con experiencias de trabajos en el área de confiabilidad y mantenimiento, sino también a veteranos trabajadores de la industria de PDVSA.

- *Experimentos:* En la elaboración de prototipos funcionales, con el objetivo de comprobar los Casos de Uso que se fueron analizando.
- *Método de modelación:* Para la caracterización de las nuevas funcionalidades que tendrá la Metodología ACIA cuando se estaban definiendo los requisitos funcionales.

El presente documento está estructurado en tres capítulos:

- En el **Capítulo 1** se describe la concepción general del sistema, se hace referencia a las características del SCIA. Se tratan los principales conceptos de la Ingeniería de Software que tienen que ver con la investigación.
- En el **Capítulo 2** se hace referencia a las tendencias y tecnologías existentes en la actualidad que se deben considerar para hacer la selección de aquellas que se van a utilizar en el proyecto.
- En el **Capítulo 3** se describe la solución propuesta, realizándose la modelación del sistema para la Metodología ACIA. Se muestran los requisitos, casos de usos, diagramas de actividades y el prototipo de interfaz de usuario.

FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se analizará la concepción general del sistema SCIA, haciendo énfasis en el Módulo de Herramientas Integrales de Confiabilidad Operacional, principalmente en la herramienta de Análisis de Criticidad Integral de Activos. Además se realizará un estudio de diferentes metodologías y se detallarán algunos conceptos de Ingeniería de Software.

1.2 Concepción General del sistema

El Sistema de Confiabilidad Integral de Activos (SCIA) para los pueblos del Alba es una aplicación de software que persigue la mejora continua de sistemas y/o procesos e incorpora, en forma sistemática, herramientas para el manejo probabilístico de información, avanzadas herramientas de diagnóstico, metodologías basadas en confiabilidad y el uso de nuevas tecnologías, en la búsqueda de optimizar el costo del ciclo de vida de los activos.

El sistema está constituido por un grupo de disciplinas, métodos, procedimientos y herramientas para optimizar el Impacto Total de Costos, desempeño y exposición al riesgo en la Vida del Negocio asociados con confiabilidad, disponibilidad, mantenibilidad, eficiencia, longevidad y regulaciones de cumplimiento en seguridad y ambiente de PDVSA esto beneficiará a todas las áreas de negocio: operaciones, ingeniería, mantenimiento, finanzas, servicios, etc. **(MARTINEZ, 2008)**

1.2.1Módulo de Herramientas Integrales de Confiabilidad Operacional

El proyecto SCIA cuenta con el módulo: "Herramientas Integrales de Confiabilidad Operacional" visualizado en la figura 1, éste consiste en un compendio de metodologías organizadas en tres Etapas. Etapa I (Diagnóstico), Etapa II (Control) y Etapa III (Optimización). A continuación se describe con mayor detalle la organización de las tres etapas del primer módulo relacionado con las herramientas de confiabilidad integral.

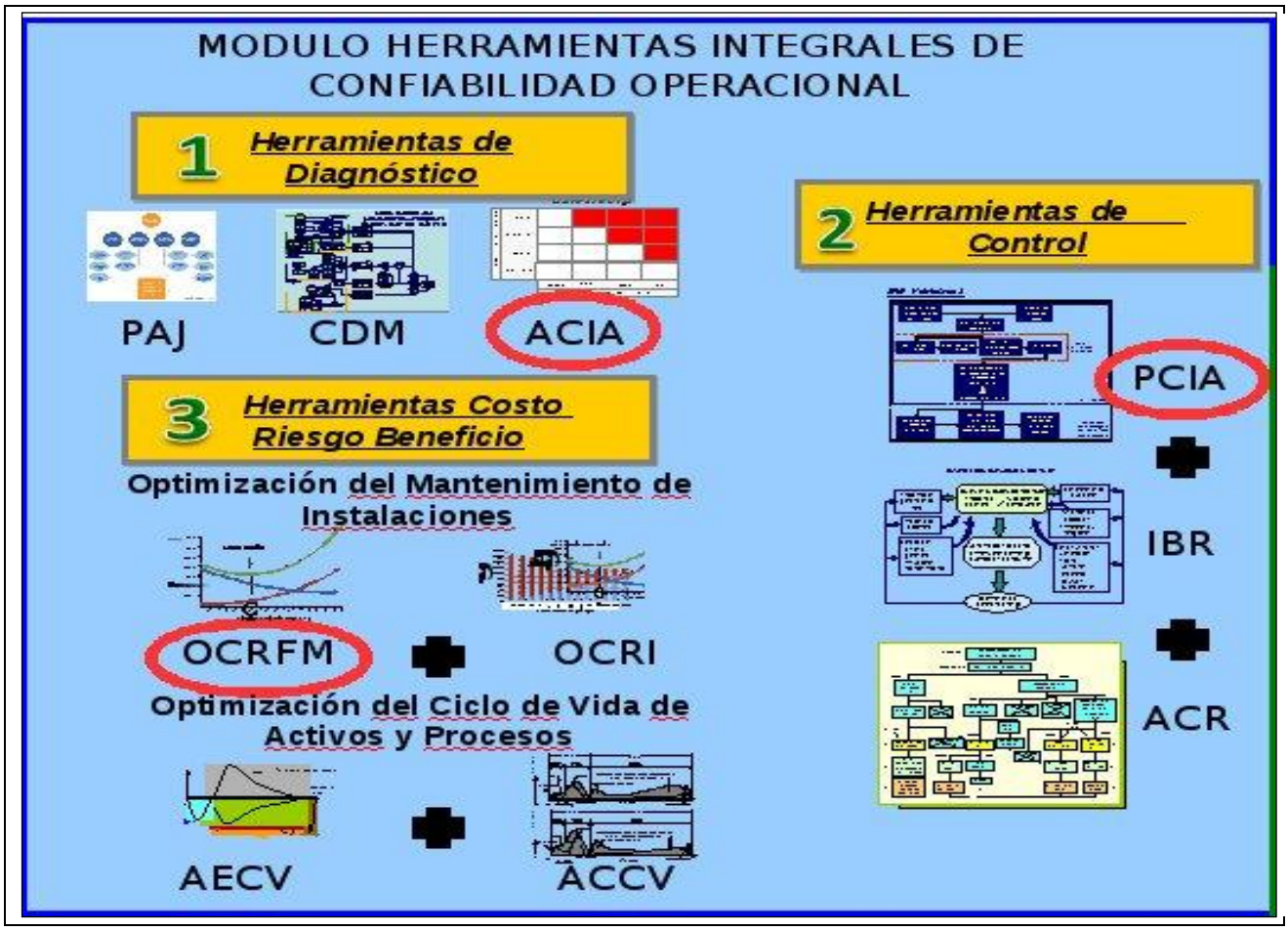


Figura 2 Módulo de Herramientas Integrales de Confiabilidad Operacional

1.2.1.1 Etapa de Diagnóstico

En la etapa de diagnóstico se contemplará la jerarquización de Unidades de Negocio, Áreas, Instalaciones, Sistemas, Grupo de Equipos y Equipos, a fin de mejorar las acciones correctivas que reducen los costos del ciclo de vida útil del proceso, mejorando la seguridad y la confiabilidad de los sistemas. El resultado del diagnóstico permitirá

establecer la ruta crítica a seguir para aplicar las metodologías requeridas. El método utilizado para la jerarquización es el más adaptable a las necesidades de los negocios a fin de homologar los procesos productivos.

Para la jerarquización de los Negocios e Instalaciones se utilizará la **Metodología Proceso Analítico de Jerarquización (PAJ).**

Para el caso de las evaluaciones de los plantas, sistemas y/o subsistemas se utilizará la **Metodología Análisis de Confiabilidad, Disponibilidad y Mantenibilidad (CDM)**, ya que esta se sustenta fundamentalmente en determinar el factor de servicio esperado o producción diferida de un proceso basado en la configuración, la disponibilidad de sus componentes y la filosofía de mantenimiento tomando en cuenta la fallas simultáneas o singulares que deriven en la interrupción de la operación a nivel de sistema.

Para la estimación de los niveles de criticidad de los Grupos de Equipos y Equipos se realizará mediante la **Metodología Análisis de Criticidad Integral de Activos (ACIA)** la cual tendrá como objetivo principal establecer una jerarquía o prioridades de instalaciones, sistemas, grupos de equipos y equipos, mediante la cuantificación del impacto global de su comportamiento en el negocio, denominada Riesgo, creando una estructura que facilita la toma de decisiones y el direccionamiento del esfuerzo y los recursos; constituye igualmente, una técnica de fácil manejo y comprensión, cuya mayor virtud es proveer una figura de merito cuyo valor es proporcional al riesgo. Esta metodología está basada en un diseño propio tomando como base los mejores criterios de los métodos de Ciliberti, Mantenimiento Basado en Criticidad, las normas API 580 y API 581 para equipos estáticos, y experiencias en el desarrollo de este tipo de estudios. **(Romero, 2009)**

- *Método de Ciliberti:* es un método semi-cuantitativo que determina las consecuencias potenciales asociadas a un equipo específico y la probabilidad de ocurrencia que ésta pueda tener, tanto en seguridad, higiene y ambiente como el impacto en el proceso. Este enfoque combina dos matrices de criticidad; una construida desde la óptica de seguridad de los procesos y otra construida desde la óptica del impacto en producción. Ambas matrices se integran en una matriz de

criticidad global. Es uno de los métodos más completos, ya que considera las probabilidades de falla y consecuencias en las áreas de SHA y Producción.

- *Mantenimiento Basado en Criticidad:* Esta metodología emplea un proceso de análisis de criticidad para priorizar el mantenimiento de activos en las industrias del Gas y Petróleo. Adicionalmente, analiza equitativamente el impacto en el proceso y seguridad para establecer la criticidad del activo. Este enfoque optimiza la eficacia de los programas de mantenimiento enfocado en los activos más importantes o críticos. Este método emplea algunos criterios particulares para determinar el nivel de criticidad, como por ejemplo, disminuye el nivel de criticidad a uno menor si el equipo cuenta con un respaldo o equipo en espera
- *Criticidad Basada en las Normativas API 580/581:* Esta metodología se basa en la Matriz de Riesgo de la Norma API 581 (Etapa I: Análisis Cualitativo de Riesgo) y solo aplica aquellos sistemas-equipos sometidos a los mecanismos de deterioro como corrosión. Esta metodología permite la ubicación del nivel de riesgo de los sistemas-equipos analizados en una matriz, que presenta cuatro niveles de clasificación de riesgo que son: riesgo bajo representado típicamente en color blanco o verde, riesgo medio presentado en amarillo, riesgo medio – alto graficado en naranja y alto riesgo mostrado en rojo.

1.2.1.2 Etapa de Control

En la etapa de control se tendrán en cuenta tres metodologías, la primera obtendrá las mejores políticas de mantenimiento para la familia de equipos Dinámicos, Eléctricos, Electrónicos e instrumentos utilizando el método **Política de Cuidado Integral de Activos (PCIA).** El segundo método se basa en establecer políticas de mantenimiento para la familia de equipos Estáticos utilizando el método de **Inspección Basa en Riesgo (IBR)** y el tercer y último método permitirá establecer acciones de mejoras a fin de mitigar el riesgo utilizando el método **Análisis Causa Raíz (ACR).**

1.2.1.3 Etapa de Optimización Costo Riesgo

En esta etapa se analiza la fase de **Optimización Costo – Riesgo (OCR)** como una vía altamente efectiva y eficiente para ejecutar estudios en un tiempo relativamente rápido con resultados de gran impacto en el negocio medular. Estos resultados permitirán optimizar el proceso de toma de decisiones de los diferentes procesos de gestión de la organización. Las herramientas de OCR ayudan a modelar y analizar los distintos escenarios que se puedan presentar, con el fin de establecer las frecuencias óptimas para realizar una actividad de mantenimiento menor, mantenimiento mayor y/o inspección; determinar el número óptimo de equipos instalados e inventario de partes y repuestos. Por otro lado, el análisis **Costo Riesgo Beneficio (OCRB)** es una herramienta para la evaluación, jerarquización y/o selección de portafolio de oportunidades, lo que a su vez permite determinar la viabilidad económica de los proyectos propuestos, una vez que se ha determinado su factibilidad técnica.

1.3 Sistema de Confiabilidad existentes en la actualidad.

1.3.1 APT-Maintenance

El APT Maintenance, es una de las herramientas de Optimización Costo Riesgo diseñada por el grupo de mantenimiento del proyecto MACRO el cual es utilizado para definir intervalos óptimos de mantenimiento, gerencia del deterioro, confiabilidad, desempeño y efectos del ciclo de vida. Esta herramienta está diseñada para calcular el mejor intervalo de mantenimiento preventivo o equipos de sustitución. Utiliza sofisticados análisis para optimizar la fiabilidad, el rendimiento, la eficiencia, los costos de mantenimiento y el impacto. Es una herramienta privativa, con un elevado costo.

1.4 Conceptos de la Ingeniería de Software.

1.4.1 Análisis del Sistema.

El análisis del Sistema es la etapa donde se deberán utilizar las herramientas para la recolección de datos como Cuestionarios, Entrevistas, Revisión de Documentos, Observación (a sistemas de mayor, menor o igual complejidad), y representar la información recabada en diagramas de procesos como Yourdon, Hipo, Tablas, Árboles, etc. **(SANGUINETTI)**

Un Análisis del Sistema se lleva a cabo teniendo en cuenta los siguientes objetivos:

- **Identificación de Necesidades**

  Es el primer paso del análisis del sistema. En este proceso el Analista se reúne con el cliente y/o usuario (un representante institucional, departamental o cliente particular), e identifican las metas globales, se analizan las perspectivas del cliente, sus necesidades y requisitos, sobre la planificación temporal y presupuestal, líneas de mercado y otros puntos que puedan ayudar a la identificación y desarrollo del proyecto.

  Este paso es llamado también por otros autores como "Análisis de Requisitos" y lo dividen en cinco partes:

  - Reconocimiento del problema.
  - Evaluación y Síntesis.
  - Modelado.
  - Especificación.
  - Revisión.

  Antes de su reunión con el analista, el cliente prepara un documento conceptual del proyecto, aunque es recomendable que este se elabore durante la

comunicación Cliente –Analista, ya que de hacerlo el cliente sólo de todas maneras tendría que ser modificado, durante la identificación de las necesidades.

- **Estudio de Viabilidad**

La viabilidad y el análisis de riesgos están relacionados de muchas maneras, si el riesgo del proyecto es alto, la viabilidad de producir software de calidad se reduce, sin embargo se deben tomar en cuenta cuatro áreas principales de interés:

1. **Viabilidad económica**

   Es la evaluación de los costos de desarrollo, comparados con los ingresos netos o beneficios obtenidos del producto o Sistema desarrollado.

2. **Viabilidad Técnica**

   Es el estudio de funciones, rendimiento y restricciones que puedan afectar la realización de un sistema aceptable.

3. **Viabilidad Legal**

   Es determinar cualquier posibilidad de infracción, violación o responsabilidad legal en que se podría incurrir al desarrollar el Sistema.

4. **Alternativas**

   Una evaluación de los enfoques alternativos del desarrollo del producto o Sistema.

Otros objetivos:

- Identificar las necesidades del cliente.
- Evaluar los conceptos que tiene el cliente del sistema para establecer su viabilidad.

- Realizar un análisis técnico y económico.
- Asignar funciones al hardware, software, personal, base de datos y otros elementos del Sistema.
- Establecer restricciones de presupuestos y planificación temporal.

