Universidad de las Ciencias Informáticas
Facultad 3

“Desarrollo de un sistema basado en casos para la gestión de errores en el Proyecto ERP-Cuba”.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor: Juan Alberto Caballero Portelles

Tutores:

- Ing. Yusnier Matos Arias
- Ing. Joisel Pérez Pérez

Co-tutor:

- Ing. Yanay Hernández Sosa

Ciudad de la Habana, Junio de 2011. “Año 53 de la Revolución”.

"...aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana. Y en esta producción, en esta dirección, está comprendida la producción de sí mismos..."

Che

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Juan Alberto Caballero Portelles	**Yusnier Matos Arias**
Firma del Autor	Firma del Tutor

Agradecimientos

Agradecer especialmente a toda mi familia por estar siempre ahí para mí.

Agradecer a todos mis amigos, especialmente a Aylin, Elizabeth, Adnerys, Lianet, Manuel, Vingut y a todos los demás que son como hermanos míos y siempre los recordaré como personas que compartimos grandes momentos durante todos estos años.

Agradecer también a mi tutor, al tribunal y todos aquellos que me ayudaron para que este trabajo se realizara y que mi deseo de graduarme de una carrera universitaria se cumpliera.

Agradecer a todos.

Dedicatoria

Dedico este trabajo primeramente a mis padres por todo el apoyo que siempre me han dado y porque sin ellos yo no sería la persona que soy.

A mi maravillosa familia, que sin su apoyo el camino recorrido hasta aquí tal vez no sería tan maravilloso y lleno de momentos felices como lo ha sido.

A mis tías María, Paquita, Elia, Cachita a todas, a todos mis tíos, a Teresa y Carlos, a mis primos, Ramiro, Rogelio, La Nena, Amaury, Rober, Vicente, Rafael, Ricardo, Pipo, y a todos los demás.

Resumen

Las empresas cubanas no están al margen de las exigencias que el avance tecnológico actual demanda, por lo que se enmarcan en el proceso de informatización de sus entidades. Uno de estos procesos es la creación de un Sistema de Gestión Integral denominado Cedrux destinado para la gestión económica de las entidades del país que actualmente se encuentra en desarrollo por la Universidad de las Ciencia Informáticas y otras entidades nacionales. Dentro del desarrollo de este software, como con cualquier otro, se presentan a menudo situaciones que muchas veces se hace necesario dedicarle tiempo y personal para solucionarlas.

Para buscar que los procesos en este desarrollo de software sobrevivan a las personas se hace necesario que estas situaciones queden registradas. En el presente trabajo se hace una propuesta de solución de un sistema que tiene como primicia dar de forma rápida mediante consulta, una solución a problemas que se presenten durante el proceso de desarrollo del Proyecto ERP-Cuba debido a que los errores que se producen en este proyecto actualmente no son gestionados por ningún software, haciendo engorroso y tardío el proceso para resolverlos. El sistema que se propone se construirá en base al uso de técnicas de la Inteligencia artificial y utilizando diferentes algoritmos y funciones para el cálculo de distancia que son utilizadas en muchos sistemas expertos existentes en la actualidad.

PALABRAS CLAVES

Sistema experto, software, soluciones, errores.

Índice

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC's), han alcanzado un elevado desarrollo, desempeñando un importante papel dentro de las estrategias de negocio y las distintas formas en que se desarrollan los procesos dentro de las empresas. En toda empresa, es necesario realizar toma de decisiones que son a menudo cruciales para el correcto desempeño de la misma y muchas veces estas decisiones son tomadas gracias a la ayuda de software especialmente diseñado para este fin. Estos programas generalmente son sistemas expertos (SE) y logran su objetivo gracias a que basan su funcionamiento en la simulación del razonamiento humano. (1)

Esta tecnología ha demostrado sus posibilidades en áreas como en los diagnósticos médicos con el (MYCIN), o que combinan el diagnóstico y facturación para hospitales como el (HELP), o la exploración de minerales y búsqueda de petróleo con el (PROSPECTOR), o el diseño de ordenadores con (XCOIM), entre otros. (2)

Una de las áreas donde estos sistemas han incursionado con gran fuerza es dentro del sector económico, siendo este sector, debido a la gran utilización del recurso información, uno de los motores del desarrollo de las Tecnologías de la Información y ha sido uno de los pioneros en el desarrollo de los primeros Sistemas Expertos. (2)

Se puede señalar la importancia que estos sistemas consiguen tener para las sucursales bancarias, entidades financieras, fábricas o empresas de cualquier tipo, independientemente de las aplicaciones específicas que puedan desarrollarse dentro de las mismas para los más diferentes usos como por ejemplo, la recopilación y el análisis de grandes cantidades de datos entre otras.

La Universidad de las Ciencias Informáticas (UCI) en conjunto con el Ministerio de Finanzas y Precios y la participación de especialistas de otras entidades desarrolladoras de software actualmente se encuentran vinculados en la realización de un Sistema Integral de Gestión de Entidades denominado CEDRUX, el cual es un producto del Proyecto ERP-Cuba de la UCI, con el objetivo de responder a las necesidades contables de las entidades cubanas de modo que modele y automatice la mayoría de sus procesos. Su misión es facilitar la planificación de todos los recursos y la integración de la información de la empresa.

El proyecto está organizado por subsistemas y módulos que tributan a algunas de las áreas de negocio de una empresa como son Contabilidad, Capital humano, Logística, Planificación, Costos y Procesos entre otros. Dentro del desarrollo del mismo interviene un número bastante amplio de desarrolladores, analistas y funcionales expertos en las diferentes áreas de proceso de una empresa sin mencionar toda la tecnología y recursos usados dentro del mismo. Debido al gran flujo de trabajo y las grandes cantidades de información que se mueven dentro del proyecto, surgen a menudo problemas de diferentes tipos y magnitudes, que una vez que son solucionados, no quedan registros de dicha solución para una posterior revisión o reutilización de la misma en caso de que volviese a ocurrir el mismo problema o uno similar en otra área de desarrollo.

Estos problemas se dan en todas las líneas de trabajo, y muchas veces se hacen bastante comunes y repetitivos, siendo necesario en algunas ocasiones, poner a varias personas a buscar una solución a los mismos, teniendo estas que abandonar el trabajo que realizan para dedicarse a ello, significando esto la pérdida de tiempo útil de desarrollo que muchas veces es tan escaso y necesario para el proyecto. Durante el desarrollo de un proyecto no puede haber pérdida de tiempo de trabajo ni atraso en la producción porque el personal que trabaja en él está enfocado en la corrección de errores que posiblemente ya hayan sucedido alguna vez y que alguien en ese momento lo solucionó pero no dejó constancia de ello.

En el Proyecto ERP-Cuba, se tomó como iniciativa llevar la gestión de los errores en un software desarrollado para este fin, pero dicho software solo almacenaba la descripción del problema y la solución que se le había dado a este, no siendo esta la mejor forma para llevar la gestión de los errores. Por ejemplo, si ocurre un error o problema, se introduce la descripción de dicha situación en el programa y si en la base de datos no existe una situación con la misma descripción, este no será capaz de devolver una solución pues estará frente a algo nuevo, a algo que no tiene precedentes para él.

Ahora bien, si el personal del proyecto contase con algún software que fuese capaz de, dada una situación, buscar todos los casos posibles, exactos o similares; teniendo en cuanta las condiciones, datos o parámetros introducidos, y este software brindara la o las soluciones más cercanas o acercadas a la que se necesitan, dicho personal estaría frente a una solución bastante buena para el proyecto, pues, los

recursos humanos y el tiempo dedicado a la búsqueda de soluciones para los errores que puedan ocurrir sería en la mayoría de los casos el menor posible.

En el presente trabajo de diploma, para dar solución a la situación descrita con anterioridad se define el siguiente **problema a resolver**: La gestión insuficiente de los errores que ocurren durante el proceso de desarrollo del Proyecto ERP-Cuba afecta el tiempo de desarrollo e incrementa el trabajo de los integrantes del proyecto, afectando además los cronogramas de entrega en algunos casos.

Se define por lo tanto como **objeto de estudio**: Los errores que se producen durante el proceso de desarrollo del Proyecto ERP-Cuba.

El **objetivo general** del presente trabajo de diploma es: Desarrollar un sistema basado en casos para la gestión de los errores que ocurren durante el proceso de desarrollo del Proyecto ERP-Cuba.

Objetivos específicos:

- ❖ Fundamentar la investigación, mediante la elaboración del Marco Teórico.
- ❖ Diseñar e implementar el sistema basado en casos para la gestión de los errores durante el proceso de desarrollo del Proyecto ERP-Cuba.
- ❖ Validar el sistema implementado.

El **campo de acción** se enmarca en los errores de integración producidos en el proceso de desarrollo del sistema CEDRUX del Proyecto ERP-Cuba.

Para darle cumplimiento a los objetivos propuestos, han sido definidas las siguientes **tareas**:

- ❖ Estudio del estado del arte referente a soluciones informáticas asociadas a los sistemas basados en casos.
- ❖ Realización del marco teórico conceptual de la investigación.
- ❖ Investigación de patrones de diseño, estándares de codificación, herramientas y tecnologías definidos por el Proyecto ERP-Cuba.
- ❖ Diseño e implementación del sistema.

- ❖ Elaborar los casos de pruebas para la aplicación.
- ❖ Validación del sistema implementado a través de pruebas al finalizar el proceso de desarrollo.

Idea a defender: Si se desarrolla un sistema basado en casos para la gestión de errores en el Proyecto ERP-Cuba, se mejorará la gestión de los errores y el tiempo para resolver los mismos.

En cumplimiento a las distintas tareas antes mencionadas, se pusieron en práctica los siguientes **métodos de investigación:**

Métodos teóricos:

- ❖ **Analítico – sintético:** Posibilitando procesar toda la información enfocada hacia la investigación de los sistemas basados en casos, permitiendo organizar y simplificar el análisis de todo el volumen de datos a recopilar en fracciones más factibles.

- ❖ **Histórico – lógico**: Para conocer los antecedentes y tendencias actuales en los sistemas basados en casos, las plataformas y lenguajes de implementación.

Métodos empíricos:

- ❖ **Experimento**: Favoreciendo el desarrollo de pruebas para la verificación de las funcionalidades implementadas, con el fin de detectar errores y comprobar el correcto funcionamiento del negocio.

- ❖ **Observación:** Para percibir y planificar como quedaría concebido el sistema.

Capítulo 1: Fundamentación teórica

1.1 Introducción.

En el presente capítulo se hará mención a grandes rasgos de lo que es la inteligencia artificial y se centrará específicamente en la explicación de lo que es un sistema experto basado en casos. También se procederá en la realización del análisis y obtención de información actualizada de algunos algoritmos, métodos y funciones utilizadas en la recuperación de casos de las bases de conocimiento. Se valorarán las tendencias actuales partiendo de que continuamente surgen aplicaciones novedosas con mejoras que se van imponiendo en la industria del software. Será explicado el Modelo de desarrollo elaborado a partir del cual estará orientada la solución. Se realizará un análisis de la Arquitectura definida para el Sistema y se especificarán las tecnologías y herramientas de desarrollo a utilizar durante la implementación para facilitar el cumplimento de los requerimientos tanto técnicos como funcionales de la aplicación.

1.2 Sistemas expertos (SE).

Antes de realizar un análisis sobre los sistemas expertos es importante abordar lo que llamamos Inteligencia Artificial (IA), la cual es la base que dio origen a los SE. Existen varias definiciones de lo que es la Inteligencia Artificial, como por ejemplo:

"*La Inteligencia Artificial es la interesante tarea de lograr que las computadoras piensen... máquinas con mente, en su amplio sentido literal*". (John Haugeland, 1995).

"Un campo de estudio que se enfoca a la explicación y estimulación de la conducta inteligente en función de procesos computacionales". (Schalkoff, 1990).

Pero uno de los conceptos más reconocidos y aplicados para esta definición es el que plantea a la Inteligencia Artificial como:

"El estudio de cómo lograr que las computadoras realicen tareas que por el momento, los humanos hacen mejor". (Rich & Knight, 1991).

Los Sistemas Expertos constituyen una técnica de la Inteligencia Artificial y basan su funcionamiento en la simulación del razonamiento humano, que tiene para ellos un doble interés, primeramente el de analizar el razonamiento que seguiría un experto humano en la materia con el fin de almacenarlo, y por otro lado, la síntesis artificial, de tipo mecánico, de los razonamientos de tal manera que estos sean semejantes a los empleados por el experto en la resolución de una cuestión dada. Estos sistemas cuentan con el apoyo del almacenamiento del conocimiento relativo a un campo muy concreto y que, mediante una serie de inferencias, producen la respuesta que un experto daría.

La tecnología representada por los SE actuales, surge de las técnicas de Inteligencia Artificial que han sido objeto de amplias e intensivas investigaciones desde finales de los años 50. Ya para mediados de los años 60 se comenzó el estudio de los SE aún con un alcance limitado, el cual se orientaba principalmente hacia los juegos o temas altamente académicos e idealizados. Posteriormente se iniciaron desarrollos en los campos de la medicina, la química, la industria y la administración. (3)

Estos sistemas son empleados en la ejecución de una variedad de tareas que en el pasado solamente podían llevarse a cabo por un número limitado de personas expertas y actualmente pueden ser realizadas por cualquier persona siempre y cuando se apoye en estos sistemas. Aunque los componentes de un Sistema Experto no se manejan de forma estándar, se cuenta con un modelo tradicional en la estructura de los mismos:

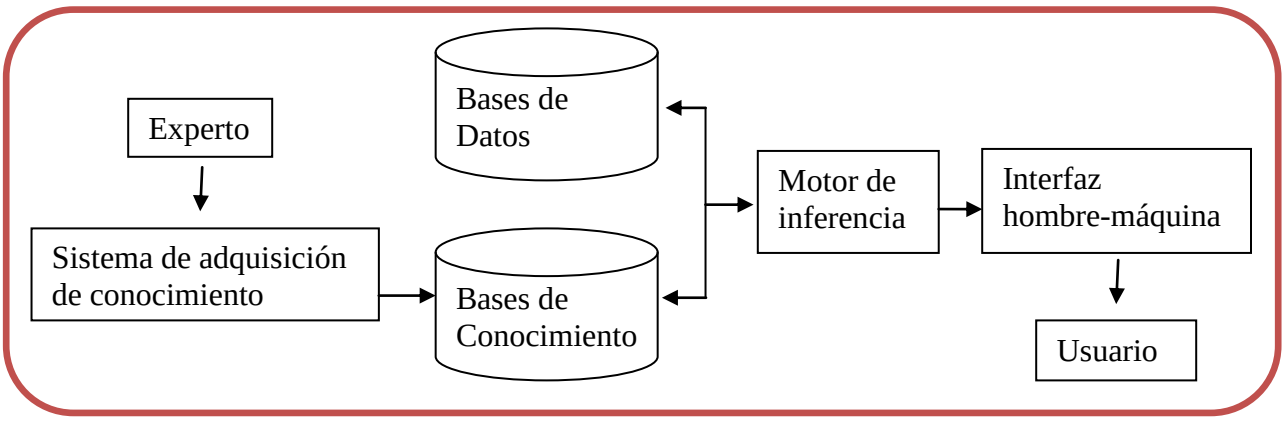

Figura 1. Esquema representativo de la composición de un sistema experto. (4)

El conocimiento del experto se obtiene a través del Subsistema de adquisición de conocimientos y se organiza en una base de conocimientos, y en función de los datos disponibles de la aplicación en la base de hechos o base de datos se imita la forma de actuar del experto explorando en la base de conocimientos hasta encontrar la solución mediante el motor de inferencia. Los resultados finales y la forma en que se obtienen se expresan a través de la interfaz hombre-máquina. Además algunos sistemas expertos manejan un componente de explicación. A continuación se describen dichos componentes:

- Subsistema de adquisición de conocimientos

El módulo de adquisición del conocimiento permite que se puedan añadir, eliminar o modificar elementos de conocimiento (lo que serian nuevos casos) en el sistema experto. Si el entorno es dinámico es muy necesario, pues el sistema funcionará correctamente sólo si se mantiene actualizado su conocimiento. El módulo de adquisición permite efectuar ese mantenimiento, anotando en la base de conocimientos los cambios que se producen. Todos los conocimientos que se obtienen deben ser estructurados de una forma correcta, todo este conocimiento se almacena en lo que se conoce como la base de conocimientos.

