Universidad de las Ciencias Informáticas

Cedrux

Implementación del proceso de Traspasos en el módulo Costos y Procesos del Sistema Integral de Gestión Cedrux: Operaciones con Secuencias de Traspasos

Trabajo para optar por el título de:

Ingeniero en Ciencias Informáticas

Autor: Luis Jorge Alva Rodríguez

Tutores: Ing. José Luis Céspedes Martínez
Ing. Joisel Pérez Pérez

Ciudad de La Habana, Junio 2009

Dedicatoria

A mis padres y abuelos por el apoyo y cariño que me han brindado a lo largo de todos estos años…

Agradecimientos

Le agradezco a mi novia Yissell porque con su amor y su paciencia ha estado siempre dispuesta a apoyarme.

A mis suegros Mayito y Mary por haberme acogido como a un hijo todo este tiempo.

A mis padres, abuelos, hermanos, tíos, primos; a mi familia en general por su amor, su dedicación, su ayuda, sus consejos y su apoyo incondicional.

A mis amigos Manolo, Dasiel y Osnier por su amistad.

A mis tutores Jose y Joisel, a mis compañeros de la universidad y al colectivo del proyecto ERP que compartió conmigo este tiempo por su apoyo en el desarrollo del trabajo de diploma.

Resumen

El costo es un elemento normativo y evaluador de la gestión de la entidad. Es importante destacar que la determinación de los costos no corresponde solo a las empresas, debiendo constituir actividades obligadas en aquellas unidades presupuestadas autofinanciadas o no, que tengan autorizadas actividades productivas y comerciales con peso económico significativo. El subsistema de costos es el encargado de automatizar toda la gestión de los costos de la entidad que lo utilice, pudiendo medir la suma de gastos de toda naturaleza monetaria, que se aplica a una producción o servicio determinado en dicha entidad.

Dentro de las responsabilidades de la gestión de costos se encuentra el proceso de Cierre y Traspaso, en el cual se realizarán todos los cierres de período y de ejercicio contables de la entidad, así como operaciones relacionadas con los traspasos de gastos.

La implementación del proceso “Traspasos” brinda la posibilidad de completar el módulo de Gestión de Costos, actividad fundamental dentro de la contabilidad financiera de una empresa. Debido a diferentes problemas que presentan los distintos software que utilizan las entidades cubanas y la necesidad del país de no depender de tecnologías propietarias se hizo imprescindible el desarrollo de una aplicación que facilitara dichos trámites económicos.

El presente trabajo de diploma expone como se implementó dicho proceso; herramientas utilizadas, lenguajes de programación, arquitecturas y demás aspectos de interés.

Índice

Índice de Tablas

Índice de Figuras

Introducción

Actualmente en las Empresas, la Contabilidad de Costos constituye la piedra angular para medir su eficiencia. La Gestión de los Costos constituye uno de sus principales procesos en el control de los recursos disponibles. Uno de los procesos fuertemente implicados en la Gestión de los Costos lo constituye “Cierres y Traspasos”, este posibilita ajustar con más precisión los gastos generados por concepto de producción o servicios brindados.

Existen varios factores que influyen en el correcto funcionamiento de la Gestión de los Costos, por solo mencionar algunos ejemplos: el trabajo resulta engorroso realizarlo manualmente y las aplicaciones existentes no cumplen con todos los requisitos, las empresas cubanas utilizan diferentes herramientas automatizadas importadas y otras nacionales. La variedad de las mismas y la imposibilidad de brindarles el mantenimiento que requieren dificulta la gestión eficiente de la economía en las entidades, esto unido a las características económicas del Sistema Socialista Cubano y a la dualidad de moneda existente en el país provoca que muchas de estas aplicaciones que funcionan perfectamente para otras empresas en el mundo sean inadecuadas para las cubanas, pues responden a otros tipos de economías, repercutiendo negativamente en la actividad contable de las mismas.

En la Universidad de las Ciencias Informáticas actualmente se encuentra en desarrollo un Sistema de Gestión Empresarial que responda a las necesidades contables de las entidades cubanas. El sistema se implementa a través de módulos correctamente diseñados que permitan la comunicación eficiente entre los diferentes subsistemas. El módulo de Costo y Procesos, responsable de gestionar las operaciones asociadas al Costo, no cuenta con la posibilidad de realizar el traspaso de gastos indirectos entre cuentas de gastos patrimoniales.

Ante la situación descrita con anterioridad el presente trabajo de diploma se propone darle solución al siguiente **problema** expresado mediante la siguiente incógnita: ¿Cómo garantizar la realización de operaciones con Secuencias de Traspasos entre Cuentas de Gastos Patrimoniales en el módulo de “Costos y Procesos” del Sistema Integral de Gestión para hacer más eficiente la gestión de los costos en las Entidades Cubanas?

De ahí que el **objeto de estudio** sea entonces: Los procesos contables de Secuencias de Traspasos en la Contabilidad de Costos para el cual se define el siguiente **campo de acción:** Implementación del proceso de Secuencias de Traspasos del Sistema Integral de Gestión Cedrux.

Para dar respuesta al problema planteado se trazó el siguiente **objetivo general**: Obtención del proceso de Secuencia de Traspaso mediante su implementación en el Sistema Integral de Gestión Cedrux.

Objetivos Específicos:

1. Caracterizar y evaluar la realización de operaciones con Secuencias de Traspasos entre Cuentas de Gastos Patrimoniales en el módulo de "Costos y Procesos" del Sistema Integral de Gestión de Entidades.
2. Fundamentar la investigación, mediante la elaboración del Marco Teórico.
3. Obtener mediante la implementación el diseño de clases de los requerimientos funcionales "Configurar Secuencias de Traspasos entre Cuentas de Gastos Patrimoniales" y "Ejecutar Secuencias de Traspasos entre Cuentas de Gastos Patrimoniales" del proceso Cierres y Traspasos.

Tareas para cumplir los objetivos:

1. Estudio del estado del arte referente a soluciones informáticas asociadas al área de conocimientos de "Costos y Procesos" de la Contabilidad General y de Costos.
2. Estudio de la Arquitectura de Software del Módulo "Costos y Procesos".
3. Estudio del análisis del requerimiento funcional "Configurar Secuencias de Traspasos entre Cuentas de Gastos Patrimoniales" del proceso Cierres y Traspasos".
4. Estudio del análisis del requerimiento funcional "Ejecutar Secuencias de Traspasos entre Cuentas de Gastos Patrimoniales" del proceso Cierres y Traspasos".
5. Estudio del diseño de clases del requerimiento funcional "Configurar Secuencias de Traspasos entre Cuentas de Gastos Patrimoniales" del proceso Cierres y Traspasos".
6. Estudio del diseño de clases del requerimiento funcional "Ejecutar Secuencias de Traspasos entre Cuentas de Gastos Patrimoniales" del proceso Cierres y Traspasos".
7. Implementación del requisito funcional "Crear Secuencias de Traspasos".

8. Implementación del requisito funcional “Modificar Secuencias de Traspasos”.
9. Implementación del requisito funcional “Eliminar Secuencias de Traspasos”.
10. Implementación del requisito funcional “Consultar Secuencias de Traspasos”.
11. Implementación del requisito funcional “Ejecutar Secuencias de Traspasos”.
12. Implementación del requisito funcional “Visualizar Comprobante de Traspasos Generados”.

Para el correcto desarrollo del presente trabajo de Diploma se emplearon métodos científicos de corte teóricos y empíricos tratando de mantener siempre un equilibrio entre lo cualitativo y lo cuantitativo. Entre los métodos teóricos se utilizaron los de Análisis – Síntesis, Inducción – Deducción para el estudio y análisis de la bibliografía, la formulación del problema y los objetivos, así como para la elaboración de las conclusiones y recomendaciones, a fin de alcanzar los objetivos de la investigación. Además se utilizó el método teórico Histórico – Lógico permitiendo hacer un estudio del comportamiento actual del desarrollo de sistemas de gestión empresarial.

Se utilizó también el método Sistémico, empleado en el establecimiento de las jerarquías funcionales que deben mantener los métodos implementados para dar solución a los requerimientos funcionales del sistema automático a desarrollar, permitiendo un desglose de los elementos que contendrá el mismo.

El método empírico utilizado fue la Observación, a través del cual se pudo percibir y planificar como quedaría concebido el sistema.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En el presente capítulo se realiza una breve descripción del proceso de Secuencias y Traspasos haciendo un estudio de algunos sistemas que actualmente son usados tanto a nivel nacional como internacional. Se mencionan los lenguajes, tecnologías y herramientas de desarrollo a utilizar en la implementación del proceso.

1.2 Conceptos básicos asociados al dominio del problema

Centro de Costo: Es la subdivisión mínima en el proceso de registro contable de los gastos en la entidad con el objetivo de permitir la medición de los recursos utilizados y los resultados económicos obtenidos. (1)

Elemento del Gasto: Es un concepto económico asociado al gasto que permite la cuantificación de los recursos materiales, laborales y monetarios en los cuales se expresan los gastos de trabajo vivo y pretérito para un período en el conjunto de la actividad empresarial. (2)

Las **cuentas de gastos** son aquellas que recogen los gastos que se producen como consecuencia de la actividad empresarial y que originan una disminución del neto patrimonial anotándose en el debe. Estas cuentas se regularizarán al cierre del ejercicio económico y se englobarán en cuentas de resultados, las cuales a su vez se recogen en la cuenta única de pérdidas y ganancias que ofrece la diferencia global lograda en el período. (3)

Los **gastos indirectos** de producción comprenden los importes de los gastos que se incurren en las actividades asociadas a la producción, no identificables con un producto o servicio determinado. Incluyen los gastos de las actividades de mantenimiento, reparaciones corrientes y explotación de equipos, dirección de la producción, control de calidad, depreciación de Activos Fijos Tangibles de producción y servicios auxiliares a ésta, entre otros. (4)

Los **gastos directos** de producción comprenden los importes de los gastos que se incurren en las actividades asociadas a la producción, identificados con un producto o servicio determinado, ya sea en la materia prima o mano de obra. (5)

Llevar a cabo una **Secuencia de Traspaso** constituye traspasar los saldos de las cuentas de gastos indirectas a cuentas de gastos directas.

1.3 Sistemas existentes vinculados al campo de acción

Se investigó sobre las principales aplicaciones web y de escritorio, similares a la que se desea desarrollar, caracterizándolas y describiendo sus principales funcionalidades. A continuación, se analizan sistemas relacionados con la Contabilidad de Costo en el ámbito internacional y nacional.

1.3.1 Open Bravo

Software opensource creado en el año 2005 desarrollado en España, presenta el módulo Gestión de la producción el cual cubre la planificación de la producción, aprovisionamientos, órdenes de fabricación, partes de trabajo, cálculo de los costes de producción, notificación de incidencias de trabajo y partes de mantenimiento.

Características:

- Implementado en el lenguaje Java.
- Aplicación completamente web desarrollada siguiendo el modelo MVC (Model, View, Control).
- Soporte para bases de datos PostgreSQL y Oracle.
- Se ejecuta sobre Apache y Tomcat.

1.3.2 OpenERP

Software opensource desarrollado en Bélgica, cuenta con un módulo Gestión contable y financiera el cual provee contabilidad general, analítica y presupuestaria, y cuenta con todas las funcionalidades para llevar los libros contables en forma rigurosa, puede definir centros de costos, gestionando de una manera eficiente la contabilidad analítica, posee también contabilidad automática de doble entrada, que se combina con una gran cantidad de herramientas de informes y análisis totalmente integradas.

Características:

- Implementado en Python.
- Tiene componentes separados en esquema Cliente-servidor.
- Soporte para bases de datos PostgreSQL.
- Es multiplataforma.

1.3.3 SAP

Software desarrollado en Alemania, por antiguos empleados de IBM. Actualmente es la quinta compañía más grande en el mundo del software. Cuenta con el módulo Controlling (CO), que permite el control de los gastos generales, costes de producto, cuenta de resultados y centros de beneficio. También se realiza directamente en el módulo CO la imputación (subreparto y distribución) de gastos generales que se lleva a cabo en el desplazamiento de períodos y se refleja en los datos de la Contabilidad de centros de beneficio. Las funciones de imputación Subreparto y Distribución permiten transferir costes e ingresos plan o reales (como por ejemplo el recargo de gastos generales) desde un centro de beneficio a otros.

Características:

- Implementado en .NET y WebSphere.
- SAP también ofrece una nueva plataforma tecnológica denominada SAP NetWeaver, esta plataforma tecnológica convierte a SAP en un programa Web-enabled, lo que significa que estaría totalmente preparado para trabajar con él mediante la web.
- Trabaja sobre el sistema operativo Windows.
- Soporte para bases de datos Oracle.

1.3.4 Versat-Sarasola

Es un software cubano de contabilidad confiable desarrollado por la entidad cubana TEICO en Villa Clara. Versat consta de 12 módulos entre ellos Costos y Procesos el cual es un complemento del módulo Contabilidad General, pues además del registro contable de gastos, incluye dos actividades muy relevantes: El traspaso o distribución de los gastos indirectos hacia los centros de costo y el ajuste o costeo, según los volúmenes de producción o servicios y sus destinos.

Características:

- Es una aplicación de escritorio.
- Implementado en Delphi.
- Trabaja sobre el sistema operativo Windows.
- Soporte para bases de datos SQL Server 2000.

1.3.5 SISCONT5

Sistema cubano creado por la empresa Tecnomática en el año 2007, el cual se aviene a las definiciones y conceptos del Ministerio de la Industria Básica (MINBAS) aunque por las acciones contables financieras que permite puede ser utilizado en otras entidades nacionales. Está formado por varios módulos, entre ellos se encuentra el de Contabilidad de Costos en el cual se registran todos los gastos atendiendo al objeto de costo que afectan, la distribución y el cálculo de costos unitarios, también permite definir las bases de distribución de los costos indirectos de un objeto de costo a otros, definir el método de traspaso de gasto y procesos tecnológicos con su fase de producción.

SISCONT5 puede ser explotado en régimen mono usuario y multiusuario. Se define para mono entidad y multientidad, para esta última existe el control de su acceso para las entidades en un mismo equipo de cómputo como servidor.

Características:

- Soporte para bases de datos SQL Server 2000.
- Trabaja sobre el sistema operativo Windows.
- Hecho en la herramienta de desarrollo de software basada en conocimiento GeneXus.

1.4 Valoración del estado del arte

Después de haber estudiado los sistemas anteriores con sus respectivos módulos relacionados con el traspaso o distribución de gastos indirectos podemos resumir que no todos son libres, teniendo que pagar licencias para su uso, la mayoría son aplicaciones de escritorio las cuales tienen que tener espacio en disco suficiente para ser instaladas y si el usuario no cuenta con una computadora con buenos recursos de hardware el trabajo en la parte del cliente se sobrecargaría, debido a que es en ésta donde se realiza la mayor parte del procesamiento. Otro factor a tener en cuenta es que al ser aplicaciones de escritorio dependen del sistema operativo para poder funcionar, esto implica que si no es multiplataforma no va a funcionar sobre otro lo cual no ocurre con una aplicación web. Por otra parte los dos sistemas más utilizados en Cuba (SISCONT5 y Versat-Sarasola) corren sobre gestores de Bases de Datos propietarios. En cuanto a las características de los lenguajes utilizados en la implementación de los sistemas se puede decir que el lenguaje java en aplicaciones web a la hora de acceder a las páginas es más lento debido a la sobrecarga que genera la máquina virtual; de .Net, podemos señalar que es un lenguaje propietario y lento para aplicaciones web. Sería entonces factible que el proyecto y el país

en el proceso por lograr su independencia tecnológica trabajara sobre herramientas y lenguajes que no sean propietarios siendo una aplicación web y el trabajo con PHP lo más indicado para desarrollar cualquier aplicación.

1.5 Tendencias y tecnologías actuales utilizadas

Se desarrollará una **Aplicación Web** la cual los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador, ésta tiene entre sus ventajas:

Multiplataforma: una misma versión de la aplicación puede correr sin problemas en múltiples plataformas como Windows, Linux, Mac, etc.

Actualización: las aplicaciones web siempre se mantienen actualizadas y no requieren que el usuario deba descargar actualizaciones y realizar tareas de instalación.

Acceso inmediato: las aplicaciones basadas en tecnologías web no necesitan ser descargadas, instaladas y configuradas, estas pueden ser accedidas desde cualquier computadora conectada a la red con solo tener un navegador para levantar la aplicación.

Menos requerimientos de hardware: las aplicaciones web consumen poco espacio en disco y también es mínimo el consumo de memoria RAM, tampoco es necesario disponer de computadoras con poderosos procesadores ya que la mayor parte del trabajo se realiza en el servidor en donde reside la aplicación.

Seguridad: los servidores donde se guardan los datos son altamente confiables.

1.6 Lenguajes y plataformas de desarrollo

1.6.1 Lenguajes

Un lenguaje de programación puede ser utilizado para controlar el comportamiento de una computadora, este consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. 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. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural.

1.6.2 Lenguajes del lado del servidor

Estos lenguajes de programación en la tecnología cliente-servidor son clasificados así, ya que se ejecutan del lado del servidor y los usuarios sólo obtienen el beneficio del procesamiento de la información.