1.4.2 El Analista de Sistema.

A raíz del surgimiento y explotación del uso de las nuevas tecnologías informáticas, la necesidad de organizar aplicaciones, modificar procedimientos, crear nuevos métodos para dirigir los negocios y buscar y aplicar metodologías adecuadas para la creación y desarrollo del producto en tiempo de mercado, nace el Analista de Sistemas.

En este epígrafe se harán referencias y citas a conceptos dados por especialistas a cerca del rol "Analista de Sistema".

- **Jacobson, Booch y Rumbauch, definen al Analista de Sistema como:**

  "el responsable del conjunto de requisitos que están modelados en los casos de usos, lo que incluye todos los requisitos funcionales y no funcionales que son casos de uso específicos. El analista de sistemas es responsable de delimitar el sistema, encontrando los actores y los casos de uso y asegurando que el modelo de casos de uso es completo y consistente. Para la consistencia, el analista de sistemas puede utilizar un glosario para conseguir un acuerdo en los términos comunes, nociones, y conceptos durante la captura de los requisitos. "**(RUMBAUGH, y otros, 2000)**

Aunque el analista de sistemas es responsable del modelo de casos de uso y de los actores que contiene, no es responsable de cada caso de uso en particular. Esto es una responsabilidad aparte, que pertenece al trabajador Especificador de casos de uso. El analista de sistemas es también el que dirige el modelado y coordina la captura de requisitos.

Hay un analista de sistemas para cada sistema. No obstante, en la práctica, este trabajador está respaldado por un equipo (en talleres o eventos similares) que incluye otras personas que también trabajan como analistas.

- **Otra definición es realizada por Ernesto Santos. (1980)**

  "…el analista de problemas en computación deberá conocer procedimientos para indagar sobre lo existente y para saber proponer un verdadero sistema racionalizado, pero también deberá conocer sobre modernos sistemas de información, base del diseño, sobre todo en computación… Estos últimos factores son los que justifican tal especialidad, porque realmente debieron existir los analistas de sistemas, aunque no existieran computadores siempre hay sistemas para organizar, que posiblemente no se difundieron, pues no existieron esos dos factores que hoy prevalecen: el computador y la información." **(TORRES)**

- **La definición de Analista de Sistemas de Senn (1992, p. 12), agrega:**

  "…Los analistas hacen mucho más que resolver problemas. Con frecuencia se solicita su ayuda para planificar la expansión de la organización…", es decir, el papel de los analistas sobrepasa los límites impuestos por la definición inicial, también cumplen el papel de asesores, ya sea en sistemas manuales o informatizados, o cualquier otro sistema donde la empresa tenga que invertir en información, después de todo esa es la razón de ser del analista." **(TORRES)**

1.4.3Metodologías de desarrollo de software

Las metodologías para el desarrollo de software describen el conjunto de fases, etapas, actividades y tareas asociadas a la producción de software (aplicación) de calidad. La finalidad del uso de metodologías es lograr el desarrollo de un software de calidad.

Rumbaugh realizó la siguiente definición:

> "Una metodología de ingeniería de software es un proceso para la producción organizada del software, empleando para ello una colección de técnicas predefinidas y convencionales en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso. Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo" **(RUMBAUGH et al. 2000)**

Existen diversas metodologías para el desarrollo de software donde cada una tiene ventajas y desventajas, y se utilizan en aquellas situaciones donde una metodología resulta más apropiada que otra.

En el contexto de este proyecto se evaluaron las siguientes:

- Metodología Scrum.
- Metodología XP.
- Metodología Orientada a objetos: Rational Unified Process (RUP).

1.4.3.1 Metodología Scrum.

SCRUM ha sido desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años.

Martin en su libro sobre la esencia de la metodología expresa:

> "Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos: El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración". **(SCHWABER, 1999).**

Por su parte Juan Palacio puntualiza:

> "Es una metodología de desarrollo muy simple, que requiere trabajo duro porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto. Es un modo de desarrollo de carácter adaptable más que predictivo, orientado a las personas más que a los procesos y emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones". **(PALACIO, 2006).**

La metodología propicia el desarrollo iterativo e incremental, y por ende asume las ventajas de este modelo de proceso de desarrollo. Conduce a que el cliente no tiene que esperar hasta que el sistema se entregue completo para usarlo, se pueden aclarar los requisitos, disminuye el riesgo de fracaso de todo el proyecto, puede aplicarse a cualquier proyecto sin importar la complejidad. Tiene como limitante la definición de los requisitos para cada Sprint. Al igual que XP no centra su desarrollo en la arquitectura.

1.4.3.2 Metodología XP

Programación Extrema, XP por sus siglas en inglés es la metodología ágil más difundida. Sobre ella Manuel Calero Solin expresa:

> "Germina como nueva disciplina para desarrollar software cerca de unos seis años atrás, pero ha causado una gran conmoción para los desarrolladores a nivel mundial. Su objetivo es muy simple, solo satisfacer al cliente e incrementar colosalmente el trabajo en equipo, donde los líderes del proyecto, clientes y programadores en si forman parte del grupo y están involucrados en todo el desarrollo del software". **(CALERO SOLIS, 2003).**

Así mismo Beck explica:

> "Está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida

> entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico". **(BECK, 2000).**

Las principales características de esta metodología es que se centra en resolver el problema lo más rápido posible. Cada miembro del equipo debe estar listo para enfrentar cualquier problema y la instrumentación de los comentarios "El cliente se introduce en el equipo de desarrollo", "Hago algo y lo pruebo", "Termino todo y después integro".

Los obstáculos más comunes surgidos en proyectos XP al decir de Larman:

> "son la "fantasía" de pretender que el cliente se quede en el sitio y la resistencia de muchos programadores a trabajar en pares". **(LARMAN, 2004).**

Craig Larman señala como factores negativos la ausencia de énfasis en la arquitectura durante las primeras iteraciones (no hay arquitectos en XP) y la consiguiente falta de métodos de diseño arquitectónico. Un estudio de casos de Bernhard Rumpe y Astrid Schröder sobre 45 proyectos reveló que las prácticas menos satisfactorias de XP han sido "la presencia del usuario en el lugar de ejecución y las metáforas, que el 40% de los encuestados no comprende para qué sirven o cómo usarlas". **(BERNHARD, 2001).**

1.4.3.3 Metodología Orientada a Objetos: RUP

La metodología Proceso Unificado de Desarrollo (RUP), de sus siglas en inglés, Rational Unified Process*,* fue desarrollada en 1998 por Grady Booch, Ivar Jacobson y James Rumbaugh, prominentes metodologistas en la industria de la tecnología y sistemas de información. RUP se caracteriza por ser: dirigido por casos de uso, centrado en la arquitectura e iterativo incremental.

Una metodología de desarrollo de software Orientada a Objeto (OO) consta de los siguientes elementos:

- Conceptos y diagramas (Modelo).
- Etapas y definición de entrega en cada una de ellas.
- Actividades y recomendaciones.

Las principales disciplinas de esta metodología son:

Disciplinas	Descripción
Modelado del negocio	Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización. Propone comprender los problemas de la organización e identificar posibles mejoras, evaluar el impacto del cambio introducido con la automatización, asegurar que todos los miembros tienen una misma visión de la organización.
Requerimientos	Esta disciplina se encarga de convertir las peticiones de los clientes en un conjunto de requerimientos de software, que definan el alcance y lo que debe hacer el producto a ser construido. Propone mantener un acuerdo a los interesados de lo que el sistema debe hacer, provee a los desarrolladores de una mejor visión de los requerimientos del sistema, define la frontera del sistema, crea las bases para la planificación y estimación.
Análisis y Diseño	Esta disciplina define como transformar artefactos resultantes del flujo de requerimientos de software en especificaciones del diseño del proyecto a desarrollar, en el se hace evolucionar la arquitectura de modo que se adapte al entorno del usuario final.
Implementación	Define como desarrollar, organizar realizar pruebas de unidad e integración de todos los componentes implementados basado en las especificaciones del sistema.
Pruebas	Este flujo se centra en encontrar y documentar los defectos en la calidad del software, validar y probar todos los supuestos hechos durante el diseño, verificar que el producto se desempeña como fue diseñado y cubre todos los requisitos pactados durante el

	flujo de “Captura de Requerimientos”.
Instalación	Se encarga de garantizar que el producto está disponible para los usuarios en su ambiente, se realizan actividades como empaque, instalación, asistencia a usuarios.
Administración del proyecto	Se centra los aspectos esenciales del desarrollo iterativo como son la planificación, control de riesgos, seguimiento del proceso de desarrollo de software y aplicación de métricas. Las restantes serán utilizadas durante el proceso de administración.
Administración de configuración y cambios	Se encarga de controlar y sincronizar la evolución de todos los productos generados, introduce el uso de herramientas que faciliten el trabajo colaborativo en el equipo.
Ambiente	Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Tabla 1: Principales etapas de RUP

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo como se muestra en la siguiente figura:

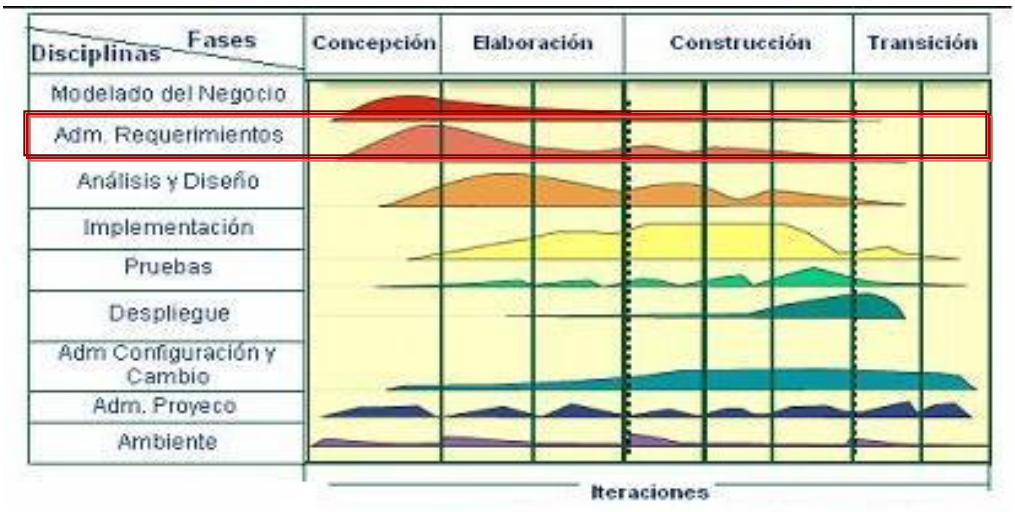


Figura 3 Metodología de desarrollo de software: RUP

Cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

- **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requisitos (funcionales y no funcionales) identificados de acuerdo al alcance definido.

- **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene uno o varios entregables del producto que han pasado las pruebas. Se ponen estos entregables a consideración de un subconjunto de usuarios.

- **Transición:** El entregable ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

1.5 Conclusiones del Capítulo

En este capítulo se ha realizado un análisis general del sistema SCIA y de las herramientas que componen el módulo de herramientas integrales de confiabilidad operacional.

También se mostraron conceptos aplicables en la Ingeniería de Software que se llevó a cabo en la herramienta de Análisis de Criticidad Integral , entre éstos se tuvo en cuenta algunas metodologías de desarrollo de software, concluyendo con la selección de RUP como la más adecuada dada las características del grupo de trabajo y la complejidad del sistema.

TENDENCIAS Y TECNOLOGÍAS ACTUALES

2.1 Introducción

En el desarrollo del capítulo, se hace un análisis de los métodos, técnicas y herramientas empleadas para la captura y especificación de requisitos así como de las tecnologías que se emplean en el desarrollo de los prototipos.

2.2 Sistema Operativo GNU/Linux.

Este proyecto será desarrollado a partir de las herramientas que brinda el software libre. Buscando la independencia tecnológica que este brinda al posibilitar la libertad de uso y distribución de los programas sin incurrir en litigios de licenciamiento o asuntos legales.

El Sistema operativo GNU/LINUX es una implementación de libre distribución UNIX para computadoras personales (PC), servidores, y estaciones de trabajo. Como sistema operativo, GNU/Linux es muy eficiente y tiene un excelente diseño. Es multitarea, multiusuario, multiplataforma y multiprocesador.

2.2.1Distribución de Linux: Debian GNU/Linux

El Proyecto Debian es una asociación de personas que han hecho causa común para crear un Sistema Operativo (SO) libre, bajo la licencia Licencia Pública General (GPL), de sus siglas en ingles General Public Licence. Basado en el conocido y distribuido núcleo de Linux la distribución es llamada Debian.

Una gran parte de las herramientas básicas que completan el sistema operativo, vienen del proyecto GNU; de ahí el nombre: GNU/Linux.

Debian es la única distribución importante de GNU/Linux mantenida solamente por voluntarios, es decir, sin un enfoque comercial, esto tiene ventajas y desventajas.

En primer lugar, las personas que se dedican a Debian tienen una alta motivación en participar en la

misma, y se actualiza la distribución diariamente, apareciendo paquetes nuevos de software constantemente. Al mismo tiempo, existe un compromiso de calidad.

En segundo lugar, dada su actitud abierta a la participación de todos, en el mismo espíritu original de Linux, constantemente hay personas que se unen a Debian para participar aportando su granito de arena, no solamente haciendo paquetes de programas, sino colaborando en el Servidor de Web, traduciendo documentación de Debian, documentando fallos, o ayudando a los usuarios a través de las listas de correo que mantiene la comunidad.

Como desventajas tiene un mayor componente técnico que otras distribuciones. También, es posible que ciertos paquetes no estén tan actualizados como debieran, quizás porque sus desarrolladores han dejado de actualizarlos y nadie se ha hecho cargo. Sin embargo esto es algo que todos los desarrolladores tratan de evitar y, aunque cada desarrollador mantiene sus paquetes, no es raro que otro desarrollador (incluso un usuario) envíe una nueva versión del paquete para arreglar un problema o actualizarlo. **(DEBIAN 2009)**.

Se decidió usar la distribución Debian en su versión estable 5.0 bajo el nombre "Leny". Esta versión incluye una gran cantidad de paquetes de programas actualizados y se encuentra disponible para ser instalada en 12 arquitecturas diferentes. A diferencia de otras distribuciones tiene un magnífico soporte de estabilidad en las aplicaciones (no requieren ser compiladas en la máquina que las esté usando).

2.3 Ingeniería de Requisitos

La definición de las necesidades de un sistema es un proceso complejo, pues en él hay que identificar los requisitos que se deben cumplir para satisfacer las necesidades de los usuarios finales y de los clientes.

El proceso de especificación de requisitos se puede dividir en tres grandes actividades:

1- Captura de requisitos.

2- Definición de requisitos.

3- Validación de requisitos.

A continuación se explicarán brevemente algunas técnicas clásicas para realizar las actividades expuestas anteriormente.

2.3.1Captura de requisitos

La captura de requisitos es la actividad mediante la cual el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema.

El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas, y depende mucho de las personas que participen en él. Por la complejidad que todo esto puede implicar, la ingeniería de requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa.

Las técnicas siguientes, de forma clásica han sido utilizadas para esta actividad en el proceso de desarrollo de todo tipo de software.

1. **Entrevistas:** resultan una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Existen muchos tipos de entrevistas y son muchos los autores que han trabajado en definir su estructura y dar guías para su correcta realización.

   Básicamente, la estructura de la entrevista abarca tres pasos: identificación de los entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados (protocolo de la entrevista).