- Base de conocimientos

La base de conocimientos contiene el conocimiento especializado extraído del experto en el dominio. Es decir, contiene conocimiento general sobre el dominio en el que se trabaja.

- Base de datos o Base de hechos

La base de datos o base de hechos es una parte de la memoria del ordenador que se utiliza para almacenar los datos recibidos inicialmente para la resolución de un problema. Contiene conocimiento sobre el caso concreto en que se trabaja. También se registrarán en ella las conclusiones intermedias y los datos generados en el proceso de inferencia. Todos estos datos no son suficientes, si no se tiene un sistema encargado de procesar y manipular toda la información para generar los resultados deseados, este sistema es conocido como Motor de inferencia.

- Motor de inferencia

El motor de inferencia es el "supervisor", un programa que está entre el usuario y la base de conocimientos y que extrae conclusiones a partir de los datos simbólicos que están almacenados en las bases de hechos y de conocimiento.

- Interfaz hombre-máquina

La interfaz establece una comunicación sencilla entre el usuario y el sistema. El usuario puede realizar consultas con el sistema a través de diferentes medios como son menús o algún otro tipo de interfaces y éste le responde con resultados. También es interesante mostrar la forma en que extrae las conclusiones a partir de los hechos. En sistemas productivos se cuida la forma de presentar al operador las órdenes obtenidas del sistema experto, debido a que información excesiva o confusa dificulta la actuación en tiempo real.

- Componente de explicación

El componente de explicación es el que permite justificar y explicar el análisis completo del problema y las soluciones propuestas, así como la semejanza o diferencia entre dicha solución y las de los casos históricos.

Existen varios tipos de Sistemas Expertos, entre los que se destacan:

- Basados en reglas.
- Basados en casos o CBR (Case Based Reasoning).
- Basados en redes bayesianas.

En cada uno de ellos, la solución a un problema planteado se puede obtener aplicando reglas heurísticas apoyadas generalmente en lógica difusa para su evaluación y aplicación, aplicando el razonamiento basado en casos, donde la solución a un problema similar planteado con anterioridad se adapta al nuevo problema o aplicando redes bayesianas, basadas en estadística y el teorema de Bayes. [1]

[1] **Thomas Bayes,** (1702-1761) matemático británico. Estudió el problema de la determinación de la probabilidad de las causas a través de los efectos observados. El teorema que lleva su nombre se refiere a la probabilidad de un suceso condicionado por la ocurrencia de otro suceso.

Los Sistemas Expertos poseen ciertas características que los hacen muy útiles, entre ellas podemos encontrar las siguientes:

- ❖ Pueden explicar su razonamiento o decisiones sugeridas: La capacidad de explicar cómo se llegó a una decisión o solución.
- ❖ Pueden mostrar un comportamiento "inteligente": Al examinar un grupo de datos, un sistema experto puede proponer nuevas ideas o métodos para la solución del problema, o proporcionar asesoramiento en el trabajo para los trabajadores.
- ❖ Pueden obtener conclusiones de relaciones complejas: Evaluar relaciones complejas para llegar a conclusiones y solucionar problemas, por ejemplo: un Sistema Experto propuesto trabajará con un sistema de fabricación flexible para determinar la mejor utilización de las herramientas, y otro sugerirá los mejores procedimientos de control de calidad.
- ❖ Pueden proporcionar conocimientos acumulados: Se puede usar para capturar conocimientos de humanos que de lo contrario podrían perderse.
- ❖ Pueden hacer frente a la incertidumbre: Tienen capacidad para enfrentar conocimientos incompletos o inexactos en su totalidad mediante el uso de las probabilidades, las estadísticas y las heurísticas.

Pero también existen algunas características que limitan su utilidad y que debemos tener presentes:

- ❖ No se han usado o probado en forma extensa.
- ❖ Algunos sistemas expertos son difíciles de controlar y usar.
- ❖ No pueden enfrentar con facilidad conocimientos "mixtos".
- ❖ Sólo manejan áreas precisas del conocimiento.
- ❖ Tienen una capacidad limitada de aprendizaje.
- ❖ No poseen sentido común.
- ❖ Posibilidad de error.
- ❖ Dificultad de mantenimiento.

1.3 Sistemas, algoritmos y funciones existentes vinculados al campo de acción.

Anteriormente se mencionaron las partes fundamentales que componen un SE y dentro de ellas se hizo mención al motor de inferencia. Dicho motor de inferencia para ejercer su función hace uso de la recuperación de los casos dentro de la base de conocimientos o base de datos, por lo que dicho proceso es crucial para el sistema, pues de él depende la elección de los mejores casos para solucionar un determinado problema.

Se puede decir que al contrario de los procesos de búsqueda en las bases de datos, donde se tiene un objetivo bien determinado, en los Sistemas basados en casos (SRBC), el proceso de búsqueda, debe emplear herramientas capaces de devolver casos semejantes, no necesariamente iguales. Esto significa que un caso recuperado no tiene que ser necesariamente igual a la consulta realizada (nuevo caso o caso a comparar). El resultado de este proceso en los SE, debe devolver el o los casos que sean suficientemente semejantes al nuevo caso. La búsqueda de los casos semejantes puede realizarse utilizando funciones de semejanza o distancia como también son llamadas.

Existen varias funciones de distancias que han sido propuestas y utilizadas para recuperar casos semejantes. Muchas de ellas trabajan con atributos numéricos y no son apropiadas para atributos simbólicos, otras permiten buscar semejanza en una estructura determinada. A continuación veremos algunas de estas funciones. (5)

1.3.1 Funciones de distancia para casos con características numéricas e información completa.

En las funciones que se analizan a continuación se considera que:

- ***x*** e ***y***, son los casos a comparar.
- x_i, y_i, son los valores de ***x*** e ***y***, para el atributo ***i.***
- ***n*** es la cantidad de características de los casos que se analizan.

1. **Distancia Euclidiana.**

Para el cálculo de semejanza en los dominios numéricos, la función más comúnmente utilizada es la distancia euclidiana, la cual se define como: (6)

$$\mathrm{d}(\mathrm{x},\mathrm{y}) = \sqrt{\left[\sum_{i=1}^{n}(xi - yi)^2\right]}$$

En esta expresión se puede observar que todas las características influyen de igual manera en la semejanza, o sea tienen igual peso.

2. **Distancia de Manhattan.**

Una función alternativa de la distancia euclidiana es la distancia absoluta, (llamada de Manhattan) que se representa como la diferencia absoluta sobre todas las dimensiones. En la mayoría de las aplicaciones, esta genera resultados similares a la distancia Euclidiana, pero con más bajo nivel computacional. Está definida por la siguiente expresión:

$$\mathrm{d}(x,y) = \sum_{i=1}^{n}|xi - yi|$$

1.3.2 Funciones de distancia para los casos donde las características que los representan son simbólicas.

Cuando todos los atributos son simbólicos o nominales una solución es recurrir a una métrica general, la cual cuenta el número de atributos que tienen valores diferentes para los casos. Una de estas distancias es la distancia de Hamming.

1. **Distancia de Hamming**

Para obtener la semejanza entre los casos, se cuenta el número de atributos diferentes.

Por ejemplo sean 2 vectores de 4 componentes:

c1 = (a, b, a, c) y *c2 = (a, f, a, c)*

La distancia entre c1 y c2 es 1.

Esta distancia también puede ser utilizada cuando todos los atributos son booleanos. La cantidad de atributos que tienen en común los ejemplos, indican los valores de semejanza. Los casos con mayor cantidad de atributos iguales, serán los más semejantes.

1.3.3 Funciones para calcular la distancia en dominios de problemas donde las características son numéricas y no numéricas incluyendo información incompleta.

En las distintas aplicaciones, las características o rasgos que describen a los problemas no necesariamente son todas numéricas, pueden existir casos con todas las características representadas como valores simbólicos o una combinación de valores numéricos y simbólicos con o sin información incompleta, considerándose como información incompleta, los valores desconocidos en un momento dado para un atributo. Para estos casos son usadas funciones de similitud como la que será explicada a continuación:

1. **Métrica Heterogénea Euclidiana Generalizada (HEOM)**

Una de las funciones desarrolladas para analizar los problemas con atributos numéricos y simbólicos con o sin información incompleta es la Métrica Heterogénea Euclidiana Generalizada (HEOM). (7)

En este caso la distancia entre ***x*** e ***y*** para un valor de atributo *i* se define como:

$$\mathrm{d}(\mathrm{x},\mathrm{y}) = \begin{cases} 1 \text{ si } xi \text{ o } yi \text{ son desconocidos.} \\ \text{overlap } (xi, yi) \text{ si } i \text{ es un atributo nominal.} \\ \text{valor del rango normalizado}\big(\text{rn\_difi } (xi, yi)\big) \text{ en cualquier otro caso} \end{cases}$$

Donde:

Los valores desconocidos del atributo, se manejan devolviendo el valor de distancia 1 (máximo valor de distancia).

La función ***overlap***, analiza los atributos nominales (simbólicos).

Se definen como:

$$\text{overlap}(x, y) = \begin{cases} 0 \text{ si } xi = yi \\ 1 \text{ en otro caso} \end{cases}$$

Para los valores numéricos se calcula el valor del rango normalizado, el cual está dado por la siguiente expresión:

$$\text{rn\_dif}(x, y) = \frac{|xi - yi|}{\text{rango}i}$$

Donde $\text{rango}_i = \text{max}_i - \text{min}_i$

Donde $\mathbf{max_i}$ y $\mathbf{min_i}$ son los valores máximos y mínimo, observados en el conjunto de entrenamiento para el atributo ***i***.

Para la comparación entre todos los atributos que conforman el caso, se utiliza la siguiente expresión.

$$\text{HEOM}(x, y) = \sqrt{\sum_{i=1}^{n} \text{d}i(xi, yi)^2}$$

Como se puede apreciar esta función utiliza diferentes funciones de distancia para diferentes tipos de atributos, si todos los atributos son simbólicos la simplificación de la función conduce a la distancia de Hamming.

Dichas funciones por sí solas no son las únicas encargadas de la búsqueda de los casos. Los SE hacen uso de algoritmos que se auxilian en estas y otras funciones para realizar las búsquedas de los resultados requeridos. Como ejemplos de estos algoritmos a continuación hacemos referencia a varios de ellos.

1.3.4 Método del vecino más cercano o Knn (K nearest neighbors)

El Knn es un método simple y calcula la semejanza entre casos almacenados y una entrada nueva, basándose en las características de los mismos. Este método hace uso de la distancia que existe entre un par de casos, dígase el caso buscado (x) y el caso almacenado (y) donde mientras más pequeña sea esa distancia, más se parecerá el caso (x) al caso (y). Es decir, este método recuperará casos semejantes, en dependencia del valor de la distancia entre los dos puntos. Es en el cálculo de dicha distancia donde entran a jugar su papel las funciones antes descritas.

Este método posee las siguientes características:

- Las reglas de clasificación por vecindad están basadas en la búsqueda en un conjunto de prototipos de los k prototipos más cercanos al patrón a clasificar. (8)
- No hay un modelo global asociado a los conceptos a aprender.
- Las predicciones se realizan basándose en los ejemplos más parecidos al que hay que predecir.
- El coste del aprendizaje es 0, todo el coste pasa al cálculo de la predicción.
- Se conoce como mecanismo de aprendizaje perezoso (lazy learning[2]).

Ventajas que presenta

- El coste del aprendizaje es nulo. (8)
- No necesitamos hacer ninguna suposición sobre los conceptos a aprender.
- Podemos aprender conceptos complejos usando funciones sencillas como aproximaciones locales.
- Es muy tolerante al ruido.

Inconvenientes

- No hay un mecanismo para decidir el valor óptimo para k (depende de cada conjunto de datos).
- Su rendimiento baja si el número de descriptores crece.
- Su capacidad de ser comprendido es nula (no hay una descripción de los conceptos aprendidos).

[2] **Lazy learning** es un método de aprendizaje en el que la formación de datos no es generada hasta que se realiza una consulta al sistema, contrario al aprendizaje ansioso, donde el sistema trata de generar los datos de entrenamiento antes de recibir consultas.

1.3.5 Algoritmo de las hormigas

El algoritmo hormiga o algoritmo de las hormigas es una técnica probabilística utilizada para solucionar problemas de cómputo; este algoritmo está inspirado en el comportamiento que presentan las hormigas para encontrar las trayectorias desde la colonia hasta el alimento.

Su funcionamiento se basa en que cuando una hormiga sale en busca de comida, va dejando un rastro de feromonas a su paso, siendo así como esta podrá regresar a casa sin problemas además de que indica el camino a las demás hormigas para que la sigan. Mientras más cantidad de hormigas pasen por ese camino más fuerte será el nivel de feromonas, por el contrario, mientras menos pasen se hará más débil hasta que desaparece. El camino que más cantidad de feromonas contenga será el mejor de todos, por lo que ese será el camino que seguirán todas las hormigas.

Los algoritmos de la optimización de la colonia de la hormiga se han utilizado para producir soluciones casi óptimas al problema del viajante de comercio. El algoritmo de la colonia de la hormiga puede funcionar continuamente y adaptarse a los cambios en tiempo real. Un ejemplo claro se puede observar en el problemas de enrutamiento de redes y sistemas urbanos del transporte.

1.3.6 Sistemas expertos existentes

Las funciones y algoritmos antes mencionados son utilizados por muchos sistemas expertos existentes en la actualidad donde van desde sistemas para diagnósticos de diferentes padecimientos médicos hasta sistemas de atención a clientes y sistemas usados por órganos judiciales.

- **Internacionales**

El campo de la medicina ha sido de gran interés desde que surgieron los SE siendo esta un área donde han prosperado bastante. Estos sistemas también se han introducido en diferentes aéreas de la sociedad, ejemplo de ellos serán mencionados a continuación:

PROTOS: Sistema que clasifica enfermedades del oído a partir de una descripción de los síntomas, historiales y resultados de los análisis de los pacientes.

CASEY: Es un sistema que diagnostica paradas cardíacas y a partir de los síntomas del paciente, este produce una red de posibles estados que provocaron los síntomas. Cuando aparecen casos nuevos, se buscan casos de pacientes con síntomas parecidos y se adapta el diagnóstico recuperado en función de las diferencias entre los síntomas de los casos.

COACH: Es un sistema que trabaja en el dominio del fútbol. Genera nuevas jugadas a base de mejorar jugadas anteriores. Tiene muchas estrategias de adaptación para formar nuevas jugadas.