1.6.2.1 Lenguaje PHP

PHP es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Es un acrónimo recursivo que significa "PHP Hypertext Pre-processor". 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. Es un potente lenguaje de secuencia de comandos diseñado específicamente para permitir a los programadores crear aplicaciones en la web con distintas prestaciones de forma rápida. Su interpretación y ejecución se realiza en el servidor en el cual se encuentra almacenada la página y el cliente sólo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página web, enriquecida con código PHP, el servidor interpretará las instrucciones mezcladas en el cuerpo de la página y las sustituirá con el resultado de la ejecución antes de enviar el resultado a la computadora del cliente. Además es posible utilizarlo para generar archivos PDF, Flash o JPG, entre otros. Permite la conexión a numerosas bases de datos de forma nativa tales como Postgres, MySQL, Oracle, ODBC, IBM DB2, Microsoft SQL Server y SQLite, lo cual permite la creación de Aplicaciones web muy robustas.

PHP tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX, Linux, Windows y Mac OS X, y puede interactuar con los servidores de web más populares.

Ventajas:

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.

- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones.

Desventajas:

- No posee una abstracción de base de datos estándar, sino bibliotecas especializadas.
- Por sus características promueve la creación de código desordenado y complejo de mantener.
- Todo el trabajo lo realiza el servidor, por tanto puede ser más ineficiente a medida que aumenten las solicitudes.
- La orientación a objetos es aún muy deficiente para aplicaciones grandes.

1.6.2.2 Lenguaje C#

Es un lenguaje de programación orientado a objetos desarrollado, diseñado y estandarizado por Microsoft a propósito general para su plataforma .NET. Sus principales creadores son Scott Wiltamuth y Anders Hejlsberg, éste último también conocido por haber sido el diseñador del lenguaje Turbo Pascal y la herramienta RAD Delphi.

Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que programarla usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes ya que C# carece de elementos heredados innecesarios en .NET. Por esta razón, se suele decir que C# es el lenguaje nativo de .NET.

La sintaxis y estructuración de C# es muy parecida a la de C++ o Java, puesto que la intención de Microsoft es facilitar la migración de códigos escritos en estos lenguajes a C# y facilitar su aprendizaje a los desarrolladores habituados a ellos. Sin embargo, su sencillez y el alto nivel de productividad son comparables con los de Visual Basic.

Ventajas:

- Su código se puede tratar íntegramente como un objeto. Se ahorra tiempo en la programación ya que tiene una librería de clases muy completa y bien diseñada.

- Tipos de datos: en C# existe un rango más amplio y definido de tipos de datos que los que se encuentran en C, C++ o Java.
- Métodos virtuales y redefiniciones: antes de que un método pueda ser redefinido en una clase base, debe declararse como virtual. El método redefinido en la subclase debe ser declarado con la palabra override.
- Propiedades: un objeto tiene intrínsecamente propiedades, y debido a que las clases en C# pueden ser utilizadas como objetos, C# permite la declaración de propiedades dentro de cualquier clase.

Desventajas:

Las desventajas que posee el uso de este lenguaje de programación son que se tiene que conseguir una versión reciente de Visual Studio .NET lo cual por motivos de la compra de la licencia incurriría en una cuantiosa inversión, por otra parte es necesario tener algunos requerimientos mínimos del sistema para poder trabajar adecuadamente tales como contar con Windows NT 4 o superior y tener como mínimo 4 gigas de espacio libre en disco para su instalación.

1.6.2.3 Lenguaje Java

Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

La tecnología Java está compuesta básicamente por 2 elementos: el lenguaje Java y su plataforma. Con plataforma nos referimos a la máquina virtual de Java (Java Virtual Machine).

Una de las principales características del lenguaje Java es su capacidad de que el código funcione sobre cualquier plataforma de software y hardware. Esto significa que nuestro mismo programa escrito para Linux puede ser ejecutado en Windows sin ningún problema. Además es un lenguaje orientado a objetos que resuelve los problemas de complejidad en los grandes sistemas, entre otras.

Ventajas:

- Un lenguaje multiplataforma potente.
- Se puede descargar de forma gratuita.
- Lo más destacado de Java a la hora de ponerse a programar con él es la cantidad de paquetes (o librerías) que tiene disponible y que añaden una potencia increíble

al lenguaje.

Desventajas:

La principal desventaja que tiene este lenguaje es su compilador pues éste te dice si un programa al compilarlo tiene errores pero no sabe decir con precisión el lugar donde está el error. La Máquina Virtual de Java consume muchos recursos, por lo que debemos tener un ordenador con muy buenas propiedades sino notaremos mucho la recarga de memoria.

1.6.3 Lenguajes del lado del cliente

Un lenguaje del lado del cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. Es necesario tener en cuenta que en dependencia del navegador algunas páginas no se verán bien si el ordenador cliente no tiene instalados los plug-in adecuados. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y ello puede afectar a la seguridad.

1.6.3.1 HTML

HTML (HyperText Markup Language) Lenguaje de Marcas de Hipertexto, 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.

1.6.3.2 XML

XML (Extensible Markup Language) lenguaje de marcas es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML (Standard Generalized Markup Language) y permite definir la gramática de lenguajes específicos (de la misma manera que HTML (HyperText Markup Language) es a su vez un lenguaje definido por SGML)). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML (eXtensible Hypertext Markup Language), SVG (Scalable Vector Graphics), MathML (Mathematical Markup Language).

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

1.6.3.3 Javascript

JavaScript es un lenguaje de scripts desarrollado por Netscape para incrementar las funcionalidades del lenguaje HTML. Sus características más importantes son:

- JavaScript es un lenguaje interpretado, es decir, no requiere compilación. El navegador del usuario se encarga de interpretar las sentencias JavaScript contenidas en una página HTML y ejecutarlas adecuadamente.
- JavaScript es un lenguaje orientado a eventos. Cuando un usuario pincha sobre un enlace o mueve el puntero sobre una imagen se produce un evento. Mediante JavaScript se pueden desarrollar scripts que ejecuten acciones en respuesta a estos eventos.
- JavaScript es un lenguaje orientado a objetos. El modelo de objetos de JavaScript está reducido y simplificado, pero incluye los elementos necesarios para que los scripts puedan acceder a la información de una página y puedan actuar sobre la interfaz del navegador.

Javascript es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros.

1.6.3.4 Tecnología AJAX

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). 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. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

AJAX es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores ya que está basado en estándares abiertos como JavaScript y Document Object Model (DOM). AJAX es una combinación de cuatro tecnologías ya

existentes:

- XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.
- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XMLHttpRequest para intercambiar datos de forma asíncrona con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.
- XML es el formato usado generalmente para la transferencia de datos solicitados al servidor, aunque cualquier formato puede funcionar, incluyendo HTML pre formateado, texto plano, JSON y hasta EBML.

1.6.4 Control de versiones

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. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas, estos proporcionan un mecanismo de almacenaje de los elementos que deba gestionar y un registro histórico de las acciones realizadas con cada elemento o conjunto de elementos (normalmente brindando la posibilidad de volver o extraer un estado anterior del producto) entre otros aspectos. Todos los sistemas de control de versiones se basan en disponer de un repositorio, que es el conjunto de información gestionada por el sistema. Este repositorio contiene el historial de versiones de todos los elementos gestionados. Cada uno de los usuarios puede crearse una copia local duplicando el contenido del repositorio para permitir su uso.

1.6.4.1 Subversion

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.

SVN surge con la intención de sustituir y mejorar al conocido CVS (Concurrent Versions System). SVN mantiene las ideas fundamentales de CVS pero suple sus carencias y evita

sus errores. Las principales características de SVN son:

- Mantiene versiones no sólo de archivos, sino también de directorios
- Mantienen versiones de los metadatos asociados a los directorios.
- 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.
- Atomicidad de las actualizaciones, una lista de cambios constituye una única transacción o actualización del repositorio, esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
- Posibilidad de elegir el protocolo de red, además de un protocolo propio (svn), puede trabajar sobre http (o https) mediante las extensiones WebDAV. WebDAV (más conocido como DAV) es un protocolo que amplía las posibilidades del HTTP/1.1 añadiendo nuevos métodos y cabeceras. La capacidad de funcionar con un protocolo tan universal como el http simplifica la implantación (cualquier infraestructura de red actual soporta dicho protocolo) y universaliza las posibilidades de acceso (si se quiere, puede utilizarse a través de Internet).
- Soporte tanto de ficheros de texto como de binarios.
- Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos.
- Mayor eficiencia en la creación de ramas y etiquetas que en CVS.

1.7 Sistemas de gestión de base de datos

Los sistemas de gestión de base de datos (SGBD); (en inglés: Database Management System, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

1.7.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. PostgreSQL es una derivación libre (open source) de este proyecto. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. Algunas de sus características son:

- Implementación del estándar SQL92/SQL99.
- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP…), cadenas de bits. También permite la creación de tipos propios.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

1.8 Herramientas de desarrollo

1.8.1 Zend Studio para Eclipse

Es un IDE (Integrated Development Environment) entorno integrado de desarrollo para el lenguaje de PHP escrito en Java destinado a desarrolladores profesionales, propietario, compatible con las plataformas Linux, MAC y Windows. Incluye todos los componentes necesarios durante el ciclo de vida de una aplicación en PHP. Incluye editor, análisis, depuración, optimizadores de código y herramientas de base de datos. Zend Studio nos permite agilizar el desarrollo web y permite simplificar proyectos complejos. Posibilita un excelente completamiento de código, coloreado en la sintaxis del código, administración avanzada de proyectos, múltiples lenguajes, incorpora el Framework de Zend, manual de PHP, integración con subversión, integración avanzada con FTP, soporte para Web Services, PHP4 y PHP5, entre otras características están:

- No requiere la instalación previa de PHP ni del entorno de ejecución de Java.
- Detección de errores de sintaxis en tiempo real.
- Manual de PHP integrado.
- Soporte para navegación en bases de datos y ejecución de consultas SQL.
- Orece soporte básico para otros lenguajes web, como HTML, Javascript y XML.
- Emparejamiento (matching) de paréntesis y corchetes (si se sitúa el cursor sobre un paréntesis (corchete) de apertura (cierre), Zend Studio localiza el correspondiente paréntesis (corchete) de cierre (apertura)).

- Código de Cobertura.
- Acceso al ecosistema de plug-ins de Eclipse.
- Mecanismo de actualización automática.

1.8.2 EMS SQL Manager PostgreSQL

Es una aplicación de alto desempeño para la administración y desarrollo de PostgreSQL Database Server. El programa trabaja con cualquier versión de PostgreSQL y soporta todas las últimas características de PostgreSQL, incluyendo espacios de tablas, nombres de argumentos en funciones y más. Su interfaz gráfica es sumamente atractiva e incluye un modo guiado de trabajo. La versión Lite incluye las herramientas básicas de mantenimiento y administración.

Características:

- Soporte completo para PostgreSQL hasta la versión 8.3
- Administración y navegación rápida de bases de datos.
- Administración fácil de todos los objetos PostgreSQL.
- Administración efectiva de seguridad.
- Capacidades de exportación e importación de datos.
- Modo guiado para labores de mantenimiento.
- Interfaz atractiva.

1.8.3 Mozilla Firefox

Mozilla Firefox es uno de los navegadores open source más usados actualmente en todo el mundo. Firefox ha sido creado por el proyecto Mozilla el cual tiene como misión preservar la elección y la innovación en Internet. El apoyo organizativo del proyecto Mozilla es proporcionado por Mozilla Foundation (EUA), Mozilla Europe y Mozilla Japón.

Algunas características que ofrece Mozilla Firefox son:

- Navegación por tabs, esta es una de las principales características que tiene Firefox, es posible navegar por pestañas.
- Es Software libre.
- Trabaja de 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) y Mac (Desde Mac OS X 10.1.x a la 10.2.x y nuevas versiones).

- También existen excelentes extensiones y skins de fácil instalación, estos mejoran la usabilidad y el aspecto del navegador.

1.9 Conclusiones

Con el estudio de algunos sistemas usados actualmente a nivel nacional e internacional en la gestión de los costos se logró identificar las deficiencias de los mismos dando paso a su erradicación con la implementación de un software que gestione los procesos de traspasos, además de establecer las tecnologías mas adecuadas para la construcción del mismo, brindando una solución informática a la situación existente en las empresas presupuestadas del país para permitir la gestión de la información referida al Sistema Contable y Financiero.

Capítulo 2: Descripción y análisis de la solución propuesta

2.1 Introducción

En el presente capítulo se hace una profunda valoración del diseño de clases propuesto por el analista del sistema dando a conocer las principales ventajas que ofreció la obtención del mismo. Se explican los beneficios que ofrece la arquitectura utilizada adecuándose a las exigencias del proyecto. Fueron analizados además, los componentes (frameworks) o módulos ya existentes y que pudieran ser usados, lo que resulta muy ventajoso dado que la reutilización ofrece ventajas como la reducción de los esfuerzos de desarrollo y el mantenimiento, mejorando la seguridad, la eficiencia y la consistencia de los diseños.

2.2 Valoración del diseño propuesto por el analista

El diseño propuesto por el analista fue valorado como bueno, fácil de entender, las clases del diseño están bien representadas ya que desde él se pueden obtener las clases fundamentales que deben ser definidas para que el sistema funcione satisfactoriamente, así como los atributos y métodos que deben tener las mismas, dando una idea clara de lo que se debe implementar. Mientras más legible sea el diseño propuesto más fácil es la adaptación de los programadores al mismo y por ende se agiliza el proceso de traducción de clases del diseño a clases de implementación. **Ver Anexo 1.**

Una característica del diseño que evidencia la claridad del trabajo es el uso de grupos de patrones de diseño como es el caso de los GRASP ("General Responsibility Assignment Software Patterns"). Los patrones “GRASP” se utilizan con el objetivo de asignar responsabilidades a las diferentes clases que se definen en el diseño.

Entre los patrones GRASP usados se encuentra el Controlador que permite asignar la responsabilidad del manejo de mensajes de los eventos del sistema a una clase y el patrón Alta Cohesión que mantiene la complejidad dentro de límites manejables, permite además obtener clases que pueden ser reutilizadas y no son vulnerables a los cambios.

2.3 Análisis de posibles implementaciones, componentes o módulos que puedan ser rehusados

2.3.1 Zend Framework

Es un framework de alta calidad y de código abierto para el desarrollo de aplicaciones y servicios web con PHP.

Construido en el verdadero espíritu de PHP, ZendFramework 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. Presenta entre otras, las siguientes características: (6) (7)

- Proporciona un sistema de caché dividido en frontend y backend, de forma que se puedan almacenar en caché diferentes datos como resultados de funciones, páginas completas, etc., y que esta información se almacene en archivos, en memoria, en base de datos, etc.
- Simplifica la gestión de archivos de configuración.
- Proporcionan los componentes que forma la infraestructura del patrón MVC.
- Proporciona una capa de acceso a base de datos, construida sobre PDO pero ampliándola con diferentes características.
- Proporciona mecanismos de filtrado y validación de entradas de datos.
- Permite convertir estructuras de datos PHP a JSON y viceversa, para su utilización en aplicaciones AJAX.
- Proporciona capacidades de búsqueda sobre documentos y contenidos.
- Permite consumir y proveer servicios web.

2.3.2 Doctrine

Doctrine es un potente y completo sistema ORM (object relational mapper) para PHP 5.2+ que incorpora una DBL (capa de abstracción a base de datos). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientado a objeto.

Esto les proporciona una alternativa poderosa a diseñadores de SQL manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. (8)

Funcionalidades:

- Exportar una base de datos existente a sus clases correspondientes y también convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos.

2.3.3 Ext

Es una librería construida con JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. Ext 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. (9)

Funcionalidades:

Dispone de un conjunto de componentes gráficos (widgets) para construir atractivos desarrollos al estilo Web 2.0, como:

- Cuadros y áreas de texto.
- Campos para fechas.
- Campos numéricos.
- Combos.
- Botones (Radiobuttons) y Cuadros de marcado (checkboxs).
- Editor HTML.
- Elementos de datos (con modos de sólo lectura, datos ordenables, columnas que se pueden bloquear y arrastrar, etc.).
- Árbol de datos.
- Pestañas.
- Barra de herramientas.
- Menús al estilo de Windows.
- Paneles divisibles en secciones.
- Dispositivos deslizantes (Sliders).

Varios de estos componentes están capacitados para comunicarse con el servidor usando AJAX. También contiene numerosas funcionalidades que permiten añadir interactividad a las páginas HTML, como:

- Cuadros de diálogo.

- Consejos rápidos (quicktips) para mostrar mensajes de validación e información sobre campos individuales.

2.4 Arquitectura

La Arquitectura de Software establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

2.4.1 Estilo híbrido basado en Capas y MVC

El estilo seleccionado para el desarrollo de la arquitectura, es un estilo en capas. Está compuesto básicamente por tres niveles o capas, este presenta algunas ventajas como son: (10)

- Desarrollos paralelos (en cada capa)
- Aplicaciones más robustas debido al encapsulamiento
- Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica)
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nuevas funcionalidades)
- Alta escalabilidad. La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.

Capa Datos: En esta capa se encuentra como servidor de base de datos PostgreSQL.

Capa Acceso a Datos: En esta capa está presente el framework Doctrine, como persistidor para la comunicación con el servidor de datos mediante el protocolo PDO.