   A pesar de que las entrevistas son esenciales en el proceso de la captura requisitos y con su aplicación el equipo de desarrollo puede obtener una amplia visión del trabajo y las necesidades del usuario, es necesario destacar que no es una técnica sencilla de aplicar. Requiere que el entrevistador sea experimentado y tenga capacidad para elegir bien a los entrevistados y obtener de ellos toda la información posible en un período de tiempo siempre limitado. Aquí desempeña un papel fundamental la preparación de la entrevista.

2. **JAD (Joint Application Development/ Desarrollo conjunto de aplicaciones):** esta técnica resulta una alternativa a las entrevistas. Es una práctica de grupo que se desarrolla durante varios días y en la que participan analistas, usuarios, administradores del sistema y clientes. Está basada en cuatro principios fundamentales: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y una filosofía de documentación "lo que ve es lo que obtiene" (WYSIWYG, What You See Is What You Get) es decir, durante la aplicación de la técnica se trabajará sobre lo que se generará. Tras una fase de preparación del JAD al caso concreto, el equipo de trabajo se reúne en varias sesiones. En cada una de ellas se establecen los requisitos de alto nivel a trabajar, el ámbito del problema y la documentación.

   Durante la sesión se discute en grupo sobre estos temas, llegándose a una serie de conclusiones que se documentan. En cada sesión se van concretando más las necesidades del sistema.

   Esta técnica presenta una serie de ventajas frente a las entrevistas tradicionales, ya que ahorra tiempo al evitar que las opiniones de los clientes se tengan que contrastar por separado, pero requiere un grupo de participantes bien integrados y organizados.

3. **Brainstorming (Tormenta de ideas):** es también una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. El grupo de personas que participa en estas reuniones no debe ser muy numeroso (máximo diez personas), una de ellas debe asumir el rol de moderador de la sesión, pero sin carácter de controlador.

   Como técnica de captura de requisitos es sencilla de usar y de aplicar, contrariamente al JAD, puesto que no requiere tanto trabajo en grupo como éste. Además suele ofrecer una visión general de las necesidades del sistema, pero normalmente no sirve para obtener detalles concretos del mismo, por lo que suele aplicarse en los primeros encuentros.

4. **Concept Mapping (Mapas Conceptuales):** son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos de relaciones se desarrollan con el usuario y sirven para aclarar los conceptos

relacionados con el sistema a desarrollar. Son muy usados dentro de la ingeniería de requisitos, pues son fáciles de entender por el usuario, más aún si el equipo de desarrollo hace el esfuerzo de elaborarlo en el lenguaje de éste. Sin embargo, deben ser usados con cautela porque en algunos casos pueden ser muy sugestivos y pueden llegar a ser ambiguos en casos complejos, si no se acompaña de una descripción textual.

5. **Sketches y Storyboards (Esbozos y Guiones Gráficos):** Está técnica es frecuentemente usada por los diseñadores gráficos de aplicaciones en el entorno Web. La misma consiste en representar sobre papel en forma muy esquemática las diferentes interfaces al usuario (esbozos). Estos esbozos pueden ser agrupados y unidos por enlaces dando idea de la estructura de navegación (guiones gráficos).
6. **Casos de Uso:** Aunque inicialmente se desarrollaron como técnica para la definición de requisitos, algunos autores proponen casos de uso como técnica para la captura de requisitos. Los casos de uso permiten mostrar el contorno (actores) y el alcance (requisitos funcionales expresados como casos de uso) de un sistema. Un caso de uso describe la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. Los actores son elementos externos (personas, otros sistemas) que interactúan con el sistema como si de una caja negra se tratase. Un actor puede participar en varios casos de uso y un caso de uso puede interactuar con varios actores.

   La ventaja esencial de los casos de uso es que resultan muy fáciles de entender para el usuario o cliente, sin embargo carecen de la precisión necesaria si no se acompañan con una información textual o detallada con otra técnica como pueden ser los diagramas de actividades.
7. **Cuestionarios y Checklists (Listas de Chequeo)**: Esta técnica requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario (listas de chequeo). Este cuestionario será cumplimentado por el grupo de personas entrevistadas o simplemente para recoger

información en forma independiente de una entrevista.

8. **Comparación de terminología:** Uno de los problemas que surge durante la licitación de requisitos es que usuarios y expertos no llegan a entenderse debido a problemas de terminología. Esta técnica es utilizada en forma complementaria a otras técnicas para obtener consenso respecto de la terminología a ser usada en el proyecto de desarrollo. Para ello es necesario identificar el uso de términos diferentes para los mismos conceptos (correspondencia), una misma terminología para diferentes conceptos (conflictos) o cuando no hay concordancia exacta ni en el vocabulario ni en los conceptos (contraste). **(KOCH, 2009)**

9. **Ingeniería Inversa:** su objetivo es obtener información técnica a partir de un producto accesible al público, con el fin de determinar de qué está hecho, qué lo hace funcionar y cómo fue fabricado.

   Es denominada ingeniería inversa porque avanza en dirección opuesta a las tareas habituales de ingeniería, que consisten en utilizar datos técnicos para elaborar un producto determinado. En general si el producto u otro material que fue sometido a la ingeniería inversa fueron obtenidos en forma apropiada, entonces el proceso es legítimo y legal. **(INGENIERA INVERSA)**

   Aplicar ingeniería inversa a algo supone profundizar en el estudio de su funcionamiento, hasta el punto de de llegar a entender, modificar, y mejorar dicho modo de funcionamiento.

   Áreas de la Ingeniería inversa:

   - ✓ *Re documentación:* es la creación o revisión de una representación equivalente semánticamente dentro del mismo nivel de abstracción relativo.
   - ✓ *Recuperación de diseño:* es un subconjunto de la ingeniería inversa, en el cual, aparte de las observaciones del sistema, se añaden conocimientos sobre su dominio de aplicación, información externa, y procesos deductivos con el objeto de identificar abstracciones significativas a un mayor nivel.
   - ✓ *Rediseño*: consiste en consolidar y modificar los modelos obtenidos, añadiendo

nuevas funciones requeridas por los usuarios.

✓ *Reingeniería de Software:* es el examen y alteración de un sistema para reconstruirlo de una nueva forma y la subsiguiente implementación de esta nueva forma. **(ALARCOS, 2009).**

Para definir los requisitos funcionales de la Herramienta de Análisis de Criticidad Integral de Activos, se usaron algunas técnicas de captura de requisitos favorables a las necesidades del equipo y que posibilitaran un dominio previo de los conceptos del negocio, permitiendo una familiarización con los mismos en un corto período de tiempo.

Una de las técnicas empleadas fue la tormenta de ideas con la participación de especialistas de empresas y universidades del hermano país de Venezuela, experimentados en esta área de trabajo. Esto permitió adquirir una visión inicial, común y general del nuevo sistema.

Se realizaron entrevistas a expertos y especialistas de PDVS. Éstas ayudaron a obtener información rápida y confiable de las necesidades. En el **Anexo 1 Guía de entrevistas realizadas para el levantamiento de requisitos** se muestra una guía de entrevistas realizadas para la captura de requisitos.

Como el dominio del sistema a desarrollar se enmarca en la automatización industrial, el equipo de analistas encargados de ejecutar las tareas relacionadas con el flujo de requisitos, tuvo que hacer un estudio profundo en cuanto a la Ingeniería de Confiabilidad, guiándose de la experiencia de los asesores en la especialidad y consultando documentación acreditada respecto al tema.

2.3.2 Definición de requisitos

Para la actividad de definición de requisitos en el proceso de ingeniería de requisitos existe un gran número de técnicas propuestas. Se describen a continuación las más relevantes:

- **Lenguaje natural**: Resulta una técnica muy ambigua para la definición de los requisitos. Consiste en definir los requisitos en lenguaje natural sin usar reglas para ello. A pesar de que son muchos los trabajos que critican su uso, es cierto que a nivel práctico se sigue utilizando.

- **Glosario y ontologías**: La diversidad de personas que forman parte de un proyecto de software hace que sea necesario establecer un marco de terminología común. Por esta razón son muchas las propuestas que abogan por desarrollar un glosario de términos en el que se recogen y definen los conceptos más relevantes y críticos para el sistema. En esta línea se encuentra también el uso de ontologías, en las que no sólo aparecen los términos, sino también las relaciones entre ellos.
- **Plantillas o patrones**: Esta técnica, recomendada por varios autores, tiene por objetivo el describir los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la ambigüedad del lenguaje natural al estructurar la información; cuanto más estructurada sea ésta, menos ambigüedad ofrece. Sin embargo, si el nivel de detalle elegido es demasiado estructurado, el trabajo de rellenar las plantillas y mantenerlas, puede ser demasiado tedioso.
- **Escenarios:** La técnica de los escenarios consiste en describir las características del sistema a desarrollar mediante una secuencia de pasos. La representación del escenario puede variar dependiendo del autor. Esta representación puede ser casi textual o ir encaminada hacia una representación gráfica en forma de diagramas de flujo. El análisis de los escenarios, hechos de una forma u otra, pueden ofrecer información importante sobre las necesidades funcionales del sistema.
- **Casos de uso**: Como técnica de definición de requisitos han sido aceptados los casos de uso, estos actualmente se ha propuesto como técnica básica del proceso RUP. Sin embargo, son varios los autores que defienden que pueden resultar ambiguos a la hora de definir los requisitos, por lo que hay propuestas que los acompañan de descripciones basadas en plantillas o de diccionarios de datos que eliminen su ambigüedad.
- **Lenguajes Formales:** Otro grupo de técnicas que merece la pena resaltar como extremo opuesto al lenguaje natural, es la utilización de lenguajes formales para describir los requisitos de un sistema. Las especificaciones algebraicas como ejemplo de técnicas de descripción

formal, han sido aplicadas en el mundo de la ingeniería de requisitos desde hace años. Sin embargo, resultan muy complejas en su utilización y para ser entendidas por el cliente. El mayor inconveniente es que no favorecen la comunicación entre cliente y analista. Por el contrario, es la representación menos ambigua de los requisitos y la que más se presta a técnicas de verificación automatizadas. **(KOCH, 2009)**

- **IEEE 1233:** Es una guía que nos da la pauta para el desarrollo de un conjunto de requerimientos que satisfacen una necesidad específica. Incluye la identificación, organización, presentación y modificación de los requerimientos. Trata las condiciones necesarias para incorporar conceptos operacionales, restricciones de diseño, y requerimientos de la configuración del diseño en la especificación. Además, trata las características y cualidades necesarias de los requerimientos individuales y del conjunto de todos los requerimientos. **(IEEE 1998)**

Mediante el análisis de las técnicas existentes para definir los requisitos, se decidió utilizar el glosario, las plantillas, los escenarios, casos de uso y la IEEE 1233.

El Glosario, para conceptualizar las terminologías y lograr un entendimiento entre equipo de desarrollo y el cliente.

Las plantillas, para estructurar de una forma estándar la información, previendo que el nivel de detalles no fuera tan estructurado evitando así las ambigüedades. Las plantillas seleccionadas fueron las que propone la metodología RUP.

Los escenarios, para la modelación del sistema, utilizando los diagramas de actividad para una mejor comprensión de los casos de uso.

Los casos de uso, se tomaron por ser considerados una técnica básica del proceso de RUP utilizado en el desarrollo del sistema. Éstos fueron descritos en plantillas estándares de RUP.

La IEEE 1233 como guía para el desarrollo de Especificaciones de Requisitos

2.3.3Validación de requisitos

Los requisitos una vez definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos en la definición de requisitos con el usuario para detectar errores o inconsistencias.

Aún así, existen algunas técnicas que pueden aplicarse para ello:

- **Reviews (Revisión):** Está técnica consiste en la lectura y corrección completa de la documentación o modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida. Más difícil es verificar consistencia de la documentación o información faltante.
- **Auditorías**: La revisión de la documentación con esta técnica consiste en un chequeo de los resultados contra una lista de chequeo predefinida o definida a comienzos del proceso, es decir sólo una muestra es revisada.
- **Matrices de trazabilidad**: Esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos.
- **Prototipos**: Algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final. **(KOCH 2009)**.

Para la validación de los requisitos se emplearon las técnicas: revisión y prototipos.

Posterior a la obtención de varios requisitos y del modelado de ellos se sostuvieron revisiones para cerciorarse que la interpretación por parte del analista fue correcta y que no faltó información.

Para obtener una idea de la interfaz de usuario se desarrolló un prototipo con parte de las

funcionalidades críticas del sistema**.**

2.4 Herramientas para el Modelado UML

El Lenguaje Unificado de Modelado (UML) por sus siglas en ingles Unified Modeling Language se define como un "lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software" **(Booch et al, 97)**. Es un lenguaje notacional (que, entre otras cosas, incluye el significado de sus notaciones) destinado a los sistemas de modelado que utilizan conceptos orientados a objetos.

El UML es un estándar para construir modelos orientados a objetos. Nació en 1994 por iniciativa de Grady Booch y Jim Rumbaugh para combinar sus dos famosos métodos: el de Booch y el OMT (Object Modeling Technique, Técnica de Modelado de Objetos). Más tarde se les unió Ivar Jacobson, creador del método OOSE (Object-Oriented Software Engineering, Ingeniería de Software Orientada a Objetos). En respuesta a una petición del OMG (Object Management Group, Grupo de Administración de Objetos) para definir un lenguaje y una notación estándar del lenguaje de construcción de modelos, en 1997 propusieron el UML como candidato **(Larman., 2004)**

2.4.1 Umbrello

Umbrello UML Modeller es una herramienta de diagramas que ayuda en el proceso del desarrollo de software, facilita la creación de un producto de alta calidad, especialmente durante fases de análisis y diseño del proyecto.

Es una herramienta libre para crear y editar diagramas UML, que ayuda en el proceso del desarrollo de software. Fue desarrollada por Paul Hensgen, y está diseñado principalmente para KDE, aunque funciona en otros entornos de escritorio.

Umbrello maneja gran parte de los diagramas estándar UML pudiendo crearlos, además de manualmente, importándolos a partir de código en C++, Java, Python, IDL, Pascal/Delphin, Ada o también Perl (haciendo uso de una aplicación externa). Así mismo, permite crear un diagrama y

generar el código automáticamente en los lenguajes antes citados, entre otros. El formato de fichero que utiliza está basado en XMI.

2.4.2 Net Beans

NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE) desarrollado usando la Plataforma NetBeans.

Es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

Dicho sistema de software permite explorar y evaluar la funcionalidad de UML en el IDE donde se pueden construir 8 diagramas UML (Diagrama de Actividades, Diagrama de Clases, Diagrama de colaboración, Diagrama de Componentes, Diagrama de Implementación, Diagrama de secuencias, Diagrama de Estados y Diagrama de Casos de Usos), además se cuenta con un generador de código mediante los diagramas e ingeniería inversa.

2.5 Plataformas para el Diseño de Prototipos No Funcionales

2.5.1 Gazpacho

Gazpacho es una herramienta para el desarrollo de interfaces, es un nuevo generador de Interfaz Gráfica de Usuarios (GUI) para la librería de controles GTK diseñado para ser muy fácil de usar. Está fuertemente inspirado en el proyecto Glade pero se ha escrito desde cero usando el lenguaje de programación Python. Uno de los objetivos del proyecto es que los ficheros .glade que Gazpacho genere deben ser completamente compatibles con la librería libglade. La licencia de Gazpacho es la LGPL por lo que se puede usar y distribuir libremente. Gazpacho es muy modular, así que facilita la integración con IDE existentes.

