Universidad de las Ciencias Informáticas
Facultad 1

Título: Herramienta de apoyo para la definición del comienzo de un programa de mejora de procesos.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

Carlos Rafael Rodríguez Martínez

Mircella Margarita Batista Pérez

Tutores:

Ing. Ailec Granda Dihigo.

Ing. Dayana C. Tejera Hernández.

Ciudad de La Habana, Cuba
Junio, 2010

Declaración de autoría

Declaramos que somos los únicos autores del trabajo titulado:

Herramienta de apoyo para la definición del comienzo de un programa de mejora de procesos y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____ Firma del autor
Mircella Margarita Batista Pérez

_____ Firma del autor
Carlos Rafael Rodríguez Martínez

_____ Firma del tutor
Ailec Granda Dihigo

_____ Firma del tutor
Dayana C. Tejera Hernández

Opinión del tutor del Trabajo de diploma.

Opinión de tutor del Trabajo de diploma

Título: Herramienta de apoyo para la definición del comienzo de un programa de mejora de procesos.

Autores: Mircella Margarita Batista Pérez
Carlos Rafael Rodríguez Martínez

Los tutores del presente Trabajo de diploma consideran que durante su ejecución las estudiantes mostraron las cualidades que a continuación se detallan.

Por todo lo anteriormente expresado consideramos que los estudiantes están aptos para ejercer como Ingenieros en Ciencias Informáticas; y propongo que se le otorgue al Trabajo de diploma la calificación de _____**.**

_____	_____	_____
Firma	Firma	Fecha
Ailec Granda Dihigo	Dayana C. Tejera Hernández	

Índice

Agradecimientos.

A mi madre, a mis abuelos, a mi hermana, por brindarme todo el amor y cariño del mundo y permitirme ser parte de un hogar el cual es el centro de mi vida, sin el que no hubiese llegado a este punto.

A mi padre, por ofrecerme su apoyo e incentivar en mí una pasión por el conocimiento.

A toda mi familia por brindarme su apoyo y reconocimiento, lo que me ha motivado a esforzarme al máximo.

A mi tutora Ailec por haberme ayudado en la realización de esta investigación, aclarando todas mis dudas.

A mi compañera de tesis Mircella Margarita Batista Pérez por ser un ejemplo de constancia y dedicación.

A mi amigo Iskael Díaz Márquez, el cual es un ejemplo a seguir para mí, al que me une una amistad de 7 años y lo considero mi hermano.

A la Revolución por construir magníficas universidades como esta y poner todos los recursos en nuestras manos, por permitirnos la oportunidad de estudiar en ella libremente y formarnos como futuros profesionales.
A todos mis amigos que de una forma u otra contribuyeron a la realización de este sueño.

Mi más sincero agradecimientos a todos; sirva estas líneas como agradecimientos a ellos.

Carlos Rafael Rodríguez Martínez.

Agradecimientos.

A mi papá que aunque no esta presente físicamente en mi vida alrededor de 5 años lo ha estado espiritualmente.

A mi mamá por confiar en mí, apoyarme y animarme para continuar en esta escuela.

A mi hermanito que siempre me estimulaba a continuar la carrera.

A mi novio Tony el cual también le debo mucho.

A toda mi familia por preocuparse siempre por mí.

A mi tutora Ailec por ayudarme con todas mis dudas.

A mi compañero de tesis Carlos por soportarme con todas mis malcriadeces.

A todos mis compañeros y amigos que he tenido durante todos estos años.

A todos los profesores que de una forma u otra me ayudaron a formarme como ingeniera.

Mircella Margarita Batista Pérez

Dedicatoria.

A mi mamá por todo su amor y dedicación, por confiar siempre en mí,

A mis abuelos,

A mi hermana.

A todos, sin distinciones ni diferencias. Por ustedes trato de superarme día a día y para ustedes es el fruto de todos mis resultados.

Carlos Rafael Rodríguez Martínez

A mi papá que aunque no está vivo se lo merece.

A mi mamá que es la mejor madre del mundo.

A mi hermano que es mi ejemplo a seguir

A mi novio por ser mi amigo y compañero durante estos años.

Mircella Margarita Batista Pérez.

Resumen

Los programas de mejoras de procesos en los proyectos productivos de cualquier organización o empresa, se pueden definir como la planificación de un esfuerzo en busca de mejorar los procesos de desarrollo. Existen varias organizaciones y empresas que se han lanzado a iniciar dichos programas y han fracasado, ya que no se encontraban preparadas. Antes de iniciarse a los proyectos de mejoras, deberían analizar una serie de factores de éxito o fracaso, que les permita tener una idea de cual será el resultado.

Ante la necesidad de crear una herramienta que permita evaluar a las organizaciones o empresas si están listas para iniciarse en los programas de mejoras, surge el trabajo de diploma: Herramienta de apoyo para la definición del comienzo de un programa de mejora de procesos, siendo su **objetivo general:** desarrollar una herramienta informática que exprese si una organización está lista para iniciar un programa de mejora de procesos. Esta herramienta ayudará a las organizaciones o empresas a saber si realmente están en condiciones de iniciar estos programas, consiguiendo que no se lancen a esta gran y costosa tarea sin estar preparados.

Este documento recoge todo el trabajo realizado en el diseño e implementación de la herramienta de apoyo. En el mismo se plasma cada uno de los pasos seguidos para lograr el desarrollo de una aplicación segura.

Palabras claves:

Herramienta, Empresas, Procesos, Mejoras, Factores.

Introducción.

Introducción:

Los negocios en el siglo XXI se han modificado y tomado valores diferentes. La existencia de clientes más informados, los avances tecnológicos, entre otras cosas, caracterizan esta etapa empresarial. Para lograr dichos cambios impuestos por este nuevo siglo es indispensable lograr el manejo de una economía del conocimiento e información de forma efectiva y eficiente.

En la sociedad moderna el *software* es una parte indispensable sin el cual muchas de las acciones que se realizan habitualmente resultarían inviables. Con estos avances en la tecnología, el *software* se plantea como el núcleo desde el cual se logra la difusión del conocimiento y se fomenta mucho más la comunicación y colaboración entre clientes y empresas. El reto fundamental en esta revolución tecnológica de las organizaciones, está basado en su adaptación a dichos cambios, enfocando sus acciones a la implementación de nuevos modelos dinámicos y herramientas que revelen la necesidad de realizar los cambios para lograr los objetivos trazados, lo cual es necesario para las organizaciones poder desarrollar producto con buena calidad y de forma eficiente, con el fin de tener una mayor ventaja competitiva ante el reto de los demás mercados.

La mejora continua de procesos, constituye hoy uno de los métodos fundamentales para lograr el éxito de una empresa desarrolladora de *software*. Lograr que los procesos que se ejecuten en la organización sean lo más eficientemente posible, con una calidad elevada, de esta forma ellos podrán obtener soluciones informáticas más efectivas con las prestaciones deseadas.

Actualmente las organizaciones en el mundo, se lanzan a dar inicio a un programa de mejoras, sin saber si realmente están en condiciones para ello. Existen un grupo de factores de éxito y elementos que pueden predeterminarte o al menos darle una idea de las posibilidades de éxito o fracaso que tienes antes de adentrarte en esta faena. Sería una buena práctica, evaluar dichos factores en una organización que quiera dar inicio a estos programas, pero existe el inconveniente de que no hay ninguna herramienta con una interfaz gráfica amigable que exprese de forma clara a los interesados, los resultados del análisis de estos factores.

Ante este inconveniente, se puede definir como **situación problémica** la no existencia de una herramienta informática con una interfaz gráfica amigable que exprese a las organizaciones si están listas para dar inicio a un programa de mejoras de procesos. Esto trae como consecuencia que diferentes empresas u organizaciones se lancen a esta gran y costosa tarea con grandes posibilidades de fracasar. Esto influye de manera negativa en las organizaciones, ya que afecta la economía, su personal pierde la confianza en su trabajo y se hace cada vez más reacio a los cambios.

Problema Científico:

¿Cómo expresar a las organizaciones si están listas para iniciar un programa de mejoras de procesos, de forma que se alcance el éxito en su aplicación?

Objeto de estudio:

El objeto de estudio lo constituyen las herramientas de apoyo al proceso de mejoras.

Campo de acción:

Lo constituyen las herramientas de apoyo a los procesos de mejoras, que utilizan razonamiento basado en casos.

Objetivo general:

Desarrollar una herramienta informática con una interfaz gráfica amigable, que exprese si una organización está lista para iniciar un programa de mejora de procesos.

Objetivos Específicos:

1. Determinar el estado del arte del tema a investigar.
2. Valorar las diferentes tecnologías existentes en el mundo, que se utilizan para aplicaciones como la que se pretende desarrollar.
3. Determinar la metodología de desarrollo de *software* que se utilizará durante el desarrollo de la aplicación
4. Desarrollar una herramienta informática que apoye el proceso de determinación del estado inicial de una organización, antes de comenzar un programa de mejoras de procesos.

Idea a defender:

El desarrollo de una herramienta con una interfaz gráfica amigable, que exprese si la organización está lista para iniciar un programa de mejoras, contribuirá al logro del éxito de los proyectos de mejora de procesos.

Tareas de la investigación

1. Elaboración del marco teórico de la investigación.

2. Investigación de las tecnologías que más se utilizan en el mundo, para llevar a cabo aplicaciones como la que se pretende desarrollar.
3. Selección de la metodología de análisis y diseño de sistemas informáticos, que facilite la creación y garantice la calidad de la aplicación.
4. Aplicación de la metodología, con todos sus flujos de trabajo en el proceso de desarrollo del sistema.
5. Implementación de la herramienta informática que a partir de los indicadores y factores ya definidos, exprese a las organizaciones su estado inicial, antes de comenzar un programa de mejoras de procesos.

Población:

Todas las herramientas y sistemas de mejoras de procesos que utilizan el razonamiento basado en casos.

Unidad de Estudio:

Herramientas que utilizan el razonamiento basado en casos.

Métodos:

El método científico de investigación es la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones. El mismo se puede clasificar en teóricos y empíricos, aunque estos se encuentran dialécticamente relacionados.

En esta investigación se utiliza como método teórico: el método Analítico-Sintético, puesto que se busca la esencia de los fenómenos, y los rasgos que lo caracterizan y distinguen. En la misma se analizan las distintas teorías y documentos, realizándose una extracción de los elementos más importantes que se relacionan con el objeto de estudio.

Otro método teórico utilizado es la modelación, ya que en la investigación se realiza una propuesta con el fin de dar solución al problema científico planteado.

Como método empírico se utiliza la observación, puesto que se identifica y evalúa la situación real que existe respecto al tema. Otro método empírico utilizado es la entrevista, a través de la cual se obtiene información necesaria para desarrollar el tema investigativo.

Análisis financiero

La realización de este trabajo, trae consigo una serie de gastos, desde el punto de vista de cantidad de recursos, cómo tiempo de trabajo. Para el desarrollo del mismo se necesita de un tiempo determinado, extendiéndose éste hasta que se logre obtener los resultados prácticos. Otros gastos que produce dicho trabajo son la utilización de materiales docentes (hojas, lápices, plumas, etc.) y locales destinados a esta actividad (departamentos y laboratorios) Esto puede incurrir en otros gastos por consumo de electricidad, en el uso de los equipamientos tecnológicos (computadora, impresora, etc.) Los resultados esperados brindarán aportes al desarrollo de la industria del *software*. Si se logra implementar una herramienta de apoyo para identificar si la organización está lista para iniciar un programa de mejora, aumentará en gran medida la calidad de los procesos de *software*. La utilización de la herramienta que se obtendrán como resultado de la investigación, conllevará al beneficio de las empresas desarrolladoras de *software* en el mundo y en nuestro país específicamente. Esto puede conllevar al beneficio económico de la UCI y del país, pues los productos de *software* serán terminados con una alta calidad, contribuyendo a la eficiencia y sostenibilidad de todo el proceso. Como se puede apreciar existe cierto costo en el desarrollo de la investigación, pero estos no le restan importancia a la misma, ni disminuyen u opacan los beneficios esperados con su realización.

El trabajo está estructurado en cuatro capítulos los cuales se explican brevemente a continuación.

Capítulo 1: Fundamentación Teórica.

El capítulo abordará todo lo referente al estudio del estado del arte en el que está enmarcada la investigación. Se describen los procesos fundamentales referentes a los programas de mejoras de procesos así como las herramientas y tecnologías a utilizar.

Capítulo 2: Característica del sistema

El capítulo enmarca la definición de los requerimientos necesarios para el cumplimiento de todas las funcionalidades del sistema, estableciendo la base para su diseño, además demostrar según el método estadístico Análisis por Puntos de Casos de Uso, la viabilidad y factibilidad del sistema propuesto.

Capítulo 3: Análisis y Diseño

El capítulo se centra en el análisis detallado del funcionamiento interno de cada uno de los procesos identificados en el capítulo 2, así como el diseño de la interacción de las clases de cada proceso para demostrar técnicamente cómo trabaja cada uno de ellos.

Capitulo 4: Implementación y Prueba

En este capítulo se muestra el diagrama de componentes, además del modelo de implementación. Se da a conocer la solución resultante del desarrollo del sistema implementado. Además se realizan las pruebas al módulo de autos para evitar la aparición de errores.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En la actualidad en las grandes empresas desarrolladoras de *software* ha aumentado en gran medida la calidad y complejidad con que se desarrollan dichos productos, por lo que resulta difícil crear productos que cumplan totalmente con las expectativas de los clientes. En este capítulo se exponen conceptos e ideas que son necesarias para el desarrollo de la investigación y se da a conocer cuáles son las causas del éxito o del fracaso de los programas de mejoras de procesos en el mundo. Además, se realiza un estudio de diferentes modelos de mejoras de procesos y herramientas para seleccionar las más acordes para llevar acabo la investigación, haciendo énfasis en el uso de la Inteligencia Artificial. También se realiza una comparación entre las metodologías de desarrollo que más se utilizan en la actualidad para seleccionar la más adecuada a la investigación.

1.2 Desarrollo de la industria del software

Con la explosión de Internet, los cambios tecnológicos en las telecomunicaciones, el progresivo avance en el intercambio de información y el desarrollo de nuevas herramientas para desarrollo de *software*, ha surgido una nueva industria estratégica con enorme potencial de impacto en el desarrollo industrial y comercial, la industria del *software.*

En la misma las tecnologías de la información y la comunicación (TIC) desempeñan un papel fundamental debido a un conjunto de factores, como: la expansión acelerada y los cambios revolucionarios en el sistema de telecomunicaciones en el mundo, los procesos involucrados en el desarrollo de la red de redes internet, el actual crecimiento de los computadores personales y la actual demanda de programas de computo especializados. Todos estos elementos están estrechamente relacionados al desarrollo y uso creciente de una tecnología multifuncional: el *software*, el cual es un elemento dual de servicio y producto, intangible e indispensable para la realización de muchas actividades. [1]

El *software* desempeña un papel fundamental en la reconfiguración de las nuevas empresas, es necesario para el buen funcionamiento de los equipos de *hardware* y se incorpora a productos de uso cotidiano.

Por su gran dinamismo económico, genera nuevas áreas y crea nuevas oportunidades de empleo. Este generó una nueva industria importante, cuyos campos significativos son la ingeniería de *software* y los servicios informáticos, que tienen una estructura compleja y gran capacidad de innovación.

La industria del *software* es la industria de más alto crecimiento y de mayor significación en el sector Electrónico – Informático en el ámbito internacional. [2]

Dicha industria es muy importante ya que desempeña un papel cada vez más significativo en la dinámica de las economías modernas, debido a que su expansión está estrechamente asociada a la introducción de nuevas tecnologías de la información y las comunicaciones, las cuales están redefiniendo de manera acelerada las formas de producir, vender y competir en prácticamente todos los sectores productores de bienes de servicios.

1.3 Proceso de desarrollo de software

Un proceso de desarrollo de *software* es un método de organizar las actividades relacionadas con la creación, presentación y mantenimiento de los sistemas de *software* [3].

Un proceso de desarrollo de *software* es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto [4].

El proceso de desarrollo de *software* es extremadamente importante porque tiene como propósito la producción eficaz y eficiente de un producto *software* que reúna los requisitos del cliente. Así el resultado final del *software* será el más óptimo para los proyectos productivos.

A lo largo de muchos años el desarrollo de *software* se ha realizado en función de la experiencia de aquellos que lo han conducido y del equipo de trabajo que se ha utilizado. Sin embargo, con los años las aplicaciones han crecido en tamaño, complejidad e importancia para la empresa que contrata el desarrollo, por lo que la necesidad de encontrar un modelo conductor para soportar el desarrollo de las aplicaciones es cada vez más patente. En la actualidad casi todas las tareas realizadas en una empresa se llevan a cabo a través de un *software,* por lo que en la industria del *software,* todas las compañías buscan desarrollar un producto con mayor calidad y eficiencia, en un tiempo más corto y reduciendo su costo de producción. Para estas empresas productoras de *software* tiene gran

importancia un buen desarrollo del proceso de producción, lo cual llevaría a la obtención de un producto con más calidad, cumpliendo con las expectativas de sus clientes y propiciando su satisfacción con el producto. El buen desarrollo de *software* trae para las empresas un incremento notable de sus ganancias.

1.4 Programas de Mejoras de Procesos

En la actualidad sin duda alguna, las necesidades y tendencias externas determinan los cambios en las organizaciones, cambios de toda índole y tamaño. Desde cambiar un poco la forma de hacer las cosas hasta realizar cambios que implican un programa formal. Dando pie al surgimiento de un programa de mejora, cuyos objetivos son el incremento de calidad, disminución de costos de producción, aumento de competitividad en el mercado, entre otros aspectos.

Las actividades de mejora de procesos y aumento de la eficiencia guían a una organización a la obtención y creación de valor a través de la mejora y control de los costos; racionalización de los procesos y el control y gestión de todas las actividades de la compañía en su ciclo de vida operativo [5].

La mejora de procesos se puede definir como la planificación de un esfuerzo en busca de mejorar los procesos de desarrollo para obtener productos con una mayor calidad y eficiencia. La misma involucra aspectos de ingeniería y administración, tanto a nivel de proyectos, como de la organización [5].

Los objetivos fundamentales perseguidos por el proceso de mejoras son:

- **Aumentar la madurez de los procesos:** grado en que están definidos, administrados, medidos y aplicados.
- **Aumentar la capacidad de los procesos:** representa los resultados esperados a partir de ser aplicado dicho proceso de mejoras.

A partir de principios de los años noventa la comunidad de Ingeniería del *Software* (industria e investigadores) ha expresado especial interés en la mejora de procesos de *software* (conocida por las siglas inglesas SPI, *Mejoras de procesos de software*). Esto está evidenciado por el creciente número de artículos que abordan sobre el tema, así como por la aparición de un gran número de estándares relacionados con SPI creados por organizaciones internacionales como el Instituto de ingeniera de *software* (SEI) e ISO. [6]

La mejora de procesos es una técnica imprescindible para el éxito en los negocios de toda organización o institución, constituyendo la clave fundamental para sobrevivir en tiempos que están marcados por los cambios.

Es importante como la mejora de proceso intenta cambiar la forma en que las personas realizan las actividades, para así satisfacer los objetivos establecidos en los principales indicadores de rendimiento tales como la rentabilidad, la satisfacción de los clientes y la utilización de recursos. Tanto si siguen una metodología formal como si implementan una iniciativa interna.