Capa Funcionamiento: En esta capa se emplea el patrón Modelo Vista Controlador (MVC), este es uno de los patrones de diseño de arquitectura de software que más se utiliza en la construcción de aplicaciones web, el cual permite la separación del código entre cada una de las capas, ayudando tanto a desarrolladores como a diseñadores a cooperar y mantener el código fuente más fácilmente, además permite que el software sea fácil de modificar.

2.4.2 Estructura del patrón MVC (11)

El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

2.5 Estrategia de Integración

Entre los componentes se aplica la integración vertical que consiste en como fluyen los datos desde la vista hacia la capa de datos y viceversa, pasando por los diferentes elementos que componen la arquitectura.

Esta consta de cuatros nodos de integración, el que se encuentra entre la vista y el controlador, el que está entre la el controlador y el modelo, el que vincula el modelo con el marco de trabajo doctrine y el que se encuentra entre el doctrine y la base de datos. Hay que señalar que este cuarto y último nodo aunque está presente no es de nuestro interés describirlo puesto que su funcionamiento es totalmente transparente para el equipo de desarrollo.

Cada componente tiene su registro de los datos de los módulos en un fichero XML que será mapeado por el frameworks para el funcionamiento del mismo, dicho fichero tiene por nombre Inversión de control (IoC). La comunicación entre los diferentes módulos y componentes se realiza mediante llamadas a la inversión de control, este registra las funcionalidades que ofrecen los métodos de las clases modelos y entidades del sistema y especifica respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que otro módulo o componente lleve a cabo las acciones de control que se requieran en el

orden necesario.

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.

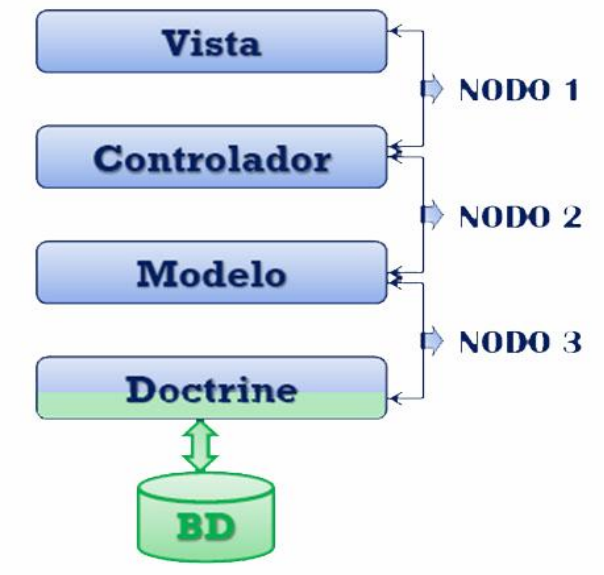

Figura # 1: Estrategia de Integración

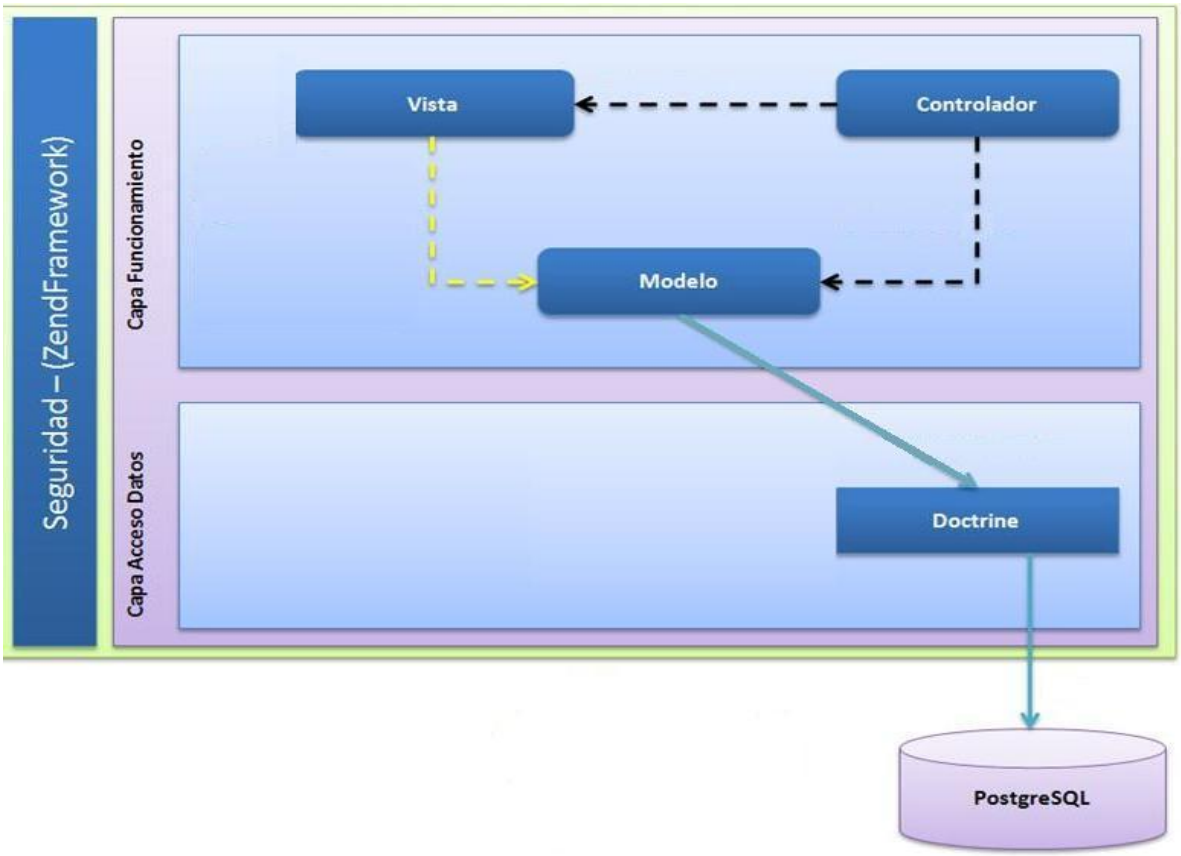

Figura # 2: Vista horizontal de la integración

2.6 Estándares de código

Un estándar de codificación comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Los estándares de codificación permiten una mejor integración entre las líneas de producción y establece pautas que conllevan a lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su capacidad de mantenimiento a lo largo del tiempo.

2.6.1 Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación **PascalCasing**. Con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: GestionarUsuario

2.6.1.1 Nomenclatura según el tipo de clases

Clases controladoras

Las clases controladoras después del nombre llevan la palabra:"Controller".

Ejemplo: GestionarUsuarioController

Clases de los modelos

- Business (Negocio)

  Las clases que se encuentran dentro de Business después del nombre llevan la palabra: "Model".

  Ejemplo: MonedaContModel
- Domain (Dominio)

  Las clases que se encuentran dentro de Domain el nombre que reciben es el de la tabla en la Base de Datos.

  Ejemplo: User
- Generated (Dominio bases)

  Las clases que se encuentran dentro de Generated el nombre comienza con la palabra: "Base" y seguido el nombre de la tabla en la Base de Datos.

  Ejemplo: BaseUser

2.6.2 Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación **CamelCasing**, y con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: insertarMoneda

En caso de ser una acción de la clase controladora se le pone el nombre y seguida la palabra:"Action"

Ejemplo: insertarMonedaAction

2.6.3 Nomenclatura de los comentarios

Los comentarios deben ser lo bastante claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando.

En caso de ser una función complicada se comentaría dentro de la misma para lograr una mejor comprensión del código.

Ejemplo:

```
public function eliminarsecuenciaAction ()
{
        // Es guardado en la variable $numero el número de la secuencia enviada del js
        $numero = $this->_request->getPost ('numero');
        // Devuelve 1 si la secuencia con ese número fue ejecutada, 0 si no fue ejecutada
        $ejecutada = DatEjecucionsecuencia::ejecutada ($numero);
        $ejecutada = $ejecutada [0] [cant];
        // Entra al if si al secuencia no fue ejecutada
        if ($ejecutada == 0)
        {
                // Es guardado en $id el id de la secuencia con ese número
                $id = ConfSecuenciasdistribucion::secuenciaid ($numero);
                // Se crea un objeto de tipo OpsecuenciatraspasoModel
                $opsec = new OpsecuenciatraspasoModel ( );
                // Es eliminada la secuencia con ese $id
                $eliminar = $opsec->eliminarsecuencia ($id);
                // Devuelve 1 si la secuencia no fue ejecutada y fue eliminada correctamente
                echo "{'resp': 1}";
        }
        else
        {
                //devuelve 0 si la secuencia fue ejecutada y no pudo ser eliminada
                echo "{'resp': 0}";
        }
}
```

Figura # 3: Ejemplo de Nomenclatura entre comentarios

2.7 Descripción de los algoritmos no triviales a implementar

Uno de las funciones fundamentales que ocurren dentro del proceso de Secuencia de Traspasos es Crear una Secuencia de Traspaso entre cuentas de gastos patrimoniales. Esta se crea por cinco criterios de distribución diferentes: Saldo, Volumen, Porciento, Subelementos y Bases Combinadas. El proceso realizado por los tres primeros criterios es común ya que todos los datos de la secuencia son similares excepto el tipo de criterio de distribución. El proceso al crear la secuencia por los dos últimos criterios se realiza igual que los primeros pero con la excepción de que si es por Subelementos se adiciona

además de los datos de la secuencia un único subelemento de gasto y en caso de ser por Bases Combinadas se adicionan tantos como el usuario desee.

2.7.1 Análisis de complejidad del algoritmo

Para conocer la complejidad del algoritmo es necesario calcular la complejidad ciclomática del mismo, para hacer dicho cálculo es necesario primero tener el código o el diseño del algoritmo, luego enmarcar cada instrucción del código con un número, que representa cada lugar del camino que puede seguir la secuencia del algoritmo, a continuación se representa el código con sus instrucciones enmarcadas.

```
public function insertarsecuenciaAction ()
{
$descripcion = $this->_request->getPost ('descripcion'); 1
$estado = $this->_request->getPost ('estado'); 1
$fechafin = $this->_request->getPost ('fechafin'); 1
$fechainicio = $this->_request->getPost ('fechainicio'); 1
$numero = $this->_request->getPost ('numero'); 1
$cbusqueda = $this->_request->getPost ('cbusqueda'); 1
$idcuenta = $this->_request->getPost ('cuenta'); 1
$idcentrocostopatrimonial = $this->_request->getPost ('centroCO'); 1
$idelementogastopatrimonial = $this->_request->getPost ('elemento'); 1
$arrayccdestino = json_decode (stripslashes ($this->_request->getPost ('centroCD'))); 1
$cdistribucion = $arrayccdestino [0] [3]; 1
$arraysecuencia = array ("cdistribucion" =>$cdistribucion, "descripcion" =>$descripcion, "estado"
=>$estado, "fechafin" =>$fechafin, "fechainicio" =>$fechainicio, "numero" =>$numero, "idcuenta"
=>$idcuenta, "idcentrocostopatrimonial" => $idcentrocostopatrimonial, "idelementogastopatrimonial"
=>$idelementogastopatrimonial, "idestructura" =>$this->idestructura, "cbusqueda" =>$cbusqueda);
1
$arrayccorigen = array ("idcuenta" =>$idcuenta, "idcentrocostopatrimonial" =>
$idcentrocostopatrimonial, "idelementogastopatrimonial" =>$idelementogastopatrimonial); 1
$idsecuencia = $this->model->insertarsecuencia ($arraysecuencia); 1
  if ($idsecuencia) 2
    {
      if ($this->model->insertarcentroorigen ($arrayccorigen)) 3
        {
        $tempval = true; 4
          foreach ($arrayccdestino as $v) 5
            {
            $centrodestino = new ConfCentrosdestino (); 6
            $centrodestino->idcuenta = $v [0]; 6
            $centrodestino->idcentrocostopatrimonial = $v [1]; 6
            $centrodestino->idelementogastopatrimonial = $v [2]; 6
            $centrodestino->idsecuenciasdistribucion = $idsecuencia; 6

                if  (¡ $this->model->insertarcentrodestino ($centrodestino)) 7
                 {
                 $tempval = false; 8
                 break; 8
                 }
                 if ($cdistribucion == 900000002) 9
```

```
                    {
                      $objsubelemento = new ConfElementos (); 10
                      $objsubelemento->idcuenta = $v [0]; 10
                      $objsubelemento->idcentrocostopatrimonial = $v [1]; 10
                        $objsubelemento->idelementogastopatrimonial =  $v [4] [0]; 10
                        $objsubelemento->idsecuenciasdistribucion = $idsecuencia; 10

                        if  (¡ $this->model->insertarsubelemento ($objsubelemento)) 11
                          {
                          $tempval = false; 12
                          break; 12
                          }
                    }

                    if ($cdistribucion == 900000003) 13
                      {
                          for ($i = 0; $i < count ($v [4]); $i ++) 14
                          {
                          $objsubelemento = new ConfElementos (); 15
                          $objsubelemento->idcuenta = $v [0]; 15
                          $objsubelemento->idcentrocostopatrimonial = $v [1]; 15
                          $objsubelemento->idelementogastopatrimonial = $v [4] [$i]; 15
                          $objsubelemento->idsecuenciasdistribucion = $idsecuencia; 15

                          if (¡ $this->model->insertarsubelemento ($objsubelemento)) 16
                             {
                                $tempval = false; 17
                                break; 17
                             }
                           }
                       }
                }
                if (! $tempval) 18
                {
                   echo '{succes: -1}'; 19
                 }
                  else 20
                  {
                     echo '{succes: 0}'; 21
                  }
          }          else 22
          {
            echo '{succes: 9}'; 23
          }
      }      else 24
     {
      echo '{succes: 10}'; 25
     }
} 26
```

Después de este paso, es necesario representar el grafo de flujo asociado, en el cual se representan distintos componentes como es el caso de:

Nodo: son los círculos representados en el grafo de flujo, el cual representa una o más

secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

Aristas: son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.

Regiones: son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

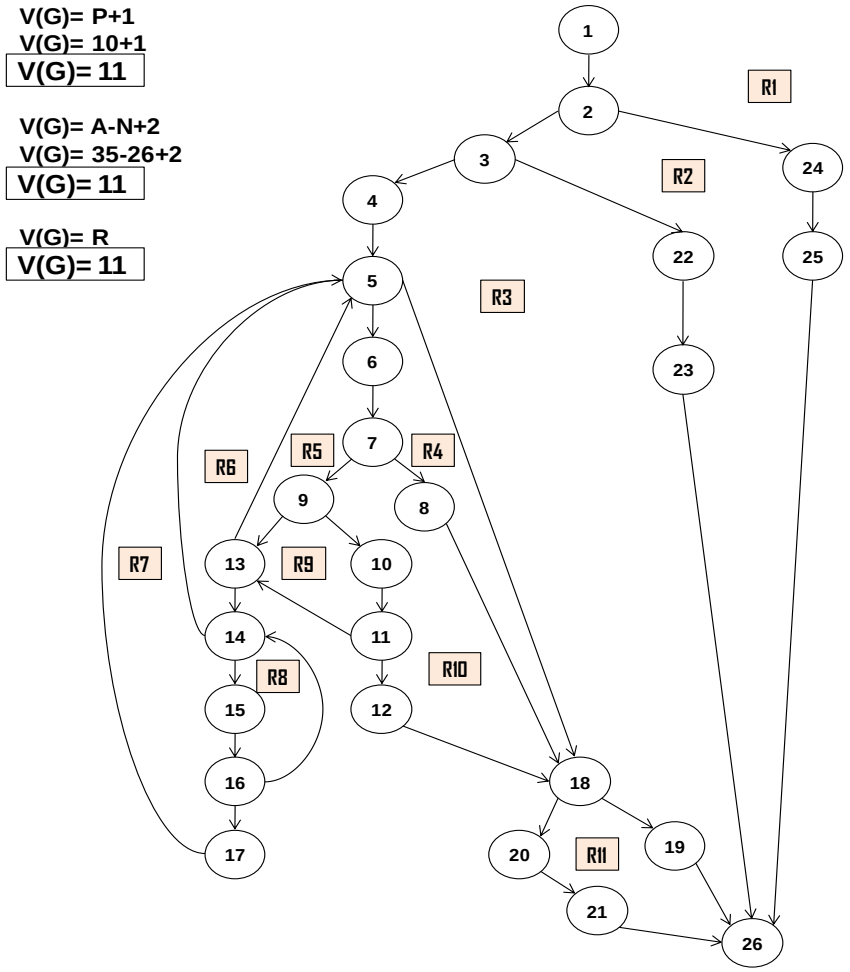

Figura # 4: Grafo de flujo asociado a función insertarsecuenciaAction

Seguidamente a la construcción del grafo de flujo se procede a efectuar el cálculo de la complejidad ciclomática del código, el cálculo es necesario efectuarlo mediante tres vías o fórmulas que, para concluir que el cálculo fue correcto es necesario que por las tres vías el resultado sea el mismo, las fórmulas para calcular son las siguientes:

V (G) = (A – N) + 2
Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.
V (G) = (35 – 26) + 2
V (G) = 11

V (G) = P + 1
Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).
V (G) = 10 +1
V (G) =11

V (G) = R
Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del cálculo.
V (G) = 11

Realizado el cálculo por las 3 vías necesarias se llega a la conclusión que el algoritmo presentado anteriormente tiene un complejidad ciclomática de 11 dando visión de que existen a lo sumo once caminos lógicos por donde recorrer el algoritmo.