2.5.2 Glade

Glade es un programa que facilita la tarea de diseño y desarrollo de interfaces gráficas de usuario, permite, diseñar estas interfaces gráficas de forma completamente visual. Los pasos a seguir en la construcción de una aplicación son los siguientes:

- Diseñar el aspecto gráfico de nuestra aplicación con Glade
- Generar el esqueleto del código fuente usando Glade (solamente en caso que usemos C o C++
- Editar el código fuente manualmente (a través de cualquier editor, como vi o xemacs)
- Compilar el proyecto

Glade almacena un conjunto de ficheros cada vez que se crea un proyecto nuevo, pero sin duda el más importante es el fichero con extensión .glade que se almacena en el directorio raíz del proyecto. Este fichero, escrito en XML representa toda la información relativa al aspecto gráfico de la aplicación.

2.5.3 QT

El Qt Designer es un creador de interfaces gráficas para la librería QT, provee un gran nivel de portabilidad tanto a nivel de lenguajes de programación soportados (aunque sea mediante ligas) como plataformas sobre las que puede correr (y para las que puede diseñar)..

Es una herramienta para el desarrollo de formularios y presentaciones gráficas para las aplicaciones. Permite acelerar el desarrollo de interfaces de alto rendimiento, a la vez que proporciona una forma fácil de diseñar interfaces gráficas de usuario avanzados generando el código fuente para las mismas, lo que permite al desarrollador ajustarlo a sus necesidades.

Este generador de interfaces fue creado inicialmente por la empresa TROLLTECH para trabajar en varias distribuciones Linux. No obstante, actualmente puede instalarse en otras plataformas como Windows y Mac OsX.

El QT Designer utiliza como base la librería gráfica de QT, que ha sido transportada a diversas plataformas, lo que permite que el código generado por el QT Designer pueda ser utilizado en diversas

plataformas. Además, el QT funciona sólo o asociándose a algunos entornos de desarrollo integrado como Visual Studio .Net o Eclipse. Esta herramienta provee características muy poderosas como la previsualización de la interfaz, soporte para widgets y un editor de propiedades bastante poderoso.

2.6 Conclusiones

Una vez analizado las características más importantes de las tecnologías existentes y las tendencias actuales para el desarrollo de software se decide que el sistema que se propone se desarrolle haciendo uso de las herramientas de software libre, el cual dará la oportunidad de implementar el Sistema de Confiabilidad Integral (SCIA) para los Pueblos de Alba a bajos costos y con soberanía tecnológica, permitiendo desarrollar e investigar para lograr nuevos productos sin limitaciones.

La herramienta CASE seleccionada para el proceso de modelado acorde al lenguaje UML 2.0, es NetBeans 6.1. Para el desarrollo de los prototipos de interfaz se utilizó el QT 4 Designer. Entre las técnicas escogidas para la captura de requisitos se encuentran el uso de las entrevistas, tormenta de Ideas, ingeniería inversa y comparación de terminología. Para la definición de los requisitos se decidió utilizar el glosario, las plantillas, los escenarios y casos de uso, posibilitando su validación a través de las técnicas: revisión y prototipos.

DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción

En este capítulo se realiza el proceso de modelación del sistema para la Metodología ACIA del producto SCIA. Para ello se identifican los actores, se describen los requisitos funcionales y no funcionales del SCIA, los casos de usos correspondientes para el funcionamiento y se visualizan los diagramas de actividades y prototipos no funcionales de la Metodología ACIA.

3.2 Modelación del sistema

En el presente epígrafe se exponen una selección del conjunto de requisitos funcionales y no funcionales de la Metodología ACIA, para un análisis detallado se puede consultar el documento "Especificación de Requisitos de Software" donde se representan todos los requisitos y especificaciones de casos de uso del sistema. En las especificaciones se incluyen las siguientes categorías: suplementarias, usabilidad, confiabilidad, desempeño, diseño, implementación, interfaces; así como los requisitos de documentación de usuario, requisitos legales y de ambiente.

3.2.1 Actores del Sistema

Los actores representan personas o sistemas externos que interactúan con el sistema; actualmente para la Metodología ACIA se consideran un solo actor:

Ingeniero de Confiabilidad: es el encargado de realizar la Metodología ACIA.

3.2.2 Requisitos Funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Para la Metodología ACIA que se está desarrollando, ellos son:

Estimar el índice de riesgo según la norma NFPA 704: El sistema brindará la posibilidad de calcular el índice de riesgo según la norma NFPA 704 (IR_{NFPA}) del equipo principal tomando en cuenta las consecuencias por inflamabilidad, reactividad y toxicidad del fluido manejado.

Estimar el índice de riesgo según las condiciones operacionales: El sistema a partir de los parámetros operacionales de velocidad rotacional, temperatura, presión de operación y daño ambiente de acuerdo al equipo principal que se está analizando determinará el índice de riesgo según las condiciones operacionales.

Determinar la estimación de las Consecuencias Seguridad Higiene y Ambiente (SHA): El sistema debe ser capaz de poder estimar las consecuencias SHA según los valores de los índices de riesgos.

Determinar si existen factores de mitigación de riesgos: el sistema debe ser capaz de mostrar diversas opciones de Detección y Aislamiento al usuario con el objetivo de disminuir el nivel de consecuencias por SHA del equipo principal bajo análisis.

Estimar las consecuencias por factor de reducción SHA: El Sistema debe permitir estimar las consecuencias por factor de reducción SHA según los criterios de Detección y Aislamiento.

Estimar la Probabilidad de Ocurrencia de un Evento no deseado por SHA: El sistema mostrará al usuario las Categorías de probabilidad de ocurrencia de un evento no deseado por SHA, para que según el comportamiento observado en el equipo bajo análisis y/o según su experiencia previa en análisis similares seleccione la que corresponde.

Determinar el nivel de criticidad por SHA del Equipo Principal: El sistema debe ser capaz de obtener el nivel de criticidad por SHA al equipo principal.

Determinar el Porcentaje de Pérdida de Producción: El sistema mostrará los diferentes niveles de las categorías de consecuencias de la Metodología de Ciliberti con el objetivo de poder seleccionar el nivel de consecuencia por impacto en procesos.

Evaluar si existen factores de mitigación de riesgo: El sistema debe permitirle al usuario decidir la existencia o no de factores de mitigación de riesgo (equipos de respaldo) que permitan disminuir el nivel de consecuencia de un equipo principal.

Estimar la Probabilidad de Ocurrencia de un Evento no deseado por Impacto en Procesos: El sistema debe ser capaz de mostrar las opciones para determinar la probabilidad de falla por procesos del equipo principal utilizando rangos de clasificación de las probabilidades por impacto en procesos teniendo como referencia el Tiempo Promedio Entre Falla (TPEF) para equipos reparables o el Tiempo Promedio Para Fallar (TPPF) para equipos no reparables.

Determinar el Nivel de criticidad por proceso: El sistema debe ser capaz de mostrar el nivel de criticidad por proceso obtenido durante todo el proceso.

Mostrar Nivel de Criticidad Final para Equipos Principales de un Grupo de Equipos: el sistema debe mostrar mediante la Matriz de Criticidad de Ciliberti, el Nivel de Criticidad Final para Equipos Principales de un Grupo de Equipos a partir del Nivel de Consecuencias por Factor de Reducción de Procesos (Co_{FRP}) y la Probabilidad de ocurrencia de un evento no deseado (Pf_P).

Asignar nivel de criticidad por SHA y por Procesos a equipos secundarios: El sistema debe ser capaz de asignarle el Nivel de Criticidad por SHA y por Procesos a equipos secundarios que posean clasificación taxonómica en el RIU y que no se encuentra asociados a los sistemas de protección y resguardo de ISGEE.

Asignar nivel de criticidad final a equipos secundarios: El sistema mediante la Matriz de Criticidad de Ciliberti deberá asignarle el nivel de criticidad final al equipo secundario bajo análisis.

Extraer información del RIU: El sistema debe ser capaz de poder extraer del RIU información técnica y operacional de los grupos de equipos y equipos.

Registrar Grupos de Equipos o Equipos en el RIU: El sistema permitirá a los usuarios debidamente autorizados, mediante una Interfaz con el RIU, la inclusión de los datos técnicos y operacionales de los equipos bajo análisis.

Guardar Información en RIU: El sistema debe tener una interfaz con el RIU que permita almacenar información.

Registrar participantes: El Sistema debe ser capaz registrar los datos de las personas que participarán en la aplicación de las diferentes metodologías. Registrando de cada participante:

1. **Nombre y Apellidos**
2. **Cédula**
3. **Cargo**
4. **Teléfono**
5. **Correo**

3.2.3 Requisitos no Funcionales.

En este epígrafe se especifican los requisitos no funcionales que se tuvieron en cuenta para el diseño y desarrollo de los prototipos. Los requisitos no funcionales son propiedades o cualidades que el producto debe cumplir. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

- **Confiabilidad**

ESCO1 Generar cálculos alineados a la especificación técnica de la metodología: Los Cálculos para la Implementación de las metodologías deben estar en correspondencia con lo establecido en los documentos de especificaciones técnicas para garantizar que los resultados sean los esperados.

- **Desempeño**

ESCO1 Tiempo mínimo de recuperación ante fallos: El tiempo mínimo para la recuperación ante aciagos fallos debe ser considerablemente corto para no perder la operatividad del sistema durante mucho tiempo.

ESDE2 Garantizar procesos y transacción masiva de datos: los procesos y transacción masiva de datos deben estar definidos hacia y desde todas las estaciones de trabajo y sistemas asociados, sin efectos negativos sobre el rendimiento del sistema.

ESDE3 Disponibilidad de los Datos en el RIU: El RIU debe permitir ser accedido, modificado, actualizado independientemente del número de datos que maneje.

ESDE4 Tiempos requerido para realizar consultas al RIU de acuerdo a la metodología aplicada: Los tiempos de consulta al RIU deben ajustarse de acuerdo al criterio y/o metodología aplicada y deben responder a las funciones requeridas por los usuarios en lapsos de espera que permitan cumplir con la funcionalidad solicitada.

ESDE5 Tiempos de cálculos operacionales: La respuestas de los cálculos de las diferentes aplicaciones deben estar dadas en menos de 3 segundos.

- **Implementación**

ESIM1 Empleo del mismo conjunto de herramientas en el desarrollo del sistema: Se debe emplear el mismo conjunto de herramientas para el desarrollo del sistema, así como para futuros desarrollos. De manera tal que garantice la escalabilidad y compatibilidad entre los diferentes desarrollos.

ESIM2 Rendimiento y alta disponibilidad: El sistema debe poseer características de rendimiento y alta disponibilidad de forma que se garantice su operatividad de forma segura, efectiva y confiable.

ESIM3 El sistema debe ser multiplataforma: El sistema debe ejecutarse en diversas plataformas de hardware y software.

ESIM4 Licencia bajo código abierto: El sistema debe cumplir con los lineamientos necesarios para la producción de software libre y la comunidad de desarrollo y soporte, manteniendo la confidencialidad del negocio y las operaciones de PDVSA.

ESIM5 Manejo de internacionalización: El sistema debe proveer soporte para el manejo de múltiples idiomas.

3.3 Diagrama de caso de uso

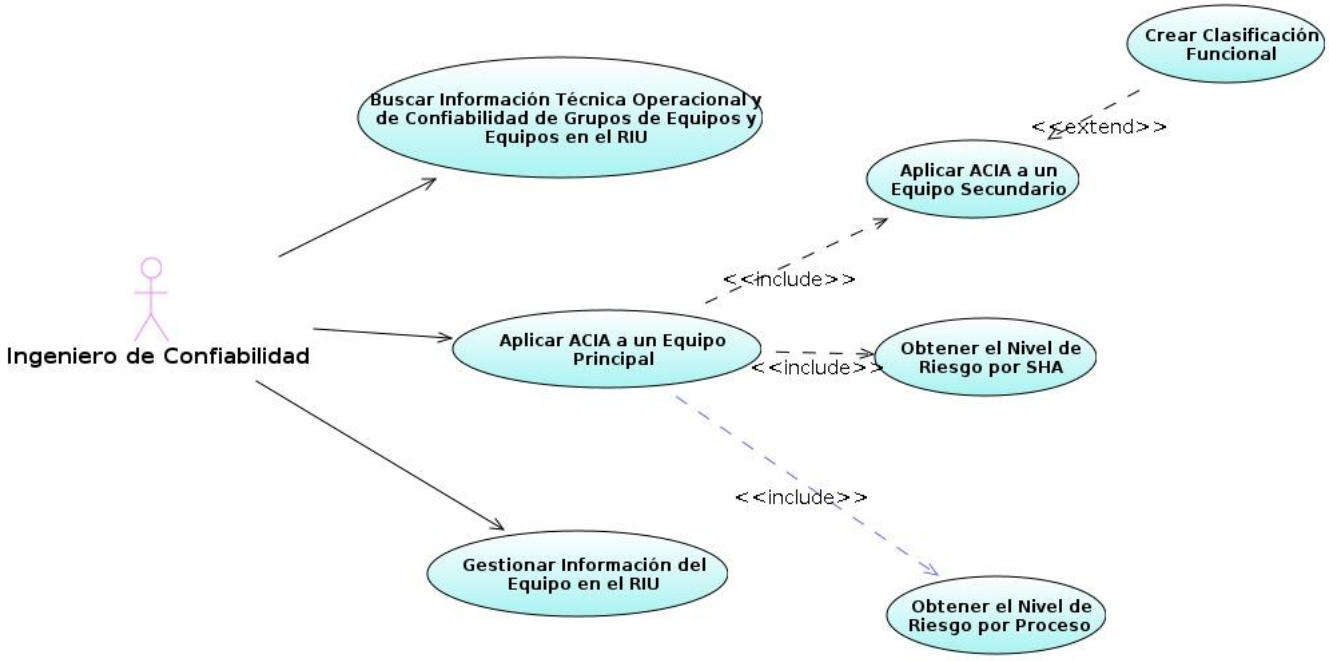


Figura 4 Diagrama de Caso de Uso de la Metodología ACIA

3.4 Descripción de casos de uso.

El objetivo principal de detallar cada caso de uso es describir su flujo de sucesos en detalle, incluyendo cómo comienza, termina e interactúan con los actores **(RUMBAUGH et al. 2000).**

A continuación se muestra la descripción del caso de uso “Aplicar Metodología ACIA a un Equipo Principal”.

Caso de Uso Aplicar Metodología ACIA a un Equipos Principal.

1 **Nombre:** Aplicar Metodología ACIA a un Equipo Principal

2 Breve Descripción

Este caso de uso le permite al Ingeniero de Confiabilidad aplicar la Metodología de Análisis de Criticidad Integral de Activos (ACIA) a un Equipo Principal de un Grupo de Equipos con el objetivo de establecer una jerarquía o prioridades de sistemas, grupos de equipos y equipos, mediante la cuantificación del impacto global de su comportamiento en el negocio, denominada riesgo, creando una estructura que facilita la toma de decisiones y el direccionamiento del esfuerzo y los recursos.

3 Actores

Ingeniero de Confiabilidad “activa”

4 Flujo básico.

4.1Seleccionar la realización de la Metodología ACIA

4.1.1 El caso de uso inicia cuando el Ingeniero de Confiabilidad le solicita al Sistema aplicar dentro de la etapa de Diagnóstico, la metodología ACIA a un Grupo de Equipos, para ello selecciona:

Etapa Diagnóstico / Metodología ACIA

4.2Registrar participantes

4.2.1 El Sistema le muestra al Ingeniero de Confiabilidad una ventana donde debe registrar los datos de las personas que participarán en la aplicación de la Metodología ACIA al equipo principal

de un Grupo de Equipos, permitiéndole ingresar cuantas personas estime conveniente. La ventana posee los siguientes campos:

- **Fecha** (Obligatorio)
- **Nombre y Apellidos** (Obligatorio)
- **Cédula** (Obligatorio)
- **Cargo** (Obligatorio)
- **Teléfono** (No Obligatorio)
- **Correo** (No Obligatorio)

4.2.2 El Ingeniero de Confiabilidad ingresa los datos obligatorios y no obligatorios (si estima conveniente) de cada participante y selecciona la opción “Guardar”.