JUDGE: Es un sistema usado para la emisión de sentencias criminales durante los juicios. Se parte de un conjunto de estrategias simples para la formación de sentencias, el cual es capaz de recuperar memorias de casos anteriores para desarrollar nuevas sentencias. Es capaz de hacer razonamientos acerca de asesinatos, homicidios y agresiones.

HYPO: En este sistema el profesor plantea el tema a estudiar y el sistema presenta un conjunto de casos relacionados pedagógicamente con el tema los cuales el profesor utiliza para mostrarlos a los estudiantes.

- **Nacionales**

Dentro del país, podemos mencionar a varios sistemas expertos, por ejemplo, en Universidad Central de las Villas se creó un sistema experto conocido como **SISI-CBE** el cual consta de 2 componentes separados, por un lado el sistema de gestión de la base de conocimientos denominada **CBE** (Case Base Editor) y por el otro el componente destinado a la inferencia de los resultados que hace uso del CBE denominada **SISI** (Sistema inteligente de Selección de Información). Este sistema ha sido probado con varias bases de datos principalmente en el campo de la medicina, y se han logrado buenos resultados pero hay que mencionar que este programa está creado para el trabajo local además de poder ser ejecutado solamente en un sistema operativo Windows. A todo esto hay que sumarle la inconveniencia del trabajo con los componentes por separado y no como un único sistema.

También dentro del país se han creado varios sistemas expertos enfocados a la medicina dentro de los cuales podemos mencionar **SAMTA** (Sistema de Medicina Tradicional Asiática) y el sistema **MALCON** destinado al estudio de las malformaciones congénitas.

1.4 Tendencias y tecnologías actuales.

La industria del software ha mostrado ser una de las áreas más dinámicas y con mayor crecimiento en los últimos años. La evolución hacia a un modelo más racional para los usuarios, con menos costes de licencia, donde se intensifique la prestación de servicios, que reduzca el tiempo de desarrollo e incremente la calidad, viene siendo lo más importante a la hora de desarrollar cualquier tipo de aplicación. A continuación se ofrecerá una valoración sobre algunas de las tendencias y tecnologías que marcan un nivel alto en el mundo del software y que bien contribuye a lo dicho anteriormente:

1. **Aplicación Web.**

La creación de aplicaciones de este tipo ha tomado gran auge debido a las funcionalidades que ofrece; algunas de estas se presentarán a continuación:

- ❖ Compatibilidad multiplataforma: Una misma versión de la aplicación puede correr sin problemas en múltiples plataformas como Windows y Linux.
- ❖ Menos requerimientos de hardware: Este tipo de aplicación no consume (o consume muy poco) espacio en disco y también es mínimo el consumo de memoria RAM en comparación con los programas instalados localmente. Tampoco es necesario disponer de computadoras con poderosos procesadores ya que la mayor parte del trabajo se realiza en el servidor donde reside la aplicación.
- ❖ Actualización: Una de las características fundamentales de las aplicaciones web es que siempre se mantienen actualizadas y no requieren que el usuario deba descargar actualizaciones ni realizar tareas de instalación.
- ❖ Acceso inmediato y desde cualquier lugar: Las aplicaciones basadas en tecnología web no necesitan ser descargadas, instaladas o configuradas. Además, pueden ser accedidas desde cualquier computadora conectada a la red en donde se accede a la aplicación.
- ❖ Seguridad en los datos: Los datos se alojan en servidores con sistemas de almacenamiento altamente fiables y se ven libres de problemas que comúnmente sufren los ordenadores de usuarios comunes como virus y roturas de disco.

Cuando se desarrolla una aplicación web existe una tendencia al uso de Arquitectura en capas y del patrón de diseño: Modelo-Vista-Controlador (MVC).

2. Arquitectura Cliente/Servidor.

La arquitectura cliente/servidor no es más que un sistema distribuido entre múltiples procesadores donde hay clientes que solicitan servicios y servidores que los proporcionan. La mayoría de las aplicaciones que se están desarrollando actualmente en la industria de software utilizan este tipo de arquitectura debido a las funcionalidades que brindan, entre ellas:

- ❖ Permite integrar y compartir información entre sistemas diferentes sin necesidad de que todos tengan que utilizar el mismo sistema operativo.
- ❖ Posibilita el mantenimiento y el desarrollo rápido de aplicaciones. (9)

La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.

Se puede asociar el uso de esta arquitectura con los principios de Arquitectura Orientada a Servicios (SOA[3]).

3. Software libre.

El software libre; extendido prácticamente en el mundo; permite crear soluciones cada vez más sofisticadas y más personalizadas debido a la libertad de los usuarios para ejecutar, copiar, distribuir y mejorar el software, tiene entre sus ventajas fundamentales:

- ❖ Independencia tecnológica: El acceso al código fuente permite el desarrollo de nuevos productos sin la necesidad de desarrollar todo el proceso partiendo de cero.
- ❖ Libertad de uso y redistribución: Las licencias de software libre existentes permiten la instalación del software tantas veces y en tantas máquinas como el usuario desee.

[3]**SOA**: Es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio.

De ahí la repercusión que tiene en el ámbito informático hoy día, siendo una tendencia que va madurando cada vez más.

1.5 Modelo de desarrollo propuesto.

Un modelo de desarrollo de software es un conjunto de pasos y procedimientos que deben seguirse para desarrollar software. Para lograr el desarrollo de la solución se hace uso del modelo de desarrollo para software tecnológico propuesto por el Departamento de Tecnología del Proyecto ERP-Cuba. El mismo proporciona una guía para regir el proceso de desarrollo de software tecnológico, centrado en la arquitectura. Este modelo propone una solución sencilla y novedosa, que se centra en el desarrollo de componentes como base tecnológica.

1.5.1 Tecnologías propuestas.

Para el desarrollo de la solución se utilizó el Marco de Trabajo Sauxe, el cual contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Este marco de trabajo está desarrollado sobre lenguaje de programación PHP y utiliza JavaScript y HTML para la capa de presentación, recurriendo al Doctrine para la capa de acceso a datos.

Sauxe implementa una arquitectura en capas donde hace uso también del Modelo-Vista-Controlador (MVC)[4]. Este marco de trabajo utiliza ExtJs en la capa de presentación para agilizar el proceso de desarrollo y mostrarle una interfaz más amigable y funcional al usuario, además de hacer uso del Zend Framework en la capa de negocio, el cual es conocido por su nivel de flexibilidad en la integración con las distintas capas de la arquitectura.

[4] **Modelo Vista Controlador** es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

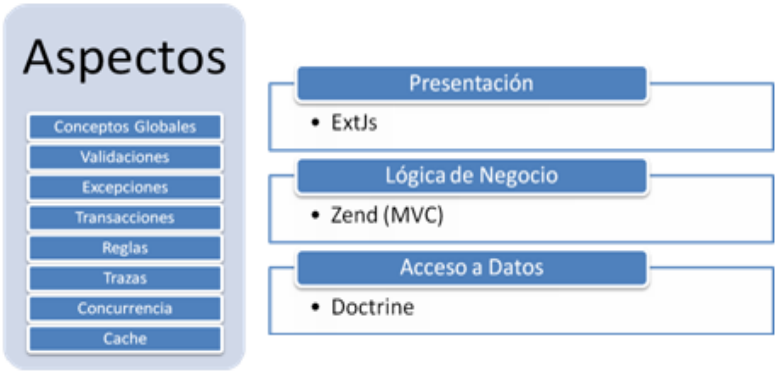

Figura 2 Arquitectura del Marco de trabajo Sauxe (10)

Debido a las características de la arquitectura que propone esta modelo de desarrollo, la cual es orientada a componentes, y al también deseo de alejar a los desarrolladores de los detalles arquitectónicos, se utiliza ZendExt Framework, extensión desarrollada para este fin.

La arquitectura del Marco de trabajo Sauxe está compuesta básicamente por cinco niveles o capas:

Capa de Presentación: En esta capa se emplea las facilidades que brinda el Marco de Trabajo ExtJS para la construcción de las interfaces de usuarios. ExtJs centra su desarrollo en tres componentes fundamentales JS-File, CSS-File y Client-page y Server-Page.

Capa de Negocio: En esta capa se emplea el Modelo Vista Controlador (MVC). De forma vertical al modelo descrito hasta este momento, estarán los aspectos fundamentales del sistema así como el encargado del tratamiento de la seguridad a nivel de aplicación Web.

Capa de Acceso a Dato: En esta capa estará presente el ORM (Object Relational Mapping) Doctrine, como Marco de Trabajo de persistencia para la comunicación con el servidor de datos mediante el protocolo PHP Data Objects (PDO)[5].

Capa de Dato: En esta capa estará ubicado como servidor de base de datos PostgreSQL y un conjunto de ficheros de configuración.

[5] **PDO** define una interfaz ligera, para tener acceso a bases de datos desde PHP.

Capa de Servicio: En esta última capa se encuentran todos los subsistemas que prestan y consumen servicios entre sí. (10)

1.5.2 Características del modelo de desarrollo:

- **Centrado en la arquitectura:**

La arquitectura determina la línea base y los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades en la producción y resuelve las necesidades tecnológicas y de soporte para el desarrollo.

- **Orientado a componentes:**

Las iteraciones son orientadas según la significación arquitectónica de los componentes, los mismos son abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan; el componente es la unidad de medición y ordenamiento de las iteraciones.

- **Iterativo e incremental:**

Las iteraciones son planificadas y coordinadas con el equipo de arquitectura, los clientes y la alta gerencia. Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo de esta manera la evolución incremental del producto.

- **Ágil y adaptable al cambio:**

El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados.

1.6 Arquitectura.

Uno de los elementos clave en todo proceso de desarrollo de software es el diseño de la Arquitectura. Básicamente sobre ella se sustentan todas las representaciones de la estructura general de la aplicación a

desarrollar. En otras palabras la arquitectura de una aplicación es la vista conceptual de la estructura de esta y donde se establecen los fundamentos básicos para que analistas, diseñadores y programadores trabajen en una línea común y así alcanzar los objetivos del sistema de acuerdo con las necesidades del cliente. La correcta definición de los estilos arquitectónicos a utilizar, patrones y mecanismos de diseño es la esencia de lo dicho anteriormente.

1.6.1 Estilos arquitectónicos:

- **Arquitectura Basada en Componentes:**

Uno de los enfoques en los que actualmente se trabaja es la arquitectura basada en componentes que tiene como objetivo hacer un uso correcto de software reutilizable, para la construcción de aplicaciones mediante el ensamblaje de partes ya existentes.

“Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y de requisitos, que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio” (11)

Una de las grandes ventajas de los componentes es la reusabilidad. Una reutilización efectiva depende no sólo de la identificación apropiada de los componentes, sino del modo en que dichos componentes son combinados y organizados. Las arquitecturas de software basadas en componentes brindan el soporte para la integración de partes en sistemas mayores, facilitando la definición de una estructura de ensamblado adecuada. El empleo de esta técnica de desarrollo de software requiere por lo tanto de un cuidadoso modelado de la arquitectura y del análisis, a los fines de asegurar reusabilidad y compatibilidad entre los componentes a interactuar.

- **Modelo-Vista-Controlador (MVC):**

El patrón arquitectónico MVC es utilizado para el desarrollo de aplicaciones web con el fin de separar en tres componentes distintos, la interfaz de usuario, la lógica de negocio y los datos persistentes, potenciando la flexibilidad y la adaptabilidad a futuros cambios.

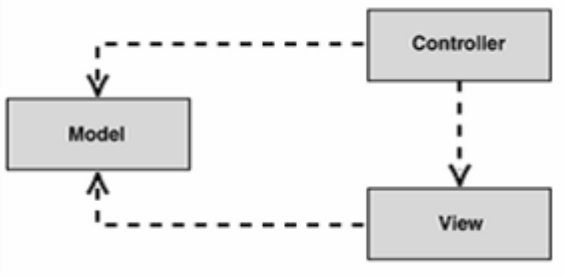

Figura 3 Representación del patrón MVC

Por su parte:

El Modelo: Es la representación de la información que maneja la aplicación. Son los datos puros que puestos en un contexto del sistema son mostrados al usuario por medio del Controlador, proveen de información al usuario o a la aplicación misma.

La Vista: Constituye la representación del modelo en forma gráfica, disponible para la interacción con el usuario. En una aplicación web la "Vista " es la página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.

El Controlador: Se encarga de responder a las solicitudes del usuario desde la Interfaz, manejando los diferentes eventos a través de las funcionalidades necesarias y la información perteneciente al Modelo. (12)

Para el desarrollo del sistema se decidió trabajar con este patrón, evidenciándose en los frameworks definidos para cada una de las capas como parte del MVC de cada componente dentro de la aplicación, es decir: para la Vista: ExtJs-Framework, el cual es muy utilizado en el desarrollo de aplicaciones Web con tecnología AJAX, para el Controlador: Zend-Framework quien emplea específicamente el estilo Modelo-Vista- Controlador como base de su funcionamiento y para agilizar el acceso a datos en el Modelo se utilizó Doctrine, un potente y completo sistema ORM (Mapeo Objeto Relacional) como antes se ha mencionado.

De forma general, según lo descrito con anterioridad, en la arquitectura a utilizar los estilos arquitectónicos que se emplean no se pueden ver de forma independiente sino como un estilo híbrido que comprende numerosas ventajas:

- ❖ Desarrollos paralelos: en cada capa.
- ❖ Aplicaciones más robustas debido al encapsulamiento.
- ❖ Mantenimiento y soporte más sencillo: es más sencillo cambiar un componente que modificar íntegramente una aplicación.
- ❖ Mayor flexibilidad: se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad.
- ❖ Alta escalabilidad: La principal ventaja de una aplicación distribuida bien diseñada es una buena escalabilidad, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. (13)

1.7 Patrones.

Se conoce como patrón a un cúmulo de información que proporciona respuesta a un conjunto de problemas similares en un contexto dado. Los patrones hacen más resistente al cambio la producción de software y establecen parejas problema-solución. También ayudan a especificar interfaces, facilitan la reutilización del código y permiten una fácil comprensión debido a la documentación estándar que estos presentan. (14)

1.7.1 Patrones de diseño:

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. (15)

Patrones GRASP: Patrones de Software para la Asignación General de Responsabilidad. En este caso las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento:

Conocer: Conocer los datos privados encapsulados, conocer los objetos relacionados, conocer las cosas que puede derivar o calcular.

Hacer: Hacer algo él mismo, como crear un objeto o hacer un cálculo, iniciar una acción en otros objetos, controlar y coordinar actividades en otros objetos.

GRASP destaca 5 patrones principales:

Experto, Creador, Alta cohesión, Bajo acoplamiento, Controlador.

Experto: Este patrón es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para ejecutar la tarea. Refuerza el encapsulamiento y esto redunda en bajo acoplamiento.

Creador: El patrón creador ayuda a identificar quién debe ser el responsable de la instanciación o creación de nuevas clases u objetos. La clase podrá crear la nueva instancia si y sólo si tiene en cuenta al menos uno de los siguientes criterios:

- Tiene la información necesaria.
- Usa directamente las instancias creadas del objeto.
- Almacena o maneja varias instancias de la clase.
- Contiene o agrega la clase.

La visibilidad entre la clase creada y la clase creadora es una de las facilidades que se deriva del uso de patrón y además conduce a un bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización así como mayor claridad.

Alta cohesión: Indica que la información que almacena una clase debe ser coherente, de manera que todos sus métodos tengan un comportamiento bien definido.