1.5 Programa de mejoras de procesos en la UCI

La Universidad de las Ciencias Informáticas (UCI) es un centro productivo, cuya misión es producir *software* y servicios informáticos a partir de la vinculación estudio-trabajo como modelo de formación. Es considerada la mayor organización productora de *software* en el país.

En la actualidad el centro está acometiendo un programa de mejora de sus procesos basado en el modelo CMMI (Modelo de Madurez de Capacidad de Integración) bajo los servicios del SIE Center (Industria del *software* Centro de Excelencia) del Tecnológico de Monterrey, México.

El proceso de mejora está encaminado a que la universidad alcance en el 2010 una certificación internacional del nivel 2 del modelo CMMI. Hecho que la convertiría en la primera empresa cubana certificada con este modelo y una de las pocas en el área del Caribe y Centro América.

CMMI es un modelo de referencia para el crecimiento de capacidades y madurez, que se enfoca tanto en procesos de administración como de ingeniería de sistemas y *software*. Con su instauración se espera alcanzar beneficios como:

- Calendarios y presupuestos predecibles en los proyectos.
- Mejora del ciclo de vida dentro del desarrollo de *software*.
- Mayor productividad.
- Mayor calidad de los productos y servicios que ofrece la universidad a sus clientes y por ende la satisfacción de los mismos.
- Mejorar la moral del personal que labora en el centro.

El servicio que ofrece el SIE Center entre otras cosas ayuda a la UCI a revisar su estrategia de mejora de procesos de *software*, para asegurar que su organización está basada en procesos y con un programa de mejora continua alineado con sus objetivos de negocio. Ayuda a la UCI a establecer las bases y fundamentos para seguir mejorando sus procesos y fortalece su cultura de calidad en el desarrollo de *software*. Además ayuda a alinear los procesos de desarrollo de *software* con los principios y requisitos del modelo CMMI, estableciendo planes de mejora con los que la organización oriente sus procesos hacia la consecución de sus metas.

CMMI, como modelo de mejora de procesos tiene dos utilidades; puede servir tanto como guía para la mejora en una organización, o como criterio para evaluar su nivel. De ahí que presente dos representaciones: una continua que permite a la organización seleccionar un área de proceso específica para hacerle una mejora usando niveles de capacidad para caracterizar dicha mejora; y la representación escalonada o por etapas, que usa un conjunto predefinido de áreas de procesos para definir un camino hacia la mejora de una organización, usando niveles de madurez, dentro de ellos el nivel dos, conocido como nivel gestionado o administrado, definido a través de siete áreas de procesos:

- Administración de Requisitos (REQM)
- Administración de la Configuración (CM)
- Aseguramiento de la Calidad de los Procesos y Productos (PPQA)
- Medición y Análisis (MA)
- Monitoreo y Control de Proyecto (PMC)
- Planificación y Control de Proyecto (PP)
- Administración de Acuerdos con Proveedores (SAM)

A continuación se presentan estadísticas de los Resultados de la evaluación de los pilotos realizadas en el mes de octubre.

Es válido aclarar que para todas las áreas de procesos que se relacionan a continuación, están completamente definidos sus procesos. Estos resultados que se muestran se refieren a la evaluación de los pilotos y dependen fundamentalmente del desempeño de los proyectos que se encuentran en el alcance del programa de mejoras.

Igualmente, las áreas de procesos no han estado trabajando el mismo tiempo. En la tabla se encuentran organizadas según el orden en que los Grupo Técnico de Trabajo (TWG) de cada área de proceso han comenzado a trabajar dentro del programa de mejoras.

AP	% **CI**	% **AI**	% **PI**	% **NI**
REQM	33	27	33	7
PPQA	21	58	21	0
PP	0	29	46	25
PMC	0	15	20	65
MA	6	17	22	55

Leyenda:

AP: Área de Procesos

CI: Completamente implementado.

AI: Ampliamente implementado.

PI: Parcialmente implementado.

NI: No implementado.

REQM: Administración de Requerimientos.

PPQA: Aseguramiento de la Calidad del

Producto y del Proceso.

PP: Planeación del Proyecto.

PMC: Monitoreo y Control del Proyecto.

MA: Medición y Análisis.

Figura 1: Esto fue realizado en los Laboratorio Industrial de Pruebas de *Software* (LIPS) [7]

En la universidad existen un gran avance en los programas de mejoras de proceso se espera poder ser representado a nivel mundial como una organización certificada internacionalmente del nivel 2 del modelo CMMI antes de terminar el 2010. Por esta causa se necesita de una herramienta capaz de evaluar si estos procesos son óptimos, y al final si diera resultado, aplicarlos en los proyectos productivos de la universidad, cosa que se desea lograr con la herramienta de apoyo.

1.6 Importancia de una Empresa al iniciar un programa de mejoras de procesos.

Estudios realizados en países latinoamericanos, reportan que en la aplicación de programas de mejoras, se interponen algunos obstáculos y dificultades, los cuales provocan de alguna manera, el fracaso o abandono del programa, entre ellos están:

• Carencia de recursos y presupuesto
– Disponibilidad del personal
• Falta de cultura de procesos en los clientes y usuario
• Pobre compromiso por parte de la dirección
– Inconsistencia en la recopilación, preparación y análisis de la información que se le presentaba
– Poca sensibilidad con las necesidades de los procesos y los recursos

• Procesos complejos
• Falta de capacitación en el uso de procesos y herramientas
• Falta de comunicación
– Desconocimiento de las metas y objetivos del proyecto
• Desconocimiento de los modelos de referencia para la mejora
• Resistencia al cambio

Se hace necesario por tanto establecer los fundamentos básicos para garantizar y dar soporte a la iniciativa de mejoras de procesos. Deben establecerse los objetivos de la empresa y del proyecto de mejora y garantizar la disponibilidad de recursos, la infraestructura y la priorización del proyecto de mejoramiento, apoyándose para todo esto en algún modelo de mejora de procesos.

Muchas de las grandes empresas desarrolladoras de *software* en el mundo han tenido en cuenta estos factores, con el fin de iniciarse en la mejora de procesos cuando realmente están preparadas para ello. La aplicación de los diferentes modelos de mejora y evaluación y la utilización de herramientas que automatizan el análisis de los factores mencionados, han contribuido directamente en el cumplimiento de sus objetivos y metas.

Hay casos en que no se evalúan ni se tienen en cuenta los elementos mencionados anteriormente, esto se debe fundamentalmente al desconocimiento de los directivos de esas organizaciones, quienes conciben al programa de mejora como un proceso inmediato, en el cual se obtienen resultados a corto plazo, sin tener en cuenta que todo ese proyecto lleva tiempo de dedicación y trabajo para obtener los beneficios a un largo plazo.

Las estadísticas del 2008 de CHAOS arrojan que:

- 76% de las organizaciones reportan pocos o ningún éxito en la mejora de procesos
- 85% de las organizaciones abandonan sus esfuerzos de mejora en los 3 primeros años
- 60% de las que abandonan los reinician en el futuro
- Solo el 2% de las organizaciones que logran éxito en los procesos de mejora reportan sus datos de mejora

Como se puede apreciar, muchas organizaciones en el mundo han fracasado en el intento de mejorar sus procesos de producción. Si la empresa estuviera segura de que está lista para iniciarse en un programa de mejoras de procesos, se evitaría correr el riesgo de colapsar o fracasar en el desarrollo del proyecto de mejoras, evitando la ocurrencia de grandes pérdidas económicas, descontento y resistencia a los cambios por parte del personal. El fracaso total en un programa de mejoras, pudiera conllevar también a la quiebra total de una organización.

Actualmente en la UCI se está desarrollando un procedimiento mediante el cual se podrá conocer si una organización esta lista o no para dar inicio a un programa de mejoras, evaluando una serie de criterios y parámetros previamente definidos. Con este procedimiento se pretende evitar que las empresas que decidan iniciar un programa de mejoras de procesos fracasen por no estar listas. Este procedimiento necesita apoyarse en una herramienta con interfaz gráfica amigable que permita mostrar si dicha organización está lista para iniciar el programa de mejoras. En el mismo se aplica la técnica de inteligencia artificial: razonamiento basado en casos, que se viene utilizando ya hace varios años, lo cual permite evaluar una situación y dar posibles soluciones dados diferentes casos ya existentes.

La automatización de este procedimiento proporcionará grandes ventajas a la organización, puesto que su uso se hace menos complejo, sin tener necesidad de tener grandes conocimientos informáticos para su utilización. Además, puede ser utilizado por cualquier empresa u organización que quiera evaluar su estado para decidirse a iniciar un programa de mejoras de procesos.

Este trabajo de diploma tiene como objetivo automatizar el procedimiento definido, creando una herramienta con una interfaz gráfica lo más amigable que pueda ser para el usuario, tratando de lograr que su utilización sea sencilla sin tener que realizar muchas acciones.

1.7 Herramientas Informáticas para apoyar la propuesta de solución

Es muy importante resaltar que la selección de las tecnologías a utilizar en dicha investigación se basa en las necesidades específicas de cada situación. Por lo que para decir que una tecnología es mejor que otra hay que hacer un análisis muy detallado, y después de terminar dicho análisis escoger la que mejor se adapte a las necesidades de la investigación.

1.7.1 Sistemas expertos similares

Desde la aparición de las computadoras hasta nuestros días, la gente ha invertido grandes esfuerzos por tratar de dar una cierta capacidad de decisión a estas máquinas, incluso un cierto grado de inteligencia.

Un sistema experto en sí no tiene verdadera inteligencia artificial; más bien, es un sistema basado en el conocimiento que, mediante el buen diseño de su base de información y un adecuado motor de inferencias para manipular dichos datos proporciona una manera de determinar resoluciones finales dados ciertos criterios.

Sistema Experto (SE): es básicamente un programa de computadora basado en conocimientos y raciocinio que lleva a cabo tareas que generalmente sólo realiza un experto humano; es decir, es un programa que imita el comportamiento humano en el sentido de que utiliza la información que le es proporcionada para poder dar una opinión sobre un tema en especial.

Otros autores lo definen como sigue: un Sistema Experto es un programa de computadora interactivo que contiene la experiencia, conocimiento y habilidad propia de una persona o grupos de personas especialistas en un área particular del conocimiento humano, de manera que permitan resolver problemas específicos de ése área de manera inteligente y satisfactoria.

Sin embargo, con los avances conseguidos hasta ahora esta definición ha cambiado, actualmente un (SE) se define de la siguiente manera:

Un SE es un sistema informático que simula los procesos de aprendizaje, memorización, razonamiento, comunicación y acción de un experto humano en una determinada rama de la ciencia, suministrando, de esta forma, un consultor que puede sustituirle con unas ciertas garantías de éxito. Así pues el razonamiento basado en casos de uso o *case base reasoning* (CBR) es un tipo de sistema experto. [8]

Razonamiento basado en caso:

El Razonamiento Basado en Casos (RBC), es un enfoque que aborda nuevos problemas tomando como referencia problemas similares resueltos en el pasado. De modo que problemas similares tienen soluciones similares, y la similitud desempeña un papel esencial. Sus componentes fundamentales son la base de casos, el módulo de recuperación de casos y el módulo de adaptación de las soluciones. La figura 3 muestra el ciclo de vida de un Sistema Basado en Casos.

Base de Casos (BC)

La BC contiene las experiencias, ejemplos o casos a partir de los cuales el sistema hace sus inferencias. Esta base puede ser generada a partir de casos o ejemplos resultantes del trabajo de expertos humanos o por un procedimiento automático o semiautomático que construye los casos desde datos existentes registrados, por ejemplo, en una base de datos.

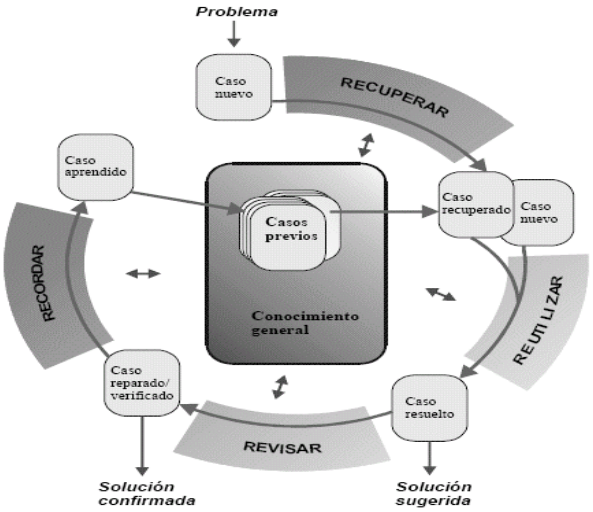

Figura 2: Ciclo de vida de un Sistema Basado en Casos

Módulo de recuperación.

En este módulo se recuperan de la Base de Casos los casos más semejantes al problema. No existe una medida de semejanza única, general, para cualquier dominio, de ahí que la eficiencia del sistema radica en la función de semejanza que se defina.

Módulo de adaptación.

Después de la determinación de los casos más semejantes, las soluciones contenidas en dichos casos pueden usarse directamente como solución al nuevo problema, pero comúnmente necesitan ser modificadas. El enfoque que utilizan los Sistemas Basado en Casos (SBC) para la adquisición de conocimiento es una de las ventajas que se le acreditan a este tipo de sistemas; pues razonan desde episodios específicos, lo cual evita el problema de descomponer el conocimiento del dominio y generalizarlo en reglas.

Otras de las ventajas de los SBC están fundamentadas; en la flexibilidad para representar el conocimiento a través de los casos, la organización de la BC y de las estrategias de recuperación y adaptación de los casos y que el usuario puede ser capaz de agregar nuevos casos a la BC sin la intervención experta.

Ventajas lo son también, el rehúso de las soluciones previas al resolver un problema, y el almacenar casos que resulto un fracaso, lo que permite advertir sobre problemas potenciales a evitar. Así como también poder fundamentar las soluciones derivadas a partir de casos reales.

Las limitantes de los SBC están en la definición de la función de semejanza y en lo difícil que resulta encontrar una estructura apropiada para describir el contenido de un caso y decidir como la memoria de casos debe ser organizada e indexada para un almacenamiento, recuperación y rehusó efectivo.[9]

Entre los sistemas expertos más destacados se puede mencionar algunos como son:

Dendral: fue el primer sistema experto que se utilizó a los problemas más reales, fue desarrollado por Edward Feigenbaum y otros programadores en la Universidad de Stanford, a mediados de los años 60, y su desarrollo duró diez años, (1965 a 1975).

Es un sistema experto que permite interpretar la estructura de las moléculas, a través de un proceso de búsqueda de generación y prueba jerárquica que se divide en tres partes fundamentales: plan, generación y prueba. Su base de conocimiento se desglosa en dos conjuntos de reglas correspondientes a cada una de las fases de desarrollo del sistema.

Mycin: es un sistema experto desarrollado a principios de los años 70 por Edgar ShortLiffe, en la Universidad de Stanford. Fue escrito en Lisp, e inicialmente estaba inspirado en Dendral, otro sistema experto que tuvo cierto éxito a finales de los años 60.

Su principal función consistía en el diagnóstico de enfermedades infecciosas de la sangre; además, Mycin era capaz de "razonar" el proceso seguido para llegar a estos diagnósticos, y de recetar medicaciones personalizadas a cada paciente (según su estatura, peso, etc.).

Su funcionamiento se basaba principalmente en un sencillo motor de inferencia, que manejaba una base de conocimiento de aproximadamente unas 500 reglas. El programa capturaba las entradas a partir de una serie de preguntas (como por ejemplo, ¿Tiene el paciente molestia en el pecho?, o ¿Ha sido operado el paciente anteriormente?), que usualmente respondía el médico del paciente. Tras este proceso, Mycin mostraba la salida por pantalla, que consistía en una serie de posibles enfermedades (ordenadas por su probabilidad asociada), la explicación del porqué de cada uno de estos diagnósticos, y una serie de recomendaciones sobre el tratamiento a seguir por el paciente.

Para calcular la probabilidad de cada uno de los resultados, los autores desarrollaron una técnica empírica basada en factores de certeza. Estos factores de certeza se calculaban de tal manera que en función de unas evidencias se asigna a la hipótesis un factor de certeza.

SISI (Sistema Inteligente de Selección de Información): que es un programa desarrollado por un grupo de investigadores de la Universidad de Las Villas, a partir de la versión 3.0 de *Borland Delphi.* Este programa es una variante del Shell diseñado por Stanfill y Waltz en 1986, y se ejecuta sobre el sistema operativo Windows 95. Su efectividad ha sido ampliamente reconocida, presenta una interfaz amigable para el usuario, quien sólo necesita conocimientos mínimos para el trabajo con el mismo, además de que ya existen diferentes expertos que lo utilizan, como son Agudo2.cbe y el *Psiq.Epilepsia.exp*, creado por la Dra. P. Piñera Tapia para su trabajo de Maestría en Informática Médica. [10]

Norma de Argelio

Según la Msc Yalila Arean Rodríguez realizando su maestría en el 2003 en el Instituto Politécnico José Antonio Echeverría (CUJAE) con titulo: Una herramienta de búsqueda inteligente en BD utilizando técnicas de RBC plateo sobre la norma de Argelio que la misma lo que realiza es el cálculo de la distancia de un caso a otro .Con esta Norma se puede realizar la evaluaciones necesarias para la realización de esta herramienta, de esta forma posibilita evaluar los factores de éxito o fracaso de cualquier organización o empresa. A continuación se muestra la forma matemática que se utiliza para aplicar esta norma. [11]

$$D(x,y) = \sum_{I=1}^{u} \left(1 - \frac{1}{1 + d(X_I, Y_I)}\right)$$

$$d(X_I, Y_I) = \begin{cases} 0 & si \ X_I = Y_I \\ 1 & si \ X_I \neq Y_I \end{cases}$$

Figura3: Ecuación de la norma de Argelio.

1.7.2 Aplicaciones de escritorio.

Actualmente para la creación de *software* se utilizan las herramientas informáticas, las cuales son las encargadas de convertir una serie de conocimientos en un *software* el cual podrá ser utilizado por empresas, personas, etc. Hoy en día se utilizan diversas herramientas informáticas para crear aplicaciones de escritorio que no son más que un sistema donde la entrada del usuario, afecta el estado del negocio, y por tanto puede elaborarse utilizando la metodología de trabajo que propone RUP, y utilizando el UML para los modelos.

Dentro de las herramientas más utilizadas para la creación de aplicaciones de escritorio se puede mencionar:

NetBeans

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de *software* llamados módulos. Un módulo es un archivo Java que contiene clases de Java escritas para interactuar con las *APIs* de NetBeans y un archivo especial que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de *software*.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun Microsystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos. La plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de *software*, proporcionan extensiones adicionales que se

integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas)
- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- Administración de ventanas
- Marco de trabajo basado en asistentes (diálogos paso a paso)

El NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Es un producto libre y gratuito sin restricciones de uso.

Eclipse

Eclipse es una plataforma universal para integrar herramientas de desarrollo, con una arquitectura abierta y basada en *plug-ins*. Además, Eclipse da soporte a todo tipo de proyectos que abarcan desde el ciclo de vida del desarrollo de aplicaciones, incluyendo soporte para modelado.

- Dentro de sus principales características se encuentran:
- Editor visual con sintaxis coloreada
- Compilación incremental de código
- Modifica e inspecciona valores de variables
- Avisa de los errores cometidos mediante una ventana secundaria
- Depura código que resida en una máquina remota
- Editor de Texto
- Resaltado de sintaxis
- Compilación en tiempo real
- Pruebas unitarias con Junit
- Control de versiones con CVS
- Integración con AntAsistentes (Wizards)

- *Plug-ins*

1.7.3 Metodologías de desarrollo de software

Todo desarrollo de *software* es riesgoso y difícil por lo que en su desarrollo siempre es necesario la utilización de una metodología que vaya guiando por pasos el proceso, logrando esto que al final no existan clientes insatisfechos con sus productos y desarrolladores más insatisfechos aun. A continuación se exponen algunas de las metodologías que actualmente se utilizan en el mundo.