4.2.3 El Sistema verifica la completitud de los campos obligatorios (Alterno 5.1) y almacena la información de cada participante en el Repositorio de Información Unificada (RIU). [T]

4.3Buscar Grupo de Equipos en el RIU

4.3.1 El Ingeniero de Confiabilidad solicita al Sistema la búsqueda de la información técnica, operacional y de confiabilidad del Grupo de Equipos.

4.3.2 El Sistema ejecuta el caso de uso “Buscar Información Técnica, Operacional y de Confiabilidad de Grupos de Equipos y Equipos en el RIU”.

4.3.3 El Sistema verifica que al equipo principal del grupo de equipos no se le ha realizado ACIA, permitiéndole al Ingeniero de Confiabilidad la realización de dicha actividad. (Alterno 5.2).

4.4 Aplicar ACIA al equipo principal de un grupo de equipos

4.4.1 El Ingeniero de Confiabilidad solicita al Sistema estimar el nivel de riesgo o criticidad a un equipo principal (Cep) del grupo de equipos señalado por él.

4.4.2 El Sistema despliega una ventana y se ejecutan los Casos de Usos “Obtener de Nivel de Riesgos por SHA del Equipo Principal” y “Obtener de Nivel de Riesgos por Procesos del Equipo Principal” (en el orden en que aparecen).

4.5Nivel de Criticidad Final para Equipos Principales de un Grupo de Equipos Nivel de Criticidad Final para el Equipo Principal de un Grupo de Equipos

4.5.1 El Sistema mediante la Matriz de Criticidad de Ciliberti asigna el Nivel de Criticidad del equipo principal (Ver Nota 11.1 “*Nivel de Criticidad del equipo*”), de acuerdo al Nivel de Riesgos por SHA y por Procesos obtenidos, de lo que resultará un código de criticidad y nivel de criticidad con su respectivo color; estos se le muestran al Ingeniero de Confiabilidad para su validación. (Ver Nota 11.2 *“Nivel de Riesgo”).*

4.6Guardar Información en RIU

4.6.3 El Ingeniero de Confiabilidad solicita al Sistema Guardar la información en el RIU.

4.6.2 El Sistema comprueba que el Equipo Principal no posee Equipos Secundarios asociados (Alterno 5.5).

4.6.3 El Sistema muestra un mensaje preguntando a los participantes ¿Están de acuerdo con el proceso realizado?

4.6.4 El Ingeniero de Confiabilidad responde que “Si”. (Alterno 5.6).

4.6.5 El Sistema Guarda la información.

4.7Fin del Caso de Uso

4.7.1 El Ingeniero de Confiabilidad le solicita al Sistema “Cerrar la ventana”.

4.7.2 El Sistema comprueba que se guardó toda la información y cierra la ventana; terminando así el Caso de Uso. (Alterno 5.8)

5 Flujos alternos

5.1Verifica campos incompletos

5.1.1 En el paso 4.2.3 del flujo básico, el Sistema verifica que hay campos obligatorios incompletos (vacíos).

5.1.2 El Sistema le muestra un mensaje de error al Ingeniero de Confiabilidad indicándole “Debe ingresar todos los campos obligatorios”.

5.1.3 El Ingeniero de Confiabilidad acepta el mensaje.

5.1.4 El Sistema retorna al paso 4.2.2 del flujo básico.

5.2Al Equipo Principal se le ha realizado ACIA, no así a Equipos Secundarios

5.2.1 En el paso 4.3.3 del flujo básico, el Sistema verifica que al equipo principal del grupo de equipos se le ha realizado ACIA, pero que hay equipos secundarios asociados a él, a los que no se le ha realizado el ACIA. (Alterno 5.3)

5.2.2 El Sistema muestra una mensaje al Ingeniero de Confiabilidad donde pregunta ¿Desea actualizar el ACIA al equipo principal?

5.2.3 El Ingeniero de Confiabilidad responde “No”. (Alterno 5.4).

5.2.4 El Sistema le muestra una notificación al Ingeniero de Confiabilidad indicándole “Se le realizará ACIA al equipo secundario señalado” y ejecuta el caso de Uso “Aplicar Metodología ACIA a Equipos Secundarios”.

5.3Se le realizó ACIA al Equipo Principal y a los Equipos Secundarios.

5.3.1 En el paso 5.2.1 del flujo alterno 5.2 el Sistema verifica que se le ha realizado el ACIA a todos los equipos secundarios.

5.3.2 El Sistema le muestra la información de ACIA del Grupo de Equipos seleccionado al Ingeniero de Confiabilidad.

5.3.3 El Sistema muestra una mensaje al Ingeniero de Confiabilidad donde pregunta ¿Desea actualizar el ACIA al equipo principal?".

5.3.4 El Ingeniero de Confiabilidad responde "No". (Alterno 5.4).

5.3.5 El Sistema retorna al paso 4.7 del flujo básico.

5.4 Actualizar ACIA a Equipo Principal

5.4.1 En el paso 5.2.3 del flujo alterno 5.2, el Ingeniero de Confiabilidad responde "Si" desea actualizar ACIA al Equipo Principal.

5.4.2 El Sistema retorna al paso 4.4 del flujo básico.

5.5 Advertencia al intentar guardar información en el RIU de Equipos Principales con Equipos Secundarios Asociados.

5.5.1 En el paso 4.6.2 del flujo básico, el Ingeniero de Confiabilidad solicita al Sistema Guardar la información en el RIU.

5.5.2 El Sistema comprueba que el Equipo Principal posee Equipos Secundarios asociados.

5.5.3 El Sistema muestra un mensaje de Advertencia al Ingeniero de Confiabilidad indicándole "Hay equipos secundarios asociados".

5.5.4 El Ingeniero de Confiabilidad acepta el mensaje.

5.4.5 El sistema ejecuta el caso de uso “Aplicar Metodología ACIA a Equipos Secundarios”.

5.6Modificar el Nivel de Criticidad Final y Código de Equipo Principal por Criterio de Expertos

5.6.1 En el paso 4.6.4 del flujo básico, el Ingeniero de Confiabilidad responde que “No” está de acuerdo con el proceso realizado.

5.6.2 El Sistema le permite editar el Nivel de Criticidad Final y Código del Equipo Principal, mostrándole un campo para que justifique el por qué de la modificación.

5.6.3 El Ingeniero de Confiabilidad modifica los campos y justifica el cambio.

5.6.4 El Sistema verifica la completitud de los campos (Alterno 5.7).

5.6.5 El Sistema guarda la información y retorna al paso 4.7 del flujo básico.

5.7Campos incompletos en la Modificación del Nivel de Criticidad Final y Código de Equipo Principal

5.7.1 En el paso 5.6.4 del flujo alterno 5.6, el Sistema verifica que hay campos incompletos y muestra un mensaje de Error al Ingeniero de Confiabilidad indicándole “Debe ingresar todos los campos obligatorios”.

5.7.2 El Ingeniero de Confiabilidad acepta el mensaje y completa los campos donde se produjo el error.

5.7.3 El Sistema guarda la información y retorna el paso 4.7 del flujo básico.

5.8Error al Finalizar el Caso de Uso

5.8.1 En el paso 4.7 del flujo básico, el Ingeniero de Confiabilidad solicita al Sistema “Cerrar la ventana”.

5.8.2 El Sistema comprueba que no se guardó la información en el RIU y le muestra un mensaje al Ingeniero de Confiabilidad indicándole “Debe guardar la información”, retornando al paso 4.6 del flujo básico.

6 Requerimientos especiales

7 Precondiciones

- El Ingeniero de Confiabilidad debe estar debidamente autorizado para poder aplicar la Metodología ACIA a un Equipo Principal.

8 Postcondiciones

Si el caso de uso finaliza correctamente, se logró:

- Registrar participantes en el RIU.
- Almacenar en el RIU el Nivel de Criticidad Final del equipo principal de un grupo de equipos.
- Ejecutar el Caso de Uso “Obtener de Nivel de Riesgos por SHA del Equipo Principal” exitosamente.
- Ejecutar el Caso de Uso “Obtener de Nivel de Riesgos por Procesos del Equipo Principal” exitosamente.
- Ejecutar el Caso de Uso ”Gestionar Información del Equipo en el RIU” exitosamente, en caso de haberse invocado.
- Ejecutar el Caso de Uso “Aplicar Metodología ACIA a Equipos Secundarios” exitosamente, en caso de haberse invocado.

Si el caso de uso no terminó correctamente se mostró el mensaje de error correspondiente.

9 Puntos de extensión

- En el paso 5.3 del flujo alterno con el caso de uso “Aplicar Metodología ACIA a Equipos Secundarios”.

10 Puntos de inclusión

- En el paso 4.4 del flujo alterno con los casos de usos “Obtener de Nivel de Riesgos por SHA del Equipo Principal”, “Obtener de Nivel de Riesgos por Procesos del Equipo Principal”.

11 Notas

11.1 Nivel de Criticidad del equipo

A continuación se muestra la Matriz de la Metodología de Ciliberti, que permite encontrar el Nivel de Criticidad Resultante del Equipo o Dispositivo en análisis, de acuerdo al Nivel de Riesgos por SHA y por Procesos.

NIVEL DE CRITICIDAD EN PROCESO							
	Muy alta	5_7	A	A	A	A	A
	Alto	4	B	B	B	A	A
	Medio	3	C	C	B	B	A
	Bajo	2	D	C	C	B	A
	Despreciable	0_1	E	D	C	B	A
			0_1	2	3	4	5_7
			Despreciable	Bajo	Medio	Alto	Muy alta
	NIVEL DE CRITICIDAD SHA						

Tabla 2 Matriz de la Metodología de Ciliberti

La siguiente Matriz de asignación, es un apoyo a la Matriz de la Metodología de Ciliberti antes mencionada.

C_{SHAEP}	C_{PEP}	Código de Criticidad	Nivel de Criticidad
0	0	E00	IV
0	1	E01	IV
1	0	E10	IV
1	1	E11	IV
0	2	D02	IV
1	2	D12	IV
2	0	D20	IV
2	1	D21	IV
0	3	C03	III
1	3	C13	III

2	2	C22	III
2	3	C23	III
3	0	C30	III
3	1	C31	III
3	2	C32	III
0	4	B04	II
1	4	B14	II
2	4	B24	II
3	3	B33	II
3	4	B34	II
4	0	B40	II

4	1	B41	II
4	2	B42	II
4	3	B43	II
0	5	A05	I
0	6	A06	I
0	7	A07	I
1	5	A15	I
1	6	A16	I
1	7	A17	I
2	5	A25	I
2	6	A26	I

2	7	A27	I
3	5	A35	I
3	6	A36	I
3	7	A37	I
4	5	A45	I
4	6	A46	I
4	7	A47	I
5	0	A50	I
5	1	A51	I
5	2	A52	I
5	3	A53	I

5	4	A54	I
5	5	A55	I
5	6	A56	I
5	7	A57	I
4	4	A44	I
6	0	A60	I
6	1	A61	I
6	2	A62	I
6	3	A63	I
6	4	A64	I
6	5	A65	I

6	6	A66	I
6	7	A67	I
7	0	A70	I
7	1	A71	I
7	2	A72	I
7	3	A73	I
7	4	A74	I
7	5	A75	I
7	6	A76	I
7	7	A77	I

Tabla 3 Matriz de apoyo de la Metodología de Ciliberti

11.2 Nivel de Riesgo

A continuación se muestra el Nivel de Criticidad para los equipos:

Nivel	Descripción
I	Muy Alta Criticidad
II	Alta Criticidad
III	Media Criticidad
IV	Baja Criticidad