- Las Clases se pueden reutilizar con mayor facilidad y flexibilidad.
- Una Clase con baja cohesión hace muchas cosas no relacionadas.
- Una Clase con alta cohesión hace lo que uno podría esperar que hiciera.

Si el sistema fallara por alguna razón es mucho más fácil encontrar responsabilidades si las Clases del sistema son cohesivas.

Bajo acoplamiento: Patrón evaluativo que asigna responsabilidades de modo que se mantenga un engranaje pobre entre las clases y objetos, reduce el impacto de los cambios y aumenta de reutilización.

Controlador: Asignar la responsabilidad a una clase de manejar mensajes correspondientes a eventos en un sistema. Encargada de recibir los datos del usuario y enviarlos a las distintas clases según el método llamado a modo de intermediario entre una determinada interfaz y el algoritmo que la implementa. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación para aumentar la reutilización de código y a la vez tener un mayor control. (16)

Patrones GOF:

Fachada: Patrón estructural que permite proveer una interfaz unificada y sencilla como intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas.

Mediador: Patrón de comportamiento que coordina las relaciones entre sus asociados. Permite la interacción de varios objetos, sin generar acoples fuertes en esas relaciones.

Cadena de Responsabilidad: La cadena de responsabilidad se encarga de evitar el acoplamiento del remitente de una petición a su receptor, dando a más de un objeto la posibilidad de manejar la petición. (17)

1.8 Lenguajes de modelado y desarrollo.

1.8.1 Lenguaje de modelado.

Se denomina lenguaje de modelado de objetos al conjunto estandarizado de símbolos y las distintas combinaciones de la disposición para modelar un diseño de software.

- **UML:**

Lenguaje que permite visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Se estará haciendo uso del mismo para todo lo que comprende el diseño del sistema.

- **BPMN:**

BPMN (Business Process Modeling Notation) es un estándar de modelado de procesos de negocio, en donde se presentan gráficamente las diferentes etapas del proceso. Su objetivo principal es coordinar la secuencia de procesos y tiene la finalidad de servir como lenguaje común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación**.** A diferencia del lenguaje UML, este orienta el modelado del sistema a los procesos.

1.8.2 Lenguajes de Programación.

El término lenguaje de programación está dado por un lenguaje que puede ser utilizado para controlar el comportamiento de una computadora. Un lenguaje de programación permite a uno o más programadores especificar de manera precisa sobre qué datos una computadora debe operar, cómo deben ser estos almacenados, transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias.

1.8.2.1 Lenguajes del lado del servidor.

Se clasifica así al lenguaje de programación en la tecnología cliente servidor que se ejecuta del lado del servidor y del cual los usuarios solo obtienen el beneficio del procesamiento de la información.

- **Lenguaje PHP:**

PHP es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Es conocido como una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones web dirigidas a bases de datos. Su interpretación y ejecución se realizan en el servidor en el cual se encuentra almacenada la página y el cliente sólo recibe el resultado de la ejecución. PHP tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX, Linux y Windows. También posee una gran capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad y es libre, lo que representa que una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente.

1.8.2.2 Lenguajes del lado del cliente.

Un lenguaje del lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y ello puede afectar a la seguridad.

- **HTML:**

Es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML. (18)

- **XML:**

Es el metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos. No es un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. (19)

- **JavaScript:**

JavaScript es un lenguaje que no requiere compilación, es decir, es un lenguaje interpretado, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM[6]. JavaScript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con JavaScript se puede crear diferentes efectos e interactuar con los usuarios. JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Netscape y Mozilla Firefox. (20)

1.9 Frameworks.

Un framework, en el desarrollo de software, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

1.9.1 ExtJs 2.2.

Librería construida empleando JavaScript con el fin de desarrollar aplicaciones web interactivas usando tecnologías como AJAX y DOM. ExtJs es muy potente ya que contiene rica colección de componentes para el diseño de Interfaces Gráficas de Usuario (GUI's) del lado del cliente haciendo uso extensivo de AJAX.

1.9.2 Doctrine.

Doctrine es un potente y completo sistema ORM[7] (en inglés Object Relational Mapper) para PHP 5.2+ que incorpora una capa de abstracción a base de datos (DBL). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientada a objeto. Esto les proporciona una alternativa poderosa a diseñadores de SQL manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario.

[6] **DOM**: El Modelo de Objetos de Documento (en inglés Document Object Model), es una plataforma que proporciona un conjunto estándar de objetos, que permite crear documentos HTML y XML, navegar por su estructura, modificar, añadir y borrar tanto elementos como contenidos. No se ajusta a un lenguaje de programación específico, lo que facilita el diseño de páginas web activas, de manera tal que proporciona una interfaz estándar para que otro software manipule los documentos.

[7] **ORM:** Componente de software que permite trabajar con los datos persistidos como si fueran parte de una base de datos orientada a objetos

Funcionalidades:

- ❖ Exporta una base de datos existente a sus clases correspondientes.
- ❖ Convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. (21)

1.9.3 Zend Framework.

Es un framework de alta calidad y de código abierto para el desarrollo de aplicaciones y servicios web con PHP. Zend Framework brinda facilidades de uso y poderosas funcionalidades. Proporciona soluciones para construir modernos, robustos y seguros sitios web. Está diseñado para Php 5 y posee buenas capacidades de ampliación. Zend proporciona un sistema de caché de forma que se puedan almacenar diferentes datos y provee los componentes que forma la infraestructura del patrón MVC y cuenta con otras ventajas que hacen de su uso muy necesario para un correcto desarrollo de productos. (22)

1.10 Tecnologías y Herramientas de desarrollo.

Para la realización de un proyecto es necesario que se posea un modelo estandarizado de las tecnologías y herramientas a utilizar para la implementación de las capas de presentación, negocio y acceso a datos.

1.10.1 Tecnologías.

- ❖ **Tecnología AJAX:**

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. Permite realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. (23)

1.10.2 Herramienta CASE.

- ❖ **Visual Paradigm:**

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Entre sus principales características podemos encontrar el soporte para aplicaciones Web, la compatibilidad entre las diferentes ediciones y la calidad de sus productos, además de ser de fácil instalación y actualización. (24)

1.10.3 Herramienta de desarrollo colaborativo.

- **Control de versiones:**

Una versión, revisión o edición de un producto, es el estado en el que se encuentra en un momento dado en su desarrollo o modificación. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Cada uno de los usuarios puede crearse una copia local duplicando el contenido del repositorio para permitir su uso. Es posible duplicar la última versión o cualquier versión almacenada en el historial.

Subversion:

También conocido como SVN, es un sistema de control de versiones que se ha popularizado bastante, en especial dentro de la comunidad de desarrolladores de software libre. Está preparado para funcionar en red y se distribuye bajo licencia libre. Entre sus funcionalidades se pueden mencionar que mantiene versiones no sólo de archivos, sino también de directorios, mantienen versiones de los metadatos asociados a los directorios y que además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre. Posee soporte tanto de ficheros de texto como de binarios y cuenta con un mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos. (25)

1.10.4 Entorno integrado de desarrollo.

Un entorno de desarrollo integrado o IDE (en inglés: Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación que permite de forma cómoda y ágil editar, compilar, ejecutar y depurar programas.

- **NetBeans IDE 6.9**

NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está destinado principalmente para el lenguaje Java, pero puede utilizarse para cualquier otro lenguaje de programación como C, C++, C#, PHP, Ruby, Phyton, HTML, JavaScript, CSS y otros. Existen un número importante de módulos para extender este IDE, siendo además un producto libre y gratuito sin restricciones de uso. Ha sido desarrollado para distintas plataformas como Linux, Mac OS X, Solaris y Windows y se integra con productos como Symfony, Zend Framework, PHP Unit Testing, MySQL y sistemas de control de versiones (SVN, CVS, Mercurial y Git).

1.10.5 Servidor de Aplicaciones web.

Es un programa que permite crear un servidor http en un ordenador de una forma rápida y sencilla. Está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Es además un programa que implementa el protocolo HTTP el cual se encarga de transferir lo que llamamos hipertextos, páginas web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

- **Servidor web Apache 2.0:**

Es una tecnología gratuita de código fuente abierta, puede ser usado en varios sistemas operativos, lo que lo hace prácticamente universal. Es un servidor altamente configurable, es decir, se pueden elegir qué características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor. (26)

1.10.6 Sistema Gestor de Base de Batos:

Se denomina Sistema Gestor de Base de Datos (siglas: SGBD) al conjunto de programas que permiten definir, construir y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad.

- **PostgreSQL 8.3:**

PostgreSQL es un servidor de base de datos relacional[8], libre. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y joins de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta de software libre PostgreSQL tiene entre otras ventajas las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y es que es un sistema multiplataforma. Fue diseñado para ambientes de alto volumen escalando muy bien al aumentar el número de CPUs y la cantidad de RAM. Soporta transacciones y desde la versión 7.0 incluye soporte para claves ajenas con comprobaciones de integridad referencial. También incluye soporte para vistas, procedimientos almacenados en el servidor, transacciones, almacenamiento de objetos de gran tamaño y además tiene ciertas características orientadas a objetos.

1.10.7 Navegador web:

Software que permite al usuario recuperar y visualizar documentos de hipertexto desde servidores web a través de Internet.

- **Mozilla Firefox 2.0.1 ó superior:**

Mozilla Firefox es el nuevo e innovador navegador open source. La misión del proyecto Mozilla es preservar la elección y la innovación en Internet. Es Software libre y se trata de un práctico y ágil navegador, que está en renovación constante. Tiene la capacidad de ser modificado a gusto del usuario y según las necesidades del mismo. Algunas características de Mozilla Firefox son su navegación por tabs, la cual es una de las principales características que tiene Firefox; además es Software libre y trabaja de

[8] **Base de datos relacional**: Modelo utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente basados en el empleo de relaciones entre las tablas.

forma excelente en computadoras sin hardware muy potente, el programa está diseñado para realizar un bajo consumo de recursos. Soporta Windows (en cualquiera de sus versiones), Linux (desde la versión 2.2 del Kernel[9]) y Mac (desde Mac OS X 10.1.x a la 10.2.x).

1.11 Conclusiones del capítulo:

El uso de los SE ha sido de gran ayuda e importancia mejorando la toma de decisiones o agilizando los procesos empresariales. Es por eso que este capítulo fue esencial para estudiar los diferentes algoritmos usados por estos sistemas, además de algunas de las distintas funciones que son usadas como ayuda a estos algoritmos en el proceso de recuperación de casos.

También se mencionaron algunos sistemas expertos existentes tanto nacional como internacionalmente y se puede decir que ninguno de estos resuelve el problema que se propone. Estos sistemas, por una parte, son aplicaciones orientadas al trabajo de escritorio y no a la web en muchos de los casos, además de que son enfocadas a áreas muy especificas como la medicina o la educación y no a la gestión de errores, y por otra parte son programas a ejecutar sobre sistemas Windows y no sobre aplicaciones libres como los sistemas Linux, aparte de que en muchos de los casos son software privativos los cuales hay que pagar para poder hacer uso de ellos.

También este capítulo ha sido esencial para la definición de un conjunto de conceptos fundamentales asociados al dominio del problema. Se explicó el modelo de desarrollo a utilizar. Se analizaron las tendencias y tecnologías actuales para el desarrollo de aplicaciones informáticas y por último se realizó una presentación de las tecnologías y herramientas conjuntamente con sus versiones propuestas por la dirección del proyecto para el desarrollo de la solución.

[9] **Kernel**: núcleo de un sistema operativo

Capítulo 2: Diseño e implementación.

2.1 Introducción

En el presente capítulo se obtendrán un conjunto de artefactos que serán de gran valor para las posteriores etapas del desarrollo como lo son: los diagramas de procesos, el diseño de clases que comprende diagramas y descripción de las mismas y el modelo de datos. Igualmente se especificará la utilización de un conjunto de patrones dentro del diseño del componente.

El capítulo comprenderá además elementos fundamentales de la implementación del sistema, en ese caso se presentan: la organización del marco de trabajo, la definición de los estándares de codificación y la estructura de datos a utilizar, la descripción de clases y algoritmos implementados.

2.2 Diseño de la solución

2.2.1 Diagramas de proceso de negocio.

El sistema a desarrollar será el encargado de la gestión de los errores dentro del Proyecto ERP-Cuba, donde el tiempo de desarrollo y las buenas prácticas que en este se usen son esenciales para su correcto funcionamiento.

Para un mayor entendimiento de los procesos que realiza el sistema se mostrará a continuación el modelado de dichos procesos haciendo uso de la Notación para el Modelado de Procesos de Negocio (BPMN).

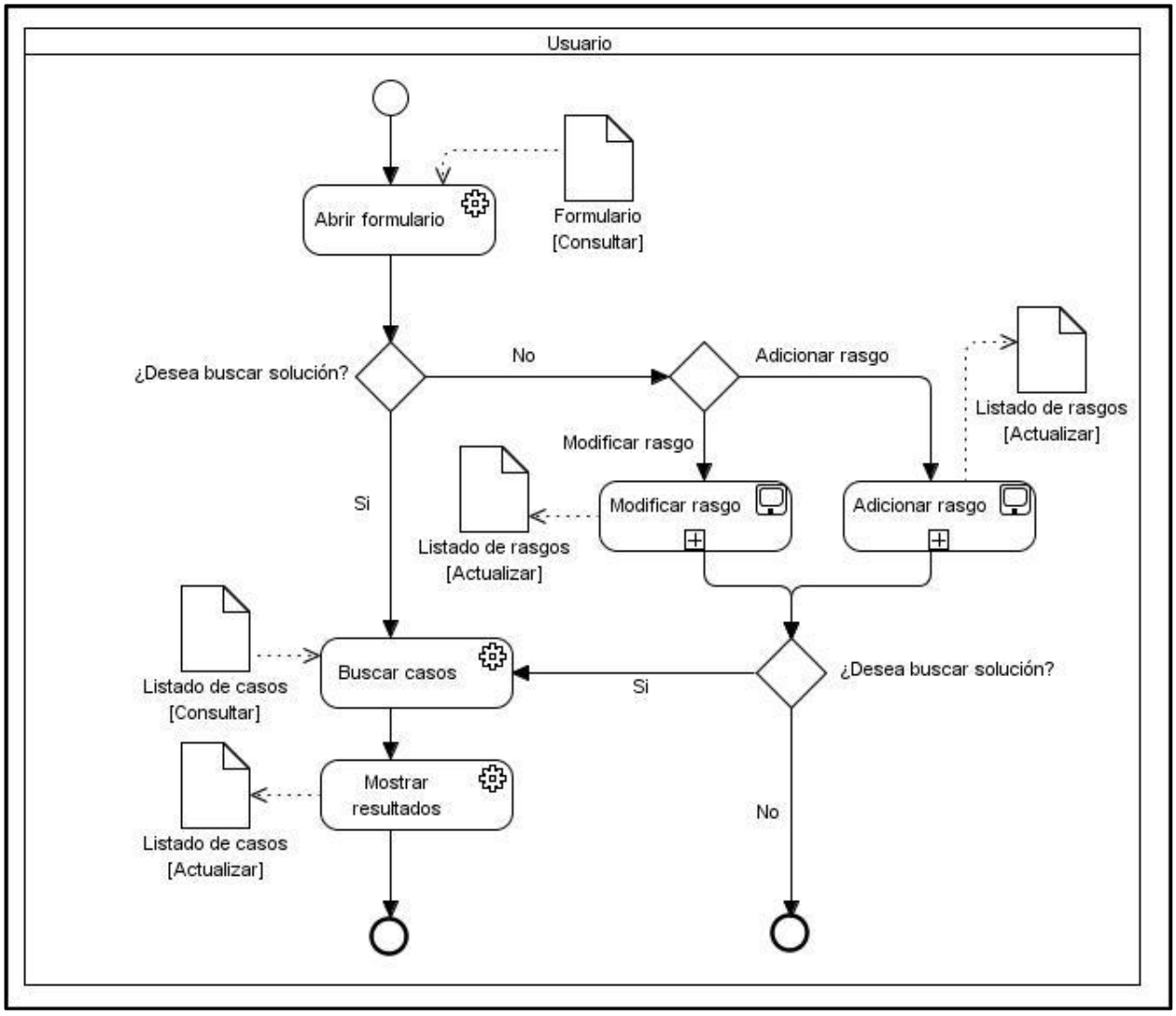

Figura 4. Diagrama de procesos Buscar solución.