2.8 Estructura de datos a usar para implementar el algoritmo

Las estructuras de datos, brindan la posibilidad de almacenar organizadamente los elementos según el lenguaje de programación y las necesidades del programador. Hay gran variedad de ellas, para escoger la que se utilizará, se debe tener en cuenta la continuidad con la que se usarán los datos que en ella se guarden, pues la simplicidad o la complejidad que las mismas poseen, pueden hacer la ejecución del sistema mucho más lento o de mayor rendimiento.

En el caso de “Costos y Procesos”, la estructura de datos a utilizar se denomina Arreglos o Arrays, los mismos son variables complejas, guardan gran cantidad de valores dentro

de un solo nombre de variable, la capacidad del mismo puede estar delimitada por el programador en caso de necesidad. Su información puede ser cambiada y manejada de manera fácil. La ventaja de usar los arreglos de PHP es que dicho lenguaje tiene una gran variedad de funciones para el uso de los mismos, entre ellas están las funciones de ordenamiento. En PHP existen dos tipos de arrays, los de índices asociativos y los de índices numéricos.

2.9 Descripción de las nuevas clases u operaciones necesarias

2.9.1 Clases Controladoras

Las clases de control coordinan el trabajo de uno o unos pocos casos de uso, coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso, por lo que definen el flujo de control y las transacciones dentro de un caso de uso delegando el trabajo a otros objetos.

Nombre: OpsecuenciatraspasoController	
Tipo de clase: Controladora	
Atributo	**Tipo**
idestructura	int
idperiodo	int
model	object
service	object
comprobante	object
idsubsistema	int
fechadelsubsistema	date
contabilizartraspaso	object
Para cada responsabilidad:	
Nombre:	**Descripción:**
cargarsecuenciaAction()	Carga todas las secuencias ejecutadas como no ejecutadas
obtenerDatosCCDModificarAction()	Obtiene los centros destinos para ser modificados
mostrardetallesecuenciaAction()	Muestra en la interfaz base, en la Crear y en la Modificar los centros destinos
mostrarcentrocostoAction()	Muestra los datos del centro de costo
verelementosAction()	Carga los elementos asociados a un centro de costo seleccionado

getSubelementosSecuenciaAction()	Devuelve los subelementos de una secuencia
verelementosseleccionadosAction()	Carga los elementos asociados a un centro de costo seleccionado en el modificar
cargarcriterioAction()	Carga el criterio de distribución
cargarcriteriobusquedaAction()	Carga el criterio de búsqueda
cargarcentrosdestinosAction()	Carga los centros destinos
insertarsecuenciaAction()	Inserta una secuencia por cualquiera de los criterios de distribución
modificarsecuenciaAction()	Modifica la secuencia por cualquiera de los criterios
eliminarsecuenciaAction()	Elimina la secuencia siempre que no esté ejecutada
getCodigoCriterioAction()	Devuelve el código del criterio
ejecutarsecuenciaAction()	Ejecuta la secuencia
insertardatejecSecuenciaAction($datocco, $datosccd)	Inserta en las tablas DatEjecucionsecuencia y ConfBaseejecucion los datos de la ejecución
obtenersaldoAction($datosccd)	Se encarga de obtener los saldos de las cuentas, centros y elementos
obtenersaldosubelementoAction($datosccd)	Obtiene el saldo del subelemento
calcularingresoAction($datosccd, $saldocuentaorigen)	Se encarga de calcular el ingreso que se debe traspasar al centro destino
calcularingresosubelementoAction($datosccd, $saldocuentaorigen)	Calcula el ingreso del subelemento
obtenerCuentasPatrimonialesAction()	Devuelve las cuentas de gastos patrimoniales
mostrarcentrosorigenAction()	Muestra todos los datos de los centros origen
obtnumsecuenciaAction()	Es generado un único numero para cada secuencia
obtenerfechactualAction()	Obtiene la fecha del servidor
getVerEjecucionAction()	Obtiene todos los datos de la secuencia ejecutada
getDatosComprobanteAction()	Devuelve el id de la ejecución de la secuencia
getLlenGridComprobAction()	Obtiene los datos necesarios de los centros destinos para ver su ejecución
getsaldo($cuenta, $centro)	Devuelve el saldo
getbases($arraycuentacentro)	Devuelve las bases
suma($array)	Suma las bases
porciento($valor, $total)	Calcula el porciento que representa valor de total
importe($porciento, $importe)	Devuelve el resultado del importe multiplicado

	por le porciento
getporcientoAction($array)	Calcula el porciento
cargarccdejecAction()	Devuelve el id del centro de costo, el di de la secuencia, id del elemento, id de la cuenta y la descripción del criterio
cargarcumAction()	Obtiene el id y la abreviatura de la unidad de medida de volumen
verejecutadasAction()	Devuelve todos los id de secuencia que estén ejecutadas

Tabla # 1: Clase controladora “OpsecuenciatraspasoController”

2.9.2 Clases Auxiliares

Son las que guardan poca o ninguna información del estado por sí misma, pero asisten en la ejecución de tareas complejas.

Nombre: OpsecuenciatraspasoModel	
Tipo de clase: Auxiliar	
Atributo	**Tipo**
idsecuencias	int
Para cada responsabilidad:	
Nombre:	**Descripción:**
insertarsecuencia($arraysecuencia)	Inserta todos los datos de la secuencia en la tabla ConfSecuenciasdistribucion
insertarcentroorigen($arrayccorigen)	Inserta todos los datos del centro origen en la tabla ConfCentrosorigen
insertarcentrodestino($objccdestino)	Inserta todos los datos del centro destino en la tabla ConfCentrosdestino
insertarconfbase($objconfbase)	Inserta todos los datos del elemento en la tabla ConfElementos
insertarsubelemento($objsubelemento)	Inserta el subelemento
insertarunidad($objunidad)	Inserta la unidad
eliminarsecuencia($id)	Elimina todos los datos en todas las tablas que relaciona la secuencia
insertardatejecutarsecuencia($idsecuenciasdistribucion, $fechasistema,$iddocprimario)	Inserta todos los datos de la ejecución de la secuencia en la tabla DatEjecucionsecuencia
insertarconfbaseejecucion($idejecucionsecuencia, $idsecuenciasdistribucion, $idcuenta, $idcentrocostopatrimonial,$importemonedabase, $importemonedabasedespues, $idelementogastopatrimonial,$tipocentro)	Inserta todos los datos de la ejecución de la secuencia en la tabla ConfBaseejecucion

modificarsecuencia($arraysecuencia)	Modifica los datos cambiados de la secuencia en la tabla ConfSecuenciasdistribucion
modificarcentroorigen($arrayccorigen)	Modifica los datos cambiados del la secuencia ConfCentrosorigen
modificarcentrodestino($centrodestino)	Modifica los datos cambiados del la secuencia ConfCentrosdestino
eliminarcentrodestino($centrodestino)	Elimina los datos de la tabla ConfCentrosdestino
eliminarcentrodestino_saldo_volumen_porciento($idsecuencia)	Elimina el centro destino dado el id de la secuencia de la tabla ConfCentrosdestino
modificarsubelemento($objsubelemento)	Modifica los datos en la tabla ConfElementos
eliminarsubelemento($idsecuencia)	Elimina los datos dado el id secuencia de la tabla ConfElementos

Tabla # 2: Clase Auxiliar “OpsecuenciatraspasoModel”

Nombre: ContabilizarSecuenciaTraspaso	
Tipo de clase: Auxiliar	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
Contabilizar($idejecucionsecuencia, $idestructura,$idsubsistema,$idperiodo)	Crea el comprobante
ConformarPases($idejecucionsecuencia,$idestructura)	Crea los anexos del comprobante
getPases($resultpases)	Crea los pases de cada anexo del comprobante

Tabla # 3: Clase Auxiliar “ContabilizarSecuenciaTraspaso”

2.9.3 Clases Entidad

Estas clases modelan información que poseen una larga vida y que a menudo son conceptos y sucesos que ocurren en el mundo real. La fuente principal de obtención son las clases entidades del negocio y el glosario de términos que se ha ido elaborando. Se encargan de modelar la información del sistema y el comportamiento asociado a una información.

Nombre: ConfBaseejecucion	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	

Nombre:	**Descripción:**
getpases($idejecucionsecuencia)	Devuelve los pases
getanexosalpases($idejecucionsecuencia,$idcuenta)	Devuelve el anexo de los pases

Tabla # 4: Clase Entidad “ConfBaseejecucion”

Nombre: ConfCentrosdestino	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
devolvercentrodestino($idsecuencia)	Devuelve los centros destinos de la tabla ConfCentrosdestino dado el id de la secuencia.
Obtenercentrosdestino($idestructura)	Devuelve el idcuenta, idcentrocosto e idelementogasto de la tabla ConfSecuenciasdistribucion dado el id de la estructura.

Tabla # 5: Clase Entidad “ConfCentrosdestino”

Nombre: ConfCentrosorigen	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
centrosorigen($idsecuencia)	Devuelve los centros origen de la tabla ConfCentrosorigen dado el id de la secuencia.
Obtenercentrosorigen($idestructura)	Devuelve el idcuenta, idcentrocosto e idelementogasto de la tabla ConfSecuenciasdistribucion dado el id de la estructura.

Tabla # 6: Clase Entidad “ConfCentrosorigen”

Nombre: ConfCriteriobusqueda	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
cargarcriteriobusqueda()	Devuelve el id y la descripción del criterio de la tabla ConfCriteriobusqueda.
getcodigocriterio($criteriobusqueda)	Devuelve el código del criterio dado el id del criterio.

Tabla # 7: Clase Entidad “ConfCriteriobusqueda”

Nombre: ConfCriteriodistribucion
Tipo de clase: Entidad

Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
cargarcriterio()	Devuelve el id y la descripción de la tabla ConfCriteriodistribucion
codigocriterio($idcriterio)	Devuelve todos los elementos de la tabla ConfCriteriodistribucion dado el id del criterio.

Tabla # 8: Clase Entidad “ConfCriteriodistribucion”

Nombre: ConfElementos	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
devolverelementosmarcados($idsecuencia, $idcentro,$idcuenta)	Devuelve todos los elementos de la tabla ConfElementos dado el id de la secuencia, del centro y de la cuenta.
getSubelementosSecuenciaMod($idsecuencia)	Devuelve los elementos de la tabla ConfElementos dado el id de la secuencia.

Tabla # 9: Clase Entidad “ConfElementos”

Nombre: ConfSecuenciasdistribucion	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
cargartraspasos($idestructura,$start,$limit)	Devuelve todos los datos de las secuencias ejecutadas y no ejecutadas.
getcountsecuencitraspaso($idestructura)	Devuelve la cantidad de secuencias dado el id de la estructura.
datosconfcentroorigen($numero)	Devuelve los datos del centro origen dado el número de la secuencia.
datosconfcentrodestino($numero)	Devuelve los datos del centro destino dado el número de la secuencia.
datosconfcentrodestinosubelemento($numero)	Devuelve los subelementos de los centros destinos dado el número de la secuencia.
secuenciaactiva($numerosecuencia, $fechainicio,$fechafin)	Devuelve 1 si la secuencia esta activa en el período y 0 en caso contrario.
getVerEjecucion ($id)	Dado el idejecucionsecuencia devuelve los datos de las tablas ConfSecuenciasdistribucion, DatEjecucionsecuencia y ConfCentrosorigen.
secuenciaid($numero)	Dado el número de la secuencia devuelve el id de la secuencia.
centrodestino($idsecuenciadistri)	Dado el id de la secuencia devuelve los datos de las tablas ConfSecuenciasdistribucion, ConfCriteriodistribucion y ConfCentrosdestino.

datosconfbaseejec($idsecuencia)	Dado el id de la secuencia devuelve la descripción del criterio y la cantidad de las mismas.
ejecutadas()	Devuelve los id de las secuencias que estén ejecutadas.

Tabla # 10: Clase Entidad “ConfSecuenciasdistribucion”

Nombre: ConfUnidades	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**

Tabla # 11: Clase Entidad “ConfUnidades”

Nombre: DatEjecucionsecuencia	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
getDatosComprobante()	Devuelve el idejecucionsecuencia de la tabla DatEjecucionsecuencia.
getLlenGridComprob($id)	Dado el idejecucionsecuencia devuelve datos de las tablas ConfSecuenciasdistribucion, DatEjecucionsecuencia y ConfBaseejecucion.
ejecutada($numero)	Devuelve la cantidad de secuencias ejecutadas.
iddatejecucion($id)	Dado el id de la secuencia devuelve el idejecucionsecuencia de la tabla DatEjecucionsecuencia.
getdatcobtabilizar($idejecucionsecuencia)	Dado el idejecucionsecuencia devuelve datos de las tablas DatEjecucionsecuencia y ConfBaseejecucion.
getpases($idejecucionsecuencia)	Devuelve los pases
getultimaejecucion()	Devuelve el máximo idejecucionsecuencia que exista en la tabla DatEjecucionsecuencia, es decir la última ejecución.

Tabla # 12: Clase Entidad “DatEjecucionsecuencia”

Nombre: BaseConfBaseejecucion	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
setTableDefinition()	Adiciona las columnas y la tabla a la clase

Tabla # 13: Clase Entidad “BaseConfBaseejecucion”

Nombre: BaseConfCentrosdestino	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
setTableDefinition()	Adiciona las columnas y la tabla a la clase

Tabla # 14: Clase Entidad “BaseConfCentrosdestino”

Nombre: BaseConfCentrosorigen	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
setTableDefinition()	Adiciona las columnas y la tabla a la clase

Tabla # 15: Clase Entidad “BaseConfCentrosorigen”

Nombre: BaseConfCriteriobusqueda	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
setTableDefinition()	Adiciona las columnas y la tabla a la clase

Tabla # 16: Clase Entidad “BaseConfCriteriobusqueda”

Nombre: BaseConfCriteriodistribucion	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
setTableDefinition()	Adiciona las columnas y la tabla a la clase

Tabla # 17: Clase Entidad “BaseConfCriteriodistribucion”

Nombre: BaseConfElementos	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
setTableDefinition()	Adiciona las columnas y la tabla a la clase

Tabla # 18: Clase Entidad “BaseConfElementos”

Nombre: BaseConfSecuenciasdistribucion	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
setTableDefinition()	Adiciona las columnas y la tabla a la clase

Tabla # 19: Clase Entidad “BaseConfSecuenciasdistribucion”

Nombre: BaseConfUnidades	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
setTableDefinition()	Adiciona las columnas y la tabla a la clase

Tabla # 20: Clase Entidad “BaseConfUnidades”

Nombre: BaseDatEjecucionsecuencia	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
setTableDefinition()	Adiciona las columnas y la tabla a la clase

Tabla # 21: Clase Entidad “BaseDatEjecucionsecuencia”

En este capítulo se realiza una valoración crítica del diseño propuesto por los analistas, teniendo en cuenta la correcta definición de las clases fundamentales del sistema a construir, se tuvo en cuenta la utilización de patrones de diseño y la reutilización de varios frameworks para optimizar el trabajo con las clases.

Se describieron todas las clases para un mejor entendimiento de estas con el objetivo de mostrar una comprensión de todo lo relacionado con los requisitos funcionales para posibilitar futuros mantenimientos de la aplicación.

Capítulo 3: Validación de la solución propuesta

3.1 Introducción

Durante el proceso de desarrollo de un software, los errores pueden comenzar a aparecer incluso desde el mismo momento en que fueron definidos los objetivos debido a que pueden haber sido especificados de forma errónea e imperfecta; lo mismo puede suceder en los posteriores pasos del diseño y desarrollo. A raíz de la incapacidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software debe ser acompañado de una actividad que garantice su calidad.

La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación, esta no es un proceso que se realiza una vez desarrollado el software, sino que debe efectuarse en cada una de las etapas de desarrollo.

La creciente inclusión del software, como un elemento más de muchos sistemas y la importancia de los costos asociados a un fallo del mismo, han motivado la creación de pruebas más minuciosas y bien planificadas.

En el presente capitulo se desarrollarán diferentes tipos de prueba, como son las pruebas de caja blanca las cuales serán aplicadas en varios métodos y las pruebas de caja negra, además se aplicarán métricas para validar el diseño de las clases basadas en atributos de calidad.

3.2 Descripción de las pruebas

Con diferentes objetivos, en variados escenarios o niveles de trabajos se enfocan las pruebas en las cuales se distinguen los diferentes niveles que a continuación se mencionan:

- Prueba de desarrollador
- Prueba independiente
- Prueba de Unidad
- Prueba de Integración
- Prueba de sistema
- Prueba de aceptación

La **Prueba de Unidad** se enfoca a los elementos testeables más pequeño del software, estos pueden ser clases, métodos, propiedades, componentes, etc. Es aplicable a

componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca.

Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del componente. Si los datos no entran correctamente, todas las demás pruebas no tienen sentido. El diseño de casos de prueba de una unidad comienza una vez que se ha desarrollado, revisado y verificado en su sintaxis el código a nivel fuente.