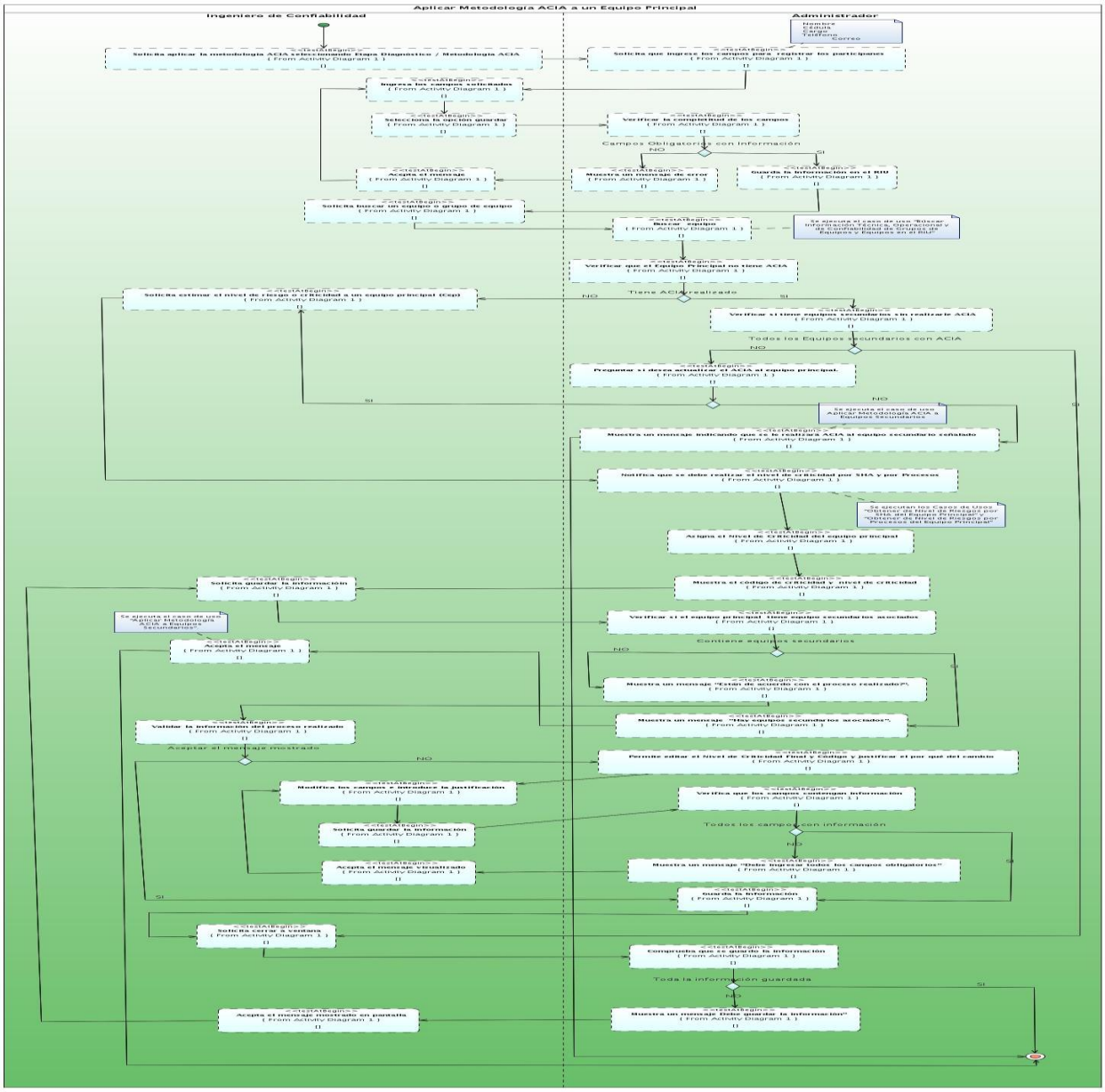
Tabla 4 Nivel de Criticidad

12. Relaciones

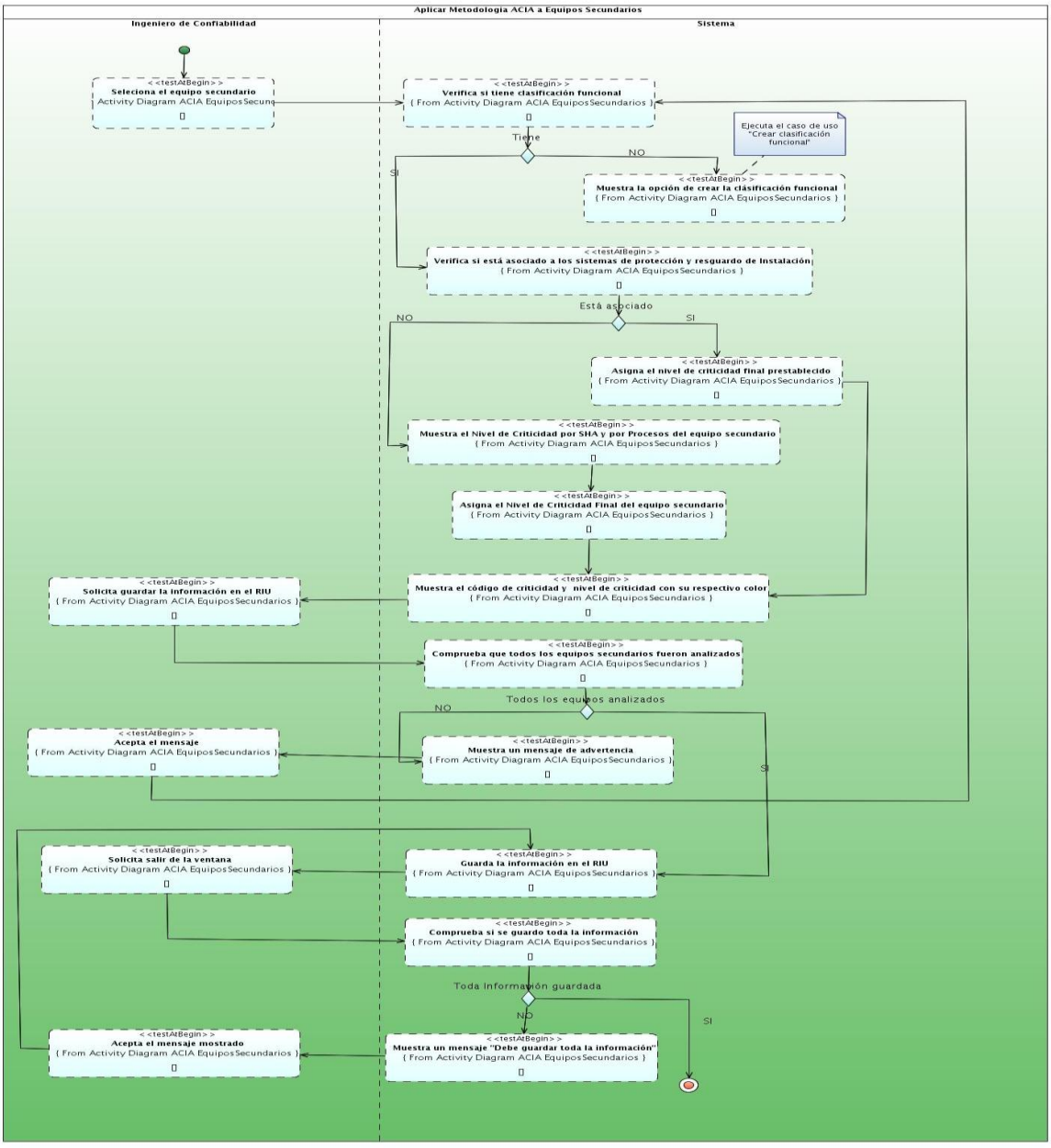
- Este caso de uso está relacionado con los casos de usos: “Autenticar Usuario”, “Cálculo de Nivel de Riesgos por SHA del equipo principal”, “Cálculo de Nivel de Riesgos por Procesos del equipo principal” ,”Gestionar Información del Equipo en el RIU” y “Aplicar metodología ACIA a Equipos secundarios”.

3.5 Diagramas de actividad

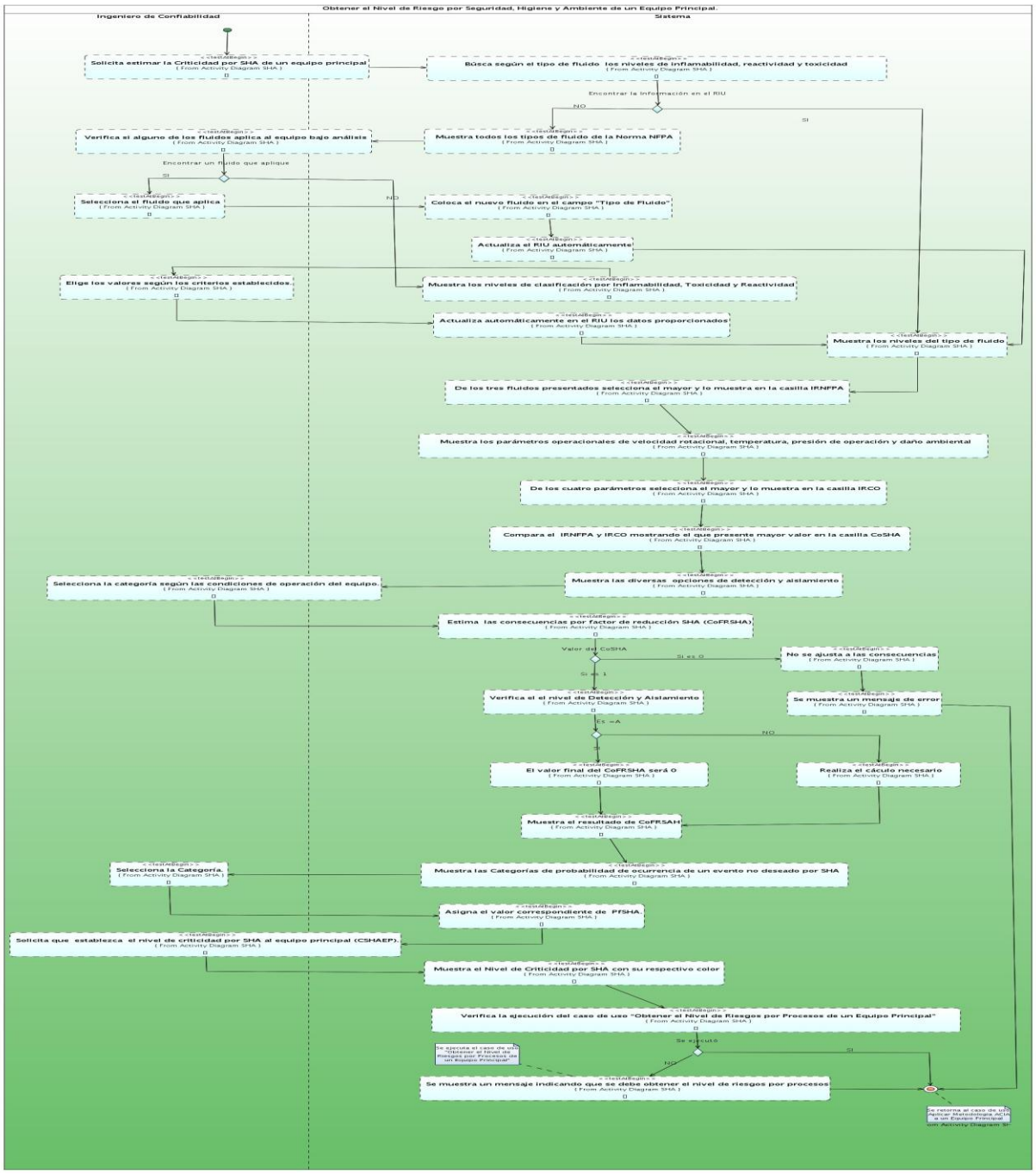
3.5.1 Diagrama de actividades del caso de uso Aplicar Metodología ACIA a un Equipo Principal.



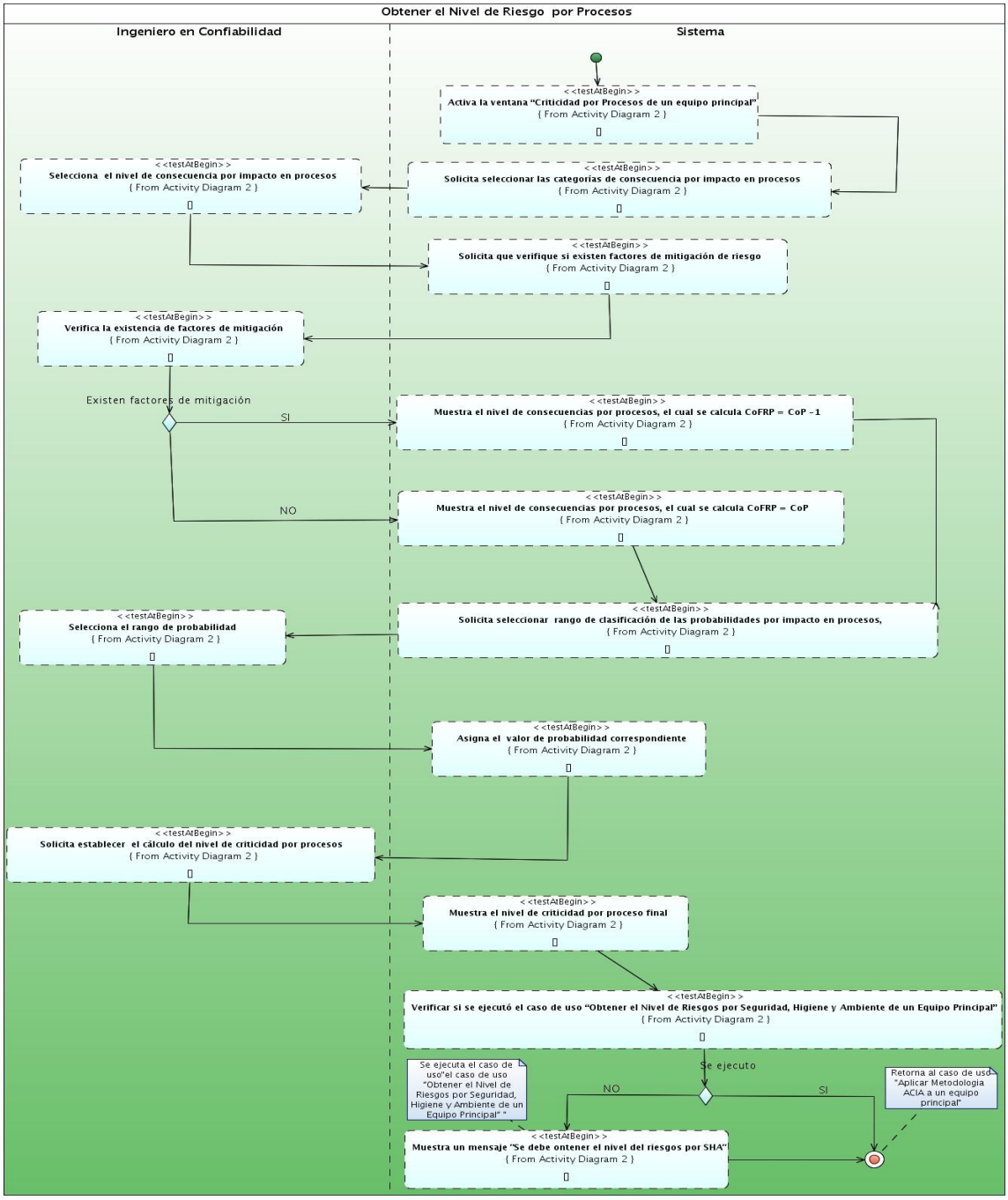
3.5.2 Diagrama de actividades del caso de uso Aplicar Metodología ACIA a un Equipo Secundario.



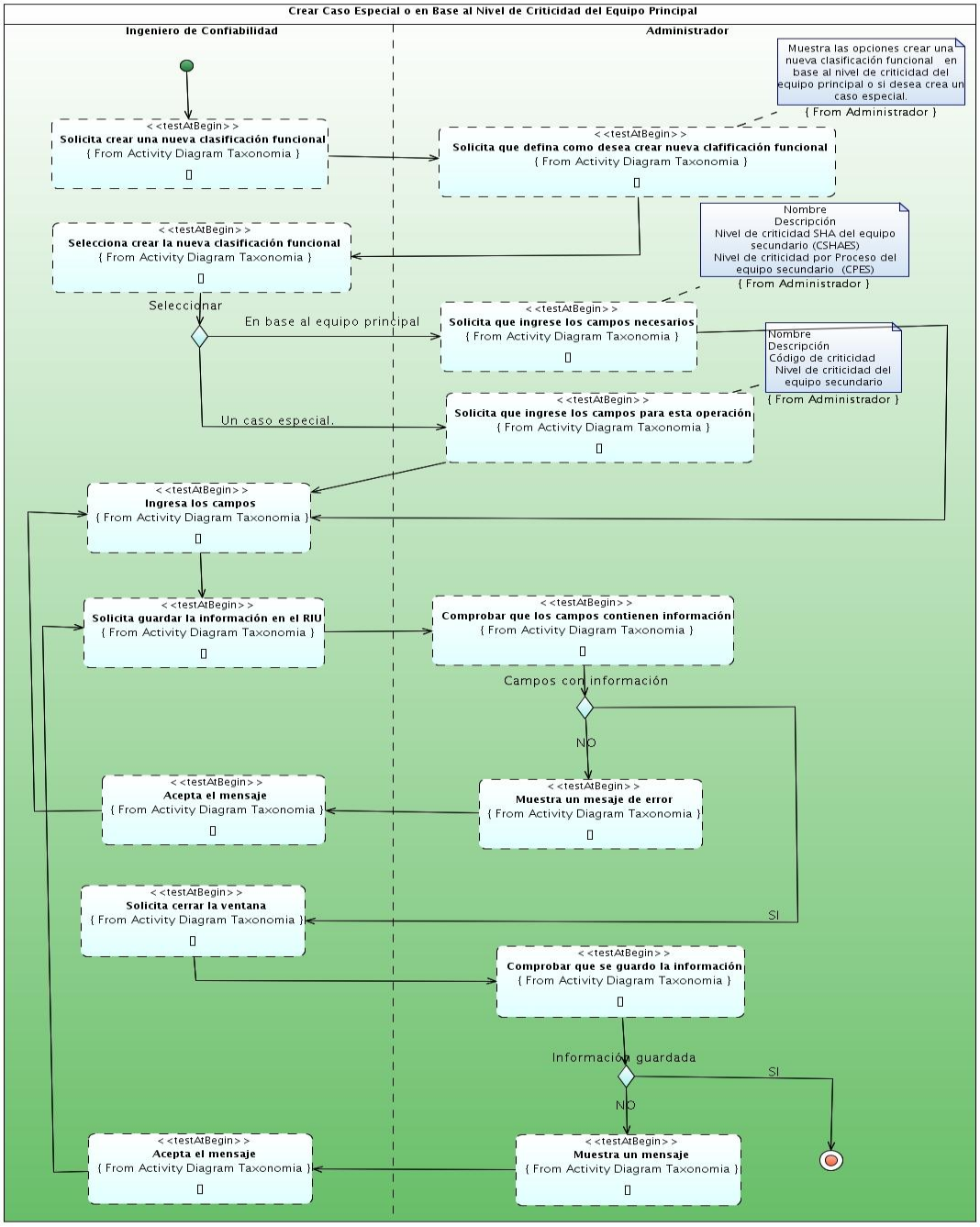
3.5.3 Diagrama de actividades del caso de uso Obtener el Nivel de Riesgos por SHA.