Ver resto de los diagramas de procesos en el Anexo 1.

2.2.2 Especificación de procesos

Tabla 1: Proceso Buscar solución.

Objetivo	El usuario introduce los datos en la aplicación del caso que necesita solucionar y el sistema a partir

	de los parámetros de entrada mostrará las posibles soluciones buscadas por el usuario.
Clientes internos	usuario
Clientes externos	N/A.
Entradas	Formulario
Flujo de eventos	
Flujo básico Buscar casos.	
1	Se introducen los datos en la aplicación.
2	Se buscan en la base de datos todos los casos que concuerdan con los parámetros de entrada.
3	Se muestran los resultados al usuario.
4	El usuario escoge la solución más factible para su problema.
5	Concluye el proceso.
Pos-condiciones	
N/A.	
Salidas	
1	Listado de casos similares al entrado por parámetros.
Flujos paralelos	
1. N/A	
Flujos alterno 1 Agregar rasgo.	
1	Se agregan nuevos rasgos y valores a la base de casos.
2	Concluye el proceso.
Flujos alterno 2 Modificar rasgo.	
1	Se agregan nuevos valores a los rasgos en la base de casos.
2	Concluye el proceso.
Pos-condiciones	
N/A	
Salidas	
N/A	

Ver resto de las tablas de descripción de procesos en el Anexo 2.

2.3 Diseño de clases:

En el Capítulo 1 fue descrito el Modelo de desarrollo propuesto por el Proyecto ERP-Cuba y dentro de este se enfatizó en el Marco de trabajo Sauxe el cual proponía la arquitectura base a seguir para el desarrollo del sistema. Esta arquitectura provee un conjunto de componentes genéricos que pueden ser reutilizados de forma que se garanticen un conjunto de escenarios arquitectónicos. En la siguiente figura

se muestra la distribución de los componentes por niveles o capas, enfatizando en la integración entre estos.

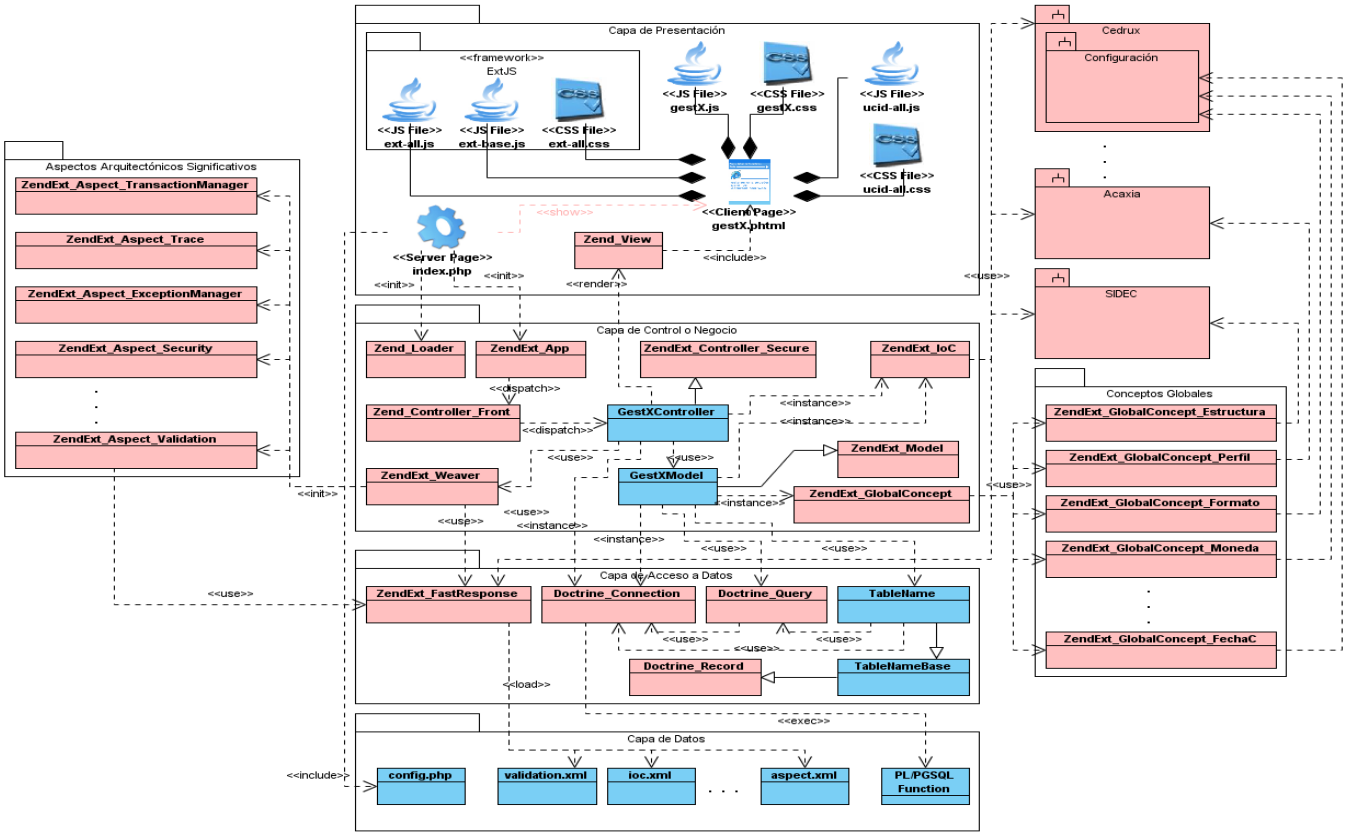

Figura 5 Taxonomía arquitectónica propuesta por Sauxe (10)

2.3.1 Diagrama de clases del diseño.

Los diagramas de clases según la clasificación UML son diagramas donde la representación de los requerimientos se lleva a cabo a través de las clases del sistema y sus interrelaciones. Representan una abstracción del dominio de modo que es formalizado el análisis de conceptos y constituyen el pilar básico del modelado, mostrando en términos generales qué debe hacer el sistema. En estos diagramas se muestran a través de atributos y métodos la estructura de las clases que después serán escritas en algún lenguaje de programación.

En cumplimiento con lo planteado por el Marco de trabajo Sauxe y teniendo como guía la Taxonomía Arquitectónica propuesta por el mismo, durante el diseño de la solución que se propone se generaron diferentes componentes los cuales serán descritos a continuación y mostrados en la figura 5.

- **Diagrama de clases del diseño.**

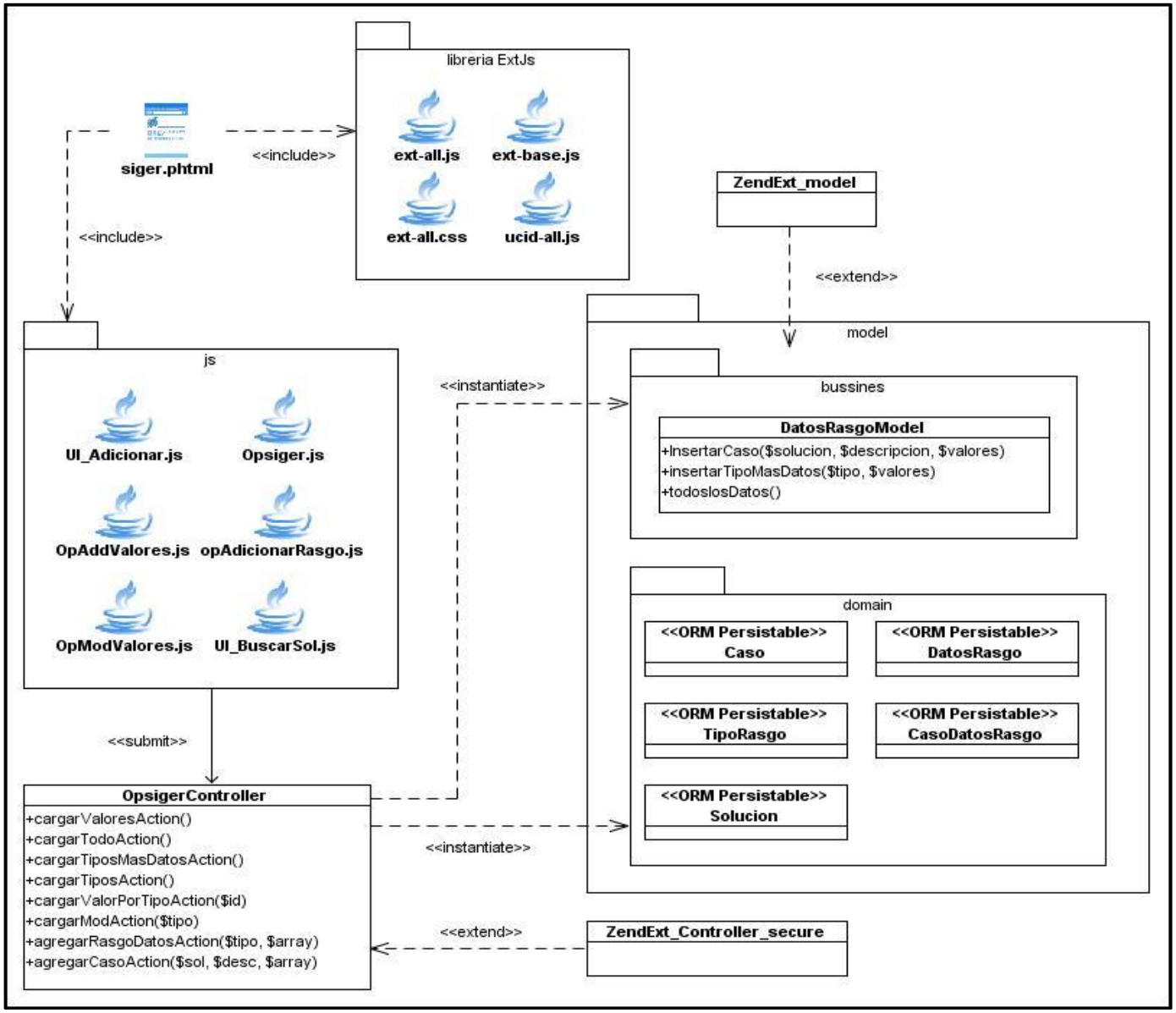

Figura 6. Diagrama de clases del diseño.

❖**Descripción del Diagrama de clases del diseño:**

Tabla 2: Descripción del Diagrama de clases del diseño.

Clases	Descripción

Librería ExtJs (ext-all.js, ext-base.js, etc.)	Contiene los componentes generados a través de la librería JavaScript ExtJs.
siger.phtml	Página encargada de visualizar, a través de los ficheros js que debe incluir, la aplicación y la información relacionada con la gestión de los errores en general.
Clases js (UI_BuscarSol.js, etc.)	Encargadas de generar de forma dinámica a través del DOM y utilizando la librería ExtJs los componentes que manejen la información en la vista necesaria para el trabajo en la aplicación. Debe enviar y recibir los datos de la controladora utilizando tecnología AJAX.
OpsigerController	Clase controladora responsable de configurar los destinos donde se deben ejecutar las diferentes funcionalidades, según las peticiones del usuario.
ZendExt_Controller_Secure	Encargada de gestionar acciones personalizadas y está integrada a la seguridad.
Paquete Model	Encargado de manejar los datos persistentes dentro del componente. Contiene el Bussines y el Domain.
ZendExt_Model	Modelo gestor de negocio que permite entre otras funcionalidades iniciar la conexión a la base de datos.

2.4 Modelo de datos.

Un modelo de datos permite describir los elementos que intervienen en una realidad o en un problema dado y la forma en que se relacionan dichos elementos entre sí. El modelo de datos propuesto en la solución cuenta con un total de 5 tablas las cuales darán soporte y solución a las necesidades del sistema.

2.4.1 Diccionario de datos:

Tabla 3: Diccionario de datos tabla dat_caso

Nombre tabla: dat_caso	
Esquema	mod_siger

Descripción	Almacena los datos propios de un caso como la descripción y demás atributos.	
Atributo	**Tipo**	**Descripción**
id_caso	numeric (19)	Llave primaria de la tabla
descripcion	varchar (255)	Descripción breve del caso
dat_solucionid_solucion	numeric (19)	FK, id de la solución que pertenece al caso
peso	numeric (19)	Peso asociado al caso
Tablas relacionadas		**Descripción de la relación**
dat_solucion		Uno a mucho
dat_caso_datos_rasgo		Uno a mucho

Tabla 4: Diccionario de datos tabla dat_solucion

Nombre tabla: dat_solucion		
Esquema	mod_siger	
Descripción	Almacena las distintas soluciones a los casos. Una solución puede estar presente en varios casos.	
Atributo	**Tipo**	**Descripción**
id_solucion	numeric (19)	Llave primaria de la tabla
valor	varchar (255)	Descripción de la solución
Tablas relacionadas		**Descripción de la relación**
dat_caso		Uno a mucho

Tabla 5: Diccionario de datos tabla dat_tipo_rasgo

Nombre tabla: dat_tipo_rasgo		
Esquema	mod_siger	
Descripción	Almacena los nomencladores de los tipos de rasgos.	
Atributo	**Tipo**	**Descripción**
id_tiporasgo	numeric (19)	Llave primaria de la tabla

denominacion	varchar (255)	Nombre del rasgo
Tablas relacionadas		**Descripción de la relación**
dat_datos_rasgo		Uno a mucho

Tabla 6: Diccionario de datos tabla dat_datos_rasgo

Nombre tabla: dat_datos_rasgo		
Esquema	mod_siger	
Descripción	Almacena los valores de los rasgos. Cada rasgo puede contener uno o varios valores asociados.	
Atributo	**Tipo**	**Descripción**
id_datosrasgo	numeric (19)	Llave primaria de la tabla
denominacion	varchar (255)	Valor asociado al rasgo
dat_tipo_rasgoid_tiporasgo	numeric (19)	FK, id del rasgo al que pertenece el valor
codigo	numeric (19)	Código designado al valor
Tablas relacionadas		**Descripción de la relación**
dat_tipo_rasgo		Uno a mucho
dat_caso_datos_rasgo		Uno a mucho

Tabla 7: Diccionario de datos tabla dat_caso_datos_rasgo

Nombre tabla: dat_caso_datos_rasgo		
Esquema	mod_siger	
Descripción	Almacena los id de los valores asociados a cada caso. Un caso tiene asociado uno o varios valores. (Valor representante del rasgo).	
Atributo	**Tipo**	**Descripción**
dat_casoid_caso	numeric (19)	FK, id del caso al que pertenecen los valores
dat_datos_rasgoid_datosrasgo	numeric (19)	FK, id del valor asociado al caso
Tablas relacionadas		**Descripción de la relación**
dat_caso		Uno a mucho

dat_ datos_rasgo	Uno a mucho

El modelo de datos es presentado en la figura 7:

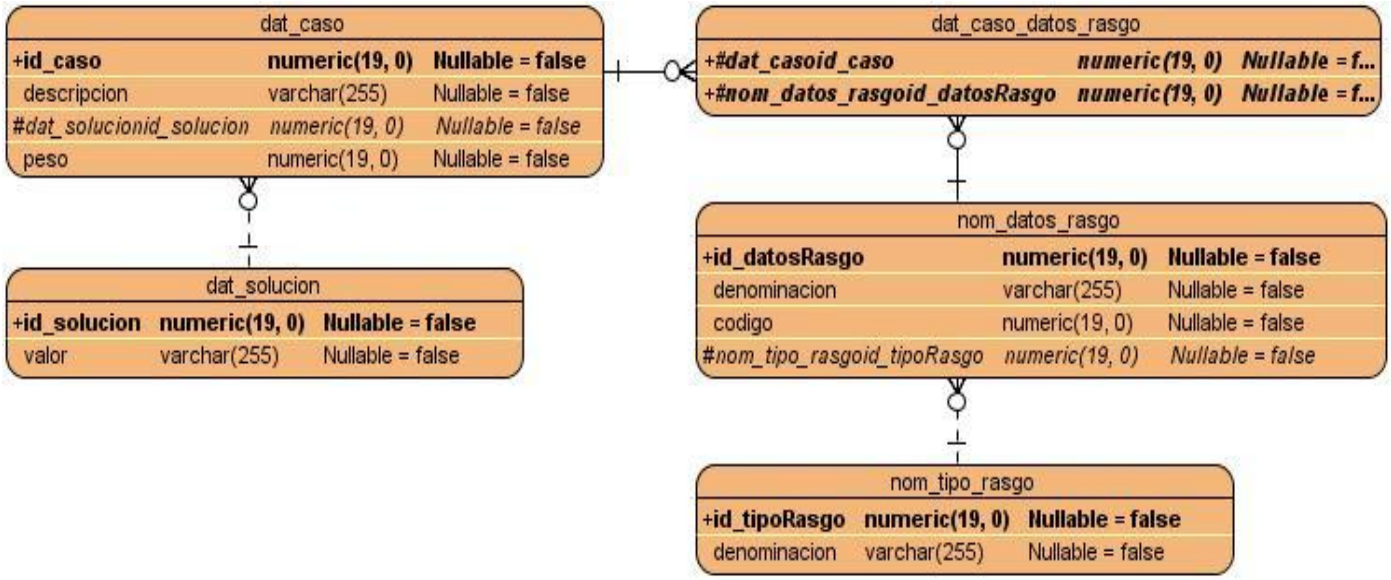

Figura 7. Modelo de datos.

2.5 Patrones de diseño utilizados

El diseño del sistema fue elaborado haciendo uso de los patrones GRASP (General Responsability Assignment Software Patterns), los que describen los principios fundamentales de la asignación de responsabilidades a objetos expresados en forma de patrones. Los patrones GRASP que se utilizaron son los siguientes:

Experto: Este patrón se evidencia en la definición de las clases de acuerdo a las funcionalidades que deben realizar a partir de la información manejada dentro del sistema, como por ejemplo las clases controladoras y las del modelo. Específicamente: la clase OpDatosRasgoModel, será la responsable de efectuar las operaciones que conciernen a las funciones: definir, eliminar y actualizar los casos dentro de la base de datos, asumiendo toda la lógica para cada una de ellas.

Alta cohesión: Este patrón fue utilizado de manera general en el diseño del sistema donde las clases fueron agrupadas según los requerimientos, siempre siguiendo la premisa de que cada clase debe implementar las operaciones que estén sobre la misma área funcional.

Creador: Este patrón se adapta a las clases del paquete Domain, quienes son las encargadas de crear los objetos de tipo Doctrine_Query, para permitir el acceso a la información almacenada en la base de datos.

Bajo acoplamiento: En el modelo de datos se definieron un conjunto de clases persistentes, entre las cuales se establecieron las relaciones necesarias de manera que fueran más independientes y reutilizables para reducir el impacto de los cambios aumentando la oportunidad de una mayor productividad dentro del sistema.

Controlador: La clase controladora: OpsigerController, es ejemplo de la aplicación de este patrón, pues la misma tendrá a cargo la responsabilidad de manejar todos los eventos dentro del sistema.

Durante el diseño del sistema se emplearon patrones GOF, específicamente los siguientes:

Mediador: La comunicación en la base de datos se puede tornar compleja debido a las dependencias entre las tablas que la componen, esto resulta difícil de manejar a la hora de acceder a un determinado valor. Para solucionar este problema se utiliza este patrón, pues se crea una nueva tabla entre todas las tablas unidas mediante una relación de muchos a muchos. De esta forma, el comportamiento entre las clases queda adaptado a las circunstancias y necesidades del diseño. Dicha operación se evidencia en el modelo de datos antes presentado entre las tablas asociadas dat_caso y dat_datos_rasgo donde la tabla mediadora resultó ser dat_caso_datos_rasgo.

Cadena de Responsabilidad: Está concebido que ante la ocurrencia de un error al realizarse una determinada consulta a la base de datos el mismo sea manejado por el Modelo, creando una nueva excepción de tipo ZendExt_Exception. Dicha excepción debe ser propagada al Controlador, el cual será el encargado de capturarla y enviarla a la Vista ya traducida, siendo esta última la que mostrará un mensaje al usuario en un lenguaje entendible notificando el error y sin especificar detalles del mismo. De esta

manera se distribuyen las responsabilidades entre los diferentes componentes, evidenciándose por lo tanto el empleo de este patrón.

2.6 Implementación del componente:

2.6.1 Estructura del Marco de Trabajo

Cumpliendo con las decisiones arquitectónicas de la dirección del proyecto se define una estructura donde van a estar ubicados cada uno de los subsistemas implementados, de manera que facilite la organización y claridad durante el desarrollo.

La carpeta denominada **apps** es donde se almacenan los controladores y el modelo de cada uno de los componentes correspondientes a los subsistemas del proyecto. En la figura se muestra dicho contenido para el componente Herramientas y el sistema en cuestión.

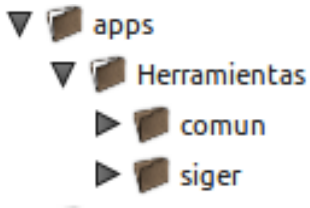

Figura 8. Carpeta de aplicación correspondiente al componente Herramientas.

común: Comprende la configuración específica de cada Subsistema, contiene la carpeta recursos y dentro de esta se encuentra una carpeta xml. Esta última incluye a los ficheros: ioc, validator, exception.

ioc: Es donde se publican los servicios que brinda cada uno de los componentes en cuanto a nombre de clases, funciones y tipo de resultado.

validator: Chequea las precondiciones antes de ejecutar una determinada función en el servidor según el tipo de parámetros, la acción y el usuario que la realice.

exception: Se define el tipo, idioma y la descripción del mensaje que va a ser mostrado cuando se lance una excepción.

El sistema siger va a contener una serie de paquetes que serán expuestos a continuación.

Figura 9. Paquete siger correspondiente a la aplicación.

En el paquete ***controllers*** se encontrarán las clases controladoras encargadas de gestionar las funcionalidades del sistema.

El paquete ***models*** estará estructurado de la siguiente forma:

Figura 10. Paquete models correspondiente a la carpeta de aplicación.

Dentro de este paquete se tienen otros dos, los cuales también agrupan clases y otros paquetes. El paquete **bussines** contendrá las clases necesarias para acceder a los datos que persisten en la base de datos. El paquete **domain** debe contener las clases generadas por el ORM Doctrine a partir de cada una de las tablas existentes en la base de datos. Cada una de estas clases heredará de una clase generada igualmente por el Doctrine las cuales se ubican en otro paquete dentro de este llamado **generated**.

El paquete **views** contendrá los paquetes idioma y scripts, que se encargan de contener el idioma en que se va a mostrar la aplicación y las páginas clientes respectivamente.

En el mismo nivel que la carpeta **apps** se debe encontrar una carpeta que es la que contiene las vistas de los diferentes subsistemas y componentes con que cuenta el proyecto. Esta carpeta se denomina **web.**

Figura 11. Carpeta de diseño correspondiente al sistema.

Figura 12. Paquete correspondiente al sistema siger y la carpeta views del diseño.

La carpeta **view** contendrá los **css** y los **js**. El paquete **css** incluirá las clases necesarias para estructurar gráficamente el componente, separando de esta forma el estilo del contenido. El paquete **js** comprenderá las clases JavaScript necesarias para que el usuario interactúe con el sistema y obtenga los resultados necesarios.

2.6.2 Descripción de las clases y funcionalidades del sistema.

Clases controladoras:

Las clases de controladoras coordinan las actividades de los objetos que implementan las funcionalidades, definen el flujo de control y las transacciones entre los objetos.

Tabla 8. Descripción de la clase OpsigerController

Nombre: OpsigerController	
Tipo de clase: Controladora	
Atributo	**Tipo**
Métodos	

Nombre	Descripción
cargarTodoAction()	La función carga todos los datos en la Base de datos.
cargarTiposAction()	La función muestra la vista que carga los rasgos
cargarValorPorTipoAction()	La función muestra la vista que carga los valores de los rasgos.
cargarModAction()	Permite cargar los valores de los rasgos a modificar
agregarRasgoDatosAction()	La función define en la base de datos los nuevos rasgos con sus respectivos valores.
agregarCasoAction()	Permite definir en la base de datos los nuevos casos.
orderMultiDimensionalArrayAction($toOrderArray, $field, $inverse)	La función permite realizar un ordenamiento de un arreglo recibiendo el arreglo, el campo por el que ordenar y si es de forma ascendente o no.
algoDistFunctionAction($casoBuscar, $casosTotal, $cantCasos)	Permite realizar una búsqueda de los casos más similares al entrado por parámetros. Recibe el caso a buscar así como la cantidad de resultados que se quieren y los casos almacenados en base de dato.

Clases Model:

Estas clases manejan la información persistente que poseen una larga vida, conceptos y sucesos que ocurren en el mundo real.

Tabla 9. Descripción de la clase DatosRasgoModel

Nombre: DatosRasgoModel	
Tipo de clase: Model	
Métodos	
Nombre	**Descripción**
insertarCaso ($solucion, $descripcion, $valores)	Inserta en la base de datos un nuevo caso.

insertarTipoMasDatos ($tipo, $valores)	Inserta en la base de datos un nuevo rasgo con los valores de este.
actualizarRasgo ($idTipo, $vTipo, $valores)	Actualiza en la base de datos un rasgo y sus valores.

Clases Domain:

Tabla 10. Descripción de la clase DatTipoRasgo

Nombre: DatTipoRasgo	
Tipo de clase: Domain	
Atributo	**Tipo**
Métodos	
Nombre	**Descripción**
todosTipos()	Devuelve todos los rasgos que hay en la base de datos.

Tabla 11. Descripción de la clase DatDatosRasgo

Nombre: DatDatosRasgo	
Tipo de clase: Domain	
Atributo	**Tipo**
Métodos	
Nombre	**Descripción**
datosPorTipo ($idTipo)	Devuelve todos los valores asociados al rasgo cuyo id es el recibido como parámetro.

Tabla 12. Descripción de la clase DatCaso

Nombre: DatCaso
Tipo de clase: Domain

Atributo	Tipo
Métodos	
Nombre	**Descripción**
todosLosDatos()	Devuelve todos los valores asociados a los casos en la base de datos.

2.6.3 Estrategia de integración.

En cada uno de los componentes del sistema el flujo de datos que va desde la vista hacia el modelo y viceversa, responde completamente a una estrategia de integración vertical concebida sobre 4 nodos y a partir de cada uno de los elementos arquitectónicos definidos. El primer nodo se sitúa entre la vista y el controlador, el segundo está entre el controlador y el modelo, el tercero vincula el modelo con el framework doctrine y el último se encuentra entre la base de datos y el doctrine.

Vista – Controlador: Los datos recogidos en un formulario son enviados al Controlador haciendo uso del protocolo de comunicación HTTP a través del método “post” para ser procesados y los resultados son enviados por el controlador a la vista en un JSON a través del método “echo”.

Controlador – Modelo: El Controlador toma los datos recibidos desde la vista, instancia una determinada clase del modelo y llama a uno de sus métodos, pasándole como parámetros los datos recibidos.

Modelo – Doctrine: El Modelo utiliza llamadas a métodos de Doctrine que le permitan crear, modificar, eliminar o actualizar los datos almacenados en las tuplas de la base de datos.

Doctrine – Base de Datos: Doctrine ejecuta las consultas a la Base de Datos utilizando programación orientada a objetos.

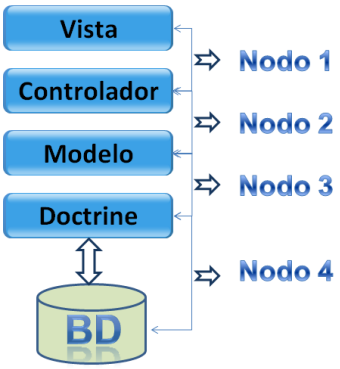

Figura 13. Nodos que intervienen en la integración.

2.6.4 Descripción de la solución

Para la creación del sistema se recurrió al uso del algoritmo Knn (K nearest neighbors) el cual fue implementado en el lenguaje Php para cumplir con las especificaciones del proyecto.

Este algoritmo cuenta con el siguiente pseudocódigo:

1. **COMIENZO**
2. **Entrada:** D = {(x1, c1),…, (x_N, c_N)}
3. **x** = (x_1,…, x_N) --> nuevo caso a clasificar
4. Para todo objeto ya clasificado (x_i, c_i)
5. **Calcular** d_i = d (x_i, x)
6. **Ordenar** di (i = 1, ... , N) en orden ascendente
7. Quedarse con los **K** casos $\mathbf{D}\frac{K}{N}$ ya clasificados más cercanos a **x**
8. Asignar a **x** la clase más frecuente en $\mathbf{D}\frac{K}{N}$
9. **FIN**

Este algoritmo recibe como parámetros la cantidad de casos a mostrar, el caso base a buscar y los casos almacenados en la base de datos y da como salida un listado de los casos más cercanos al caso base entrado por parámetro.

Durante la codificación del mismo se tuvieron que resolver problemas como: Qué función de distancia usar para calcular $\mathbf{d_i = d\ (x_i, x)}$ y como programar el algoritmo para que fuera lo más óptimo posible.

Para la implementación del algoritmo se recurrió al uso de funciones ya predefinidas en el lenguaje Php a partir de la versión 4.0 o superior las cuales facilitaron y optimizaron la codificación del algoritmo. Estas funciones son:

array_intersect: Retorna un arreglo que contiene todos los valores de un arreglo que están presentes en todos los demás arreglos introducidos como parámetros.

array_diff: Retorna la diferencia entre 2 arreglos.

array_slice: Extrae una parte de un arreglo.

También están presentes algunas funciones auxiliares que no son propias del lenguaje Php, las cuales fueron implementadas y usadas como apoyo para el trabajo del algoritmo, como es la función ***ordenarArregloMultidimencionalAction()*** la cual es usada para el ordenamiento que se muestra en el pseudocódigo del Knn en el paso #6 y la función ***convertirDatAction()***, encargada de cargar los datos de la base de conocimientos y pasarlos por parámetros en la entrada del algoritmo.