A este nivel uno de los métodos de prueba más importantes es el método de caja blanca. Se denominan pruebas de caja blanca a aquellas que examinan el código al más bajo nivel, es decir, aseguran que las operaciones internas se ajustan a las especificaciones.

Una de las técnicas para ejecutar las pruebas de caja blanca es la del “Camino básico”, el resultado de esta técnica es una medición cuantitativa de la complejidad de un programa. El valor calculado como complejidad ciclomática, define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Esta técnica fue usada anteriormente en este trabajo en el capítulo 2, epígrafe 2.7.1 “Análisis de complejidad del algoritmo”.

Otro de los métodos de prueba es el de caja negra. Esta prueba permite obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ella se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

3.3 Aplicación de pruebas de caja blanca

Para realizar el test es necesario realizar primeramente los procesos descritos en el capítulo 2, epígrafe 2.7.1 “Análisis de complejidad del algoritmo” para calcular los valores de la complejidad ciclomática de los procedimientos a los cuales se le va a aplicar la prueba. A continuación se enumeran las sentencias de código de los procedimientos.

Procedimiento #1:

```
public function eliminarsecuenciaAction ()
  {
    $numero = $this->_request->getPost (‘numero’); 1
    $ejecutada = DatEjecucionsecuencia::ejecutada ($numero); 1
    $ejecutada = $ejecutada [0] [cant]; 1
```

```
 if ($ejecutada == 0) 2
   {
     $id = ConfSecuenciasdistribucion::secuenciaid ($numero); 3
     $opsec = new OpsecuenciatraspasoModel ( ); 3
     $eliminar = $opsec->eliminarsecuencia ($id); 3
     echo "{'resp': 1}"; 4
   }
   else 5
   {
     echo "{'resp': 0}"; 6
   }
} 7
```

Procedimiento #2:

```
public function verelementosAction ()
  {
   $datos = $this->pIntegrator->nom_conf->obtenerelementos ($this->_request->getPost
  (‘idcentro’), $this->_request->getPost (‘idcuenta’)); 1
   for ($i = 0; $i < count ( $datos ); $i ++)  2
     {
       $datoss [$i] ['idelemento'] = $datos [$i] ->idelementogastopatrimonial; 3
       $datoss [$i] ['descripcion'] = $datos [$i] ->descripcionelemento; 3
     }
     echo (json_encode ($datoss)); 4
  } 5
```

Seguidamente se construye el grafo de flujo asociado a los códigos anteriores.

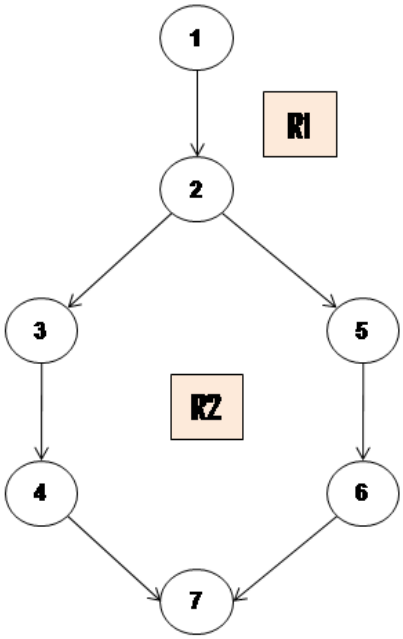

Figura # 5: Grafo de flujo asociado al Procedimiento #1 Eliminar secuencia de traspaso

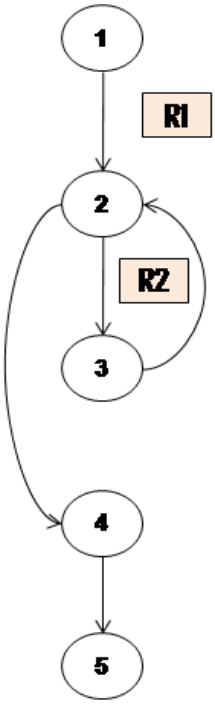

Figura # 6: Grafo de flujo asociado al Procedimiento #2 Ver elementos

Luego de haber construido el grafo, se realiza el cálculo de la complejidad ciclomática, mediante las tres fórmulas descritas en el capítulo anterior, epígrafe 2.7.1 “Análisis de complejidad del algoritmo”, las cuales tienen que arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

Fórmulas para calcular complejidad ciclomática.

1 – V (G) = (A – N) + 2.

2 – V (G) = P + 1.

3 – V (G) = R.

Aplicando estas fórmulas al grafo de flujo de la figura del **Procedimiento #1** se obtienen los siguientes resultados:

Calculando mediante la fórmula 1:

V (G) = (7 -7) + 2

V (G) = 2.

Calculando mediante la fórmula 2:

V (G) = 1 + 1

V (G) = 2.

Calculando mediante la fórmula 3:

V (G) = 2.

Aplicando estas fórmulas al grafo de flujo de la figura del **Procedimiento #2** se obtienen los siguientes resultados:

Calculando mediante la fórmula 1:

V (G) = (5 -5) + 2

V (G) = 2.

Calculando mediante la fórmula 2:

V (G) = 1 + 1

V (G) = 2.

Calculando mediante la fórmula 3:

V (G) = 2.

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código de ambos procedimientos es 2, lo cual nos da un límite superior para el número de caminos independientes que componen el conjunto básico y, consecuentemente, un valor limite superior para el número de

pruebas que se deben diseñar y ejecutar para garantizar que se cubren todas las sentencias de los procedimientos.
Es necesario representar los caminos básicos por los que puede recorrer el flujo.

Procedimiento #1
Camino básico #1:
1 – 2 – 3 – 4 – 7.
Camino básico #2:
1 – 2 – 5 – 6 – 7.
Procedimiento #2
Camino básico #1:
1 – 2 – 4 – 5.
Camino básico #2:
1 – 2 – 3 – 2 – 4 – 5.

Después de haber extraído los caminos básicos de ambos flujos, se procede a ejecutar los casos de pruebas para cada procedimiento, se debe realizar al menos un caso de prueba por cada camino básico.

Para realizar los casos de pruebas es necesario cumplir con las siguientes exigencias:
Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.

Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.

Entrada: Se muestran los parámetros que entran al procedimiento.

Resultados Esperados: Se expone resultado que se espera que devuelva el procedimiento.

Procedimiento #1
Caso de prueba para el camino básico **# 1**:
Descripción: En este caso no se hace la entrada de datos, sino que se elimina la

configuración de la secuencia de traspaso siempre y cuando no se haya ejecutado la misma.

Condición de ejecución: Que la secuencia de traspaso no haya sido ejecutada.

Entrada: No existe Entrada.

Resultados esperados: Se espera que pueda ser eliminada la secuencia de traspaso y sea lanzado el siguiente cartel informativo: 'Se realizó la operación correctamente.'.

Fue eliminada satisfactoriamente la secuencia y es mostrado el cartel informativo 'Se realizó la operación correctamente.'.

Caso de prueba para el camino básico # **2**:

Descripción: En este caso no se hace la entrada de datos, sino que se elimina la configuración de la secuencia de traspaso siempre y cuando no se haya ejecutado la misma.

Condición de ejecución: Que la secuencia de traspaso no haya sido ejecutada.

Entrada: No existe Entrada.

Resultados esperados: Se espera que la secuencia no pueda ser eliminada ya que fue ejecutada anteriormente y que sea lanzado el siguiente cartel informativo: 'Error, esta secuencia ya fue ejecutada.'.

No se pudo eliminar la secuencia de traspaso puesto que ya había sido ejecutada con anterioridad y se mostró el siguiente cartel informativo: ‘Error, esta secuencia ya fue ejecutada.'.

Procedimiento #2

Caso de prueba para el camino básico # **1**:

Descripción: En este caso no se hace la entrada de datos, sino que se cargan los elementos asociados a un Centro de Costo determinado.

Condición de ejecución: Que el Centro de Costo tenga elementos asociados.

Entrada: Se entra el Centro de Costo.

Resultados esperados: Se espera que no sea cargado ningún elemento.

No se cargó ningún elemento puesto que el Centro de Costo no tenía elementos asignados.

Caso de prueba para el camino básico # **2**:

Descripción: En este caso no se hace la entrada de datos, sino que se cargan los

elementos asociados a un Centro de Costo determinado.

Condición de ejecución: Que el Centro de Costo tenga elementos asociados.

Entrada: Se entra el Centro de Costo.

Resultados esperados: Se espera que sean cargados los elementos asociados al Centro de Costo.

Se cargaron los elementos asociados al Centro de Costo satisfactoriamente.

Luego de aplicar los distintos casos de pruebas, se pudo comprobar que el flujo de trabajo de cada procedimiento está correcto ya que cumplen con las condiciones necesarias que se habían planteado.

3.4 Aplicación de pruebas de caja negra

Para realizar este tipo de prueba se usará el requisito funcional “Crear Secuencias de Traspasos”. El objetivo de este requisito como lo dice su nombre es crear o insertar la configuración de una secuencia de traspaso ya sea por cualquiera de los criterios de distribución posibles los cuales son: Saldo, Volumen, Porciento, Subelementos y Bases Combinadas.

Requisitos para pruebas de Caja Negra

Nombre del Requisito	Descripción General	Escenarios de Pruebas	Flujo del Escenario
Crear Secuencias de Traspasos	En la interfaz Secuencias de Traspasos, se presiona el botón *Crear* y se carga la página *Crear/Modificar Secuencia de traspaso.* Los campos *No. de Traspaso* y *Fecha Inicio* se llena de manera automática, en el caso de la fecha se toma la del subsistema, el usuario es responsable de llenar los campos *Descripción* y *Fecha Fin*, así como escoger la *Cuenta de Gasto Patrimonial*, el *Estado* puede ser Activo o Inactivo en dependencia de la fecha de inicio y fin, si se vence el rango de fecha el sistema pone de manera automática el *Estado* en Inactivo. Se muestran todos los *Centros de Costo* asociados a esa cuenta con su código y descripción. Se selecciona de los Centros de Costo asociados a la cuenta de gasto el Centro de Costo que se va a definir como origen y se selecciona el elemento de traspaso y el criterio de búsqueda (existen dos criterios de búsquedas: por saldo de período y por saldo acumulado). Se presiona el botón Definir CCO. Para realizar la configuración del traspaso	**EP- 1.1:** Definir correctamente la secuencia de traspaso.	Se presiona el botón *Crear*. Se selecciona la Cuenta de Gasto. Se selecciona la Fecha de fin. Se introduce la Descripción. Se marca la opción *Estado* en caso de querer crearla Activa. Se selecciona de los Centros de Costo que están asociados a la cuenta el que se quiere definir como Centro de Costo Origen. Se selecciona el Elemento de Traspaso. Se selecciona el Criterio de Búsqueda. Se presiona el botón *Definir CCO.* Se selecciona el(os) Centro(s) Destino(s). Se selecciona el Criterio de Distribución. Se presiona el botón *Definir*. Se muestra un mensaje validando que la operación se realizó correctamente. Se presiona el botón *Aceptar.*
		EP - 1.2: Presionar el botón *Cancelar* una vez seleccionados los datos necesarios para definir el Centro de Costo Origen.	Se presiona el botón *Crear.* Se selecciona la Cuenta de Gasto. Se selecciona la Fecha de fin. Se introduce la Descripción. Se marca la opción *Estado* en caso de querer crearla Activa. Se selecciona de los Centros de Costo que están asociados a la cuenta el que se quiere definir

	para los centros destinos se selecciona el/los centro(s) de costo(s) que se va(n) a definir como destino(s), se presiona el botón Definir y se muestran en la tabla la Cuenta de Gasto, el(os) Centro(s) de Costo(s) Destino(s), el Criterio de Distribución que puede ser por (Saldo, por Subelemento, Por Bases Combinadas, por Porciento y por Volumen) y el Elemento de Traspaso. Si escojo como Criterio de Distribución Bases Combinadas se me muestra una ventana con todos los elementos que están asociados a ese(os) Centro(s) de Costo Destino(s) para escoger los elementos que se quieren traspasar, si se escoge como criterio de distribución Subelemento entonces se muestra la misma ventana pero te da solo la oportunidad de escoger un solo elemento a traspasar y cuando el criterio de distribución es por Volumen, por Porciento o por Saldo no pasa nada, es decir no se muestra ninguna ventana en la configuración. El valor del porciento y el del volumen se especifica en la ejecución. Terminado dicho procedimiento, se presiona el botón *Definir* y se crea la secuencia de traspaso.		como Centro de Costo Origen. Se selecciona el Elemento de Traspaso. Se selecciona el Criterio de Búsqueda. Se presiona el botón *Cancelar***.** Se muestra un mensaje verificando que se quiere realizar la operación. Se presiona el botón *Aceptar.* Se limpian automáticamente los campos seleccionados.
		EP - 1.3: Configurar una secuencia de traspaso con datos incompletos.	Se presiona el botón *Crear*. Se selecciona la Cuenta de Gasto. Se selecciona la Fecha de fin. Se introduce la Descripción. Se marca la opción *Estado* en caso de querer crearla Activa. Se selecciona de los Centros de Costo que están asociados a la cuenta el que se quiere definir como Centro de Costo Origen. Se selecciona el Elemento de Traspaso. Se selecciona el Criterio de Búsqueda. Se presiona el botón *Definir CCO.* Se selecciona el(os) Centro(s) Destino(s). Se presiona el botón *Definir.* Se muestra un mensaje indicando que faltan datos obligatorios para la configuración de la secuencia. Se presiona el botón *Aceptar.*
		EP - 1.4: Presionar el botón *Quitar* una	Se presiona el botón *Crear*. Se selecciona la Cuenta de Gasto.

		vez seleccionados el(os) Centro(s) de Costo Destino(s).	Se selecciona la Fecha de fin. Se introduce la Descripción. Se marca la opción *Estado* en caso de querer crearla Activa. Se selecciona de los Centros de Costo que están asociados a la cuenta el que se quiere definir como Centro de Costo Origen. Se selecciona el Elemento de Traspaso. Se selecciona el Criterio de Búsqueda. Se presiona el botón *Definir CCO*. Se selecciona el(os) Centro(s) Destino(s). Se selecciona el Criterio de Distribución. Se presiona el botón *Quitar*. Se quitan el(os) Centro(s) seleccionados de la tabla.
		EP - 1.5: Presionar el botón *Salir* cuando se está creando la secuencia.	Se presiona el botón *Crear.* Se presiona el botón *Salir.* Se muestra un mensaje verificando que se quiera realizar la operación. Se presiona el botón *Aceptar.* Se cierra la página automáticamente.
		EP - 1.6: Configurar una secuencia de traspaso sin haber seleccionado una Fecha fin.	Se presiona el botón *Crear*. Se selecciona la Cuenta de Gasto. Se introduce la Descripción. Se marca la opción *Estado* en caso de querer crearla Activa. Se selecciona de los Centros de Costo que están asociados a la cuenta el que se quiere definir como Centro de Costo Origen.

			Se selecciona el Elemento de Traspaso. Se selecciona el Criterio de Búsqueda. Se presiona el botón *Definir CCO*. Se selecciona el(os) Centro(s) Destino(s). Se presiona el botón *Definir.* Se muestra un mensaje indicando que falta un dato obligatorio para la configuración de la secuencia. Se presiona el botón *Aceptar.*
		EP - 1.7: Configurar una secuencia de traspaso sin haber seleccionado una Cuenta de Gasto.	Se presiona el botón *Crear.* Se selecciona una Fecha fin. Se introduce la Descripción. Se marca la opción *Estado* en caso de querer crearla Activa. Se selecciona de los Centros de Costo que están asociados a la cuenta el que se quiere definir como Centro de Costo Origen. Se selecciona el Elemento de Traspaso. Se selecciona el Criterio de Búsqueda. Se presiona el botón *Definir CCO*. Se selecciona el(os) Centro(s) Destino(s). Se presiona el botón *Definir*. Se muestra un mensaje indicando que falta un dato obligatorio para la configuración de la secuencia. Se presiona el botón *Aceptar*.
		EP - 1.8: Configurar una secuencia de traspaso sin haber seleccionado un	Se presiona el botón *Crear.* Se selecciona una Fecha fin. Se introduce la Descripción. Se marca la opción *Estado* en caso

		Centro de Costo Origen.	de querer crearla Activa. Se selecciona la Cuenta de Gasto. Se selecciona el Elemento de Traspaso. Se selecciona el Criterio de Búsqueda. Se presiona el botón *Definir CCO*. Se selecciona el(os) Centro(s) Destino(s). Se presiona el botón *Definir*. Se muestra un mensaje indicando que falta un dato obligatorio para la configuración de la secuencia. Se presiona el botón *Aceptar*.
		EP - 1.9: Configurar una secuencia de traspaso sin haber seleccionado un Elemento de Traspaso.	Se introduce la Descripción. Se marca la opción *Estado* en caso de querer crearla Activa. Se selecciona de los Centros de Costo que están asociados a la cuenta el que se quiere definir como Centro de Costo Origen. Se selecciona la Cuenta de Gasto. Se selecciona el Criterio de Búsqueda. Se presiona el botón *Definir CCO.* Se selecciona el(os) Centro(s) Destino(s). Se presiona el botón *Definir*. Se muestra un mensaje indicando que falta un dato obligatorio para la configuración de la secuencia. Se presiona el botón *Aceptar.*
		EP - 1.10: Configurar una secuencia de traspaso sin haber seleccionado un	Se presiona el botón *Crear*. Se selecciona una Fecha fin. Se introduce la Descripción. Se marca la opción *Estado* en caso de querer crearla Activa.