3.5.4 Diagrama de actividades del caso de uso Obtener el Nivel de Riesgos por Procesos.



3.5.4 Diagrama de actividades del caso de uso Crear Nueva Clasificación Funcional.



3.6 Prototipo de Interfaz de Usuario

Los prototipos de interfaz de usuario nos ayudan a comprender y especificar las interacciones entre los actores humanos y sistema durante la captura de requisitos. No sólo nos ayuda a desarrollar una interfaz gráfica mejor, sino también a comprender mejor los casos de usos. A la hora de especificar la interfaz gráfica de usuario también pueden utilizarse otros artefactos, como los modelos de interfaz gráfica y los esquemas de pantalla. **(RUMBAUGH et al. 2000).**

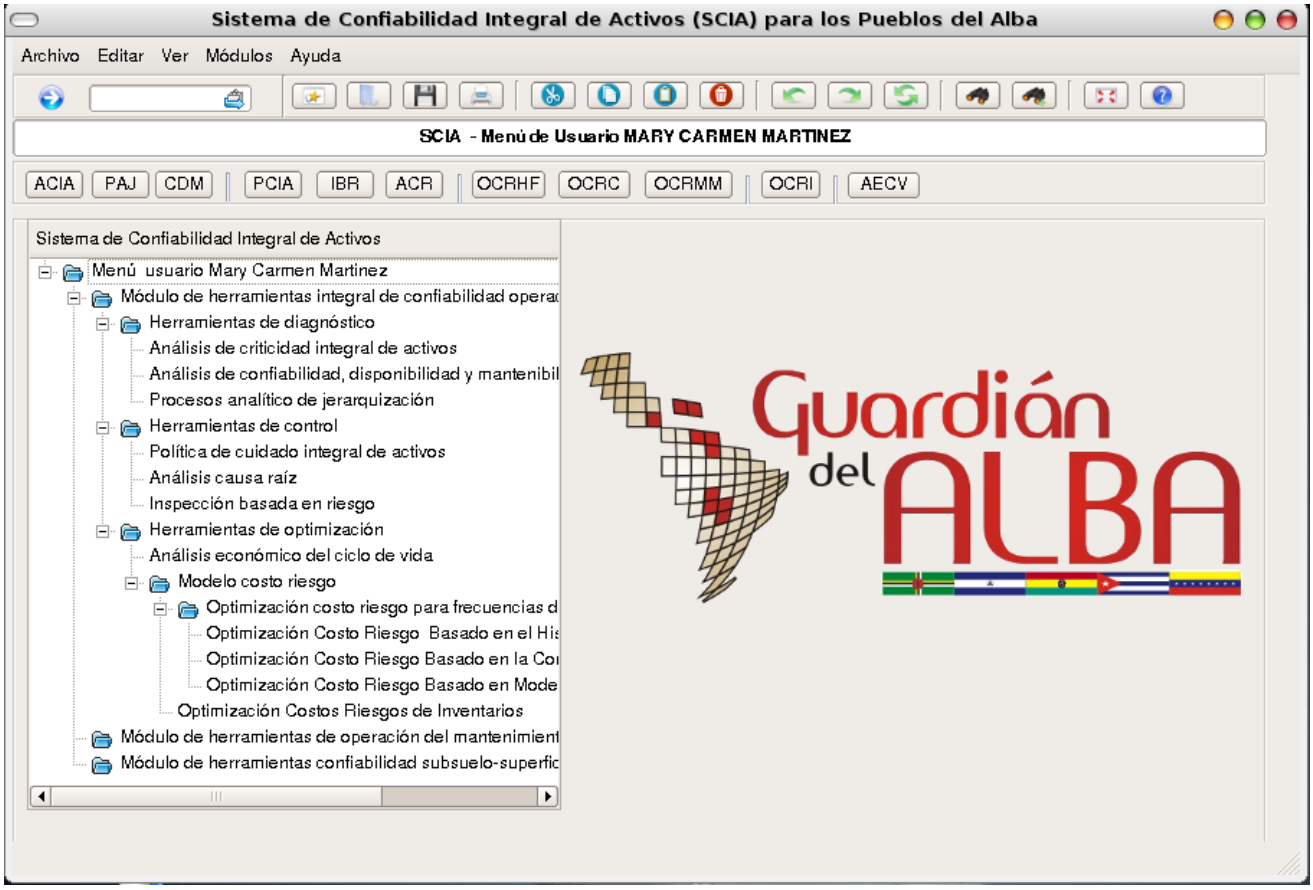


Figura 5 Vista principal de la aplicación

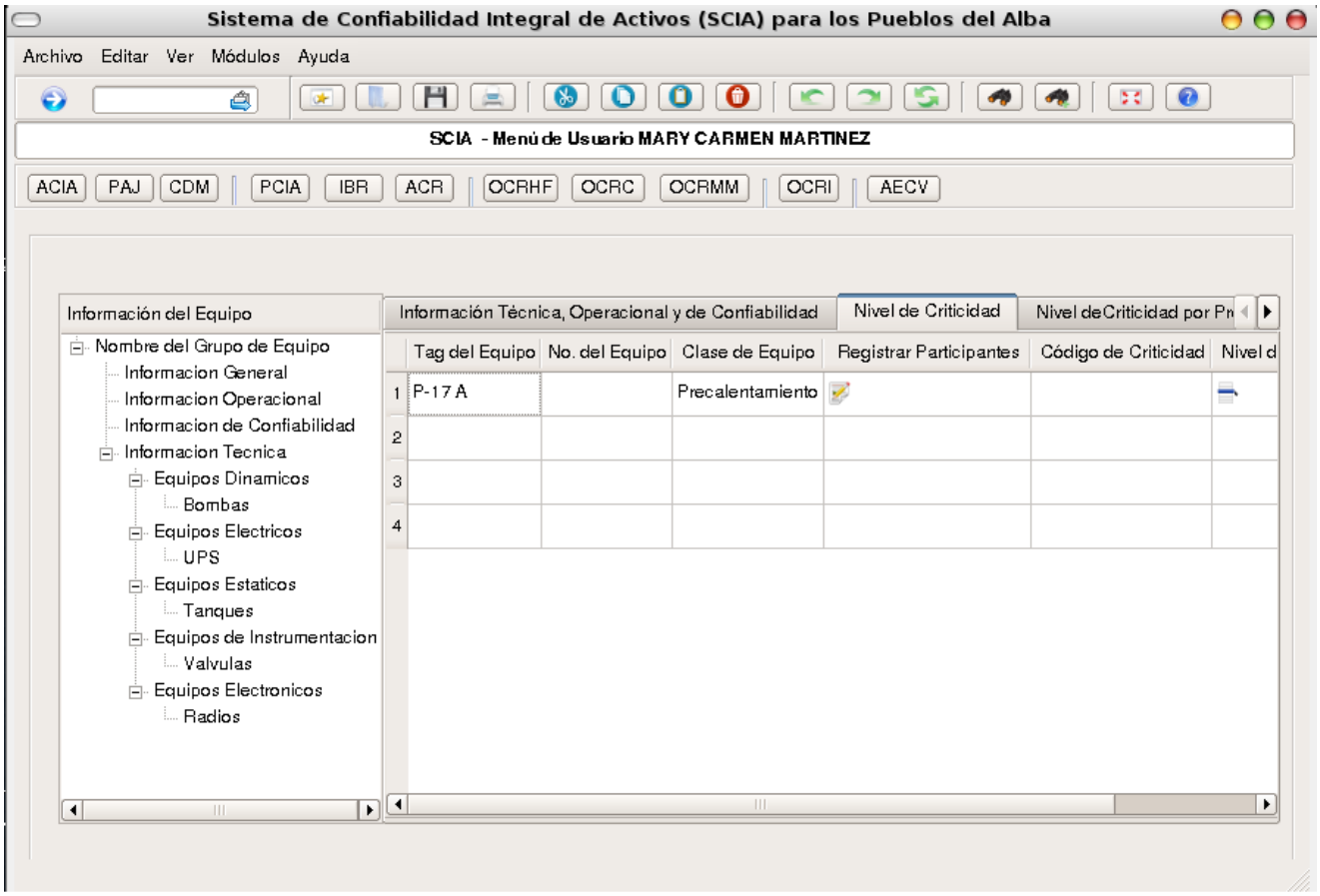


Figura 7 Vista al activar el Nivel de Criticidad

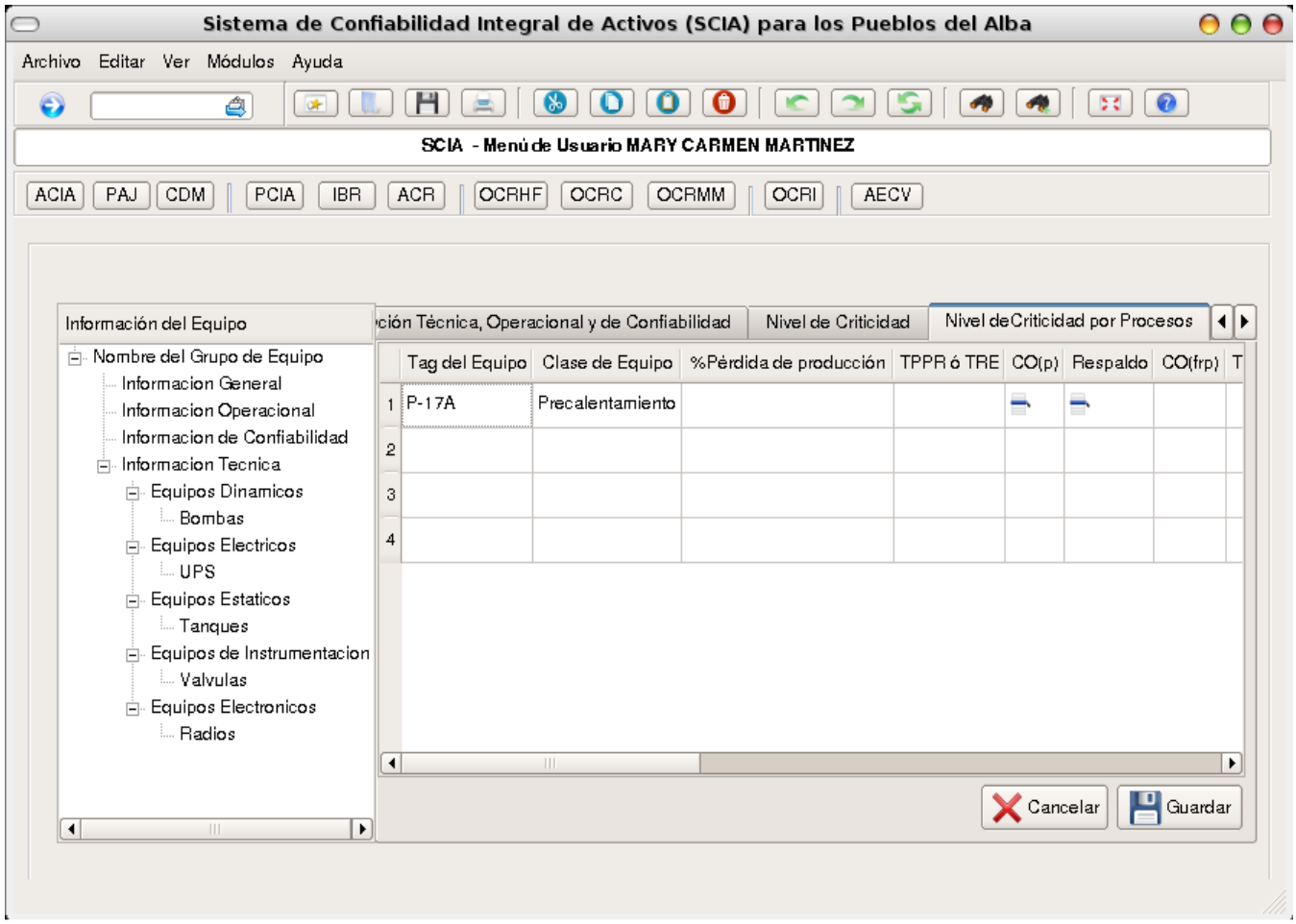


Figura 8 Nivel de Criticidad por Procesos

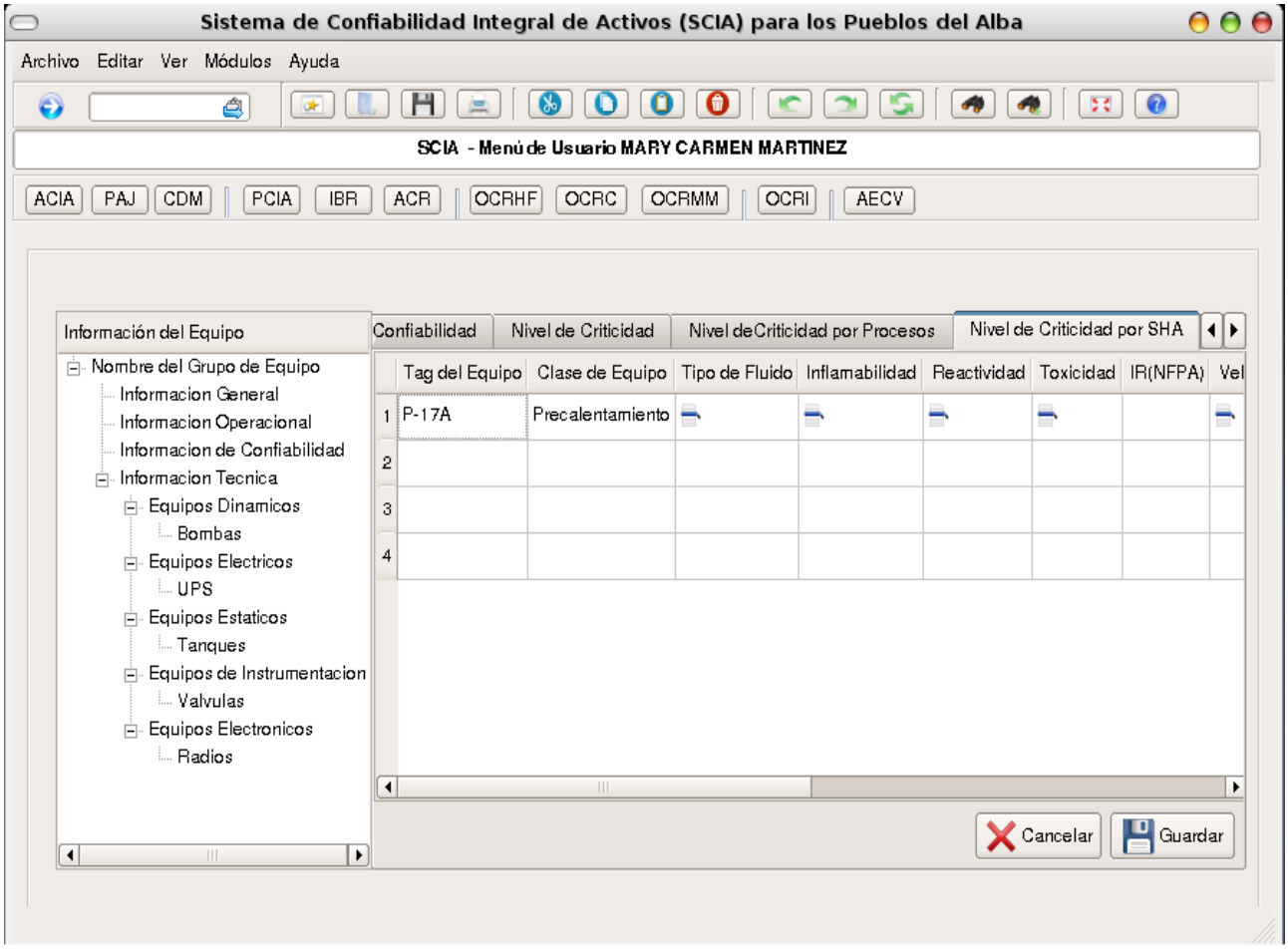


Figura 9 Nivel de Criticidad por SHA

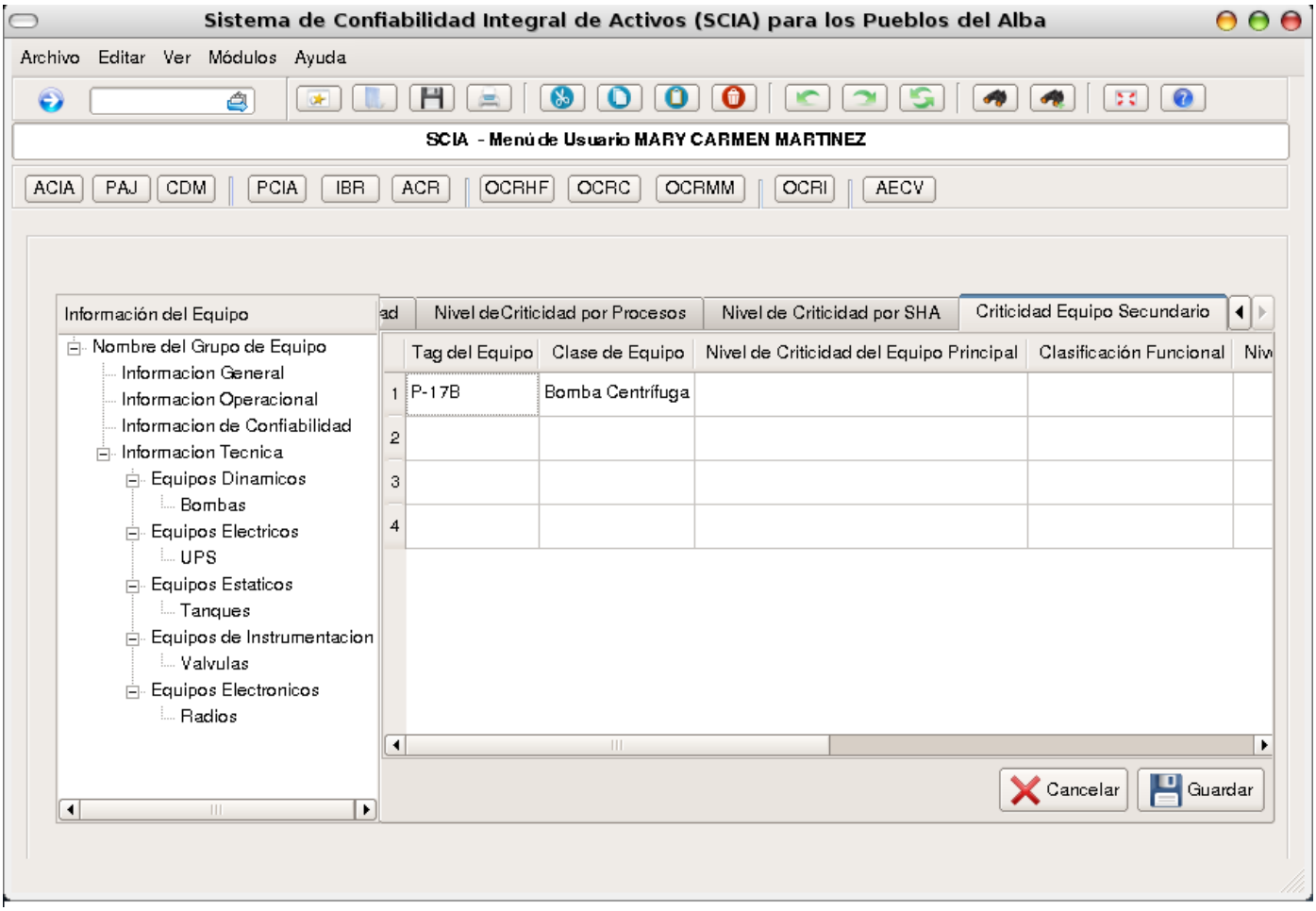


Figura 10 Nivel de Criticidad de Equipos Secundarios

3.7 Conclusiones

Haciendo uso de las técnicas y herramientas seleccionadas se obtuvo el levantamiento de requisitos de la Metodología ACIA, dándole realización a una parte de las actividades correspondiente a la etapa de análisis. Se concluyó con la validación de algunos requisitos críticos mediante un prototipo que contó con la aprobación del cliente consciente de que lo que estaba viendo era un prototipo y no el sistema final.

CONCLUSIONES:

- A partir del levantamiento de requisitos con la participación constante del cliente como especialista del tema, se logró desarrollar el flujo de requisitos completamente, el cual fue validado y aceptado.

- Con la obtención de un prototipo de interfaz de la Metodología ACIA se cumplió con el objetivo trazado de desarrollar el flujo de requisitos para la Metodología ACIA del producto SCIA.

- La tecnología escogida cumplió con las expectativas de parte del producto obtenido.

- Se realizaron validaciones con el cliente, con el objetivo de llevar una trazabilidad entre los requisitos planteados y el prototipo de interfaz de usuario presentado.

RECOMENDACIONES:

- Darle continuidad al trabajo basándose en los requisitos descritos, casos de usos, prototipos no funcionales y diseños de la Metodología ACIA, donde el paso siguiente es el análisis y diseño.

- Profundizar en el estudio de otros Sistemas de Confiabilidad buscando ampliar las funcionalidades del sistema SCIA.

- Utilizar la Metodología en proyecto como ERP y el Minbas, en los cuales realizan análisis de criticidad a sus activos.

REFERENCIAS BIBLIOGRÁFICAS:

ALARCOS, G. 2009. Ingeniería Inversa. [En línea] 2009. [Citado el: 1 de Junio de 2009.] http://alarcos.inf-cr.uclm.es.

BECK, K. 2000. *"Extreme Programming Explained. Embrace Change".* s.l. : Pearson Education,1999,Traducido al español como: "Una explicación de la programación extrema. Aceptar el cambio", Addison Wesley. 2000., 2000.

BERNHARD, RUMPE y ASTRID, SCHRÔDER. 2001. *"Quantitative survey on Extreme Programming Projects". Informe, Universidad de Tecnología de Munich.* 2001.

CALERO SOLIS, MANUEL. 2003. Una explicación de la programación extrema (XP).V Encuentro usuarios xBase. [En línea] 2003. [Citado el: 16 de Febrero de 2009.] http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/explicaxp.pdf..

DEBIAN, C. 2009. Acerca de Debian. [En línea] 2009. [Citado el: 25 de Abril de 2009.] www.debian.org.

IEEE, 1233. 1998. IEEE 1233-1998. [En línea] 1998. [Citado el: 29 de Mayo de 2009.] http://standards.ieee.org/reading/ieee/std_public/description/se/1233-1998_desc.html.

KOCH, N. 2009. Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo. [En línea] 2009. [Citado el: 30 de Abril de 2009.] www.pst.informatik.uni-muenchen.de.

LARMAN., CRAIG. 2004. *Agile & Iterative Development. A Manager's Guide. Reading, Addison-Wesley.* 2004.

MARTINEZ, MARY CARMEN. 2008. *Documento Visión "Sistema de Confiabilidad Integral de Activos (SCIA) para los Pueblos del ALBA".* Mérida, Venezuela : s.n., 2008.

PALACIO, JUAN. 2006. El modelo Scrum. *Navegapolis.* [En línea] 2006. [Citado el: 4 de Marzo de 2009.] http://www.navegapolis.net/content/view/694 .

ROMERO, MARÍA TERESA. 2009. *Especificaciones Técnicas de la Metodología ACIA.* Mérida Venezuela : s.n., 2009.

RUMBAUGH, J. y JACOBSON, I. 2000. *El proceso unificado de desarrollo de software.* La Habana : Addison Wesley Longman, 2000.

SANGUINETTI, C. *Análisis y Diseño de Sistema.*

SCHWABER K, BEEDLE M., MARTIN R.C. 1999. *"Agile Software Development with SCRUM".* s.l. : Prentice Hall. 2001.Prentice Hall. 1999., 1999.

TORRES, L. S. El analista de sistemas y el paradigma estructurado. [En línea] [Citado el: 20 de 5 de 2009.] ww.monografias.com.

BIBLIOGRAFÍA:

- "Establishing Mechanical Integrity of Process Equipment using a Criticality-Based Maintenance Program" [Libro]. - [s.l.] : National Petroleum Refiners Association, Mayo 1996..
- **Nar Lucía y Luís Tovar** MANUAL DEL PROCESO DE MANTENIMIENTO DE RUTINA ESTRUCTURA DE ACTIVOS: JERARQUIZACIÓN DE SISTEMAS Y TAXONOMÍA DE EQUIPOS EN PDVSA [Informe]. - Venezuela : [s.n.], 2009.
- National Fire Protection Asociation. Standard System for the Identification of the Hazards of Materials for Emergency Response 704 [Libro]. - Edición electrónica, 2002.
- Reliability and Risk Management, S. A. Manual de Confiabilidad Integral [Libro]. - 2008. - Vol. I.
- Risk Based Inspection Base Resource Document. API Publication 581 [Informe]. - [s.l.] : American Petroleum Institute., Mayo 2000.