La función de distancia elegida para responder al cálculo de la lejanía entre los casos que se describe en el paso #5 es la ***Métrica Heterogénea Euclidiana Generalizada (HEOM)*** descrita en el Capítulo 1, la cual trabaja en dominios de problemas donde las características son numéricas y no numéricas incluyendo información incompleta. Esta función es programada dentro del mismo algoritmo y recibe como entrada los 2 casos a comparar, el caso base y el caso almacenado en la base de conocimientos, devolviendo la distancia existente entre estos.

Una vez teniendo la distancia para cada uno de los casos, el algoritmo ordena el listado de casos de forma ascendente por el campo “valor” el cual es el que contiene la distancia entre los casos, y luego corta el arreglo devolviendo solo los que el usuario quiere que se muestren. El tiempo de ejecución de todas estas operaciones es mínimo y estará sujeto a la cantidad de casos que quiera visualizar el usuario y a la cantidad a analizar de casos existentes en la base de conocimiento.

2.6.5 Descripción general del funcionamiento del sistema.

El usuario realiza una petición en el navegador, la cual genera un evento mediante un formulario JS, que envía una notificación a la clase controladora mediante AJAX. Una vez realizada la petición el controlador es el encargado de decidir quién va a ser el responsable de obtener los datos de la misma, si una clase Bussines en caso de la petición ser para insertar o modificar (CRUD), o si lo hará una clase Domain si lo que se desea es consultar. Esta última es mapeada por el Doctrine y se conecta mediante PDO (Doctrine Record) a la capa interna de doctrine que a su vez es la que se conecta directamente a la base de datos. La clase controladora es la encargada de la lógica propia del negocio, cálculos y procesamientos dentro del sistema.

2.6.6 Prototipos de interfaz de usuario

A continuación se presentan los prototipos de interfaz funcionales y se describirá un ejemplo del flujo de las operaciones realizadas por un usuario en interacción con el sistema.

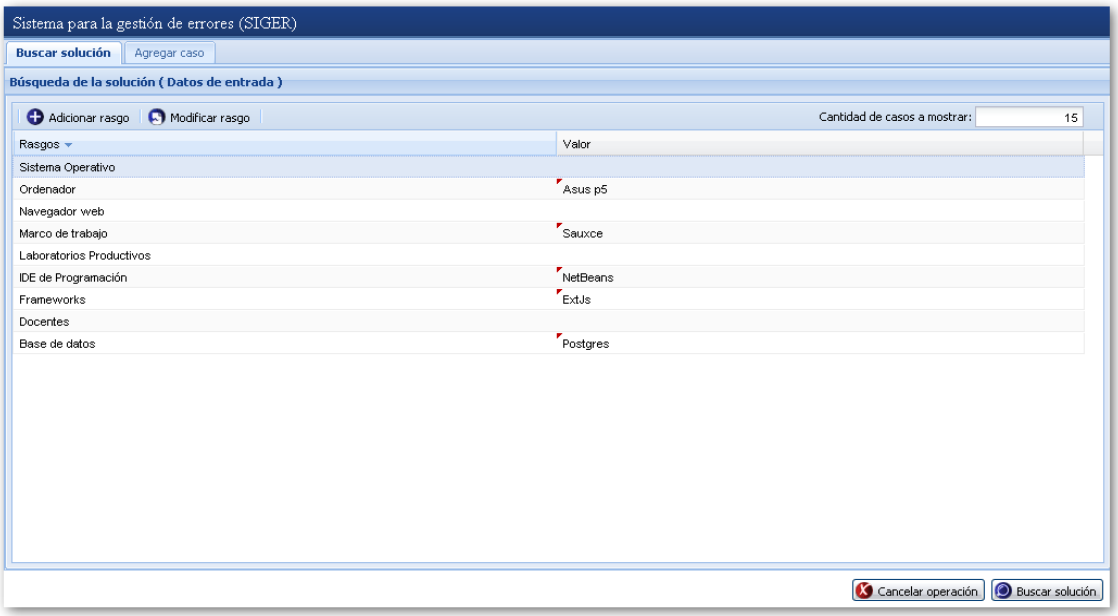

Figura 14. Prototipo interfaz de usuario Buscar solución.

Tabla 13. Descripción de las interfaces de usuario.

Figura	Funcionalidad	Descripción
1	Buscar solución.	Cuando carga el sistema es la interfaz por defecto a cargar. Se seleccionan los valores asociados a los rasgos y se presiona el botón ***Buscar solución*** mostrando un mensaje de confirmación y luego de aceptar se mostrará la interfaz **Mostrar resultados** (Anexo #6). Antes de buscar la solución se puede acceder a la interfaz **Adicionar rasgo** (Anexo #4) o a la **Modificar rasgo** (Anexo #5) si se desean agregar rasgos o modificar algunos de estos.
2	Adicionar caso	En esta interfaz se introducen los datos de un nuevo caso y se agrega a la base de datos presionando el botón ***Agregar***. Si antes de agregar el nuevo caso de desea adicionar algún rasgo nuevo o modificar uno existente se puede seleccionar las interfaces **Adicionar rasgo** (Anexo #4) o la **Modificar rasgo** (Anexo #5).
3	Adicionar rasgo	Se agrega un nuevo rasgo, se introduce el nombre de este y luego los valores asociados al mismo. Se presiona aceptar y se almacena en la base de datos.
4	Modificar rasgo	Se muestra cuando se desea modificar un rasgo, en este caso específicamente el nombre, de querer modificar los valores se presiona el botón ***Modificar valores*** y se muestra la interfaz **Modificar valores del rasgo** (Anexo #5 Fig. A5.2), de lo contrario se acepta el cambio o se cancela la operación.
5	Modificar valores del rasgo	Es mostrada cuando se desea modificar los valores asignados a un rasgo, se agregan los nuevos valores a este y se presiona ***Aceptar***, de lo contrario se cancela la operación y regresa a la interfaz **Modificar rasgo** (Anexo #5 Fig. A5.1).
6	Mostrar resultados	Se muestran los resultados de la búsqueda realizada. Si se presiona doble clic encima de una de las filas es mostrada la interfaz Mostrar detalles del caso (Anexo #6 Fig.A6.2)
7	Mostrar detalles del	Son mostrados los detalles del caso seleccionado en la interfaz Mostrar

	caso	resultados dígase la denominación, la descripción, la solución asignada a este y los valores que contiene.

2.7 Conclusiones del capítulo.

La arquitectura definida y los artefactos generados a partir del estudio de las necesidades del proyecto en cuanto a la gestión de los errores, fueron la base para la realización del presente capítulo a partir de los cuales fue posible llevar a cabo el diseño y la implementación del sistema de gestión de errores SIGER.

Capítulo 3: Validación de la solución

3.1 Introducción

Dentro del proceso de desarrollo de un software, en la medida en que aumente la complejidad y tamaño del producto, mayor es el riesgo de que contenga errores los cuales pueden ser arrastrados durante todo el camino a seguir hasta la terminación del mismo, por lo que se hace necesario el uso de herramientas para comprobar la calidad del proceso de desarrollo y ver si éste es lo suficientemente óptimo como para satisfacer las necesidades del cliente.

En el presente capítulo se realiza la validación de la solución propuesta, para lo que se aplican diferentes métricas de diseño para evaluar en qué medida se garantiza la calidad y la complejidad de la solución. Se aplican además pruebas exploratorias que tienen como objetivo comprobar la funcionalidad y validar la estructura del sistema desarrollado.

3.2 Métricas para la validación del modelo del diseño

3.2.1 Tamaño operacional de las clases (TOC)

La métrica TOC está dada por la cantidad de funcionalidades contenidas en las clases, a partir de las cuales se determina la afectación que ejerce en el diseño.

Atributos de calidad según esta métrica:

- **Responsabilidad:** Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.
- **Complejidad de implementación:** Grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización:** Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.

Ver los instrumentos y la tabla de resultados para la métrica TOC en el Anexo 7.

A continuación se muestra el rango de valores y la complejidad por cada uno de los atributos.

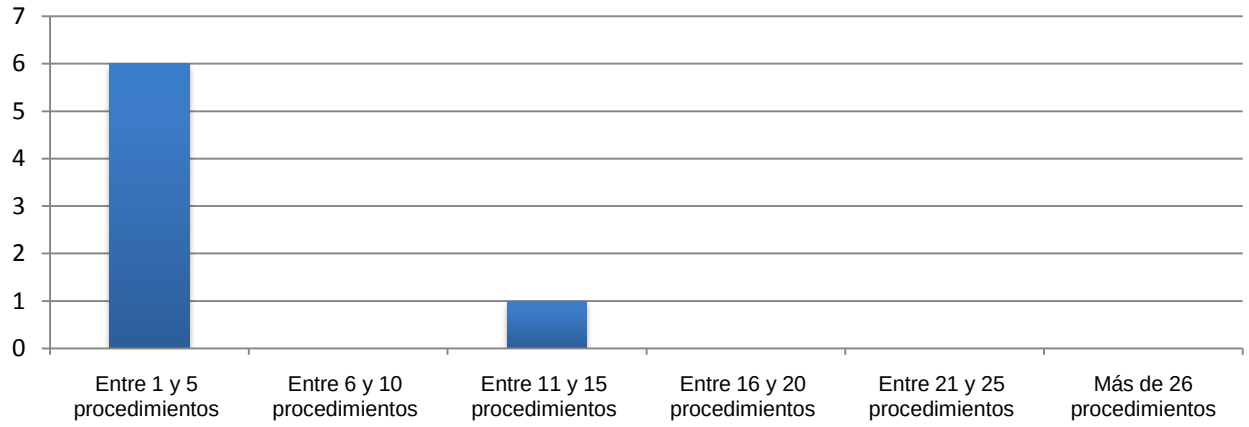

Figura 15. Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

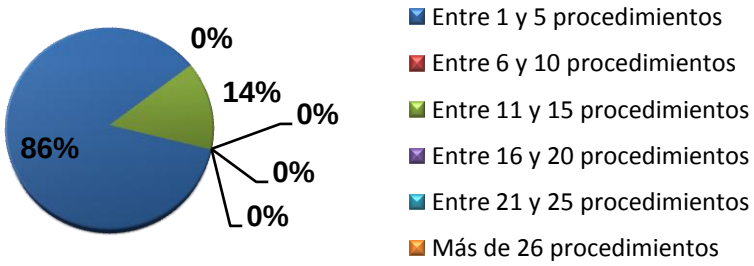

Figura 16. Representación en % de los resultados obtenidos agrupados en intervalos definidos según la métrica TOC.

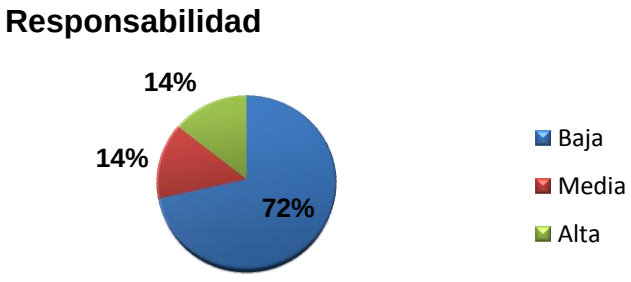

Figura 17: Representación los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

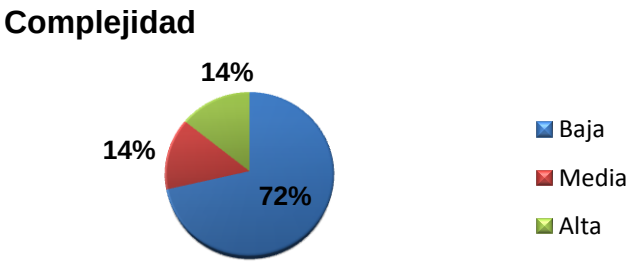

Figura 18: Representación de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

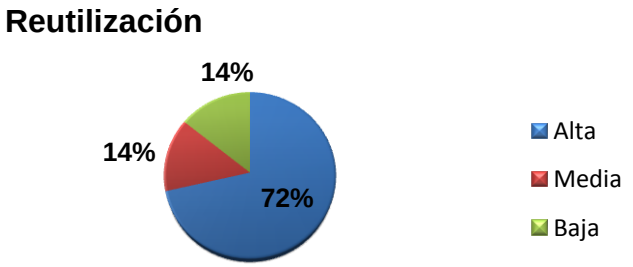

Figura 19. Representación de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Durante la evaluación de la métrica TOC los resultados obtenidos demostraron que la mayoría de las clases poseen menor cantidad de operaciones que la media registrada (86%), por lo que se encuentra dentro de los niveles aceptables de calidad. Los atributos de calidad de las clases se encuentran en un nivel medio satisfactorio (72%); de manera que se puede confirmar la elevada reutilización (elemento clave en el proceso de desarrollo de software) y cómo se reducen en menor grado la responsabilidad y la complejidad de implementación.

3.2.2 Métrica de Relaciones entre Clases (RC).

La métrica RC está dada por la cantidad de relaciones existentes entre las clases contenidas en el diseño, a partir de las cuales se determina la afectación que éstas ejercen dentro de la eficiencia del sistema.

A continuación se describen los tipos de afectaciones que se pueden observar al evaluar el diseño según la métrica RC.

❖**Acoplamiento:** Dependencia o interconexión de una clase o estructura de clase respecto a otras.

❖**Cantidad de pruebas:** Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto (componente) diseñado.

❖**Complejidad del mantenimiento**: Nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto. Puede influir significativamente en los costes y la planificación del proyecto.

❖ **Reutilización:** Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.

Ver los instrumentos y la tabla de resultados para la métrica RC en el Anexo 8.

A continuación se muestra el rango de valores y los resultados por cada uno de los atributos.

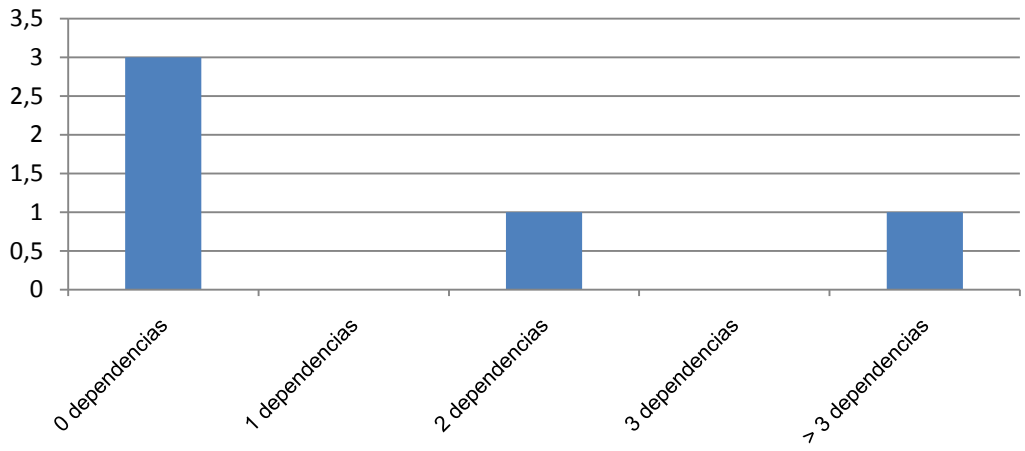

Figura 20. Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Figura 21. Representación en % de los resultados obtenidos agrupados en intervalos definidos según la métrica RC.