Microsoft Solution Framework (MSF)

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

Figura 4: Metodología MSF

MSF tiene las siguientes características:

- **Adaptable:** es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- **Escalable:** puede organizar equipos tan pequeños entre 3 ó 4 personas, así como también, proyectos que requieren 50 personas a más.
- **Flexible:** es utilizada en el ambiente de desarrollo de cualquier cliente.
- **Tecnología Agnóstica:** porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

Programación Extrema (XP)

Es una de las metodologías de desarrollo de *software* ágil más reconocidas en la actualidad, utilizada para proyectos de corto plazo y pequeño equipo de desarrollo, fue creada a mediados de la década de los 80 por Kent Beck. Esta metodología consiste en una programación ágil o extrema, cuya peculiaridad es tener como parte del equipo de desarrollo al usuario final, este es uno de los requerimientos para llegar al triunfo del proyecto. Esta organizada en 4 cuatro fases: Planificación, Diseño, Desarrollo y Pruebas.

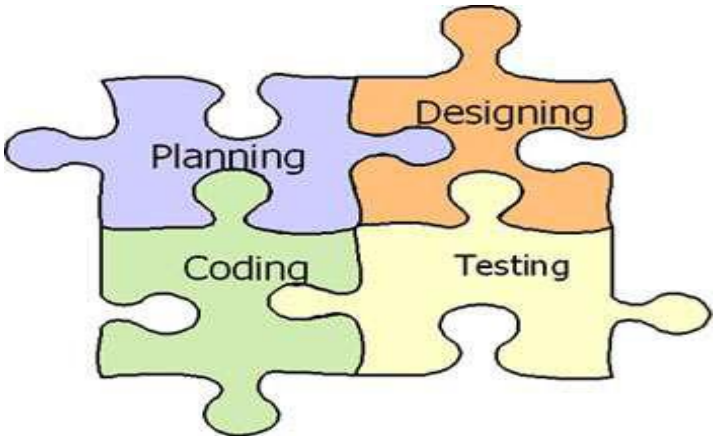

Figura 5: Metodología Programación Extrema

Posee cuatro variables principales que son:

Coste: Equipo de desarrollo, computadoras y locales.
Calidad: En el desarrollo del proyecto y en los entregables.
Tiempo: Tiempo de entrega parcial y total del proyecto.
Ámbito: Definición de problemas a resolver y cuales se dejan para futuras versiones.

Esta metodología considera como aspecto fundamental, la comunicación entre los desarrolladores y el usuario final, todos forman parte del equipo de desarrollo, tienen como objetivo primordial la simplicidad al crear y codificar los módulos del sistema, la retroalimentación constante de ideas entre el equipo de desarrollo, el cliente y los usuarios finales y la refactorización.

Algunos de los beneficios de utilizar las prácticas de la programación extrema son: programación en pares, refactorización, integración continua, pruebas de aceptación, unidad de pruebas y otras que favorecen el aprendizaje de la programación e incitan a que se realicen investigaciones que apoyen la integración de estas prácticas. [12].

Lo fundamental en este tipo de metodología es:

- La comunicación entre los usuarios y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del sistema.
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

Proceso unificado de desarrollo (RUP)

El proceso unificado de desarrollo de *software* (*Rational Unified Process - RUP*) es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes. Es una metodología robusta utilizada para grandes proyectos, es una de las más utilizadas por las grandes compañías productoras de *software*, utiliza UML como lenguaje de modelado y divide el proceso de desarrollo de *software* en cuatro fases fundamentales:

- Inicio
- Elaboración
- Construcción
- Transición

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

La metodología RUP esta dirigida por los casos de usos por lo que permite detallar cada una de las funcionalidades que se esperan del *software*, está centrado en la arquitectura del sistema y es iterativo e incremental con lo que se alcanza un desarrollo en iteraciones, en las cuales se reproduce el ciclo de vida del *software* en cascada a menor escala. Los objetivos de una iteración se elaboran en función de lo que se cumplió en las anteriores. Permite la documentación de cada uno de los pasos a seguir en el proceso de desarrollo mediante unas planillas, que genera por cada una de las disciplinas; se basa en UML (Lenguaje Unificado de Modelado) como herramienta principal.

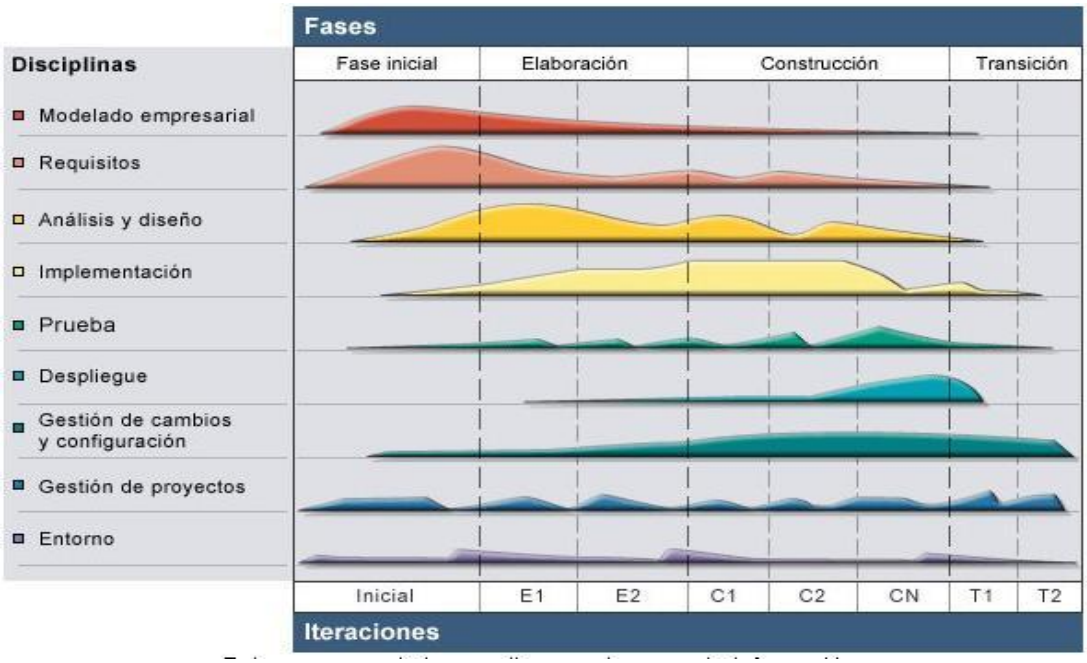

Figura 6: En esta gráfica de RUP en dos dimensiones se muestra la organización mediante flujos de trabajo

RUP se usa exitosamente en más de 1000 empresas, tiene diversos dominios de aplicación (50% *e-business*), dicha metodología de desarrollo de *software* es utilizada en proyectos grandes y cortos, formalmente o como e-coach en ingeniería de *software*. Algunos ejemplos de empresas que utilizan RUP son:

1.- Comunicaciones: Ericson, Alcatel, MCI

2- Defensa: Lockheed-Martin, British Aerospace

3- Manufactura: Xerox, Volvo, Intel

4- Finanzas: Visa, Merrill Lynch, Schwab

5- Integración de sistemas: Ernst&Young, Oracle, Deloitte&Touche

En resumen, la metodología RUP es un proceso de ingeniería de *software* y un producto de *software* para producir *software* de calidad, flexible, y en plazos y presupuestos predecibles. Incorpora las mejores prácticas de desarrollo de *software* validadas comercialmente, ayuda a alcanzar el nivel 2 de madurez del CMMI, es usado exitosamente en escenarios diversos y puede implementarse paulatinamente en una organización. Es más, no hay metodología mejor que otra ni válida al 100% para todas las personas y empresas. Esta metodología por sus características es la que mas se

ajusta para modelar la investigación. Permite una modelación global y detallada de los procesos, mediante el diagrama de casos de uso del negocio y los diagramas de actividades. Además, admite un modelado del sistema estructurado a partir de los requisitos funcionales identificados. [13]

1.8.2 Lenguajes de programación

Para el desarrollo de aplicaciones de escritorio existen diversos lenguajes, los cuales permiten desarrollar aplicaciones dinámicas, permitiendo la interacción con el usuario y la personalización de la información.

Java

Java fue diseñado como lenguaje orientado a objetos, posee una curva de aprendizaje muy rápida, muestra una colección de clases para su uso en aplicaciones de red, que permiten establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas, es compilado en la medida en que su código fuente se transforma en una especie de código maquina, fue diseñado para crear *software* altamente fiable por lo que proporciona numerosas comprobaciones en compilación y en tiempo de ejecución, soporta aplicaciones que serán ejecutadas en los más variados entornos de red, sobre arquitecturas distintas y con sistemas diversos, especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.

El lenguaje Java puede ser usado para crear dos tipos de programas: aplicaciones independientes y *applets*.

Las aplicaciones independientes se comportan como cualquier otro programa escrito en cualquier lenguaje.

Por su parte, las *applets* son pequeños programas que aparecen embebidos en las páginas web, como aparecen los gráficos o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer conexiones de red, presentar menús y cuadros de diálogo para luego emprender acciones. [14]

C++

Es uno de los lenguajes más cercanos al código de máquina, este es capaz de proveer los recursos de los lenguajes de alto nivel, como la programación orientada a objeto.

Este es un lenguaje imperativo orientado a objetos derivado del C. En realidad un súper conjunto de C, que nació para añadirle cualidades y características de las que carecía. Debido a que su ancestro, sigue muy ligado al *hardware* subyacente, manteniendo una considerable potencia para programación a bajo nivel, se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción.

Él provee muchos mecanismos y algoritmos con niveles de atracción muy similares a C# y Java, no todo se tiene que hacerse de 0, en la actualidad se ha desarrollado un gran número de librerías estándar que facilitan el trabajo con estructuras de datos de alta complejidad, manipulación de cadenas, así como interfaz gráfica de usuario por citarse algunos ejemplos. [15]

C# (C-Sharp)

Es un lenguaje orientado a objetos, elegante y con seguridad de tipos que permite a los desarrolladores crear una amplia gama de aplicaciones sólidas y seguras que se ejecutan en .NET marco de trabajo. Puede utilizar este lenguaje para crear aplicaciones cliente para Windows tradicionales, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos, y muchas tareas más.

C# es un lenguaje orientado a objetos, elegante y con seguridad de tipos que permite a los desarrolladores crear una amplia gama de aplicaciones sólidas y seguras que se ejecutan en .NET marcos de trabajo. Puede utilizar este lenguaje para crear aplicaciones cliente para Windows tradicionales, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos, y muchas tareas más.

Además de estos principios básicos orientados a objetos, C# facilita el desarrollo de componentes de *software* a través de varias construcciones de lenguaje innovadoras, entre las que se incluyen:

- Firmas de métodos encapsulados denominadas delegados, que permiten notificaciones de eventos con seguridad de tipos.
- Propiedades, que actúan como descriptores de acceso para variables miembro privadas.
- Atributos, que proporcionan metadatos declarativos sobre tipos en tiempo de ejecución.

- Comentarios en línea de documentación XML. [16]

1.8 Lenguaje de modelado

El UML nació en 1994 por iniciativa de Grady Booch y Jim Rumbaugh, para combinar sus dos famosos métodos, el de Booch y el OMT (Tecnología de Modelado de Objetos) y más tarde se le unió Ivar Jacobson creador del OOSE (Ingeniería de *Software* Orientada a Objetos).

En la actualidad el lenguaje de modelado que más se utiliza es UML (Lenguaje Unificado de Modelado).

UML es lenguaje para visualizar, especificar, construir y documentar los artefactos de sistema que involucra una gran cantidad de *software*. Además, permite la modelación de sistemas con tecnologías orientada a objetos. UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas, ya que es un lenguaje, cuenta con reglas para combinar tales elementos.

La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. El modelo UML de un sistema es similar a un modelo a escala de un edificio junto con la interpretación del artista del edificio. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

A continuación se describirán brevemente los diagramas más comunes del UML y los conceptos que representan.

El modelo gráfico de UML tiene un vocabulario en el que se identifican:

1. **Elementos** (abstracciones que constituyen los bloques básicos de construcción).

    - **Estructurales:** Partes que representan cosas.
    - **Clase:** Conjunto de objetos que comparten atributos, operaciones, relaciones y semántica.
    - **Colaboración:** De fine la interacción entre los elementos que proporcionan un comportamiento cooperativo mayor que la suma de los comportamientos de sus elementos.
    - **Casos de Uso:** Conjunto de secuencia de acciones que un sistema ejecuta y que produce un resultado observable para un actor.
    - **Clase Activa:** Clase cuyos objetos tienen uno o más procesos o hilos de ejecución.

- **Nodo:** elemento físico que dispone de memoria y con frecuencia capacidad de almacenamiento.
- **Componente:** Es una parte física y reemplazable de un sistema que conforman un conjunto de interfaces y proporciona la implementación de dicho conjunto.
- **Comportamiento:** Partes del modelo que representan el comportamiento en el tiempo y el espacio.
- **Interacción:** Conjunto de mensajes intercambiados entre un conjunto de objetos para alcanzar un propósito específico.
- **Máquina de estado:** Especifica las secuencias de estados por las que pasa un objeto o una interacción durante su vida.
- **Agrupamiento:** Cajas en las cuales puede descomponerse un modelo.
- **Paquete:** Mecanismo de propósito general para organizar elementos en grupos.
- **Anotación:** Comentarios que se pueden aplicar para describir, clarificar y hacer observaciones sobre cualquier elemento de un modelo.

2. **Relaciones**: Ligan los elementos.
   - **Dependencia**: relación semántica que indica que un cambio en un elemento afecta a la semántica de otro elemento.
   - **Asociación:** Relación estructural que describe las conexiones entre objetos.
   - **Generalización/Especialización:** Relación en la que el hijo comparte la estructura y el comportamiento del padre.
   - **Realización:** Relación semántica entre clasificadores, en donde un clasificador especifica un contrato que otro clasificador garantiza que cumplirá.

3. **Diagramas:** Es la representación gráfica de un conjunto de elementos. Visualizan un sistema desde diferentes perspectivas.

   Diagramas de estructura estática: Describen las propiedades estructurales del sistema.
   - **Diagrama de clases**: Conjunto de clases, interfaces y colaboraciones; así como sus colaboraciones.
   - **Diagrama de objetos:** Conjunto de objetos y sus relaciones.
   - **Diagrama de casos de uso:** Conjunto de casos de uso y actores y sus relaciones.

   Diagramas de comportamiento:

- **Diagramas de interacción** (secuencia y colaboración): Objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos.
- **Diagrama de estados:** Muestra una máquina de estado que consta de estados, transiciones, eventos y actividades.
- **Diagrama de actividad:** Es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.

Diagramas de implementación:

- **Diagrama de componentes:** Organización y las dependencias entre un conjunto de componentes. Diagrama de despliegue: Configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

Por todo lo anteriormente mencionado y teniendo en cuenta además, que para el desarrollo de la aplicación se va a seguir el paradigma orientado a objeto, Se ha decidido que el lenguaje de modelado a utilizar sea el UML.

1.8.1 Herramientas de modelado

Debido al gran avance que existe en la tecnología en este momento se han podido resolver muchos detalles que en años pasados retrasaban el desarrollo de un *software*. Entre estas tecnologías actuales que ayudan considerablemente al desarrollo de un *software* se encuentran las herramientas de modelado, las mismas constituyen el medio donde se modela el sistema que se desea, guiándose por una metodología y utilizando algún lenguaje de modelado. Se puede mencionar algunas de las más utilizadas hoy en la actualidad en el mundo como:

Rational Rose

Rational Rose Enterprise Edition es una herramienta CASE, importante para el Modelado Visual mediante UML de sistemas *software*. Permite especificar, analizar, diseñar el sistema antes de codificarlo. Mantiene la consistencia de los modelos del sistema *software*. Chequeo de la sintaxis UML. Generación de documentación automáticamente. Generación de Código a partir de los Modelos. Ingeniería Inversa (crear modelo a partir código); entre otras funcionalidades, facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de casos de uso, vista lógica, vista de componentes y vista de despliegue). Permite que los arquitectos y diseñadores practiquen el desarrollo orientado al modelado, permitiéndoles producir modelos independientes a la, arquitectura de la plataforma del *software* y necesidades del negocio. [17]

Visual Paradigm

Visual Paradigm es una herramienta UML CASE considerada muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Posee licencia gratuita y comercial. Es un producto de calidad, soporta aplicaciones web, se puede encontrar en varios idiomas, permite la generación de código para Java y exportación como HTML, es fácil de instalar y actualizar y posee compatibilidad entre ediciones.

Ofrece:

- Diseño centrado en casos de uso y enfocado al negocio que genera un *software* de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales *IDEs.*

Después del estudio de ambas herramientas se decidió utilizar como herramienta para el modelado, el *Visual Paradigm*. Debido a que la Universidad hoy en día está llevando una migración de todos sus sistemas a *software* libre. [18]

1.8.3 Gestores de base datos

Bases de Datos

Conjunto de datos persistentes, interrelacionados entre si de forma lógica, que representa una situación del mundo real.

Base de Datos Relacional

Una base de datos relacional es un conjunto de una o más tablas estructuradas en registros (líneas) y campos (columnas), que se vinculan entre sí por un campo en común, lo que permite velocidad y flexibilidad.

Gestor de Base de Datos

Se le denomina gestor de base de datos al tipo de *software* específico que se dedica a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición y manipulación de datos, además de un lenguaje de consulta. Su propósito general es el de manejar de manera clara, sencilla y ordenada un conjunto de información.

Es difícil pensar un una aplicación tanto de escritorio como web que no este conectada a una base de datos. Y es que precisamente los sistemas de base de datos brindan una serie de ventajas y funcionalidades a las aplicaciones.

Existen diversas formas de manejar dichas bases de datos: con gestores como Oracle, SQL Sever, MySQL, PostgreSQL entre otros.

Oracle

Oracle es básicamente una herramienta cliente/servidor para la gestión de bases de datos. Es un producto vendido en el mundo, aunque la gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes y multinacionales.

Es el mayor y más usado Sistema Manejador de Base de Datos Relacional (RDBMS) en el mundo. La corporación Oracle lo ofrece como un producto incorporado a la línea de producción. Además, incluye cuatro generaciones de desarrollo de aplicación, herramientas de reportes y utilitarios.

Garantiza la autenticidad apropiada de los usuarios y la privacidad e integridad de los datos, permite manejar la asignación de privilegios, monitorear las operaciones de la base de datos a lo largo de toda la empresa y su arquitectura ofrece escalabilidad para soportar un gran número de usuarios y cargas de trabajo de alto volumen de transacciones.

MySQL.

Gestor de base de datos muy popular, de código abierto, confiable, poderoso y multiplataforma. Debido a su gran rapidez, facilidad de uso, infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, es que alcanza gran aceptación en la comunidad de programadores, convirtiéndolo en uno de los gestores más utilizados.

Aprovecha la potencia de sistemas multiprocesadores, gracias a su implementación multihilo.

Soporta gran cantidad de tipos de datos para las columnas.

Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc.).

Gran portabilidad entre sistemas.

Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.

PostgreSQL.