- **Superintendente de Soporte; Gerente de Mantenimiento** GUIA PARA EL ANALISIS DE CRITICIDAD DE ACTIVOS DEL MEJORADOR SINCRON [Informe]. - Venezuela : [s.n.], Mayo 2005.
- **Yanez, Medardo; Nuccette, Geokena** Confiabilidad Integral un Enfoque Práctico [Libro]. - Maracaibo, Zulia, Venezuela : [s.n.], Enero 2009. - Vol. Tomo 1 Manual de Aplicación de Disciplinas y Metodologías de Ingeniería y Análisis de Riesgo.
- **Yañez M. Medardo [y otros]** Confiabilidad Integral ® SINERGIA DE DISCIPLINAS [Libro]. - Venezuela : [s.n.], Enero 2007. - Vol. I DISCIPLINAS.
- **Yañez M. Medardo [y otros]** Confiabilidad Integral ® SINERGIA DE DISCIPLINAS [Libro]. - Venezuela : [s.n.], Enero 2008. - Vol. II Metodologías.

ANEXOS

Anexo 1 Guía de entrevistas realizadas para el levantamiento de requisitos. (DST AIT Mérida 2008)

Paso 1: Establecer el perfil del Stakeholder o Usuario.

1. Nombre y Apellidos.
2. Cargo.
3. ¿Cuáles son sus responsabilidades claves?
4. ¿Qué entregables produce? ¿Para quién?

Paso 2: Comprender el ambiente del usuario.

1. ¿Quiénes son los usuarios?
2. ¿Cuál es su nivel educativo?
3. ¿Los usuarios tienen experiencia con este tipo de aplicación?
4. ¿Qué plataformas están en uso? ¿Cuáles son sus planes para futuras plataformas?
5. ¿Qué tipos de documentación escrita y en línea UD. Necesita?

Paso 3: Evaluando la confiabilidad, desempeño y necesidades de mantenimiento.

1. ¿Cuáles son sus expectativas de confiabilidad?
2. ¿Cuáles son los requerimientos especiales para la licencia?
3. ¿Conoce de algún otro requerimiento que debamos saber o tener en cuenta?

Paso 4: Resumiendo.

1. ¿Hay otras cuestiones que deba preguntarle?
2. ¿Si necesito hacer seguimiento, puedo llamarlo o escribirle al correo?
3. ¿Estaría dispuesto a participar en una revisión de los requerimientos?

GLOSARIO DE TÉRMINOS

Actor: Alguien o algo, fuera del sistema o negocio que interactúa con el sistema o negocio.

AIT: Automatización, Informática y Telecomunicaciones. Es la Dirección de Petróleos de Venezuela, S.A. que rige, provee y garantiza los servicios y soluciones integrales de tecnologías de Automatización, Informática y Telecomunicaciones (AIT) de la Corporación, por ello contribuye a mantener su continuidad operativa y a ejecutar los planes; innovando y actuando como agentes de transformación con responsabilidad social, económica y ambiental.

Artefactos*:* Una parte de la información que (1) es producida, modificada, o usada por un proceso, (2) define un área de responsabilidad, y (3) está sujeta al control de versión. Un artefacto puede ser un modelo, un elemento del modelo, o un documento. Un documento puede adjuntar otros documentos.

Código Abierto: Es una tendencia internacional del desarrollo de software que profesa la distribución del código junto a las aplicaciones, se rigen por licencias tales como GNU/GPL.
GNU/GPL: Es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

GUI: Graphical User Interface (**Interfaz gráfica de usuario**) es un método para facilitar la interacción del usuario con el ordenador a través de la utilización de un conjunto de imágenes y objetos pictóricos (iconos, ventanas) además de texto.

Herramientas CASE: Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software (Investigación Preliminar, Análisis, Diseño, Implementación e Instalación.).

IEEE: Corresponde a las siglas de The Institute of Electrical and Electronics Engineers, el Instituto de Ingenieros Eléctricos y Electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.

Ingeniería de Software: Se puede definir como el tratamiento sistemático de todas las fases del ciclo de vida del software.

Multiplataforma: es un término utilizado frecuentemente en informática para indicar la capacidad o características de poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

Norma: La norma es una regla a seguir para alcanzar un fin determinado. Las normas se crean en Comisiones Técnicas de Normalización. Una vez elaborada la norma, esta es sometida durante seis meses a la opinión pública. Transcurrido este tiempo y analizadas las observaciones se procede a su redacción definitiva, con las posibles correcciones que se estimen, publicándose luego. Todas las normas son sometidas a revisiones periódicas con el fin de ser actualizadas.

Proceso de desarrollo de software: Es la definición del conjunto completo de actividades necesarias para transformar los requisitos de un usuario en un producto. Un proceso es una plantilla para crear proyectos.

Proceso: Es un conjunto de actividades y resultados asociados que producen un resultado.

Proyecto de Software: El elemento organizativo a través del cual se gestiona el desarrollo de software. El resultado de un proyecto es una versión de un producto.

Proyecto: Combinación de recursos humanos y no humanos reunidos en una organización temporal para conseguir un propósito, tiene un punto de de comienzo definido y con objetivos definidos mediante los que se identifican.

Requerimiento: Son capacidades o características que debe tener el sistema o modelo desarrollo para satisfacer la demanda y/o necesidad del cliente.

Software Libre: Es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

Técnicas: Sucesión ordenada de acciones que se dirigen a un fin concreto, conocido y que conduce a unos resultados precisos.

Tecnología: Tecnología es una característica propia del ser humano consistente en la capacidad de éste para construir, a partir de materias primas, una gran variedad de objetos, máquinas y herramientas, así como el desarrollo y perfección en el modo de fabricarlos y emplearlos con vistas a modificar favorablemente el entorno o conseguir una vida más segura. El ámbito de la Tecnología está comprendido entre la Ciencia y la Técnica propiamente dichas.