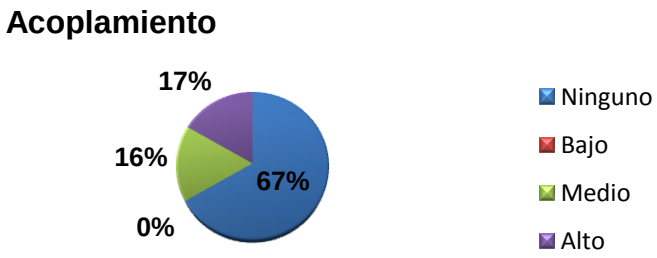

Figura 22. Representación de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

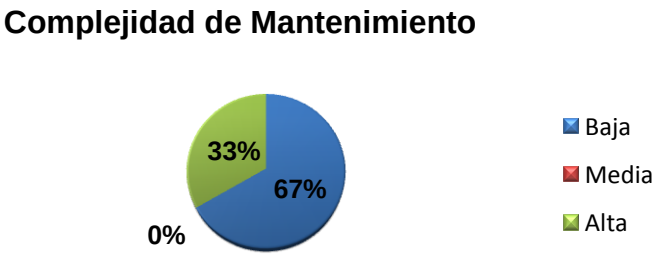

Figura 23. Representación de los resultados de la métrica RC en el atributo Complejidad de Mantenimiento.

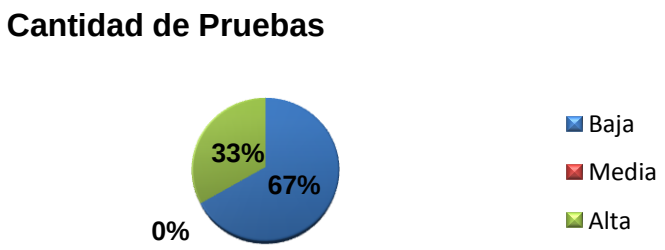

Figura 24. Representación de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

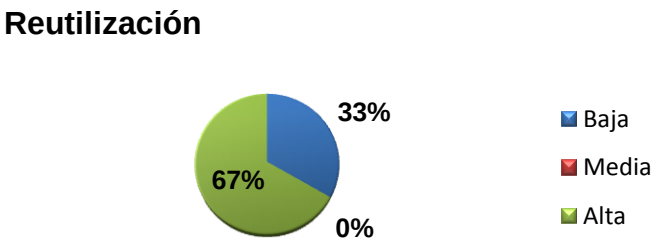

Figura 25. Representación de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Durante la evaluación de la métrica RC los resultados obtenidos demostraron que la mayoría de las clases (60%), poseen menos de 3 dependencias entre clases, por lo que se encuentra dentro de los niveles aceptables de calidad. Los atributos de calidad se encuentran en un nivel satisfactorio; en el 67% de las clases el grado de dependencia o acoplamiento es mínimo, la complejidad de mantenimiento, la cantidad de pruebas y la reutilización se comportan favorablemente para un 67% de afectación en las clases.

3.3 Pruebas de Software:

Para determinar el nivel de calidad del sistema se deben efectuar medidas o pruebas que permitan comprobar el grado de cumplimiento respecto de las especificaciones iníciales definidas para el mismo.

Los avances tecnológicos conjuntamente con la competencia en el ámbito informático a nivel mundial exigen software de calidad y es por esto que la aplicación de pruebas de este tipo durante todas las etapas de desarrollo de un proyecto evita la ocurrencia de errores y defectos en el producto final.

3.3.1 Prueba de caja negra o funcional

Este tipo de prueba se centra en lo que se espera de un sistema, es decir, intentan encontrar casos en que el sistema no se atiene a su especificación. Esta prueba se limita a brindar solo datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el sistema internamente, es decir, solo trabaja sobre su interfaz externa. En esencia permite encontrar:

- ❖ Funciones incorrectas o ausentes.
- ❖ Errores de interfaz.
- ❖ Errores en estructuras de datos o en accesos a las bases de datos externas.
- ❖ Errores de rendimiento.
- ❖ Errores de inicialización y terminación.

Dentro de la prueba de caja negra se incluyen las siguientes técnicas:

❖ **Partición de Equivalencia:** Divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

❖ **Análisis de Valores Límites:** Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

La Partición de Equivalencia divide el dominio de entrada de una aplicación en un número finito de variables de equivalencia donde para el presente caso se definen dos tipos, las válidas, que representan entradas válidas al programa, y las no válidas, que representan valores de entrada erróneos.

Tabla 14. Caso de prueba de Caja Negra para validar la funcionalidad Buscar solución.

Nombre de la funcionalidad	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Buscar solución	Se buscan los casos más parecidos al entrado por datos y se muestran los detalles	EP 1.1: Realizar la búsqueda de los casos introduciendo datos válidos.	– Se introducen los datos del caso que se desea buscar. – Se presiona el botón **Buscar solución.**

	de los mismos conjuntamente con la solución propuesta.	EP 1.2: Realizar la búsqueda de los casos introduciendo datos inválidos.	– Se introducen los datos del caso que se desea buscar. – Se presiona el botón **Buscar solución**.
		EP 1.3: Realizar la búsqueda de los casos dejando campos obligatorios en blanco.	– Se introducen los datos del caso que se desea buscar dejando algún campo obligatorio en blanco. – Se presiona el botón **Buscar solución**.
		EP 1.4: Cancelar la operación.	– Se introducen o no los datos del caso a buscar. – Se presiona el botón **Cancelar operación.**

Descripción de variable.

Tabla 15. Descripción de las variables del caso de prueba para la funcionalidad Buscar solución.

No.	Nombre de campo	Clasificación	Válido	Inválido
1	Valor	Lista desplegable	NA	NA
2	Cantidad de casos	Numérico	NA	NA

Juego de datos a probar.

Tabla 16. Datos de prueba del caso de prueba para la funcionalidad Buscar solución.

Id del escenario	Escenario	Valor	Cantidad de casos	Respuesta del sistema	Resultado de la prueba
EP 1.1	Realizar la búsqueda de los casos introduciendo datos válidos.	V(Linux)	V(10)	El sistema realiza la búsqueda de los casos más similares.	
EP 1.2	Realizar la búsqueda de los casos introduciendo datos inválidos.	NA	NA	El sistema no permite realizar la búsqueda con datos incorrectos.	
EP 1.3	Realizar la búsqueda de los casos dejando campos en blanco.	I(Vacío)	V(20)	El sistema emite una alerta: "Debe de introducir los valores asociados al caso que desea buscar."	
		V(Windows)	I(Vacío)	El sistema emite una alerta: "Debe de introducir la cantidad de casos a buscar."	

Luego de aplicados los métodos de prueba a la funcionalidad Buscar solución, es válido señalar que los resultados obtenidos hasta el momento han sido satisfactorios desde el punto de vista interno y funcional del sistema, atendiendo al correcto comportamiento del mismo ante diferentes situaciones (entradas válidas y no válidas). Ver resto de los Caso de prueba de Caja Negra en Anexo 5.

3.4 Conclusiones del capítulo.

La evaluación de la calidad del sistema desarrollado fue el elemento clave del capítulo que recién concluye. En ese sentido fueron aplicadas las métricas: Relaciones entre Clases y Tamaño Operacional de la Clase para validar y evaluar el diseño, las cuales arrojaron valores satisfactorios para cada uno de los indicadores correspondientes. También se efectuaron pruebas de software mediante casos de pruebas, para los cuales se tuvieron en cuenta las entradas, las salidas y los resultados esperados pudiéndose decir que el sistema cumple con los requerimientos y expectativas del cliente.

Conclusiones generales

Una vez terminado el presente trabajo de diploma se puede concluir que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos, para esto:

Fueron estudiadas las diferentes funciones de distancia y algoritmos más usados por los sistemas expertos además de ser mencionados los sistemas tanto nacionales como internacionales vinculados al campo de acción; evidenciándose de esta manera la no existencia de una solución informática capaz de ejecutar las funcionalidades referentes a la gestión de los errores producidos en el proceso de desarrollo dentro del Proyecto ERP-Cuba el cual cumpliera con los requerimientos de este.

Se realizó el diseño y la implementación de un sistema con el objetivo de erradicar los problemas de los sistemas existentes siempre atendiendo a los requerimientos del proyecto para lo cual fueron estudiados factores muy importantes como el marco de trabajo, el modelo de desarrollo y las diferentes técnicas y herramientas definidas por la dirección del mismo.

Se evaluó la viabilidad de la aplicación a través de pruebas de software efectuadas, las cuales arrojaron resultados favorables posibilitando dar cumplimiento a las funcionalidades previstas para el mismo.

La solución propuesta es novedosa, su importancia radica en la posibilidad de encontrar la solución de un determinado problema de forma rápida, permitiendo el ahorro de tiempo y recursos dentro del proyecto.

Recomendaciones

Al concluir el presente trabajo de diploma, se recomienda:

- ❖ Continuar realizando pruebas al sistema para detectar posibles debilidades y agregar mejoras al mismo.
- ❖ Continuar optimizando los algoritmos para un mejor funcionamiento interno del sistema.
- ❖ Incluir mejoras y funcionalidades nuevas al sistema de forma que sea más configurable y preste mejor servicio a los usuarios.
- ❖ Profundizar en temas referentes a los sistemas expertos y valorar la posibilidad de incrementar las funcionalidades del mismo a medida que se desarrolle una versión posterior.

Bibliografía

1. **Riesbeck, Christopher and Shank, Roger.** *Inside Case Based Reasoning.* New York : s.n., 1989.
2. **Corrillo Verdun, José Domingo.** *Metodología para el desarrollo de sistemas expertos.* Madrid, España : s.n.
3. **Kolodner, Janet.** *Case-Based Reasoning.* 1993.
4. **Araya, Selim Díaz.** *Sistemas Expertos, un paso en la simulación del razonamiento humano.* Costa Rica : s.n., 2004.
5. **Arean Rodrígruez, Ing. Yalila.** *Una herramienta de búsqueda inteligente en bases de datos, utilizando técnicas de Razonamiento Basado en Casos.* La Habana : s.n., 2003.
6. **Irigoien Garbizu, Itziar.** *Métodos basados en distancias estadísticas en comparación y clasificación de poblaciones.* País Vasco : s.n., 2008.
7. **Duda, R.O., Hart, P.E. and Stork, D.G.** *Pattern classification (2nd Edition).* s.l. : Wiley-Inter Science, 2000.
8. **García Cambronero, Cristina and Gómez Moreno, Irene.** *ALGORITMOS DE APRENDIZAJE: KNN & KMEANS.* Madrid, España : s.n.
9. desarrolloweb.com. [Online] 2006. [Cited: 04 20, 2011.] http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.
10. **Gómez Baryolo, Ing. Oiner, Morejón Borbón, Ing. Yoandry and García Tejo, Ing. Darien.** *ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE.* Ciudad de la Habana : s.n., 2010.
11. **Vallecillo, Antonio, Fuentes, Lidia and Troya, José M.** *Desarrollo de Software Basado en Componentes.* Málaga, España : s.n., 2007.
12. debug_mode=on. [Online] 2009. [Cited: 05 03, 2011.] http://es.debugmodeon.com/articulo/el-patron-mvc.
13. **Pérez Pérez, Joisel, Chaviano Gómez, Enrique and Vázquez Zambrano, Donel.** *Diseño de solución informática para la gestión y control de los costos en las entidades.* Ciudad de la Habana : s.n., 2009.
14. **Alexander, Christopher.** *A Pattern Language (vol II).* 1977.
15. **Gamma, Erich, et al.** *Design Patterns: Elements of Reusable Object-Oriented Software.* 1995.

16. El mundo informático. [Online] [Cited: 05 05, 2010.] http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman.
17. **Prieto, Félix.** *Patrones de diseño.* 2009.
18. Lenguaje HTML. [Online] [Cited: 03 12, 2011.] http://www.desarrolloweb.com/articulos/711.php.
19. Extensible Markup Language (XML). [Online] 2003. http://www.w3.org/XML/.
20. territoriopc. [Online] 2001. [Cited: 04 30, 2011.] http://www.territoriopc.com/javascript/tutorial_javascript_introduccion.php.
21. Doctrine. *Doctrine.* [Online] [Cited: 04 10, 2011.] http://www.doctrine-project.org.
22. Zend Framework. [Online] [Cited: 02 10, 2011.] http://framework.zend.com/manual/en/.
23. AbartiaTeam. [Online] 2006. [Cited: 03 25, 2011.] http://www.abartiateam.com/desarrollo-web/200602_uso-de-la-tecnologia-ajax-en-el-desarrollo-web.
24. visual-paradigm. *visual-paradigm.* [Online] 2005. [Cited: 04 15, 2011.] http://www.visual-paradigm.com.
25. Subversion. [Online] 2010. [Cited: 04 23, 2011.] http://subversion.apache.org/.
26. Documentación del Servidor HTTP Apache 2.0. [Online] 2009. [Cited: 04 12, 2011.] http://httpd.apache.org/docs/2.0/es/.
27. **Haugeland, John.** 1995.
28. *Aprendizaje Deductivo, Inductivo, y Abductivo con Algoritmos Genéticos.* **Boadas, Jose Alejandro Brito.** 10, s.l. : Revista Ingeniería Informática, 2004.
29. Conocimientos-La divisa del nuevo milenio. *Conocimientos-La divisa del nuevo milenio.* [Online] [Cited: 02 12, 2011.] http://conocimientosweb.net/portal/article150.html.
30. **Márquez, Juan José Samper.** REDcientífica. *REDcientífica.* [Online] 1997. [Cited: 03 03, 2011.] http://www.redcientifica.com/doc/doc199908210001.html.
31. **Borrero, Jaime Andrés Lara.** *UTILIZACIÓN DE SISTEMAS EXPERTOS PARA LA OPTIMIZACIÓN DE LA TOMA DE DESICIONES MULTICRITERIO.* Colombia : s.n., 2004.
32. **Gutiérrez, José Manuel.** *Sistemas Expertos Basados en Reglas.* España : s.n., 2000.
33. **Durking, J.** *Expert Systems: Design and Development.* New York : Maxwell Macmilan, 1994.
34. **E. Castillo, JM. Gutiérrez, H. Hadi.** *Expert Systems and Probabilistic Network Models.* New York : Versión Española editada por la Academia Española de Ingeniería, 1997.

35. **Mora, Carlos León de.** *Aplicación de los sistemas expertos a la gestión integrada de averías en redes de telecomunicaciones. .* Sevilla, España : s.n., 2003.

36. Pruebas de Software. [Online] 2010. [Cited: 04 12, 2011.] http://lsi.ugr.es/~ig1/docis/pruso.pdf.

37. GNU Operating System. [Online] 2009. [Cited: 03 07, 2011.] http://www.gnu.org/philosophy/free-sw.html.

38. **Fox, Javier E.** *Sistemas expertos y su aplicación en la medicina.* 1991.

39. **García Borroto, Milton, et al.** *Selección y construcción de objetos para el mejoramiento de un clasificador supervisado: un análisis crítico.* 2008.

40. **Béjar, Javier.** *Resolución de problemas. Algoritmos de búsqueda.* Barcelona : s.n., 2007.

41. **Sanz Cuenca, Julián.** *Reconocimiento de objetos por descriptores de formas.* Barcelona : s.n., 2008.

42. **Troncoso Lora, Alicia.** *Técnicas avanzadas de predicción y optimización aplicadas a sistemas de potencia. .* Sevilla : s.n., 2005.

43. **Anibal Bregón, Arancha Simón, et ol.** *Un sistema de razonamiento basado en casos para la clasificación de fallos en sistemas dinámicos.* España : s.n., 2005.