		Criterio de Búsqueda.	Se selecciona de los Centros de Costo que están asociados a la cuenta el que se quiere definir como Centro de Costo Origen. Se selecciona el Elemento de Traspaso. Se selecciona la Cuenta de Gasto. Se presiona el botón *Definir CCO*. Se selecciona el(os) Centro(s) Destino(s). Se presiona el botón *Definir.* Se muestra un mensaje indicando que falta un dato obligatorio para la configuración de la secuencia. Se presiona el botón *Aceptar.*
		EP - 1.11: Configurar una secuencia de traspaso sin haber introducido su descripción.	Se presiona el botón *Crear.* Se selecciona una Fecha fin. Se marca la opción *Estado* en caso de querer crearla Activa. Se selecciona de los Centros de Costo que están asociados a la cuenta el que se quiere definir como Centro de Costo Origen. Se selecciona el Elemento de Traspaso. Se selecciona la Cuenta de Gasto. Se selecciona el Criterio de Búsqueda. Se presiona el botón *Definir CCO*. Se selecciona el(os) Centro(s) Destino(s). Se presiona el botón *Definir.* Se muestra un mensaje indicando que falta un dato obligatorio para la configuración de la secuencia. Se presiona el botón *Aceptar.*

Tabla # 22: Requisito Crear secuencia de traspaso para prueba de caja negra

Descripción de variable

No	Nombre de Campo	Clasificación	Puede ser Nulo	Descripción
1	Cuenta	Combo Box	NO	Aparecen todas las Cuentas de Gastos que tienen Centros de Costos asociados.
2	Estado	Check Box	SI	Se selecciona si se va a crear la secuencia Activa.
3	Fecha fin	Data Time	NO	Se selecciona la fecha de fin de la secuencia.
4	Descripción	Campo de texto	NO	Se puede introducir descripción (combinación de letras, números) de la secuencia.
5	Elemento de Traspaso	Combo Box	NO	Los Elementos de Traspaso que aparecen son los que están asociados al Centro de Costo seleccionado, de ellos se selecciona el Elemento de Traspaso con el que se quiere configurar la secuencia.
6	Criterio de Búsqueda	Combo Box	NO	Se seleccionará para la configuración de la secuencia uno de los dos Criterios de Búsquedas definidos.
7	Criterio de Distribución	Combo Box	NO	Se seleccionará para la configuración de la secuencia uno de los dos Criterios de Distribución definidos.

Tabla # 23: Descripción de la variable

Juegos de datos a probar

Id del escenario	Escenario	Cuenta	Estado	Fecha Fin	Descripción	Elemento de Traspaso	Criterio de Búsqueda	Criterio de Distribución	Respuesta del Sistema	Resultado de la Prueba
EP 1.1	Definir correctamente la secuencia de traspaso.	V(Cuenta 700)	V()	V(2009/03/01)	V(Secuencia de enero)	V(Salario)	V(Por Saldo de Período)	V(Por Volumen)	Se muestra el mensaje: "Se realizó la operación correctamente.".	Correcto
		V(Cuenta 731)	NA	V(2009/03/10)	V(Secuencia de marzo)	V(Corriente)	V(Por Saldo Acumulado)	V(Bases Combinadas)		
EP 1.2	Presionar el botón **Cancelar** una vez selecciona dos los datos necesarios para definir el Centro de Costo Origen.	V(Cuenta 700)	V()	V(2009/03/21)	V(Secuencia de febrero)	V(Alimento)	V(Por saldo del Período)	V(Por Subelemento)	Se muestra el mensaje: "¿Está seguro que desea cancelar la operación?".	Correcto
		V(Cuenta 700)	NA	V(2009/03/28)	V(Secuencia de cuenta 731)	V(Vestuario)	V(Por Saldo Acumulado)	V(Saldo)		
EP 1.3	Configurar una secuencia de traspaso con datos incompleto	NA	V()	V(2009/03/29)	V(Secuencia de cuenta 700)	NA	V(Por saldo del Período)	V(Por Porciento)	Se muestra el mensaje: "Por favor, verifique nuevamente que hay datos incompleto	Correcto
		V(Cuenta 731)	NA	NA	V(Secuencia de cuenta 731)	V(Combustible)	V(Por Saldo Acumulado)	V(Por Subelemento)		

	s.	V(731)	V()	V(2009/05/23)	NA	V(Inmobiliario)	V(Por saldo del Período)	V(Por Volumen)	s.”.	
		V(Cuenta 700)	NA	V(2009/04/20)	V(Secuencia de abril)	NA	NA	V(Por Saldo)		
		V(cuenta 700)	NA	NA	V(traspaso 31)	V(Combustible)	NA	NA		
		V(700)	NA	NA	NA	NA	NA	NA		
EP 1.4	Presionar el botón **Quitar** una vez seleccionados el(os) Centro(s) de Costo Destino(s).	V(700)	V()	V(2009/08/23)	V(traspaso 3)	V(Energía)	V(Por Saldo Acumulado)	V(Bases Combinadas)	Se quitan automáticamente los Centro(s) de Costo Destino(s).	Correcto
		V(Cuenta 700)	NA	V(2009/05/23)	V(Secuencia de cuenta 700)	V(Combustible)	V(Por saldo Acumulado)	V(Por Saldo)		
EP 1.5	Presionar el botón **Salir** cuando se está creando la secuencia.	V(731)	V()	V(2009/06/16)	V(Secuencia de cuenta 731)	V(Transporte)	V(Por Saldo del Período)	V(Por Subelemento)	Se muestra el siguiente mensaje: “¿Está seguro que desea salir de la configuración?”.	Correcto
		V(Cuenta 700)	NA	V(2009/07/01)	V(Secuencia de julio)	V(tecnología)	V(Por saldo Acumulado)	V(Por Volumen)		
EP 1.6	Configurar una secuencia	V(731)	NA	V(2009/07/07)	V(Secuencia de julio)	V(Combustible)	V(Por saldo del Período)	V(Por Subelemento)	Se muestra el siguiente mensaje:	Correcto

	de traspaso sin haber selecciona do una Fecha fin.	V(731)	V()	V(2009/ 07/07)	V(Secuencia de julio)	V(Trans- porte)	V(Por saldo del Período)	V(Por Subelemento)	“Introduzca una Fecha fin.”.	
EP 1.7	Configurar una secuencia de traspaso sin haber selecciona do una Cuenta de Gasto.	V(Cuent a 700)	V()	V(2009/ 05/23)	V(Secuencia 3)	V(salario)	V(Por saldo Acumulado)	V(Por Volumen)	Se muestra el siguiente mensaje: “Seleccion e una Cuenta.”.	Correcto
		V(Cuent a 700)	NA	V(2009/ 05/23)	V(Secuencia 3)	V(salario)	V(Por saldo Acumulado)	V(Por Subelemento)		
EP 1.8	Configurar una secuencia de traspaso sin haber selecciona do un Centro de Costo Origen.	V(731)	V()	V(2009/ 01/23)	V(Secuencia 1)	V(Inmobilia rio)	V(Por saldo Acumulado)	V(Por Volumen)	Se muestra el siguiente mensaje: “Seleccion e un Centro de Costo.”.	Correcto
		V(731)	NA	V(2009/ 01/23)	V(Secuencia 1)	V(Inmobilia rio)	V(Por saldo Acumulado)	V(Por Volumen)		
EP 1.9	Configurar una secuencia	V(Cuent a 700)	NA	V(2009/ 07/07)	V(Secuencia de julio)	V(Transpor te)	V(Por saldo del Período)	V(Por Subelemento)	Se muestra el siguiente mensaje:	Correcto

	de traspaso sin haber selecciona do un Elemento de Traspaso.	V(Cuent a 700)	V()	V(2009/ 07/07)	V(Secuencia de julio)	V(Transpor te)	V(Por saldo del Período)	V(Por Subelemento)	“Seleccion e un Elemento de Traspaso.”.	
EP 1.10	Configurar una secuencia de traspaso sin haber selecciona do un Criterio de Búsqueda.	V(700)	V()	V(2009/ 08/03)	V(Secuencia de traspaso)	V(Salario)	V(Por saldo Acumulado)	V(Bases Combinadas)	Se muestra el mensaje: “Seleccion e un Criterio de Búsqueda.” .	Correcto
		V(700)	NA	V(2009/ 08/03)	V(Secuencia de traspaso)	V(Salario)	V(Por saldo del Período)	V(Bases Combinadas)		
EP 1.11	Configurar una secuencia de traspaso sin haber introducido su descripción .	V(Cuent a 731)	NA	V(2009/ 08/10)	V(Secuencia de traspaso 3)	V(Combust ible)	V(Por saldo Acumulado)	V(Por Volumen)	Se muestra el mensaje: “Describa la Secuencia de Traspaso.”.	Correcto
		V(Cuent a 731)	V()	V(2009/ 08/13)	V(Secuencia de traspaso 3)	V(Combust ible)	V(Por saldo Acumulado)	V(Por Volumen)		

Tabla # 24: Juego de datos a probar

3.5 Validación del modelo de diseño propuesto

Desde que el ser humano se ha visto capaz de producir en cualquier rama de la ciencia en beneficio propio, ha buscado la calidad de sus productos. En la rama de las ciencias informáticas se ha hecho necesario definir una serie de atributos los cuales miden la calidad de los productos de software, algunos de estos son el tamaño, la complejidad, la frecuencia esperada de aparición de errores, cobertura de pruebas. Los objetivos perseguidos por dichas mediciones pueden ser la búsqueda de un método de estimación de esfuerzo de desarrollo, el cálculo de la cobertura de pruebas en el aseguramiento de la calidad o como en el caso que nos ocupa, la calidad en el diseño software Orientado a Objetos. Basándonos en el estudio realizado sobre las Métricas Técnicas para sistemas Orientados a Objetos de Pressman (12) obtuvimos un esquema sencillo de implementar y que a la vez cubre los principales atributos de calidad de software, siendo esto la principal razón de la concepción de las métricas inspiradas en lo propuesto por Pressman.

Atributos de calidad que se abarcan:

- **Responsabilidad**: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación**: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización**: Consiste en el grado de reutilización de presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento**: Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- **Complejidad del mantenimiento**: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- **Cantidad de pruebas**: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, módulo, clase, conjunto de clases, etc.) diseñado.

Las métricas concebidas como instrumento para evaluar la calidad del diseño del proceso de Traspasos y su relación con los atributos de calidad definidos en este trabajo exceptuando las métricas asociadas con la herencia (Profundidad de Herencia **(PH),** Número de Descendientes **(ND)** y Número de Operaciones Redefinidas para una clase

hija **(NOR)** puesto que las clases pertenecientes al proceso de Traspaso no poseen herencia) son las siguientes:

Tamaño operacional de clase (TOC): Está dado por el número de métodos asignados a una clase.

Atributo que afecta	**Modo en que lo afecta**
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

Tabla # 25: Tamaño operacional de clase (TOC)

Relaciones entre clases (RC): Está dado por el número de relaciones de uso de una clase con otras.

Atributo que afecta	**Modo en que lo afecta**
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad del mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla # 26: Relaciones entre clases (RC)

A continuación se presentan los resultados obtenidos de la aplicación de los instrumentos de evaluación para medir las métricas anteriormente descritas, y una valoración de los mismos.

3.5.1 Resultados del instrumento de evaluación de la métrica Tamaño operacional de clase (TOC)

Ver instrumentos y tabla de resultados en (Anexo # 9: Instrumento de medición de la métrica Tamaño operacional de clase (TOC)).

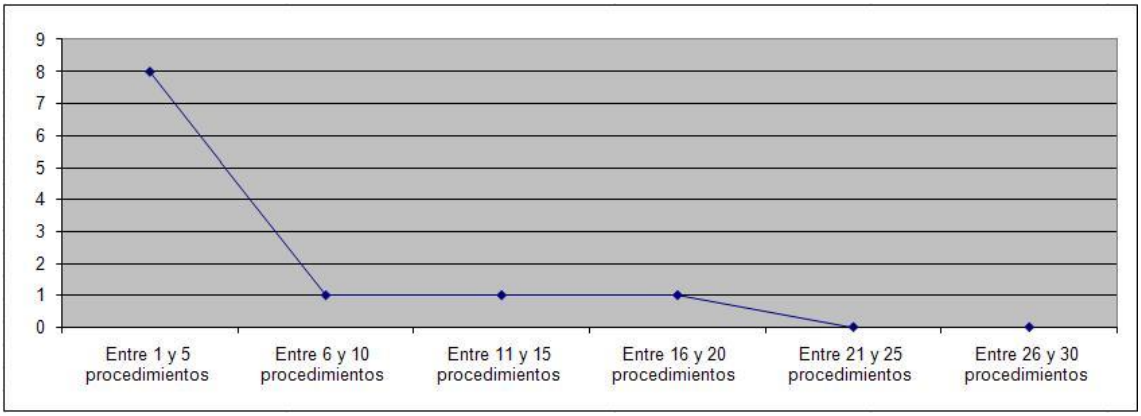

Figura # 7: Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos

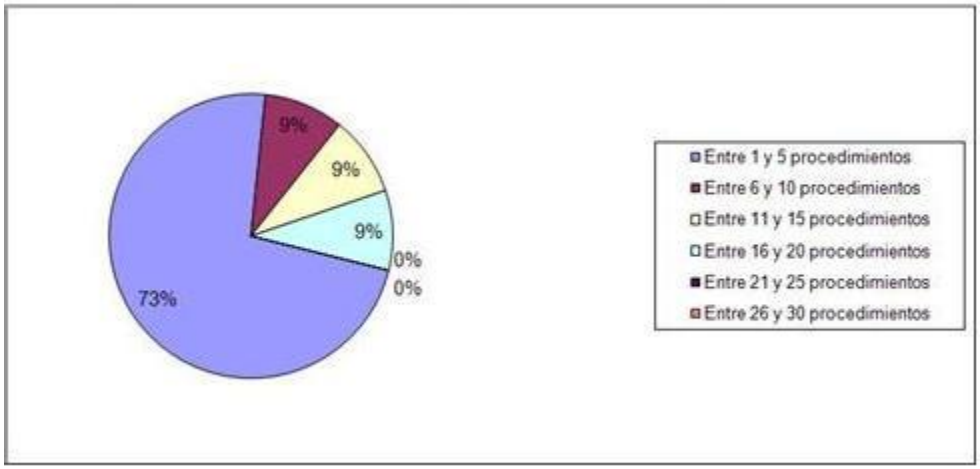

Figura # 8: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos

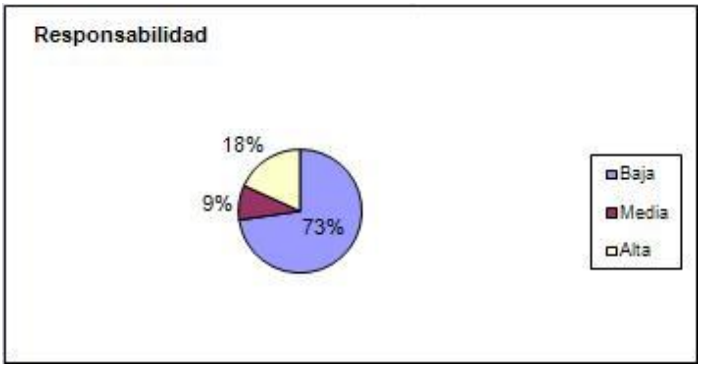

Figura # 9: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad

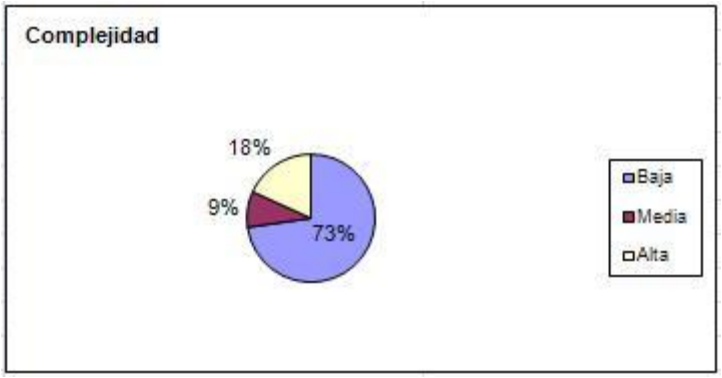

Figura # 10: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación

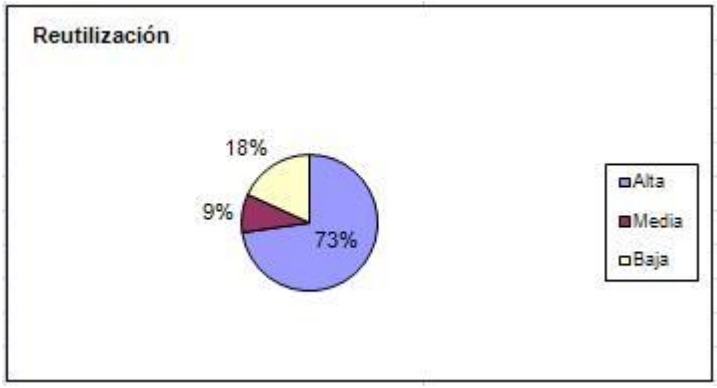

Figura # 11: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño del proceso de Traspasos tiene una calidad aceptable teniendo en cuenta que el 82% de las clases incluidas en este

proceso posee menos cantidad de operaciones que el doble del promedio calculado. Además el 82% de las clases poseen evaluaciones positivas en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).