Es un sistema gestor de bases de datos. Liberado bajo la licencia BSD, que posee una serie de características positivas respecto a otros como:

- Gran escalabilidad. Es ajustable al número de procesadores y a la cantidad de memoria que posee el sistema de la forma óptima, por este motivo es capaz de soportar una mayor cantidad de peticiones simultáneas. Teniendo en cuenta esto, es vital en la universidad, ya que no se requiere de una gran computadora para trabajar con él.

- Tiene la capacidad de almacenar procedimientos en la propia base de datos.

- Multiusuario, multiprogramado, con arquitectura cliente-servidor y control de privilegios de acceso.

- Los tipos internos han sido mejorados, incluyendo nuevos tipos de fecha/hora de rango amplio y soporte para tipos geométricos adicionales.

- La velocidad del código del motor de datos ha sido incrementada aproximadamente en un 20% - 40%, y su tiempo de arranque ha bajado el 80% desde que la versión 6.0 fue lanzada.

- Implementa el uso de *rollback's,* subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en las que *MySQL* no podría.

- Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, como puede ser Oracle.

1.8.4 Propuesta solución

Para la realización del trabajo se utilizará como metodología de desarrollo RUP, la cual permite detallar fácilmente el desarrollo de la investigación. Como lenguaje de modelado UML que se combina fácilmente con RUP. Para el modelado se escogió el Visual Paradigm ya que es una herramienta libre y hoy en día la política seguida por la Universidad de las Ciencias Informáticas (UCI), es la de migrar hacia *software* libre. Como plataforma de desarrollo se utilizará NetBeans, la cual es una de las plataformas más utilizadas en el mundo para el desarrollo de aplicaciones en Java, como lenguaje de programación se utilizará Java por ser un lenguaje orientado a objetos, libre y soporta fácilmente conexión a múltiples gestores de base de datos. Como Gestor de base de datos se utilizara PostgreSQL. El cual es libre y se integra fácilmente con Java.

1.9 Conclusiones

El valor práctico de esta investigación recae en la importancia que tiene la aplicación, la cual facilitará a las empresas poder interactuar con una base de casos que permitan ayudarlos en el proceso de toma de decisiones, exponiendo el estado en que se encuentran las empresas, para saber si están listas o no para dar inicio a un programa de mejoras de procesos, lo cual evitaría el fracaso total, perdidas económicas, descontento del personal. Además, la técnica de razonamiento de casos posibilita, que cuanto más grande sea la base de casos más acertada será la decisión que se tome.

Capítulo 2: Características del sistema

En el presente capítulo se describen detalladamente las características que debe presentar el sistema, se aborda además, una panorámica del problema en cuestión, así como el objeto de automatización y el modelo de dominio. Además, se presenta una propuesta del sistema, de la que se detallan los requisitos tanto funcionales como no funcionales a cumplir en la realización de la misma, en conjunto con la definición de actores y relaciones entre ellos, unido al del diagrama de casos de uso del sistema y las descripciones textuales de los casos de uso.

2.1 Problema

En la actualidad en ninguna institución existe una herramienta de apoyo a los programas de mejoras de procesos, la no existencia de dicha herramienta informática que exprese a las organizaciones si están listas para dar inicio a un programa de mejoras de procesos, traería como consecuencia que diferentes empresas u organizaciones se lancen a esta gran y costosa tarea con un gran porciento de fracasar. Todo esto conlleva a problemas e ineficiencias a la hora de decidir, lo que hace evidente la ausencia de una aplicación informática que gestione todos los procesos de la herramienta de apoyo.

2.2 Objeto de automatización

El objeto de automatización es el proceso de evaluación de una organización antes de comenzar un programa de mejora. En el caso de la Universidad de la Ciencias Informáticas (UCI), se refiere como organización a los proyectos productivos o Centros de Desarrollos existentes en cada facultad. Los miembros de dicho proyectos o centros, tendrán diferenciación en cuanto a la forma que interactúa con el sistema, en dependencia del papel que desempeñen. El sistema permitirá que se creen Reportes sobre las organizaciones inscritas, organizaciones con éxito y fracaso además de poder calcular la posibilidad de éxito al iniciar la mejora de procesos.

2.3 Información que se maneja

La información que se maneja es la relacionada con los proyectos productivos existentes en la universidad, y la necesidad de saber si los mismos están preparados para iniciar un programa de mejoras procesos sin causar daños económicos.

Para lograr esto se realizará una herramienta de apoyo la cual permitirá calcular la posibilidad de éxito o fracaso de cualquier institución además de mostrar los reportes de cualquier empresa u organización registrada con sus respectivos resultados.

2.4 Propuesta de sistema

En la presente investigación, con el propósito de darle cumplimiento al problema científico planteado, de acuerdo con los estudios realizados y atendiendo a las necesidades del cliente, se propone el desarrollo de una aplicación de escritorio ya que estas son más robustas y tienen una tiempo de respuesta más rápido que las aplicaciones web. Además, es más fácil sorprender al usuario con una aplicación de escritorio que con una web. Dicha aplicación evaluando una serie de parámetros ya definidos por la ingeniera en ciencias informáticas Ailec Granda Dihigo en su tesis de maestría, donde realizó un estudio de las experiencias obtenidas por diferentes organizaciones, en la realización de programas de mejoras de procesos, así como el análisis de los principales factores que han provocado o influido en el éxito o fracaso de organizaciones, que se adentran en estos proyectos de mejoras; permitirá conocer el estado de una organización antes de iniciar un programa de mejoras de procesos.

Esta aplicación escritorio será capaz de brindar varias funcionalidades que apoyarán el programa de mejoras de procesos, dentro de las que se encuentran:

1. Autenticar usuario: permitirá a los usuarios autenticarse y poder ver todos los reportes.
2. Autenticar usuario_organización: permitirá crear un usuario para cada organización el cual va a ser el encargado de trabajar con su organización y podrá además de ver los reportes, calcular el éxito o fracaso de su organización.
3. Calcular éxito o fracaso en el programa de mejoras de procesos: está funcionalidad permitirá al administrador y al usuario_organización poder calcular el éxito o fracaso a una organización en el programa de mejoras de procesos.
4. Ver reportes: permitirá a todos los usuarios poder ver todos los reportes ya sea el de éxito, fracaso o el de las organizaciones inscritas.

La aplicación de escritorio permitirá a los usuarios con el papel de Administrador ser el encargado de crear los usuarios, usuarios _organizaciones, las organizaciones y de gestionar los reportes.

2.5 Modelo de dominio

El negocio estudiado tiene muy bajo nivel de estructuración, se puede llegar a esta conclusión después de haber estudiado todos los procesos que se van a efectuar, donde los flujos de información se encuentran difusos, y cuando se desea realizar una actividad, múltiples personas intervienen en la misma, lo que implica un solapamiento de responsabilidades, además es difícil establecer las reglas de funcionamiento, por lo que se propone realizar un modelo de dominio.

Se realizará el modelo de dominio, porque permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo. Esto ayuda a los usuarios, clientes, desarrolladores e interesados a utilizar un vocabulario común para poder entender el contexto en que se enmarca el sistema.

Es necesario saber como debe funcionar el proceso en cuestión para capturar correctamente los requisitos y así poder construir un sistema con las características que el cliente desee.

Como primera tarea hay que identificar todos los conceptos que se utilizarán en el diagrama, mediante un glosario de términos sobre los nombres:

UCI: Universidad de las Ciencias Informáticas
Proyectos Productivos: Son todos los proyectos existentes en la universidad.
Cliente: Es la persona que patrocina en el desarrollo del proyecto.
Calisoft: Un centro de referencia de calidad y órgano de certificación de *software*.
Personas: Todas las personas que interactúan con los proyectos**.**
Roles: Son las responsabilidades que tiene cada persona dentro del proyecto.
Proceso de desarrollo: Es un mecanismo que utiliza el centro de Calisoft para evaluar a las empresas.
Productos: Los productos realizados por los proyectos o centro de la universidad.

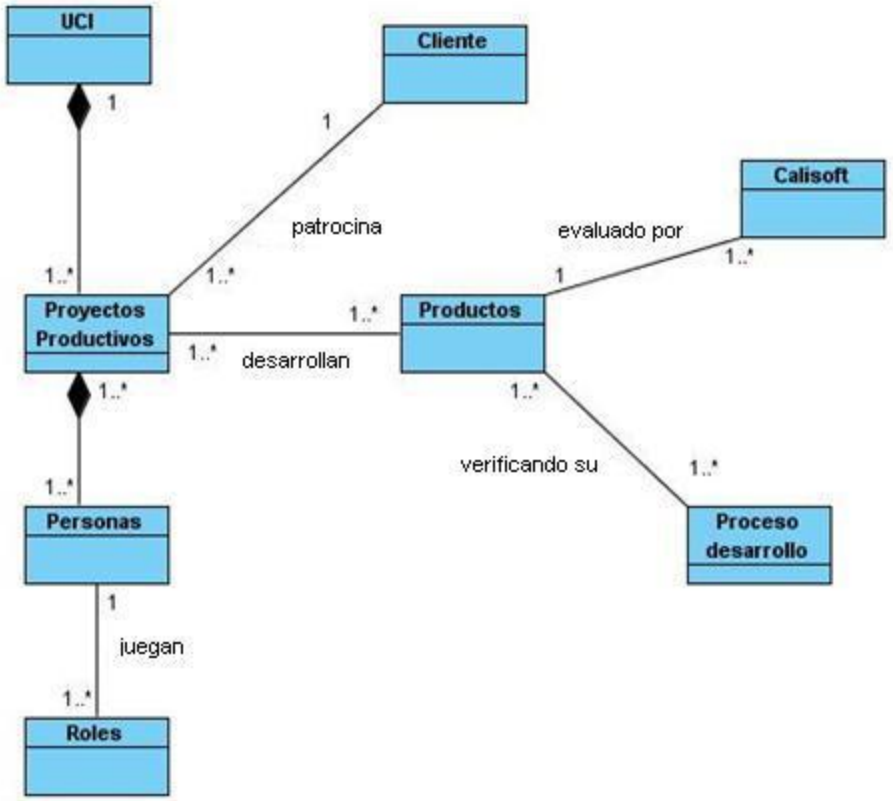

Figura 2.1 Modelo de dominio.

2.6 Especificación de los requisitos de software

La IEEE (Instituto de Ingenieros Eléctricos y Electrónicos) Norma Glosario y Terminología de la Ingeniería del *Software* (1990), define los requerimientos de *software* como condiciones o capacidades que tienen que ser alcanzadas o poseídas por un sistema o componente de un sistema para satisfacer un contrato, estándar u otro documento impuesto formalmente.
Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen. Es una característica que el sistema debe tener para cubrir alguna de las necesidades de los usuarios que lo motivan para resolver un problema o lograr un objetivo.

2.6.1 Requisitos funcionales

R1: Autenticar usuario

- permite autenticar usuario, con el cual se podrá acceder al sistema a través de nombre y contraseña.

R2: Crear Usuario _ organización

- permite crear un usuario que es el encargado de una organización, con el cual se podrá trabajar en dicha organización.

R3: Modificar Usuario _ organización

- permite modificar los datos de un usuario_organización ya sea nombre, contraseña.

R4: Buscar Usuario _ organización

- permite buscar un usuario_organización que este inscrito en el sistema.

R5: Eliminar Usuario _ organización

- permite eliminar un usuario_organización del sistema.

R6: Crear reporte de cantidad organizaciones inscritas

- permite crear un reporte con todas las organizaciones inscritas en el sistema.

R7: Crear reporte de cantidad organizaciones con éxito

- permite crear un reporte con todas las organizaciones inscritas en el sistema que hayan tenido éxito en el programa de mejoras de procesos, el cual podrá ser visto por todos los usuarios.

R8: Crear reporte de cantidad organizaciones con fracaso

- permite crear un reporte con todas las organizaciones inscritas en el sistema, que hayan fracasado en el programa de mejoras de procesos y será visto por todos los usuarios autenticados en el sistema.

R9: Calcular posibilidad de éxito de un programa de mejoras

- permite calcular el éxito o fracaso que podrá tener una organización de las que están inscritas en el sistema, mostrando los resultados.

R10: Crear Organización

- permite crear las organizaciones con las que el sistema trabajará

R11: Modificar Organización

- permite modificar los datos de las organizaciones que están inscritas en el sistema.

R12: Buscar Organización

- permite la búsqueda de una organización determinada.

R13: Eliminar Organización

- posibilita la de eliminar una organización que este inscrita en el sistema.

2.6.2 Requisitos no funcionales.

Los requerimientos no funcionales detallan las propiedades o cualidades que el producto debe tener, aumentándole funcionalidad al sistema, pues hacen al producto atractivo, fácil de usar, rápido y confiable, los cuales se encuentran separados por categorías que ahora se mencionan.

Usabilidad:

El sistema podrá ser usado de forma fácil por cualquier persona, con pocos conocimientos de computación.

Apariencia o interfaz externa:

La interfaz del usuario deberá ser tan familiar como sea posible a los clientes.

Diseño orientado a llamar la atención del usuario y con una navegación sencilla.

Rendimiento:

El sistema demora en una transición aproximadamente 5 segundos como máximo, por detrás de la aplicación de escritorio se realizarán consultas a bases de datos. El sistema será estable no presentará ningún tipo de error que lo haga inestable.

Confiabilidad:

La información contenida en el sistema debe ser totalmente confiable. Deben implementarse mecanismos para garantizar la respuesta ante posibles fallos lo más rápido posible.

Seguridad:

El acceso será controlado con nombres de usuario y contraseñas. Solo los usuarios con derechos de administrador podrán acceder las funciones administrativas, los usuarios normales no podrán.

Soporte:

Servidor de base de datos PostgreSQL.

Máquina Virtual de Java

***Software*:**

PostgreSQL 8.3

PgAdmin III

NetBeans 6.8

Legales: El producto es general y puede ser usado en otras aplicaciones similares.

Empleo del CMS bajo licencia GPL.

Eficiencia:

Garantizar velocidad de respuesta.

Tener base de datos normalizada, para garantizar la integridad de la información y reducir los tiempos de respuesta.

2.7 Definición de los casos de uso del sistema

2.7.1 Actores del sistema

Nombres	Justificación
Administrador	Realiza todas las acciones que realiza el sistema como son crear usuario, gestionar usuario_ organización, ver reportes de organizaciones inscritas, organizaciones con éxito, organizaciones con fracaso y calcular éxito o fracaso de las organizaciones en el programa de mejoras.
Usuario_organización	El usuario_organización es el encargado de calcular el éxito o fracaso de su organización, además de ver los reportes.

Tabla 2.1 Actores del sistema

2.7.2 Listado de los casos de uso del sistema.

CU- 1	Autenticar_ usuario
Actores:	Usuario
Descripción:	Permite al administrador crear un usuario
Referencias:	R1

Tabla 2.2 Caso de uso Autenticar usuario.

CU- 2	Gestionar usuario_organización
Actores:	Administrador
Descripción:	Permite al administrador crear, modificar, buscar y eliminar al usuario_organización.
Referencias:	R1,R2,R3,R4y R5

Tabla 2.3 Caso de uso Gestionar usuario_organización

CU- 3	Crear reporte de organización con éxito
Actores:	Administrador, usuario_organización.
Descripción:	Permite al administrador y al usuario_organización la opción de crear reporte de organización con éxito.
Referencias:	R1 y R11

Tabla 2.4 Caso de uso Crear reporte de las organización con éxito

CU- 4	Crear reporte de organización con fracaso
Actores:	Administrador, usuario_organización.
Descripción:	Permite al administrador y al usuario_organización la opción de crear reporte de organización con fracaso.
Referencias:	R1 y R12

Tabla 2.5 Caso de uso Crear reporte de organización con fracaso

CU- 5	Crear reporte de organización inscrita
Actores:	Administrador, usuario_organización.
Descripción:	Permite al administrador y al usuario_organización la opción de crear reporte de organización inscrita.
Referencias:	R1y R10

Tabla 2.6 Caso de uso Crear reporte de organización inscrita

CU- 6	Calcular éxito o fracaso.
Actores:	Administrador, usuario_organización.
Descripción:	Permite al administrador y al usuario_organización la opción de poder calcular el posible éxito o fracaso de una organización.
Referencias:	R1 y R13

Tabla 2.7 Caso de uso Calcular éxito o fracaso

CU- 7	Gestionar organización
Actores:	**Administrador**
Descripción:	**Permite al administrador crear, modificar, buscar y eliminar a la Organización**
Referencias:	**R1,R6,R7,R8 y R9**

Tabla 2.8 Caso de uso Gestionar organización

2.7.3 Diagrama de casos de uso del sistema.

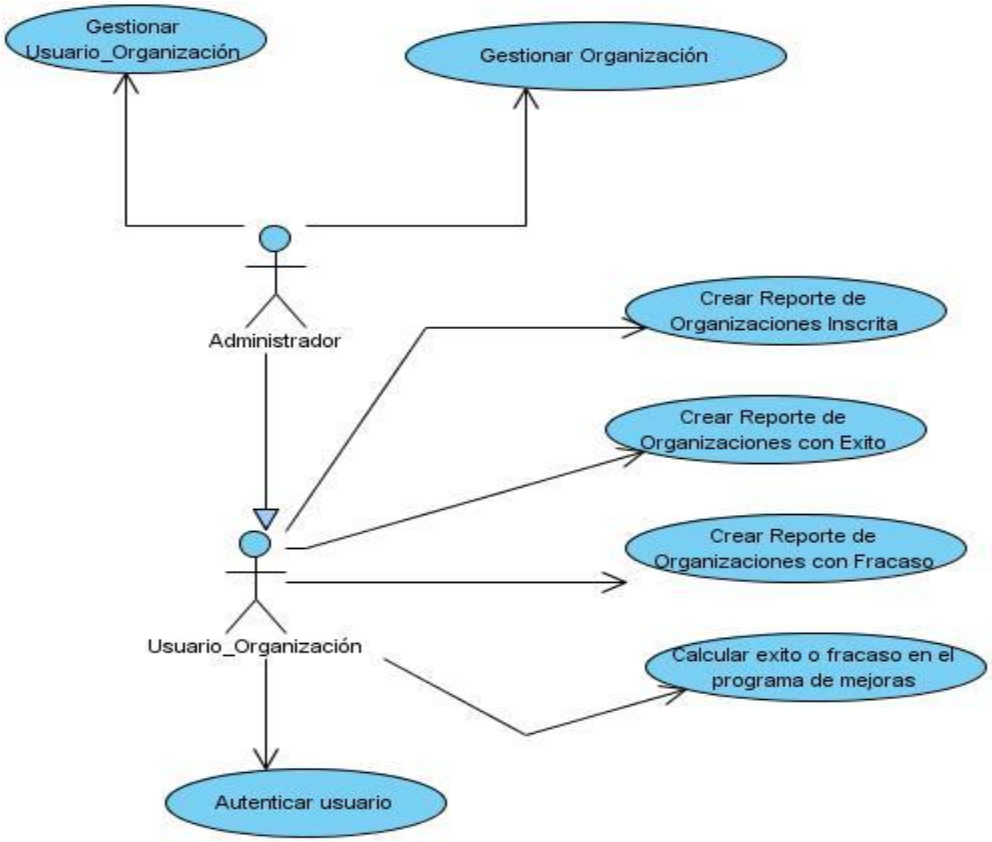

Figura 2.2 Diagrama del caso de uso del sistema

2.7.4 Descripción de los casos de uso del sistema.

CU_1	Autenticar usuario
Actores:	**Usuario**
Resumen:	**El caso de uso se inicia cuando el usuario accede al menú autenticar usuario y se autentica.**
Precondiciones:	**El usuario debe de estar creado**
Referencias:	**R1**
Poscondiciones:	**Se autenticó el usuario.**
Flujo Normal de Eventos	
Sección " "	
Acción del Actor	**Respuesta del Sistema**
1.0 El usuario se dirige al menú autenticar usuario. **1.2 El usuario introduce los datos (nombre y contraseña).**	**1.1 El sistema muestra el formulario autenticar usuario.** **1.3 El usuario se autentica en el sistema.**
Prototipo de Interfaz	
Flujo Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.2.1 Muestra mensaje para que llene los campos obligatorios. **1.2.1 Muestra mensaje de usuario o contraseña incorrecta.**

Tabla 2.9 Descripción del casos de uso Autenticar_usuario.