3.5.2 Resultados del instrumento de evaluación de la métrica Relaciones entre Clases (RC)

Ver instrumentos y tabla de resultados en (Anexo # 10 Instrumento de medición de la métrica Relaciones entre clases (RC)).

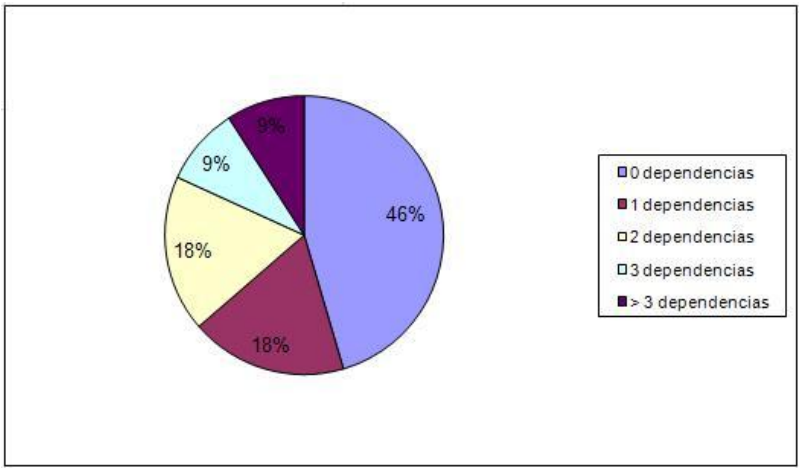

Figura # 12: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos

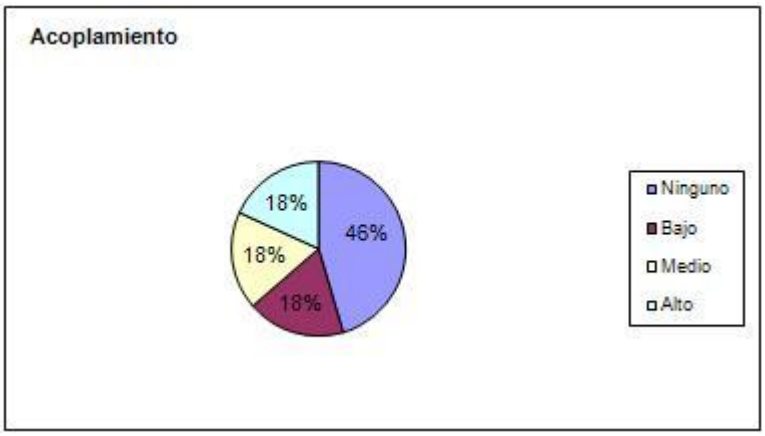

Figura # 13: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento

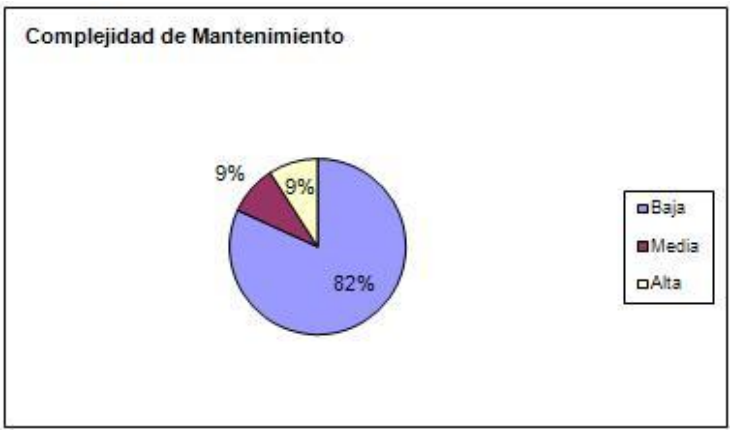

Figura # 14: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento

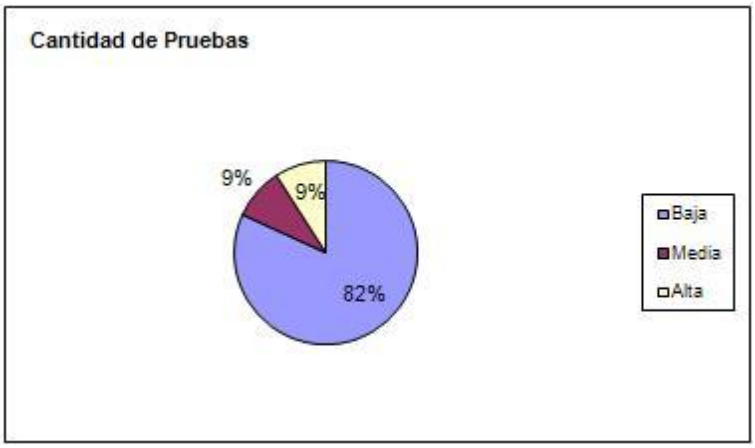

Figura # 15: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas

Figura # 16: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del proceso de Traspasos tiene una calidad aceptable teniendo en cuenta que el 82% de las clases incluidas en el proceso posee menos dependencias de otras clases que el doble del promedio calculado. Además el 46% de las clases no poseen acoplamiento con otras y el 82% posee índices aceptables en cuanto a Acoplamiento. Así mismo los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización se comportan satisfactoriamente en un 91 % de las clases.

0.1 a 0.3	Mal
0.4 a 0.7	Regular
0.8 a 1	Bien

Tabla # 27: Umbrales para definir la calidad del diseño

Atributos\Métricas	Tamaño Operacional de Clase **(TOC)**	Profundidad de Herencia **(PH)**	Relaciones entre Clases **(RC)**	Número de Descendientes **(ND)**	Número de Operaciones Redefinidas **(NOR)**	**Total**
Complejidad Implementación	0.5	-				0.5
Reutilización	0.5	-	1	-		0.75
Acoplamiento			0.5			0.5
Complejidad Mantenimiento		-	1		-	1
Cantidad de Pruebas			1	-	-	1
Responsabilidad	0.5			-	-	0.5
Total	0.5	-	0.9	-	-	0.7

Tabla # 28: Resultado final del proceso de aplicación de las métricas (TOC) y (RC)

3.6 Conclusiones

Con la realización de este capítulo, se ha podido arribar a la conclusión que la elaboración y aplicación de las pruebas resulta de gran importancia para la validación de la calidad del software. Fueron utilizadas las métricas Tamaño Operacional de Clase y Relaciones entre Clases para verificar el diseño propuesto arrojando resultados aceptables en la mayoría de los atributos de calidad los cuales estas métricas miden.

Se aplicaron correctamente dos tipos de prueba a la aplicación, donde se obtuvieron resultados satisfactorios, pues no se encontraron errores de implementación lo que muestra que el trabajo de desarrollo se llevó a cabo con el cuidado requerido.

Conclusiones Generales

El módulo de Costos y Procesos se encuentra actualmente en proceso de desarrollo, este trabajo de diploma tenía como objetivo obtener mediante la implementación uno de sus procesos, el de “Traspasos”, para ello se tuvo en cuenta la experiencia adquirida del estudio de sistemas a nivel nacional e internacional que presentan módulos con la gestión de traspasos o distribución de gastos indirectos identificando ventajas y deficiencias de los mismos, permitiendo erradicar en éste los problemas de los sistemas existentes y fusionar las mejores prácticas y funcionalidades.

Se le aplicaron métricas al diseño propuesto por los analistas, las cuales arrojaron resultados positivos, lo que permitió ahorrar tiempo al equipo de desarrollo, pues se partió de un diseño satisfactorio ya existente.

Una vez concluida la investigación realizada, se dio cumplimiento al objetivo y a las tareas planteadas logrando implementar una aplicación web que permite llevar a cabo todas las funcionalidades descritas en los requisitos establecidos por el cliente.

Se realizaron diferentes tipos de pruebas con el objetivo de comprobar la viabilidad de la solución obteniendo resultados favorables.

De forma general esta aplicación se ajusta a las nuevas tecnologías basadas en software libre.

Recomendaciones

Se recomienda realizar el despliegue de la aplicación en varias entidades para comprobar que cumple desde el punto de vista tecnológico con las expectativas del cliente.

Se aconseja optimizar un poco más el diseño de clases propuesto con el objetivo de realizar una mejor distribución de la responsabilidad y el acoplamiento entre las clases.

Se exhorta a seguir profundizando en el tema abordado para ampliar los conocimientos con vistas a detectar posibles debilidades en la implementación de la aplicación.

Bibliografía Referenciada

1. **Anónimo.** Costo de producción. 2007. http://www.monografias.com/trabajos29/costo-produccion/costo-produccion.shtml.

2. **Brango, Beatriz Elena Genes.** Contabilidad de Costos y Presupuesto. 5 27, 2009. http://www.mailxmail.com/curso-acumulacion-costos/elementos-gasto.

3. La gran enciclopedia de economía. http://www.economia48.com/spa/d/cuentas-de-gastos/cuentas-de-gastos.htm.

4.**Anónimo.** 2 25, 2009. http://www.cubaindustria.cu/contadoronline/contabilidad/Uso%20y%20Contenido/731%20Gastos%20Indirectos%20de%20Producci%C3%B3n.htm.

5. **Anónimo**. 1 10, 2009. http://www.mailxmail.com/curso/empresa/fundamentoscostos/capitulo5.htm.

6. **Leopoldo, Carlos.** Zend Framework, una introducción. 2008. http://www.carlosleopoldo.com/post/zend-framework-una-introduccion/.

7. **Arroyo, Cristian R.** Zend Framework 1.5. 2008. http://www.vivaphp.com.ar/frameworks/zend-framework-1.5.

8. **Garnet, Joan.** Doctrine: ORM Open Source para PHP 5.2+. 2007. http://www.joangarnet.com/blog/?p=415.

9. Ext JS: Cross-Browser Rich Internet Application Framework. 2009. http://extjs.com/products/extjs/.

10. **Peláez, Juan.** Arquitectura basada en capas. 2009. http://geeks.ms/blogs/jkpelaez/archive/2009/05/29/arquitectura-basada-en-capas.aspx.

11. **Catalani, Exequiel.** ARQUITECTURA Modelo/Vista/Controlador. 08 2007. http://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador/.

12. **Pressman, Roger S.** *Ingeniería del Software. Un Enfoque Práctico.* 2002.

Bibliografía Consultada

1. **Sampieri, Roberto Hernández.** *Metodología de la Investigación.*

2. **León, Rolando Alfredo Hernández and Coello González, Sayda.** *El paradigma cuantitativo de la investigación científica.*

3. **León, Rolando Alfredo Hernández and Coello González, Sayda.** *El paradigma cualitativo de la investigación social.*

4. **Meyer, Eric A.** *Cascading Style Sheets: The Definitive Guide.*

5. http://ebml.sourceforge.net/.

6. http://www.mysql.com/.

7. http://www.postgresql.org/.

8. http://www.microsoft.com/spain/sql/default.mspx.

9. http://www.microsoft.com/spain/windows/.

10. **Marc.** 6.5.8 Openbravo. 16 7, 2008. http://www.yourerpsoftware.com/content/658-openbravo.

11. **libre, Foro de Software.** ¿Qué es un ERP? (Enterprise Resource Planning para Linux). 11 27, 2007. http://www.versvs.net/anotacion/que-es-un-erp-enterprise-resource-planning-linux.

12. Estudio42. http://www.estudio42.com.ar/index.php/openerp.

13. Portal de ayuda del SAP. http://help.sap.com/saphelp_40b/helpdata/es/eb/13771c43c411d1896f0000e8322d00/content.htm.

14. **CONTADOR, EL ECO DEL.** El VERSAT-Sarasola: Sistema cubano de Gestión Contable-Financiero. 10 18, 2008. http://elecodelcontador.blogspot.com/2008/10/el-versat-sarasola-sistema-cubano-de.html.

15. GeneXus. 4 13, 2009. http://es.wikipedia.org/wiki/GeneXus.

16. PHP. 5 25, 2009. http://es.wikipedia.org/wiki/.php.

17. **Sitio Web universitario, Madrid, España.** http://gsyc.es/~agutierr/pfc-tecnica-html/node22.html.

18. Artículos de ASP.Net. 2005. http://www.devjoker.com/asp/ver_contenidos.aspx?co_contenido=125.

19. **Marañón, Gonzalo Álvarez.** Características del lenguaje Java. 2009. http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html.

20. ¿Que es Java? 2008. http://java.ciberaula.com/articulo/que_es_java/.

21. Lenguaje de Programación. 5 28, 2009. http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java.

22. Introducción al lenguaje html. 2008. http://www.hooping.net/faq-html.aspx.

23. **Rizo, César Labañino.** http://www.ispcmw.rimed.cu/sitios/digbiblio/cont/CI/Biblioteca_Virtual/GLOS_143.HTM.

24. Java Script. [Online] 12 2008. http://www.elcodigo.net/tutoriales/javascript/javascript1.html.

25. **Valdelli, Ilario.** Aspectos y características generales. http://www.htmlpoint.com/javascript/corso/js_02.htm.

26. Javascript. 5 27, 2009. http://es.wikipedia.org/wiki/Javascript.

27. **García, Luis.** Sistema de control de versiones: SUBVERSION. 1 17, 2008. http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=548&mode=thread&order=0&thold=0.

28. Documentación de la Comparativa MySQL vs PostgreSQL. 06 7, 2002. http://www.netpecos.org/docs/mysql_postgres/x15.html.

29. Base de datos SQL objeto-relacional, versión 8.3 del servidor. 2009. http://packages.debian.org/es/lenny/postgresql-8.3.

30. Los más comunes editores web. 5 29, 2009. http://www.diginota.com/index2.php?option=com_content&do_pdf=1&id=1996.

31. EMS PostgreSQL Manager. 04 3, 2006. http://www.sharewareconnection.com/ems-postgresql-manager.htm.

32. Mozilla Firefox, el navegador web del momento. 6 16, 2004. http://www.maestrosdelweb.com/editorial/firefox/.

33. Inversión de Control. 05 7, 2009. http://es.wikipedia.org/wiki/Inversi%C3%B3n_de_Control.

34. Patron (MVC) Sugerido. 7 13, 2007. http://orfeogpl.org/ata/node/242.

35. **Foro, Usuarios del.** ¿Comparativa entre php y asp, cual es mejor opción? 02 2007.http://foros.cristalab.com/comparativa-entre-php-y-asp-cual-es-mejor-opcion-t34769/.

36. **Covelo, Abraham.** Unit testing (pruebas unitarias). 05 2009. http://www.novanebula.net/blog/archives/99-Unit-testing-pruebas-unitarias.html.

37. **Group, W3C HTML Working.** ¿What is XHTML? 08 2002. http://www.w3.org/TR/xhtml1/#xhtml.

38. **Hickson, Ian.** Sending XHTML as text/html. http://hixie.ch/advocacy/xhtml.

39. DHTML. 5 21, 2009. http://es.wikipedia.org/wiki/DHTML.

40. **Alvarez, Miguel Angel.** Explica rápidamente lo que es HTML Dinámico y diferencia entre DHTML del lado del cliente y del servidor. 09 5, 2005. http://www.desarrolloweb.com/articulos/391.php.

41. **Aguila, Lic. Yoandry Pacheco.** AJAX un nuevo acercamiento a las aplicaciones Web. 02 8, 2007. http://www.monografias.com/trabajos43/ajax/ajax2.shtml.

42. Oracle. 05 12, 2009. http://es.wikipedia.org/wiki/Oracle.

43. Microsoft Sql Server. 05 23, 2009. http://es.wikipedia.org/wiki/Microsoft_SQL_Server.

44. **Pérez, Elizabeth Romero.** ¿Qué es Linux? 2008. http://www.monografias.com/trabajos14/linux/linux.shtml.

45. Portal GNU/Linux. 05 17, 2008. http://es.wikipedia.org/wiki/Portal:Linux.

46. Unix. 05 22, 2009. http://es.wikipedia.org/wiki/Unix.

47. **Carrasquel, Dorela.** UNIX. http://www.monografias.com/trabajos36/sistema-unix/sistema-unix.shtml.

48. Windows . 05 26, 2009. http://es.wikipedia.org/wiki/Windows.

49. **C., Benjamín González.** Descubriremos todo lo concerniente al protocolo SOAP: para que sirve, sus ventajas y la estructura de los mensajes. 07 7, 2004. http://www.desarrolloweb.com/articulos/1557.php.

50. Soap. 05 9, 2009. http://es.wikipedia.org/wiki/SOAP.

51. **Arregui, Juan José Olmedilla.** Revisión Sistemática de Métricas de Diseño Orientado a Objetos. 09 2005. http://www.scribd.com/doc/408149/Metricas-basadas-en-el-Diseno-Orientado-a-Objetos-Ingenieria-de-Software.