Para ver descripciones de los caso de uso remitirse al (**Anexo 1**).

2.8 Estudio de factibilidad

Es importante calcular la factibilidad de un proyecto para saber si es favorable llevarlo a cabo. Se propone en este trabajo el uso de la técnica de estimación basada en puntos de casos de usos. La estimación basada en casos de usos permitirá crear un aproximado del tiempo en que se desarrollará el sistema. A continuación, se detallan los pasos a seguir para la aplicación de este método.

2.8.1 Cálculo de puntos de casos de uso sin ajustar

El primer paso para la estimación es el cálculo de los puntos de casos de uso sin ajustar, este valor se obtiene aplicando la siguiente ecuación matemática:

UUCP=UAW+UUCW

UUCP: Puntos de casos de uso sin ajustar.

UAW: Factor de peso de los actores sin ajustar.

UUCW: Factor de peso de los casos de uso sin ajustar.

Para calcular el peso de los actores sin ajustar se hace un análisis de la cantidad de actores presentes en el sistema y la complejidad de cada uno de ellos como se muestra en la tabla.

Tipo de actor	Factor de peso	Cantidad de actores	Total
Simple	1	0	0
Medio	2	0	0
Complejo	3	2	6

Tabla 2. 16 Factor de peso de los actores sin ajustar

UAW = Σ cant actores * Peso = 3 * 2

UAW=6

Luego de obtener este valor se pasa a calcular el valor del factor de peso de los casos de uso sin ajustar (UUCW). Este valor se calcula mediante un análisis de la cantidad de casos de uso presentes en el sistema y la complejidad de cada uno de ellos.

Tipo	Descripción	peso	Cant * peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones	5	3*5
Medio	El Caso de Uso contiene de 4 a 7 transacciones	10	2*10
Complejo	El Caso de Uso contiene más de 8 transacciones	15	2*15
Total			75

Tabla 2.17 Factor de peso de los casos de uso sin ajustar

UUCW = Σ cant CU * Peso = 5 * 3 + 10 * 2 + 15 *2
UUCW=75

Calculando los puntos de casos de uso sin ajustar (UUCP):

UUCP = 6 + 75 = 81

2.8.2 Cálculo de los puntos de casos de uso ajustados

UCP= UUCP*TCF*EF

Donde:
UCP: Puntos de casos de uso ajustados.
UUCP: Puntos de casos de uso sin ajustar.
TCF: Factor de complejidad técnica.
EF: Factor de ambiente.

El factor de complejidad técnica (TCF) se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante. En la tabla se muestra el peso de cada uno de estos factores.

Factor	Descripción	Peso	Valor	Comentario	Σ (Pesoi * Valor i)
T1	Sistema distribuido	2	1	El sistema no es distribuido.	2
T2	Objetivos de performance o tiempo de respuesta	1	4	El tiempo de respuesta es rápido.	4
T3	Eficiencia del usuario final	1	4	No requiere de conocimientos profundos para tratar con la aplicación.	4
T4	Procesamiento interno complejo	1	2	No hay cálculos complejos	2
T5	El código debe ser reutilizable	1	5	Se requiere que el código sea reutilizable	5
T6	Facilidad de instalación	0.5	4	Escasos requerimientos de facilidad de instalación	2
T7	Facilidad de uso	0.5	5	Normal	2.5

T8	Portabilidad	2	0	No se requiere que el sistema sea portable	0
T9	Facilidad de cambio	1	3	Se requiere un costo moderado de mantenimiento	3
T10	Concurrencia	1	0	No hay concurrencia	0
T11	Incluye objetivos especiales de seguridad	1	2	Seguridad normal	2
T12	Provee acceso directo a terceras partes	1	0	No presenta terceras partes.	0
T13	Se requieren facilidades especiales de entrenamiento a los usuarios	1	5	Pocos usuarios internos, sistema fácil de usar.	5
Total					31.5

Tabla 2.18 Factor de complejidad técnica

TCF = 0.6 + 0.01 * Σ (Peso * valor asignado)

TCF= 0.6 + 0.01 * 31.5 = 0.91

Las habilidades y el entrenamiento del grupo involucrado en el desarrollo tienen un gran impacto en las estimaciones de tiempo. Estos factores son los que se contemplan en el cálculo del **factor de ambiente (EF)**. El cálculo del mismo es similar al cálculo del factor de complejidad técnica, es decir, se trata de un conjunto de factores que se cuantifican con valores de 0 a 5. En la tabla se muestra los pesos correspondientes al sistema que se propone acompañados de un breve comentario:

Factor	Descripción	Peso	Valor	Comentario	Σ (Peso i * Valor i)
E1	Familiaridad con el modelo de proyecto utilizado	**1.5**	**3**	El grupo está familiarizado con el modelo, pero faltan algunos detalles.	**4.5**
E2	Experiencia en la aplicación	**0.5**	**4**	La mayoría del grupo ha trabajado mucho tiempo en este tipo de aplicación	**2**
E3	Experiencia en orientación a objetos	**1**	**4**	La mayoría del grupo programa orientado a objetos	**4**

E4	Capacidad del analista líder	**0.5**	**4**	El analista está muy bien capacitado	**2**
E5	Motivación	**1**	**5**	Existe motivación del grupo.	**5**
E6	Estabilidad de los requerimientos	**2**	**2**	Se esperan cambios	**4**
E7	Personal part-time	**-1**	**2**	Todo el grupo es full-time	**-2**
E8	Dificultad del lenguaje de programación	**-1**	**3**	Se usará lenguaje Java.	**-3**
Total					**16.5**

Tabla 2.19 Factor de ambiente

EF= 1.4 – 0.03 * Σ (Peso * valor asignado)

EF= 1.4 – 0.03 * 16.5 = 0.90

UCP=UUCP*TCF*EF

= 81*0.91*0.90

= 66.33

2.8.3 Estimación de esfuerzo a través de los puntos de casos de uso

Los factores que afectan el factor ambiente entre E1 y E6 es 1 porque están por debajo de 3. Por lo que el total es 2 o menos, se utiliza el factor de conversión 20 horas-hombres. El esfuerzo en horas hombres se calcula con la siguiente ecuación matemática:

E=UCP*CF

E=66.33 *20

E=1326.6 Horas-Hombres

Calcular el esfuerzo de todo el proyecto

Actividad	% esfuerzo	Valor esfuerzo
Análisis	10%	331.65
Diseño	20%	663.3
Implementación	40%	1326.6
Prueba	15%	497.47
Sobrecarga	15%	497.47
Total	100%	3316.5

Tabla 2.20 Esfuerzo de todo el proyecto

El trabajo realizado comprende las fases de diseño, implementación y prueba donde el esfuerzo total (ET) de estas fases es de 3316.5 **horas-hombre** y por cada 240 horas se tiene 1 mes, eso daría un esfuerzo total de 13.818 mes-hombre**.**

Si:

Tiempo = **ET / CH**

Tiempo = 13.818 / 2

Tiempo = 6.90

Esto quiere decir que 2 personas realizando el mismo esfuerzo, pueden realizar el sistema analizado en aproximadamente 7 meses.

2.9 Conclusiones del capítulo

- Establecer las características del sistema a partir de las entrevistas a expertos o clientes permitió implementar y poseer dominio de cómo se desarrollan los procesos que intervienen en el negocio.

- La definición del modelo de dominio facilitó el establecimiento de las relaciones entre los componentes que intervienen en los proyectos productivos.

- La captura de los requerimientos del sistema constituye el punto de partida para el desarrollo de la aplicación propuesta en la investigación, pues permitió agruparlos en casos de uso similares y de esta forma expresarlos en el diagrama de caso de uso del sistema.

- Además de realizar el estudio de factibilidad con la intención de para saber si es favorable llevarlo a cabo o no.

Capítulo 3: Análisis y diseño del sistema

En este capítulo se tendrán en cuenta los procesos de análisis y diseño del sistema, en el cual se presentan los distintos artefactos y diagramas de clases del análisis, los diagramas de clases del diseño, y por cada realización de casos de uso se muestran los diagramas de colaboración. También se realiza la descripción de las clases, se diseña la base de datos y se describen las tablas de la misma. Además, el flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo con los nodos específicos en el modelo de despliegue. Los diagramas de despliegue y componentes conforman lo que se conoce como un modelo de implementación al describir los componentes a construir, además la organización y dependencia entre nodos físicos en los que funcionará a aplicación.

3.1 Patrones de diseño

Para realizar el diseño de la herramienta de apoyo se aplicaron los patrones de diseño GRASP (patrones generales de *software* para asignar responsabilidades).

Experto: Es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen.

Creador: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

Alta cohesión: Asignar una responsabilidad de modo que la cohesión siga siendo alta. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. La ventaja que significa una gran funcionalidad también soporta un aumento de la capacidad de reutilización.

Bajo acoplamiento: Asignar una responsabilidad para mantener bajo acoplamiento (una clase con bajo (o débil) acoplamiento no depende de muchas otras; una clase con alto (o fuerte) acoplamiento recurre a muchas otras.). Este patrón estimula asignar una responsabilidad de modo

que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento.

Controlador: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. La mayor parte de los sistemas reciben eventos de entrada externa, En cualquiera de los casos que puedan existir, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen esos eventos de entrada.

3.2 Descripción de la arquitectura

Existen diferentes patrones arquitectónicos dentro de los que se pueden mencionar el patrón arquitectónico modelo vista controlador (MVC),el patrón arquitectónico basado en componentes y el patrón arquitectónico de tres capas. A continuación se explican las características principales:

MVC:

Es un patrón de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la lógica de negocio.

Arquitectura basada en Componentes:

Esta describe una aproximación de ingeniería de *software* al diseño y desarrollo de un sistema. Se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado.

Arquitecturas en Capas

Donde se define cómo organizar el modelo de diseño a través de capas, que pueden estar físicamente distribuidas, lo cual quiere decir que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores. Este patrón es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las

dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. Además, nos ayuda a identificar qué puede reutilizarse, y proporciona una estructura que nos ayuda a tomar decisiones sobre qué partes comprar y qué partes construir. Cuando trabajamos en capas, podemos usar diferentes niveles o cantidad de capas (2 o más capas)

"Modelo-Tres Capas"

Capa de aplicación

Esta capa contiene todas las clases de interfaz de usuarios que representan las pantallas de la aplicación que el usuario ve. Esta capa depende de la capa de servicios de negocio y de la capa intermedia.

Capa de Servicios de Negocio

La capa de servicios de negocio tiene todas las clases controladoras que representan el comportamiento de la aplicación, según los casos de uso. Esta capa representa la frontera del cliente con la capa intermedia. La capa de servicios de negocio depende de la capa intermedia.

Capa Intermedia

La capa intermedia apoya el acceso al SGBD (Sistema Gestor de Base de Dato).

Luego de haber realizado un estudio de los patrones arquitectónicos se escogió para la realización de la aplicación, la arquitectura de tres capas la cual tiene como ventajas:

- La interfaz del usuario, el servidor de capa intermedia, son más flexibles que en el diseño de dos capas, ya que la estación solo necesita transferir parámetros a la capa intermedia.
- Con la arquitectura de tres capas, la interfaz del cliente no es requerida para comprender o comunicarse con el receptor de los datos. Por lo tanto, esa estructura de los datos puede ser modificada sin cambiar la interfaz del usuario en la PC.
- El código de la capa intermedia puede ser reutilizado por múltiples aplicaciones si esta diseñado en formato modular.

- La separación de roles en tres capas, hace mas fácil reemplazar o modificar una capa sin afectar a los módulos restantes.

3.3 Modelo de análisis

El modelo de análisis es una aproximación al modelo del diseño. En este modelo hay un refinamiento de los requisitos, sin embargo, no se tiene en cuenta el lenguaje de programación que se va a utilizar en la construcción de la aplicación, debido a que el objetivo del análisis es comprender perfectamente los requisitos del *software* y no precisar como se implementará la solución. Es uno de los flujos de trabajo que se realizan en el proceso de *software* durante la fase de elaboración. A continuación se hace una descripción de las clases que se utilizarán en la realización de dichos diagramas y que permitirán obtener una mejor visión del sistema. Los prototipos para identificar las diferentes clases que participan en el análisis son los siguientes:

CI_<Nombre de la clase>: estas son las Clases Interfaz, las cuales modelan la interacción entre los actores y el sistema, ventanas, formularios, comunicación con otros sistemas o dispositivos.

CC_<Nombre de la clase>: estas son las Clases Controladoras, las cuales coordinan el trabajo de las clases encapsulan el comportamiento de cada caso de uso y las funciones más complejas.

CE_<Nombre de la clase>: estas son las Clases Entidad, las cuales modelan toda la información del sistema que posee una vida larga y que puede ser persistente además de modelar el comportamiento asociado a una información.

3.3.1 Diagrama de clases del análisis

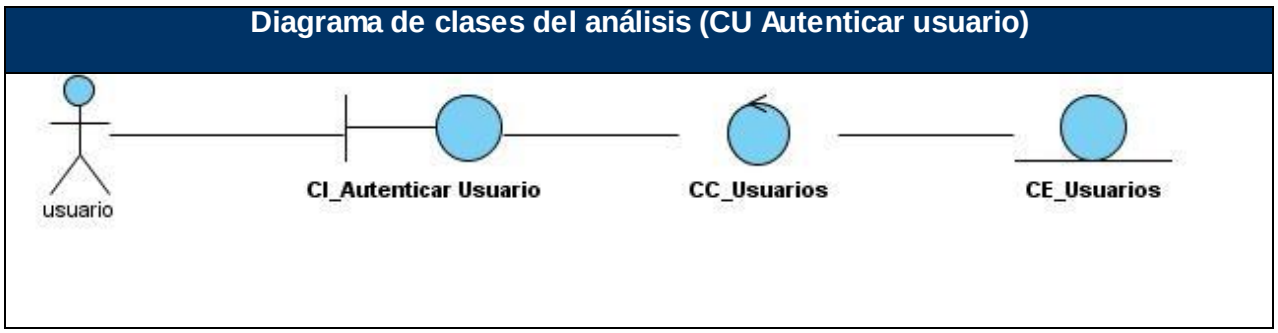

Figura 3.1 Diagrama de clase del análisis CU Autenticar usuario.

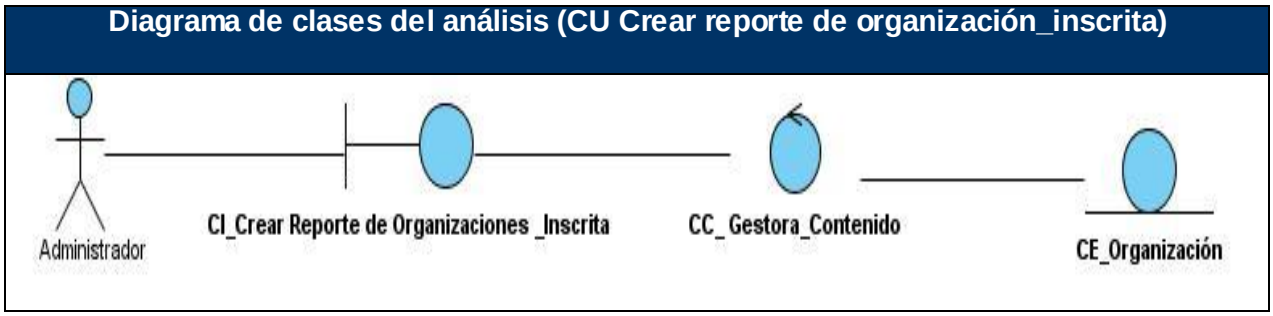

Figura 3.2 Diagrama de clases del análisis CU Crear reporte de organización inscrita.

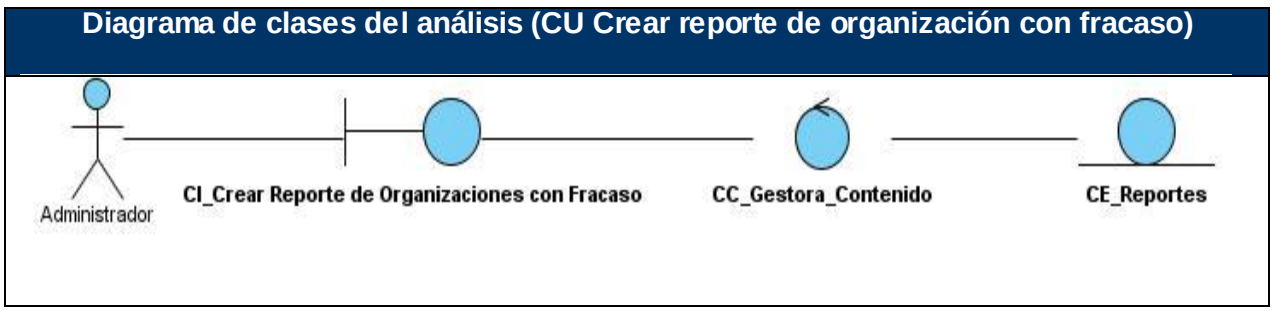

Figura 3.3 Diagrama de clases del análisis Crear reporte de organización con fracaso.

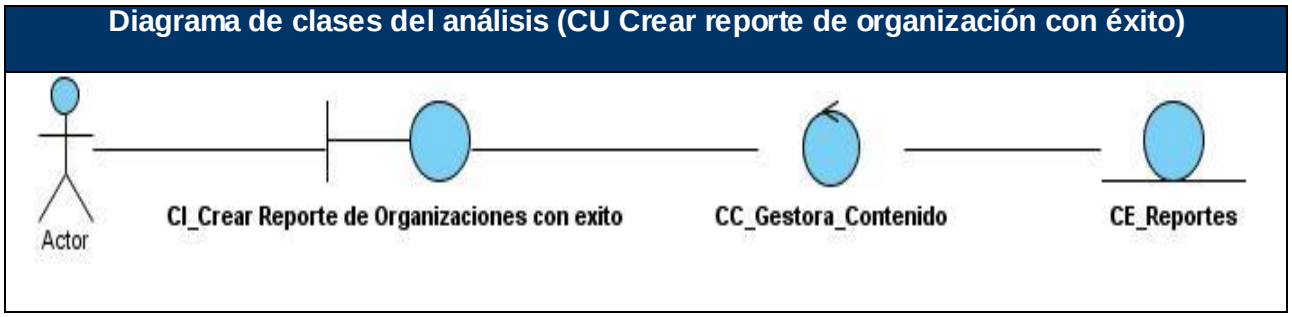

Figura 3.4 Diagrama de clases del análisis CU Crear reporte de organización con éxito.

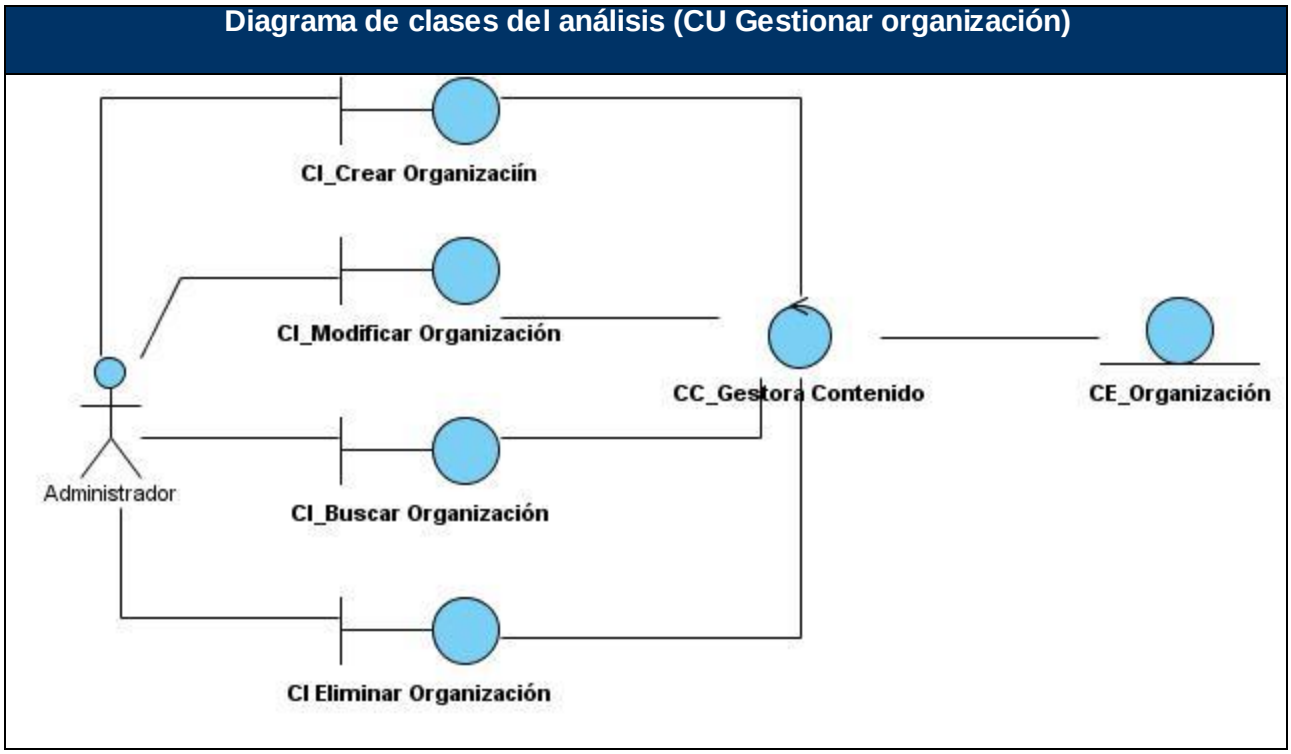

Figura 3.5 Diagrama de clases del análisis CU Gestionar organización.

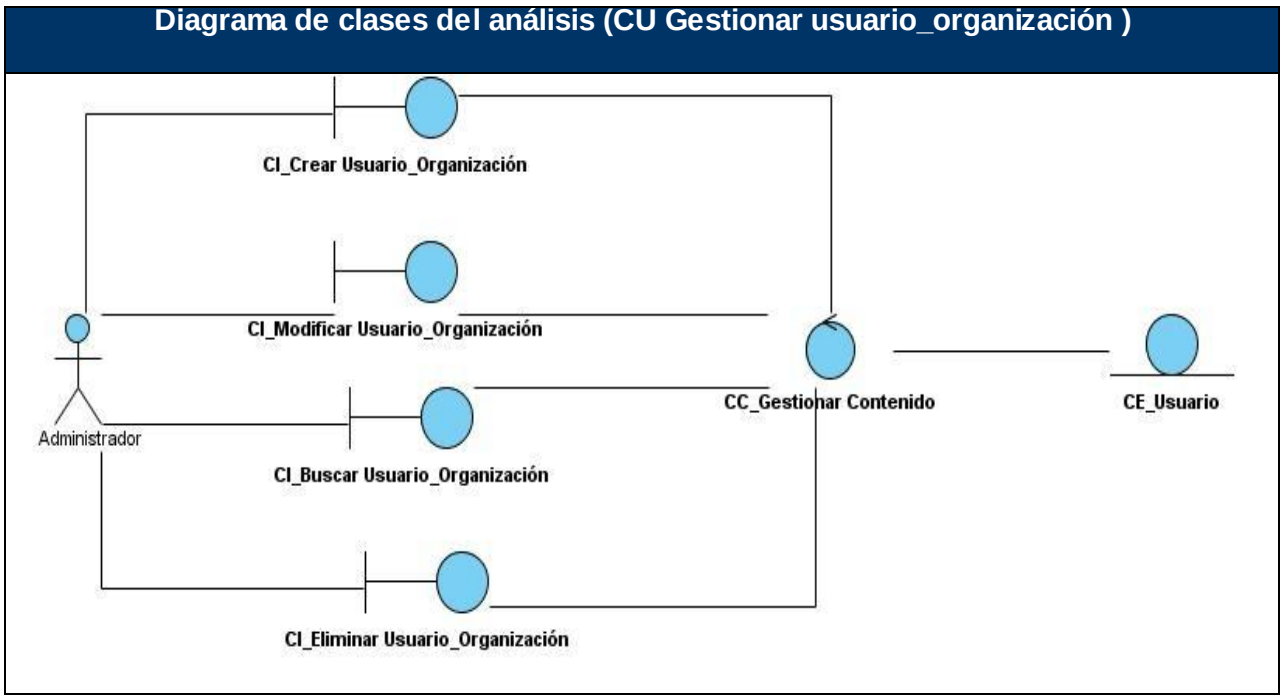

Figura 3.6 Diagrama de clases del análisis CU Gestionar usuario_organización.

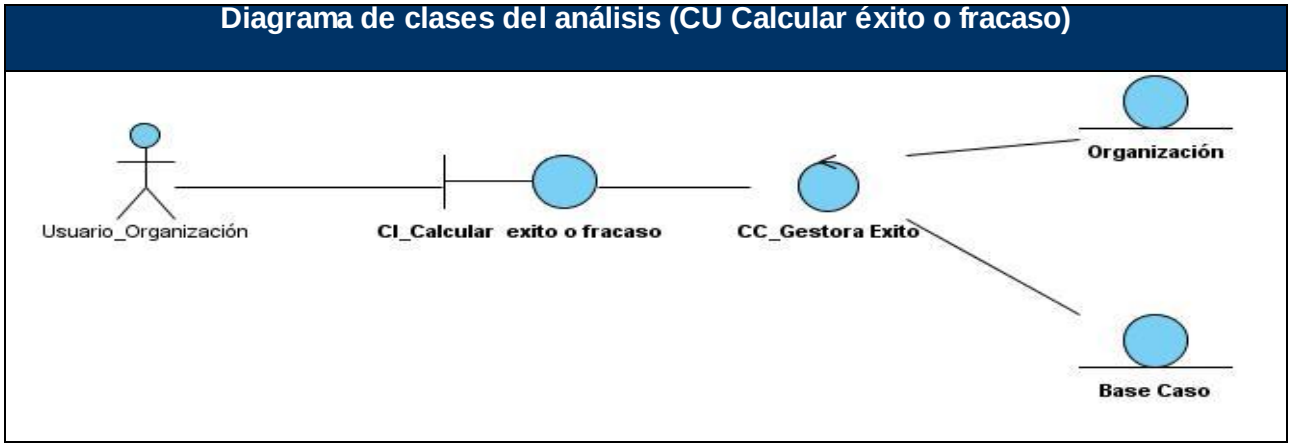

Figura 3.7 Diagrama de clases del análisis CU Calcular éxito o fracaso.

3.3.2 Diagrama de interacción

Los diagramas de interacción realizados en el análisis fueron los de colaboración (Ver en los **Anexos 2**)

3.4 Diseño

A continuación se desarrolla el flujo de trabajo de diseño, a través del cual se modela el sistema que se pretende construir para que este sea capaz de soportar todos los requisitos. El diseño es el centro de atención final de la fase de elaboración y el comienzo de las iteraciones de construcción, no es más que el refinamiento del análisis, el cual contribuye a una arquitectura estable y sólida, además de crear un plano del modelo de implementación. Mas tarde durante la fase de construcción, cuando la arquitectura es estable y los requisitos están bien entendidos, el centro de atención se desplaza a la implementación.

3.4.1 Diagrama de clases del diseño

Para la elaboración del diseño, se comenzara explicando todas las clases existentes, en la elaboración del sistema:

Autenticar usuario: el cual se encarga de la autenticación de los usuarios para darles los permisos para cada tipo de usuario. En esta aplicación existen dos tipos de usuario, esta el administrador y el usuario_organización, en este caso de uso el que más prioridad tiene es el administrador, porque el usuario_organización solo puede revisar los reportes y utilizar la interfaz del calcular éxito o fracaso.

Gestionar usuario organización: el cual se encarga de realizar las operaciones básicas sobre los datos: crear, modificar, buscar y eliminar además de que le da algunos permisos al usuario para poder interactuar con la aplicación.

Gestionar organización: el cual se encarga de realizar las operaciones básicas sobre los datos: crear, modificar, buscar y eliminar además que permite registrar la organización que utilicen la herramienta y almacenar eso datos para utilizarlos como experiencias pasadas, y agregarlos a la base de caso existente.

Calcular éxito o fracaso: esta clase se encarga de realizar los cálculos de una organización o institución para poder evaluar si esta se encuentra en condiciones de iniciar un programa de mejoras de procesos utilizando cálculos matemáticos con el algoritmo matemático llamado Argelio que es una especie de sistema experto.

Crear reporte de organizaciones inscrita: esta clase se encarga de almacenar todos los resultados obtenidos cuando una organización o empresa utiliza la aplicación y los muestra en una lista.

Crear reportes de organización con éxito: esta clase se encarga de almacenar todos los resultados obtenidos de organizaciones con éxito y los muestra en una tabla.

Crear reportes de organización con fracaso: esta clase se encarga de almacenar todos los resultados obtenidos de organizaciones con fracaso y los muestra en una tabla.

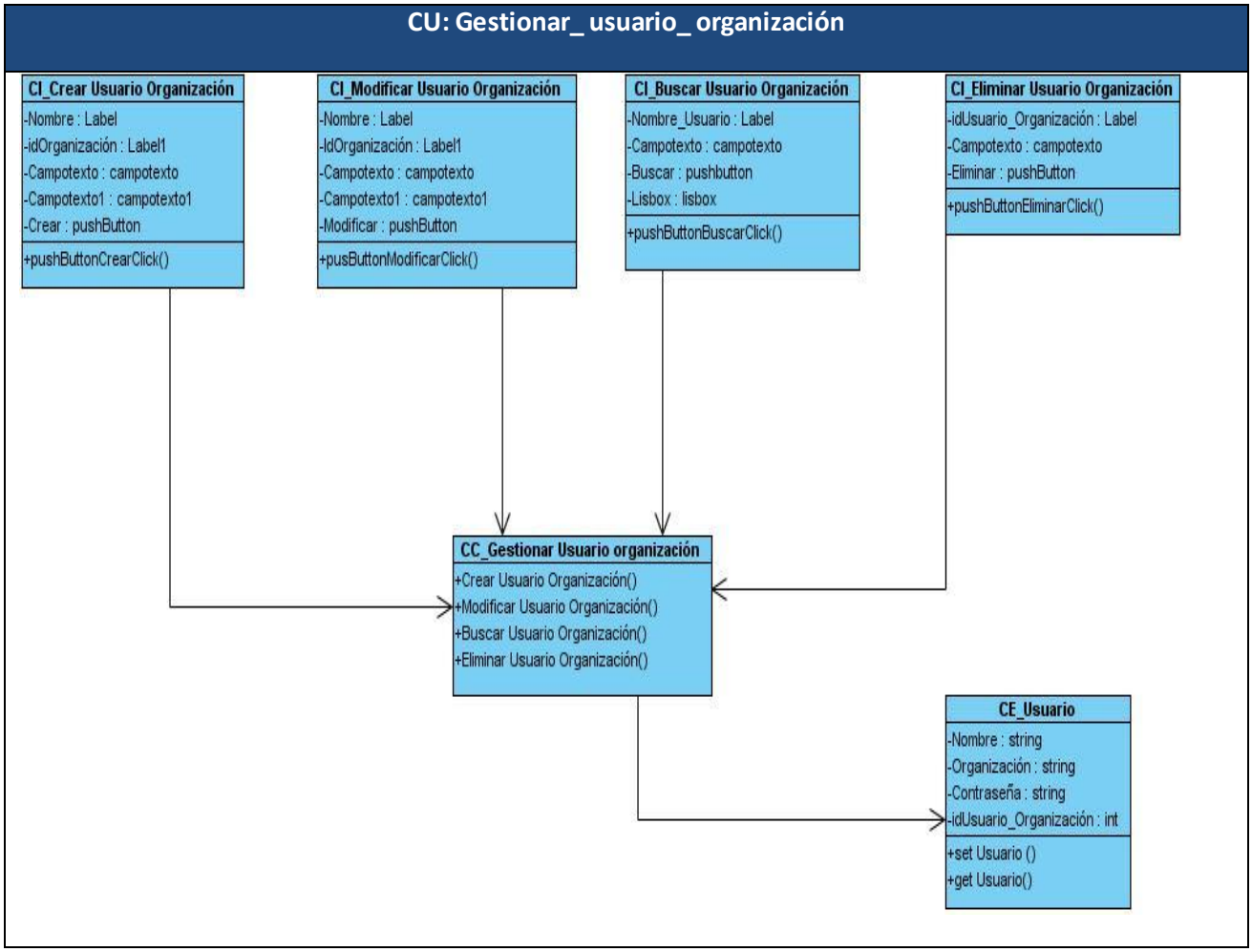

Figura 3.8 Diagrama del diseño del caso de uso Gestionar usuario_organización

Para ver los diagramas del diseño remitirse al (**Anexo 4**).

3.4.2 Diagramas de interacción en el diseño

Los diagramas de interacción son usados para modelar los aspectos dinámicos de un sistema. Los diagramas de interacción pueden utilizarse para visualizar, especificar, construir y documentar la dinámica de una sociedad particular de objetos, o se pueden utilizar para modelar un flujo de control particular de un caso de uso. Un diagrama de interacción muestra una interacción, que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. El diagrama de interacción que se modelará es el de secuencia, es uno de los más efectivos para modelar la interacción entre objetos en un sistema.

Para ver los diagramas de secuencia dirigirse a los (**Anexos 3**).

3.4.3 Diagrama de despliegue

El modelo de despliegue describe cómo una aplicación se despliega a través de una infraestructura. La intención del modelo de despliegue no es describir la infraestructura, sino el camino en cual los componentes específicos deben corresponder a una aplicación que despliega a través de él. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de *hardware* y otra información relacionada al despliegue del sistema propuesto. El diagrama de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. A continuación se muestra el diagrama de despliegue que se propone para el sistema.

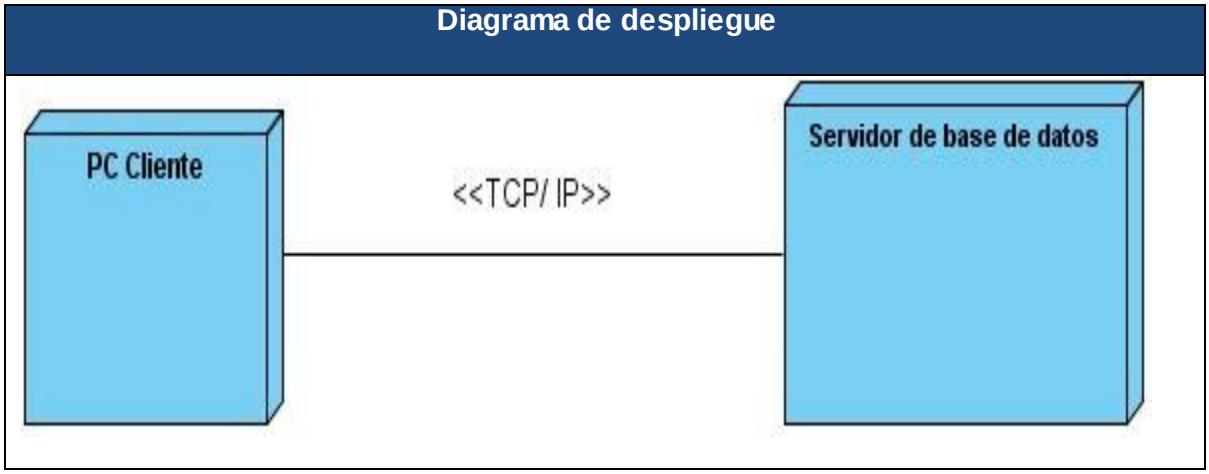

Figura 3.15 Diagrama de despliegue

3.4.4 Diseño de base de datos

La base de datos es el sistema utilizado para el almacenamiento de datos y acceso controlado a los que han sido almacenados. En este epígrafe se muestra el diseño de la base de datos del sistema propuesto a través del diagrama de clases persistentes y el esquema de la base de datos generados a partir de este, el modelo de datos.

3.4.4.1 Diagrama de clases persistentes

Las clases persistentes son aquellas que necesitan ser capaz de guardar su estado en un medio permanente, la necesidad de guardar su estado está dado por el almacenamiento físico permanente de la información de la clase, para la copia de seguridad en caso del fracaso del sistema, o para el intercambio de información. A continuación se muestra el diagrama de clases persistentes.

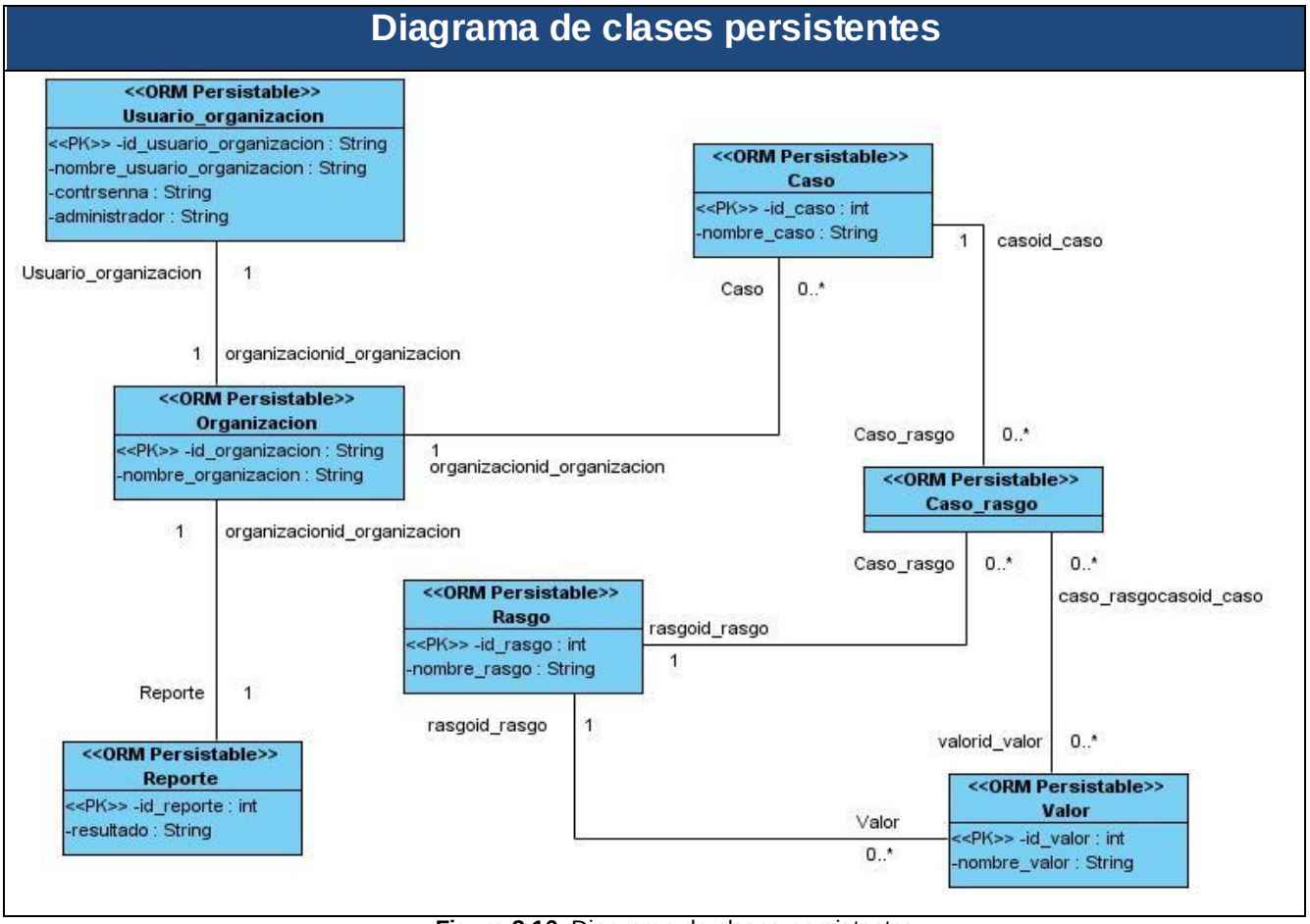

Figura 3.16 Diagrama de clases persistentes

3.4.4.2 Modelo de datos

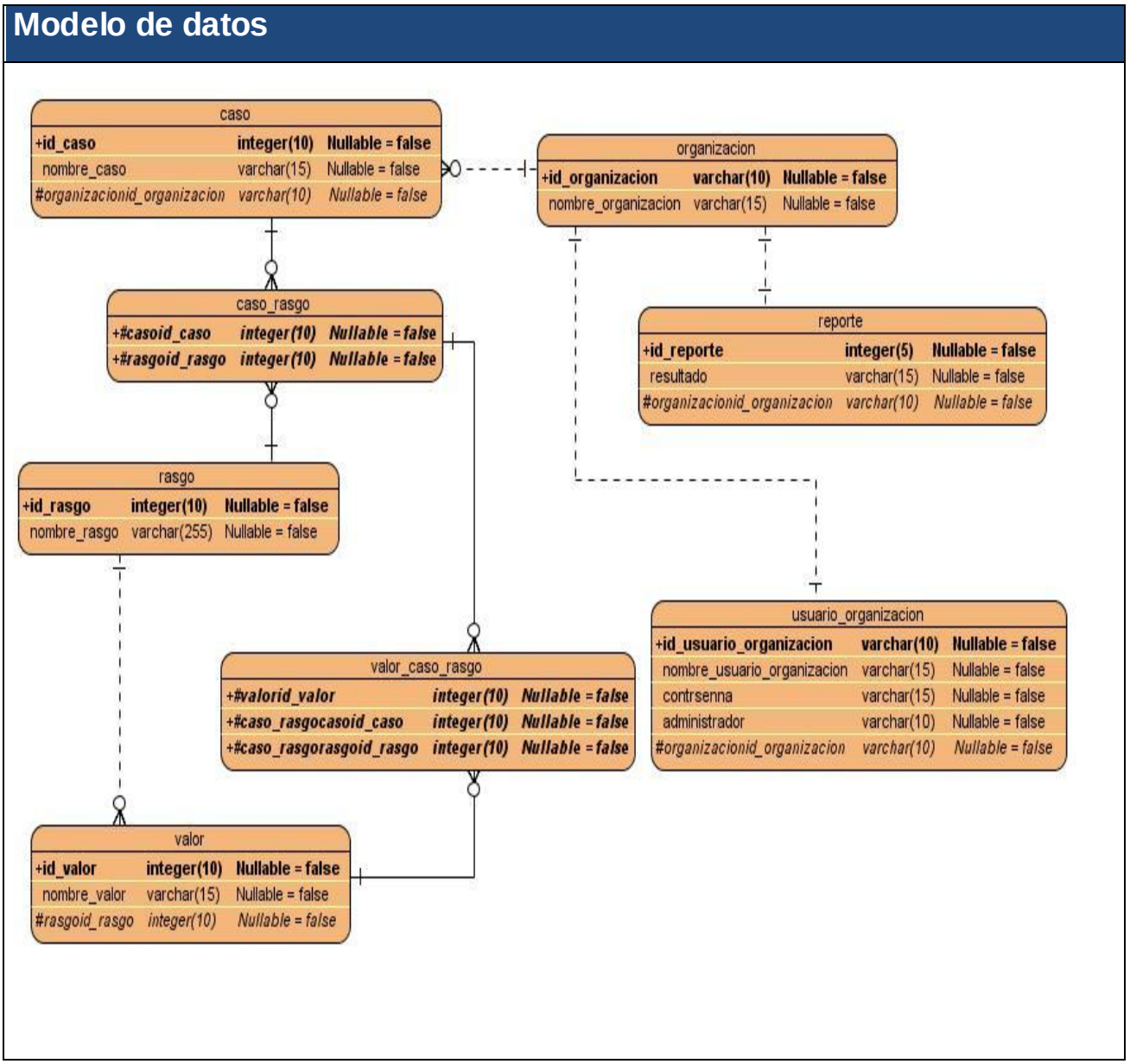

Figura 3.17 Diagrama modelo de datos

3.4.4.3 Descripción de las tablas

Nombre: Organización		
Descripción: Contiene todos los contenidos de las organizaciones.		
Atributo	**Tipo**	**Descripción**
id_organización	String	Identificador de la tabla de organización
nombre_organización	varchar	Nombre de la organización

Tabla 3.1 Descripción de la tabla organización

Nombre: Rasgo		
Descripción: Contiene todos los contenidos de los rasgos		
Atributo	**Tipo**	**Descripción**
Id_rasgo	int	Identificador de la tabla de rasgos
nombre_rasgo	varchar	Nombre de los rasgos
#valorid_valor	int	Identificador de valores

Tabla 3.2 Descripción de la tabla rasgo

Nombre: Caso		
Descripción: Contiene todos los contenidos de los rasgos		
Atributo	**Tipo**	**Descripción**
Id_caso	int	Identificador de la tabla de casos
nombre_caso	varchar	Nombre de los casos

Tabla 3.3 Descripción de la tabla caso

Nombre: Valor		
Descripción: Contiene todos los contenidos de los valores.		
Atributo	**Tipo**	**Descripción**
Id_valor	int	Identificador de la tabla de valores
nombre_valor	varchar	Nombre de los valores

Tabla 3.4 Descripción de la tabla valor

Nombre: Valor _rasgo _caso		
Descripción: Contiene todos los contenidos de los valores ,rasgos y casos es decir es una base de casos		
Atributo	**Tipo**	**Descripción**
#valorid_valor	int	Identificador de la tabla valor
#rasgo_casorasgoId_rasgo	int	Identificador de la tabla rasgo
#rasgo_casocsoId_caso	int	Identificador de la tabla caso

Tabla 3.5 Descripción de la tabla valor_rasgo_caso.

Nombre: Rasgo _caso		
Descripción: Contiene todos los contenidos de los rasgos y casos		
Atributo	**Tipo**	**Descripción**
#rasgosId_rasgo	int	Identificador de la tabla rasgo
#casoId_caso	int	Identificador de la tabla caso

Tabla 3.6 Descripción de la tabla rasgo_caso.

Nombre: Reporte		
Descripción: Contiene todos los contenidos de los reportes realizados		
Atributo	**Tipo**	**Descripción**
id_reporte	int	Identificador de la tabla de Reportes
resultado	varchar	Resultado que obtendrá la organización
Id_organización	string	Organización

Tabla 3.7 Descripción de la tabla reporte

Nombre: Usuario _organización		
Descripción: Contiene todos los contenidos de los usuarios y las organizaciones a donde pertenecen.		
Atributo	**Tipo**	**Descripción**
id_usuario_organización	String	Identificador de la tabla de usuario_organización
nombre_usuario_organización	varchar	Nombre de los usuario_ organización
contraseña	varchar	Contraseña del usuario
administrador	String	Para saber si es administrador o no
#organizaciónid_organización	String	Organización a la que pertenece el usuario_organización

Tabla 3.8 Descripción de la tabla usuario_organización.

3.5 Conclusiones del capítulo

- Con el análisis del sistema, se logró un refinamiento y mejor comprensión de los requisitos, así como una aproximación al modelo de diseño, realizando los diagramas de clases del análisis.
- La propuesta del diseño permitió modelar el sistema que se pretende construir, de manera que soporte todos los requisitos.
- Se realizó el diagrama de clases de diseño, logrando así definir las relaciones entre las interfaces, las clases, y la base de datos.
- Se realizó los diagramas de interacción para modelar los aspectos dinámicos del sistema.
- -Se realizó además el diagrama de despliegue el cual permitió describir cómo una aplicación se despliega a través de una infraestructura.
- Para garantizar el almacenamiento de la información del sistema se realizó el diseño de la base de datos, a partir de una identificación de las clases persistentes expresadas en su diagrama correspondiente y la generación del modelo de datos.

Capítulo 4: Implementación y prueba

Durante este capítulo se abordarán algunos aspectos importantes necesarios para la implementación como el diagrama de componentes. También se reflejarán en este capítulo las pruebas a la herramienta de apoyo para así comprobar la calidad con que ha sido llevada a cabo la proyección del sistema.

4.1 Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre sus elementos. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo, especificando los subsistemas de implementación y sus dependencias.

El modelo de implementación describe cómo se implementan los elementos del modelo de diseño en términos de componentes. Un componente se define como el empaquetamiento físico de los elementos de un modelo. El modelo de implementación describe además como se organizan los componentes y se interrelacionan entre sí.

Algunos estereotipos estándar de componentes son los siguientes:

- Ejecutable: Es un programa que se puede ejecutar en un nodo.
- Biblioteca: Es una biblioteca de objetos estática o dinámica.
- Tabla: Es una tabla de una BD.
- Archivo: Es un fichero que contiene código fuente o datos.
- Documento: Es un documento.
- Página Web: Es una página que se obtiene de la ejecución del sistema.

Los componentes tienen las siguientes características:

- Tienen relaciones de traza con los elementos del modelo que implementan.
- Pueden implementar varios elementos. Por ejemplo, varias clases. Sin embargo, la forma exacta en que se crea esta traza depende de cómo van a ser estructurados y modularizados los ficheros de código fuente, dado el lenguaje de programación que se esté usando.

A continuación se describe por detallados el diagrama de componente del sistema desarrollado. Donde se explica el propósito de cada componente utilizado en la aplicación:

Componente	Propósito
Autenticar	Permite a los usuarios autenticarse para poder interactuar con el sistema.
Gestionar Usuario_Organización	Permite al administrador poder crear, eliminar, buscar, actualizar usuarios organización, los cuales van hacer los encargados de trabajar con sus respectivas organizaciones.
Gestionar Organización	Permite crear, eliminar, buscar y modificar los datos de las organizaciones
Calcular éxito o fracaso	Permite a los usuarios que trabajan con sus organizaciones saber si están listas para iniciar un programa de mejoras
Reporte de Organizaciones	Muestra un listado con todas las organizaciones que hay inscritas en el sistema
Reporte Organizaciones con éxito	Muestra un listado con todas las organizaciones que tuvieron éxito en el programa de mejoras
Reporte Organizaciones con fracaso	Muestra un listado con todas las organizaciones que tuvieron fracaso en el programa de mejoras
Clase Acceso a datos	Controla la conexión hacia la base datos

Tabla 4.1 Descripción de los componentes

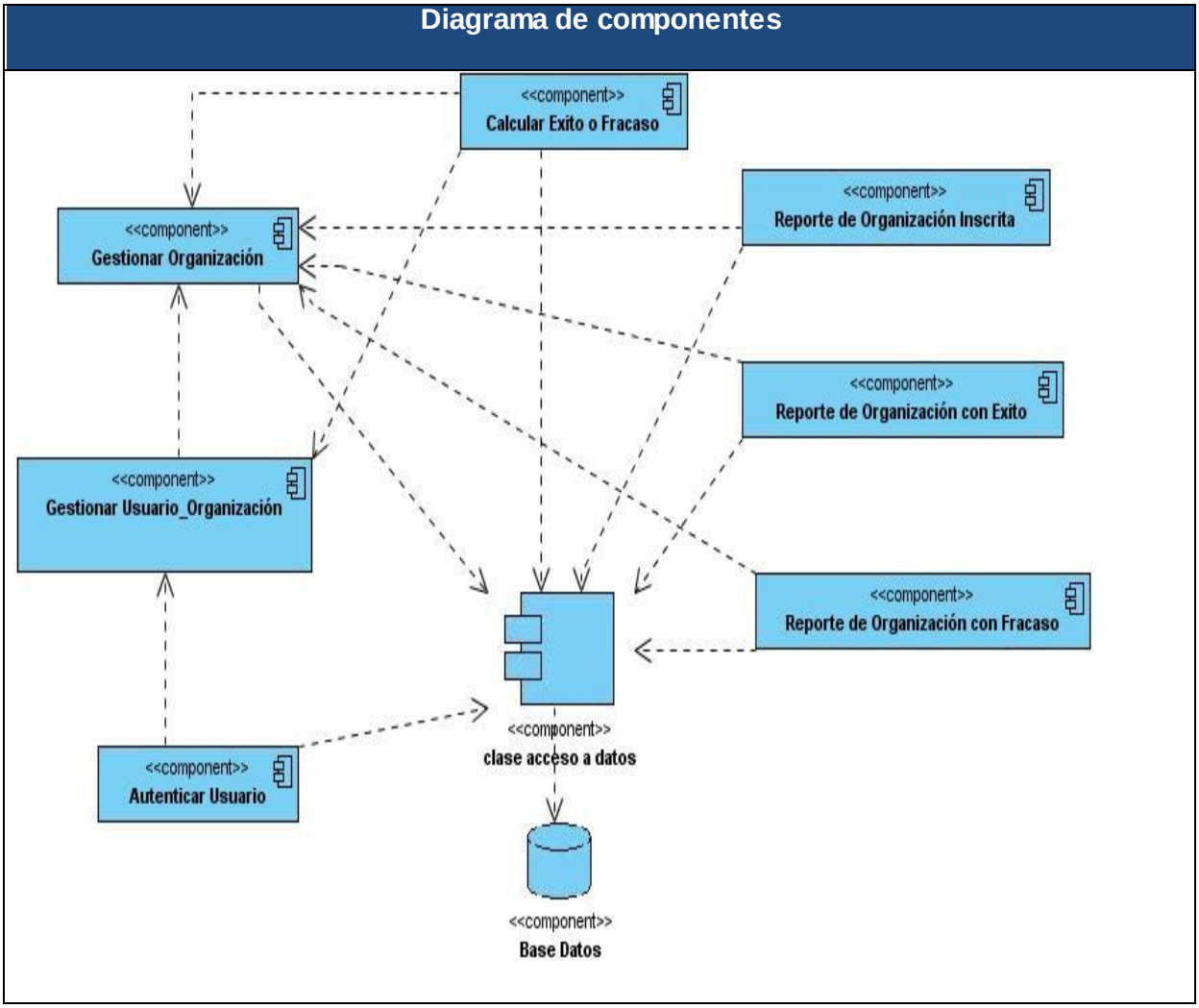

Figura 4.1 Diagrama de componentes.

4.2 Prueba

Luego de haber realizado un estudio de los tipos de pruebas existentes para probar el software desarrollado en esta tesis, se utilizó el paradigma de las prueba de tipo de caja negra, específicamente se implementaron prueba de particiones equivalente ya que la misma tiene como objetivo reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software. Se toma un riesgo porque se escoge no probar todo. Así que se necesita tener mucho cuidado al escoger las clases.

Caso de Prueba: Caso de Uso Autenticar Usuario

Caso de prueba	
Nombre: Autenticarse correctamente	**Nombre de la historia:** Autenticar Usuario
Descripción: El usuario intenta autenticarse insertando los datos correctamente	
Entrada: EL usuario llena los campos “Usuario” y “Contraseña” y pulsa el botón “Aceptar”.	
Resultados: EL sistema permite que el usuario entre correctamente	
Evaluación de la prueba: Satisfactoria	

Tabla 4.2 Caso de Prueba Autenticarse correctamente

Caso de prueba	
Nombre: Autenticarse Incorrectamente	**Nombre de la historia:** Autenticar Usuario
Descripción: El usuario intenta autenticarse insertando la contraseña incorrecta	
Entrada: EL usuario llena los campos “Usuario” y “Contraseña” y pulsa el botón “Aceptar”.	
Resultados: El sistema muestra el mensaje: “Usuario o contraseña incorrectos”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.3 Caso de Prueba Autenticarse incorrectamente (1)

Caso de prueba	
Nombre: Autenticarse Incorrectamente	**Nombre de la historia:** Autenticar Usuario
Descripción: El usuario intenta autenticarse insertando el usuario incorrecto	
Entrada: EL usuario llena los campos “Usuario” y “Contraseña” y pulsa el botón “Aceptar”.	
Resultados: El sistema muestra el mensaje: “Usuario o contraseña incorrectos”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.4 Caso de prueba Autenticarse incorrectamente (2)

Caso de Prueba: Caso de Uso Gestionar usuario _organización.

Caso de prueba	
Nombre: Crear correctamente	**Nombre de la historia: Crear**
Descripción: El administrador después de autenticarse llena los datos correctamente para crear a un usuario_organización y pulsa el botón Crear	
Entrada: El administrador llena todos los campos necesarios y pulsa el botón “Crear”	
Resultados: El sistema muestra un mensaje “Ha sido creado correctamente”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.5 Caso de Prueba Crear usuario _organización.

Caso de prueba	
Nombre: Crear Incorrectamente	**Nombre de la historia:** Crear
Descripción: El administrador después de autenticarse en el sistema selecciona la opción de “Crear”. El administrador introduce números en el campo del id usuario y pulsa el botón “Crear”.	
Entrada: El administrador introduce todos los datos en los campos y pone números en el campo de id usu y pulsa el botón “Crear”.	
Resultados: El sistema muestra el mensaje “No se puede escribir números”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.6 Caso de Prueba Crear usuario _organización incorrectamente (1)

Caso de prueba	
Nombre: Crear Incorrectamente	**Nombre de la historia:** Crear
Descripción: El administrador después de autenticarse en el sistema selecciona la opción de "Crear", el administrador introduce números en el campo del id usuario y pulsa el botón "Crear".	
Entrada: El administrador introduce todos los datos en los campos y pone un id _usuario o un nombre de Usuario que ya existe en la base datos y pulsa el botón "Crear".	
Resultados: El sistema muestra el mensaje "Ya existe un usuario _organización con ese id o con el mismo nombre".	
Evaluación de la prueba: Satisfactoria	

Tabla 4.7 Caso de Prueba Crear usuario _organización incorrectamente (2).

Caso de prueba	
Nombre: Crear Incorrectamente	**Nombre de la historia:** Crear
Descripción: El administrador después de autenticarse en el sistema selecciona la opción de "Crear", el administrador deja campos vacíos y pulsa el botón "Crear".	
Entrada: El administrador deja algunos campos vacíos y pulsa el botón "Crear".	
Resultados: El sistema muestra el mensaje "Debe llenar los campos vacíos".	
Evaluación de la prueba: Satisfactoria	

Tabla 4.8 Caso de prueba Crear usuario_organización incorrectamente (3)

Caso de Prueba: Caso de Uso Gestionar usuario organización.

Caso de prueba	
Nombre: Modificar Correctamente	**Nombre de la historia:** Modificar
Descripción: El administrador después de autenticarse en el sistema debe de seleccionar el usuario que quiere modificar y luego le da a la opción modificar. El administrador llena los datos correctamente y pulsa el botón “Modificar”	
Entrada: El administrador llena los datos correctamente y pulsa el botón “Modificar”	
Resultados: El sistema muestra el mensaje “Se modificaron los datos correctamente”	
Evaluación de la prueba: Satisfactoria	

Tabla 4.9 Caso de Prueba Modificar usuario _organización.

Caso de prueba	
Nombre: Modificar Incorrectamente.	**Nombre de la historia:** Modificar.
Descripción: El administrador después de autenticarse en el sistema debe seleccionar el usuario q quiere modificar y luego le da a la opción modificar. El administrador deja campos sin llenar y pulsa el botón “Modificar”.	
Entrada: El administrador deja algunos campos vacíos y pulsa el botón “Modificar”.	
Resultados: El sistema muestra el mensaje “Debe llenar los campos vacíos”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.10 Caso de Prueba Modificar usuario _organización incorrectamente (1).

Caso de prueba	
Nombre: Modificar Incorrectamente	**Nombre de la historia: Modificar**
Descripción: El administrador después de autenticarse en el sistema debe seleccionar el usuario que quiere modificar y luego le da a la opción modificar. El administrador entra números en el campo id usuario y pulsa el botón “Modificar”	
Entrada: El administrador entra números en el campo id usuario y pulsa el botón “Modificar”	
Resultados: El sistema muestra el mensaje “No se puede escribir números”.	
Evaluación de la prueba: Satisfactoria.	

Tabla 4.11 Caso de Prueba Modificar usuario _organización incorrectamente (2).

Caso de prueba	
Nombre: Eliminar correctamente	**Nombre de la historia:** Eliminar
Descripción: El administrador después de autenticarse en el sistema debe seleccionar el usuario que quiere eliminar y luego le da a la opción “Eliminar”. Luego de darle al botón eliminar sale una ventana confirmando que la eliminación ha sido satisfactoria.	
Entrada: El administrador luego de seleccionar el id _usuario pulsa el botón “Eliminar”.	
Respuesta: El sistema muestra el mensaje “Ha sido eliminado satisfactoriamente”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.12 Caso de Prueba Eliminar usuario _organización.

Caso de prueba	
Nombre: Buscar Correctamente	**Nombre de la historia:** Buscar
Descripción: El administrador después de autenticarse en el sistema tiene la opción de buscar los usuarios por id.	
Entrada: El administrador entra correctamente el nombre usuario y pulsa el botón “Buscar”.	
Resultados: El sistema muestra el usuario encontrado.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.13 Caso de Prueba Buscar usuario _organización.

Caso de prueba	
Nombre: Buscar incorrectamente	**Nombre de la historia:** Buscar
Descripción: El administrador después de autenticarse en el sistema tiene la opción buscar usuario por el id.	
Entrada: El administrador deja el campo nombre usuario vacío y pulsa el botón “Buscar”.	
Resultado: El sistema muestra el mensaje “Debe llenar los campos vacíos”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.14 Caso de Prueba Buscar Usuario _organización incorrectos (1).

Caso de prueba	
Nombre: Buscar incorrectamente	**Nombre de la historia:** Buscar
Descripción: El administrador después de autenticarse en el sistema tiene la opción buscar usuario por el id.	
Entrada: El administrador introduce números en el campo de nombre usuario y pulsa el botón “Buscar”.	
Resultado: El sistema muestra el mensaje “No se puede escribir números”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.15 Caso de Prueba Buscar Usuario _organización incorrectos (2).

Caso de prueba	
Nombre: Buscar incorrectamente	**Nombre de la historia:** Buscar
Descripción: El administrador después de autenticarse en el sistema tiene la opción buscar usuario por el id.	
Entrada: El administrador introduce un nombre usuario que no existe en la base datos y pulsa el botón “Buscar”.	
Resultado: El sistema muestra el mensaje “No se encuentra en la base datos”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.16 Caso de Prueba Buscar usuario _organización incorrectos (3).

Caso de Prueba: Caso de Uso Gestionar organización.

Caso de prueba	
Nombre: Crear Correctamente	**Nombre de la historia:** Crear
Descripción: El administrador después de autenticarse en el sistema selecciona la opción “Crear”. Luego aparece una ventana donde se llena los campos necesarios para adicionar una organización. Después de administrador llena todos los campos correctamente y pulsa el botón “Crear”.	
Entrada: El administrador llena los campos necesarios y pulsa el botón “Crear”.	
Resultados: El sistema muestra el mensaje “Se ha creado correctamente la Organización”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.17 Caso de Prueba Crear organización.

Caso de prueba	
Nombre: Crear Incorrectamente	**Nombre de la historia:** Crear
Descripción: El administrador después de autenticarse en el sistema selecciona la opción “Crear”. Luego aparece una ventana donde se llena los campos necesarios para adicionar una organización. El administrador deja algún campo vacío y pulsa el botón “Crear”.	
Entrada: El administrador deja algún campo vacío y pulsa el botón “Crear”.	
Resultados: El sistema muestra el mensaje “Error debe llenar los campos vacíos”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.18 Caso de Prueba Crear organización incorrectamente (1).

Caso de prueba	
Nombre: Crear Incorrectamente	**Nombre de la historia:** Crear
Descripción: El administrador después de autenticarse en el sistema selecciona la opción “Crear”. Luego aparece una ventana donde se llena los campos necesarios para adicionar una organización. El administrador introduce en el campo id Organización números y pulsa el botón “Crear”.	
Entrada: El administrador introduce en el campo id Organización números y pulsa el botón “Crear”.	
Resultados: El sistema muestra el mensaje “No puede escribir números”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.19 Caso de Prueba Crear organización incorrectamente (2)

Caso de prueba	
Nombre: Crear Incorrectamente	**Nombre de la historia:** Crear
Descripción: El administrador después de autenticarse en el sistema selecciona la opción “Crear”. Luego aparece una ventana donde se llena los campos necesarios para adicionar una organización. El administrador introduce en el campo id Organización números y pulsa el botón “Crear”.	
Entrada: El administrador introduce id _organización o nombre organización que ya existen en la base datos y pulsa el botón “Crear”.	
Resultados: El sistema muestra el mensaje “Ya existe una Organización con ese Id o Nombre”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.20 Caso de Prueba Crear organización incorrectamente (3)

Caso de prueba	
Nombre: Modificar Correctamente	**Nombre de la historia:** Modificar
Descripción: El administrador después de autenticarse en el sistema selecciona la opción “Modificar”. Luego cuando abra la ventana de modificar, selecciona la organización a modificar y llena los nuevos datos correctamente y pulsa el botón “Modificar”.	
Entrada: El **administrador** selecciona la organización a modificar y llena los nuevos datos correctamente y pulsa el botón “Modificar”.	
Resultados: El sistema muestra el mensaje “Se modificaron los datos correctamente”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.21 Caso de Prueba Modificar organización.

Caso de prueba	
Nombre: Modificar Incorrectamente	**Nombre de la historia:** Modificar
Descripción: El administrador después de autenticarse en el sistema selecciona la opción "Modificar". Luego cuando abra la ventana de modificar, selecciona la organización a modificar y al llenar los nuevos datos deja campos en blanco o introduce números y pulsa el botón "Modificar".	
Entrada: El administrador selecciona la organización a modificar y al llenar los nuevos datos deja campos en blanco o introduce números y pulsa el botón "Modificar".	
Resultados: El sistema muestra el mensaje "Debe llenar los campos en blanco o verificar que no haya números".	
Evaluación de la prueba: Satisfactoria.	

Tabla 4.22 Caso de Prueba Modificar organización incorrectamente (1)

Caso de prueba	
Nombre: Eliminar	**Nombre de la historia:** Eliminar
Descripción: El administrador después de autenticarse en el sistema debe seleccionar la organización que quiere eliminar y luego le da a la opción "Eliminar". Luego de darle al botón eliminar sale una ventana confirmando que la eliminación ha sido satisfactoria.	
Entrada: El administrador luego de seleccionar el id_ organización pulsa el botón "Eliminar".	
Resultados: El sistema muestra el mensaje "Ha sido eliminado satisfactoriamente".	
Evaluación de la prueba: Satisfactoria	

Tabla 4.23 Caso de Prueba Eliminar organización.

Caso de prueba	
Nombre: Buscar Correctamente	**Nombre de la historia:** Buscar
Descripción: El administrador después de autenticarse en el sistema tiene la opción de buscar las organizaciones por id.	
Entrada: El administrador entra correctamente el nombre de organización y pulsa el botón "Buscar".	
Resultados: El sistema muestra la organización encontrada.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.24 Caso de Prueba Buscar organización.

Caso de prueba	
Nombre: Buscar Incorrectamente	**Nombre de la historia:** Buscar
Descripción: El administrador debe entra correctamente el id y pulsa el botón “Buscar”.	
Entrada: El administrador deja el campo nombre_ organización vacío y pulsa el botón “Buscar”.	
Resultados: El sistema muestra el mensaje “Debe llenar los campos vacíos”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.25 Caso de Prueba Buscar organización incorrectamente (1)

Caso de prueba	
Nombre: Buscar Incorrectamente	**Nombre de la historia:** Buscar
Descripción: El administrador debe entra correctamente el id y pulsa el botón “Buscar”.	
Entrada: El administrador introduce números en los campos y pulsa el botón “Buscar”.	
Resultados: El sistema muestra el mensaje “No se puede escribir números”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.26 Caso de Prueba Buscar organización incorrectamente (2)

Caso de prueba	
Nombre: Buscar Incorrectamente	**Nombre de la historia:** Buscar
Descripción: El administrador debe entra correctamente el id y pulsa el botón “Buscar”.	
Entrada: El administrador introduce un nombre de Organización que no existe en la base datos.	
Resultados: El sistema muestra el mensaje “No se encuentra en la base datos”.	
Evaluación de la prueba: Satisfactoria	

Tabla 4.27 Caso de Prueba Buscar organización incorrectamente (3)

4.3 Conclusiones del capítulo

En este capítulo se obtuvo como resultado el diagrama de componentes y se vio el diseño de los casos de prueba de los procesos más importantes del *software* con la intención de detectar los posibles errores que este pueda tener para así erradicarlos.

Conclusiones

- El sistema informático facilita a la organización poder evaluar una serie de parámetros previamente definidos, lo cual le permitirá a dicha organización poder conocer su estado antes de empezar un programas de mejoras de procesos, teniendo esto gran importancia ya que evitaría , el fracaso del mismo, grandes perdidas económicas por parte de la empresa. Además evitaría que el personal de organización se volviera reacio a los cambios.
- La utilización de los métodos teóricos y empíricos permitió conocer el estado del objeto de estudio.
- La utilización de la metodología de desarrollo RUP permitió realizar un modelo de dominio debido a que los procesos involucrados en el negocio no se encontraban bien definidos y estaban difusos.
- La captura de requisitos constituye el punto de partida para el desarrollo de la aplicación propuesta en la investigación.
- Las tareas investigativas proporcionaron mayor organización durante la investigación.

Recomendaciones

Los objetivos de este trabajo han sido logrados, teniendo en cuenta que se cumplieron todos los requerimientos planteados. No obstante para futuras investigaciones y proyectos que guarden relación con este trabajo se hacen las siguientes recomendaciones:

- Continuar el estudio de este sistema adicionándole nuevas funcionalidades a la aplicación.
- Implementar el sistema en dominio genérico de manera que pueda ser utilizado en cualquier tipo de organización.
- Desarrollar el sistema en una plataforma web de manera tal que se mejore la interfaz gráfica de dicho sistema.

Referencia bibliográfica

[1].Los Alamitos, C. (2008). *"Software Requirements Engineering", 2nd Edition, IEEE Computer Society.*

[2].WITSA. (15 de 02 de 2006) World Information Technology and Services Alliance. Recuperado el 30 de 01 de 2010, de World Information Technology and Services Alliance: URL: http://www.witsa.org/

[3].Larman. (2005). *UML y el proceso de desarrollo.*

[4].Jacobson Ivar, Booch Grady, Rumbaugh James. El Proceso Unificado de Desarrollo de *Software*. Addison Wesley. (2000) (R, 2007)

[5].R, R. (2007). *Mejora de Procesos y Desempeño.* Recuperado el 17 de 03 de 2010, de Mejora de Procesos y Desempeño.: http://www.pwc.com/extweb/service.nsf/docid

[6].Villa M, R. M. (2008). *Modelos de Evaluación y Mejora de Procesos: Análisis Comparativo.*

[7].Martínez, I. S. (2009). *InfoCali*. Recuperado el 15 de 04 de 2010, de InfoCali: http://calisoft.uci.cu

[8].De la Cruz A.V., V. J. (1993). *Fundamentos y Práctica de la Construcción de Sistemas Expertos",.* La Habana: Editorial Academia.

[9].Yanet Kolodner, M. K. (1999). *Case-Based Reasoning.*

[10].Piñera Tapia, P.(1998) Razonamiento basado en caso: una herramienta para el abordaje del paciente epiléptico con trastorno psiquiátrico. Tesis para optar por Máster en Informática Médica.

[11].Rodríguez, M. Y. (2003). *Una herramienta de búsqueda inteligenteen bd utilizando técnica de RBC.* Ciudad Habana .

[12].Sanchez, M. A. (07 de 06 de 2004). *Metodologías De Desarrollo De Software*. Recuperado el 28 de 02 de 2010, de Metodologías De Desarrollo De *Software*: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_*software*_07062004.html

[13].Wesley, A. (2000). *Kruchten, P., The Rational Unified Process: An Introduction.*

[14].James Gosling, B. J. (2005). *The Java language specification, tercera edición. Addison-Wesley.*

[15].Stroustrup, M. A. (2000). *The Annotated C++ Reference Manual.*

[16].Sierra, J. C. (2008). *EL LENGUAJE DE PROGRAMACIÓN C#.* Ra_ma.

[17].Wendy Boggs, M. B. (2002). *“Mastering UML with Rational Rose”.*

[18].Zhao, J. y. (2005). *"Comparación de Herramientas de modelado UML".*

Bibliografía

1. QUIÑONES, E. *Introducción a PostgreSQL.* [Consultado el: 20 de marzo de 2010] Disponible en: http://www.postgresql.org.pe/articles/introduccion_a_postgresql.pdf.

2. SÁNCHEZ, J. I. P. *Metodología para el Desarrollo de Software.* 2005, Disponible en: www.lcc.uma.es/~jignacio/index_archivos/TEMA4.pdf.

3. TIGRIS.ORG COMMUNITY. *Open Source Software Engineering Tools* [Consultado el: 14 enero de 2010]. Disponible en: http://argouml.tigris.org/.

4. VISUAL PARADIGM INTERNATIONAL. *Visual Paradigm for UML* [Consultado el: 14 enero de 2010]. Disponible en: http://www.visual-paradigm.com/product/vpuml/.

5. BASE DE DATOS ORIENTADA A OBJETOS. [Citado el: 30 de 01 de 2010.] http://www.mhproject.org/media/blogs/mhpenlaces/Interno/Presentaciones/db4objects/db4objects.pdf.

6. El equipo de desarrollo de PostgreSQL. [Citado el: 25 de 02 de 2010.] http://sdi.bcn.cl/desarrollo/doctos/PostgreSQL%20-%20Tutorial.pdf.

7. Herramientas case. [Citado el: 07 de 12 de 2009.] http://www1.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf.

8. Herramientas CASE para BD. [Citado el: 10 de 12 de 2009.] http://www.monografias.com/trabajos24/herramientas-case/herramientas-case.shtml.

9. PostgreSQL. Una Alternativa de DBMS Open Source. [Citado el: 16 de 01 de 2009.] http://www.lgs.com.ve/pres/PresentacionES_PSQL.pdf.

10. Rational Rose. [Citado el: 16 de 12 de 2009.] http://www.slideshare.net/vivi_jocadi/rational-rose.

11. Visual Paradigm For Uml. [Citado el: 07 de03 de 2010.] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

12. Visual Paradigm for UML (ME). [Citado el: 25 de 03 de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/.

13. Introducción a la inteligencia artificial: Los sistemas expertos [Citado el: 28 de 04 de 2010] http://www.doi.icai.upco.es/simio/transpa/t_se_bv.pdf

14. Inteligencia Artificial y Sistemas Expertos [Citado el: 28 de 03 de 2010]
http://www.slideshare.net/CarlosPalacios/inteligencia-artificial-y-sistemas-expertos

15. STUART, RUSSELL; PETER, NORVIG
1996 Inteligencia artificial, un enfoque moderno. Ed. Prentice Hall. ISBN 0-13-103805-2

16. PATRICK, WINSTON 1984 Inteligencia artificial. Ed. Addison-Wesley ISBN 0-201-51876-7

17. ELAINE, RICH 1988 Inteligencia artificial. Ed McGraw-Hill ISBN 0-07-450364-2

18. La idea, el programa Mycin y las Reglas de Inferencia [Citado el: 05 de 04 de 2010]
http://www.unav.es/asignaturas/ia/tsld012.htm

19. Recursión, Backtracking y Unificación, explicación de búsquedas en PROLOG [Citado el: 07 de 02 de 2010]
http://www.uhu.es/nieves.pavon/p2/tema2/2punto12.html.

20. Información sobre Base de Conocimientos [Citado el: 07 de 01 de 2010]
http://www.cs.cinvestav.mx/SC/publica/chapa/intro_lm/node46.html#SECTION00611100000000000000

21. Información sobre Máquina de Inferencia [Citado el: 23 de 03 de 2010]
http://www.cs.cinvestav.mx/SC/publica/chapa/intro_lm/node47.html#SECTION00611200000000000000.

22. Características de los Sistemas Expertos [Citado el: 01 de 05 de 2010]
http://iwia.sis.epn.edu.ec/~hbanda/heuristica/pagina6_6.html

Glosario de términos

UCI: Universidad de las Ciencias Informáticas

TIC: Tecnologías de la información y la comunicación

CHAOS: Organización internacional de carácter inconformista cuyos componentes se definen como portavoces de los piratas informáticos (hackers) de todo el mundo y demandan una sociedad con libertad ilimitada y flujos de información sin censuras.

SE: Sistemas expertos.

CMMI: Integración de Modelos de Madurez de Capacidades (CMMI) es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de *software*.

RBC: Razonamiento basado en casos.

BD: Bases de datos

BC: Bases de casos.

RUP: El Proceso Unificado Racional (habitualmente resumido como RUP) es un proceso de desarrollo de *software* y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

UML: Lenguaje Unificado de Modelado (UML). Es un lenguaje de modelado de sistemas de *software*.

TWG: Grupo técnico de trabajo.

SEI: (Instituto de Ingeniería de *Software*) Es la universidad estadounidense Carnegie Mellon, es un instituto independiente, que goza de un elevado prestigio internacional.

TCP/IP: Se refieren a dos protocolos de red: Transmission Control Protocol (Protocolo de Control de Transmisión) e Internet Protocol (Protocolo de Internet).