Glosario de Términos

CGI:(Por sus siglas en inglés “Common Gateway Interface”) es un método para la transmisión de información hacia un compilador instalado en el servidor. Su función principal es la de añadir una mayor interacción a los documentos web que por medio del HTML se presentan de forma estática. Esta tecnología tiene la ventaja de correr en el servidor cuando el usuario lo solicita por lo que es dependiente del servidor y no de la computadora del usuario.

XHTML: acrónimo inglés de eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web. En su versión 1.0, XHTML es solamente la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML. Su objetivo es avanzar en el proyecto del World Wide Web Consortium de lograr una web semántica, donde la información, y la forma de presentarla estén claramente separadas.

CSS: Las hojas de estilo en cascada (*Cascading Style Sheets*, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

XML: siglas en inglés de ***Ex****tensible* ***M****arkup* ***L****anguage* («lenguaje de marcas), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML. XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

ECMAScript: es una especificación de lenguaje de programación publicada por ECMA International. Actualmente está aceptado como el estándar ISO 16262. Define un lenguaje de tipos dinámicos ligeramente inspirado en Java y otros lenguajes del estilo de C. Soporta algunas características de la programación orientada a objetos mediante objetos basados en prototipos y pseudoclases.

DOM: El *Document Object Model* (una traducción al español no literal, pero apropiada, podría ser Modelo en Objetos para la representación de Documentos), abreviado DOM, es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML, que es para lo que se diseñó principalmente. En efecto, el DOM es una API para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como ECMAScript (Javascript).

XMLHttpRequest: También referida como XMLHTTP (Extensible Markup Language / Hypertext Transfer Protocol), es una interfaz empleada para realizar peticiones HTTP y HTTPS a servidores web. Para los datos transferidos se usa cualquier codificación basada en texto, incluyendo: texto plano, XML, JSON, HTML y codificaciones particulares específicas. La interfaz se presenta como una clase de la que una aplicación cliente puede generar tantas instancias como necesite para manejar el diálogo con el servidor. El uso más popular de esta interfaz es proporcionar contenido dinámico y actualizaciones asíncronas en páginas web mediante tecnologías construidas sobre ella como por ejemplo AJAX.

JSON: Acrónimo de "JavaScript Object Notation", es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

EBML: Sigla de Extensible Binary Meta Language (Meta Lenguaje Binario Extendible), fue diseñado como una extensión binaria simplificada de XML, con el propósito de almacenar y manipular datos de forma jerárquica con campos de longitud variable. Usa los mismos paradigmas que podemos encontrar en un archivo XML, separando sintaxis y semántica.

DHTML: El HTML Dinámico o DHTML (del inglés Dynamic HTML) designa el conjunto de técnicas que permiten crear sitios web interactivos utilizando una combinación de lenguaje HTML estático, un lenguaje interpretado en el lado del cliente (como JavaScript), el lenguaje de hojas de estilo en cascada (CSS) y la jerarquía de objetos de un DOM. Una página de HTML Dinámico es cualquier página web en la que los scripts en el lado del cliente cambian el HTML del documento, después de que éste haya cargado completamente, lo cual afecta a la apariencia y las funciones de los objetos de la página. La característica dinámica del DHTML, por tanto, es la forma en que la página interactúa con el usuario cuando la está viendo, siendo la página la misma para todos los usuarios.

AJAX: acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). 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. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. AJAX es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

MySQL: Es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

PostgreSQL: Es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD. Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (***PostgreSQL Global Development Group***).

Oracle: Es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), desarrollado por Oracle Corporation.

Microsoft SQL Server: Es un sistema de gestión de bases de datos relacionales (SGBD) basado en el lenguaje Transact-SQL, y específicamente en Sybase IQ, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

Constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, PostgreSQL.

Linux: Es el núcleo o kernel del sistema operativo libre denominado GNU/Linux (coloquial pero erróneamente llamado Linux). Lanzado bajo la licencia pública general (GPL - General Public License) de GNU y desarrollado gracias a contribuciones provenientes de todo el mundo, Linux es uno de los mejores ejemplos de software libre cuyos desarrolladores originales siguieron la filosofía de ese movimiento. Linux fue creado por Linus Torvalds en 1991.

UNIX: (registrado oficialmente como UNIX®) es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T.

Windows: Es una familia de sistemas operativos desarrollados y comercializados por Microsoft. Existen versiones para hogares, empresas, servidores y dispositivos móviles, como computadores de bolsillo y teléfonos inteligentes. Hay variantes para procesadores de 16, 32 y 64 bits. Incorpora diversas aplicaciones como Internet Explorer, el Reproductor de Windows Media, Windows Movie Maker, Windows Mail, Windows Messenger, Windows Defender, entre otros. Desde hace muchos años es el sistema operativo más difundido y usado del mundo; de hecho la mayoría de los programas (tanto comerciales como gratuitos y libres) se desarrolla originalmente para este sistema.

SOAP: (siglas de Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

IBM: International Business Machines o IBM (conocida coloquialmente como el Gigante Azul) es una empresa que fabrica y comercializa herramientas, programas y servicios relacionados con la informática. Tiene su sede en Estados Unidos y está constituida como tal desde junio de 1911, pero lleva operando desde 1888.

WebSphere: WebSphere es una familia de productos de software propietario de IBM, aunque el término se refiere de manera popular a uno de sus productos específicos: WebSphere Applications Server (WAS). WebSphere ayudó a definir la categoría de software middleware y está diseñado para configurar, operar e integrar aplicaciones de e-

business a través de varias plataformas de red usando las tecnologías de la web. Esto incluye componentes de run-time (como el WAS) y las herramientas para desarrollar aplicaciones que se ejecutarán sobre el WAS.

Apache: El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

Tomcat: Es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache. Tomcat puede funcionar como servidor web por sí mismo.

Python: Es un lenguaje de programación interpretado creado por Guido van Rossum en el año 1991. Python se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation.

GeneXus: Es una herramienta de desarrollo de software basada en conocimiento, orientada principalmente a aplicaciones de clase empresarial para la web y plataformas Windows. El desarrollador especifica sus aplicaciones en alto nivel (de manera mayormente declarativa), a partir de lo cual se genera código para múltiples entornos.

Mac: En redes de computadoras la dirección MAC *(Media Access Control address* o dirección de control de acceso al medio) es un identificador de 48 bits (6 bytes) que corresponde de forma única a una tarjeta o interfaz de red.

Java Virtual Machine: Una Máquina virtual Java (en inglés *Java Virtual Machine*, **JVM**) es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.

SGML: Son las siglas de ***S****tandard* ***G****eneralized* ***M****arkup* ***L****anguage* o "Lenguaje de Marcado Generalizado". El lenguaje SGML sirve para especificar las reglas de etiquetado de documentos y no impone en sí ningún conjunto de etiquetas en especial.

SVG (Scalable Vector Graphics): Es un lenguaje para describir gráficos vectoriales bidimensionales, tanto estáticos como animados (estos últimos con ayuda de SMIL), en XML.

MathML (Mathematical Markup Language): El MathML o Mathematical *Markup Language* es un lenguaje de marcado basado en XML, cuyo objetivo es expresar notación matemática de forma que distintas máquinas puedan entenderla, para su uso en combinación con XHTML en páginas web, y para intercambio de información entre programas de tipo matemático en general.

Netscape: Netscape Navigator es un navegador web y el primer resultado comercial de la compañía Netscape Communications, creada por Marc Andreessen, uno de los autores de Mosaic, cuando se encontraba en el NCSA (Centro Nacional de Aplicaciones para Supercomputadores) de la Universidad de Illinois en Urbana-Champaign. Netscape fue el primer navegador comercial.

Opera: Es un navegador web y suite de Internet creado por la empresa noruega Opera Software. Es reconocido por su gran velocidad, seguridad, soporte de estándares (especialmente CSS), tamaño reducido, internacionalidad y constante innovación.

RIA: Acrónimo de ***R****ich* ***I****nternet* ***A****pplications* (Aplicaciones de Internet Enriquecidas) son un nuevo tipo de aplicaciones con más ventajas que las tradicionales aplicaciones web. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones web y las aplicaciones tradicionales.

DOM: El Document Object Model (una traducción al español no literal, pero apropiada, podría ser Modelo en Objetos para la representación de Documentos), abreviado DOM, es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML.

CVS: El Concurrent Versions System (CVS), también conocido como Concurrent Versioning System, es una aplicación informática que implementa un sistema de control de versiones. Mantiene el registro de todo el trabajo y los cambios en los ficheros (código

fuente principalmente) que forman un proyecto (de programa) y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren. CVS se ha hecho popular en el mundo del software libre. Sus desarrolladores difunden el sistema bajo la licencia GPL.

WebDAV: Es un grupo de trabajo del Internet Engineering Task Force. El término significa "Edición y versionado distribuidos sobre la web", y se refiere al protocolo (más precisamente, la extensión al protocolo) que el grupo definió. El objetivo de WebDAV es hacer de la World Wide Web un medio legible y editable.

IDE (Integrated Development Environment): Un entorno de desarrollo integrado o, en inglés, *Integrated Development Environment* ('*IDE*'), es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

GRASP: En diseño orientado a objetos, GRASP son patrones generales de software para asignación de responsabilidades, es el acrónimo de "General Responsibility Assignment Software Patterns.

PDO: PHP Data Objects (o PDO) es una extensión que provee una capa de abstracción de acceso a datos para PHP 5, con lo cual se consigue hacer uso de las mismas funciones para hacer consultas y obtener datos de distintos manejadores de bases de datos.

Anexos

Anexo 1: Modelo de diseño

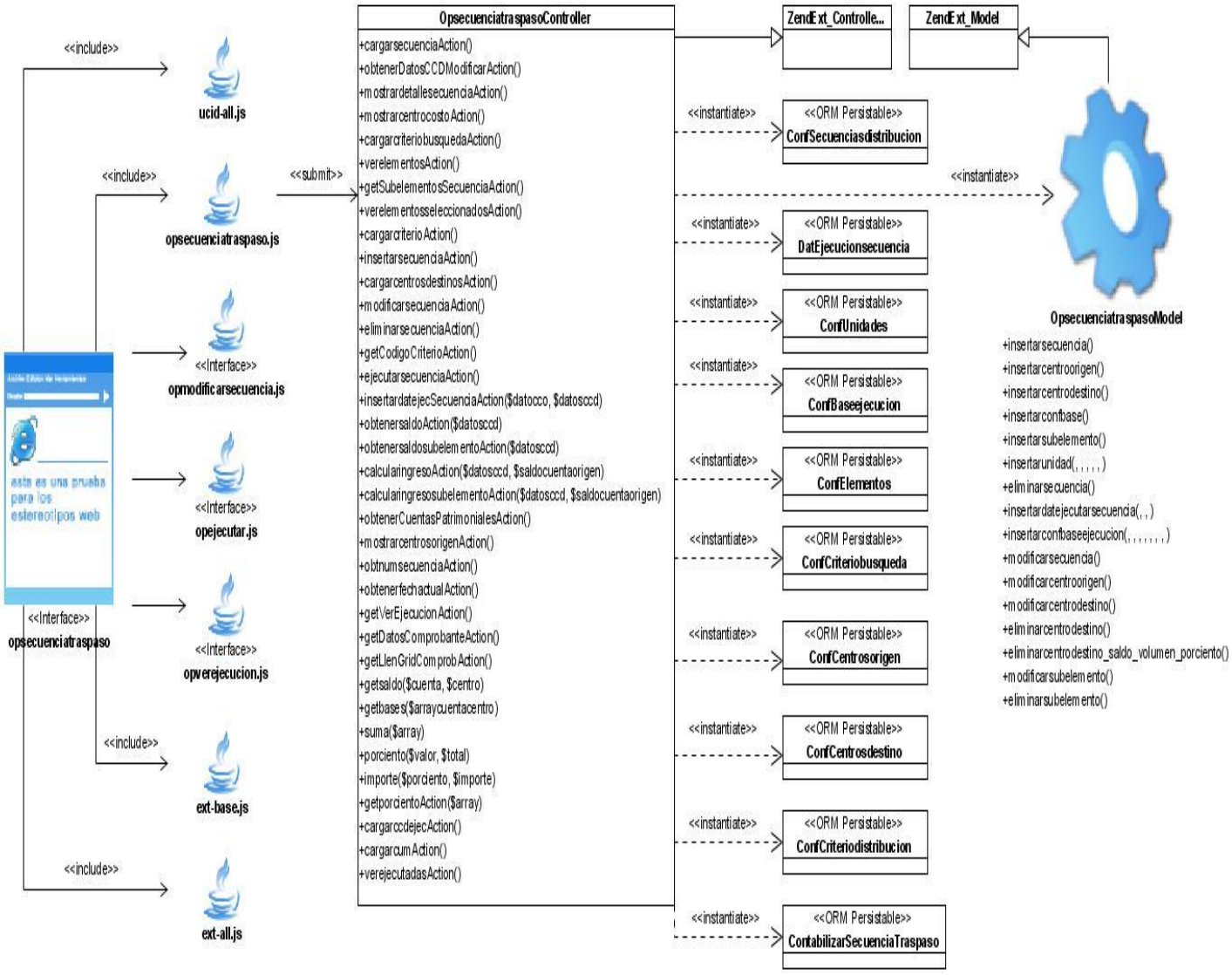

Figura # 17: Modelo de diseño.

Anexo 2: Interfaz Base

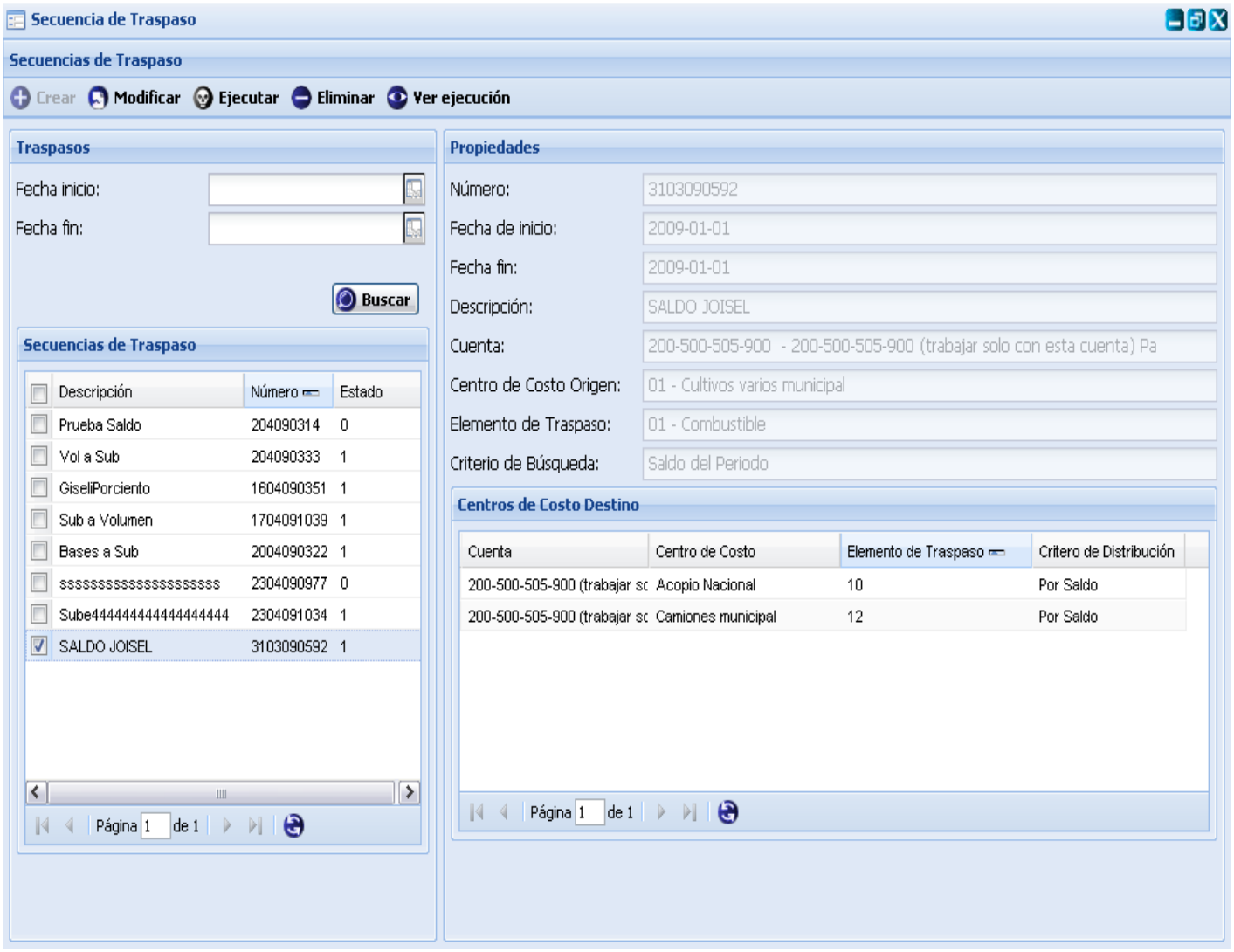

Figura # 18: Interfaz Base de la secuencia de traspaso.

Anexo 3: Interfaz Crear

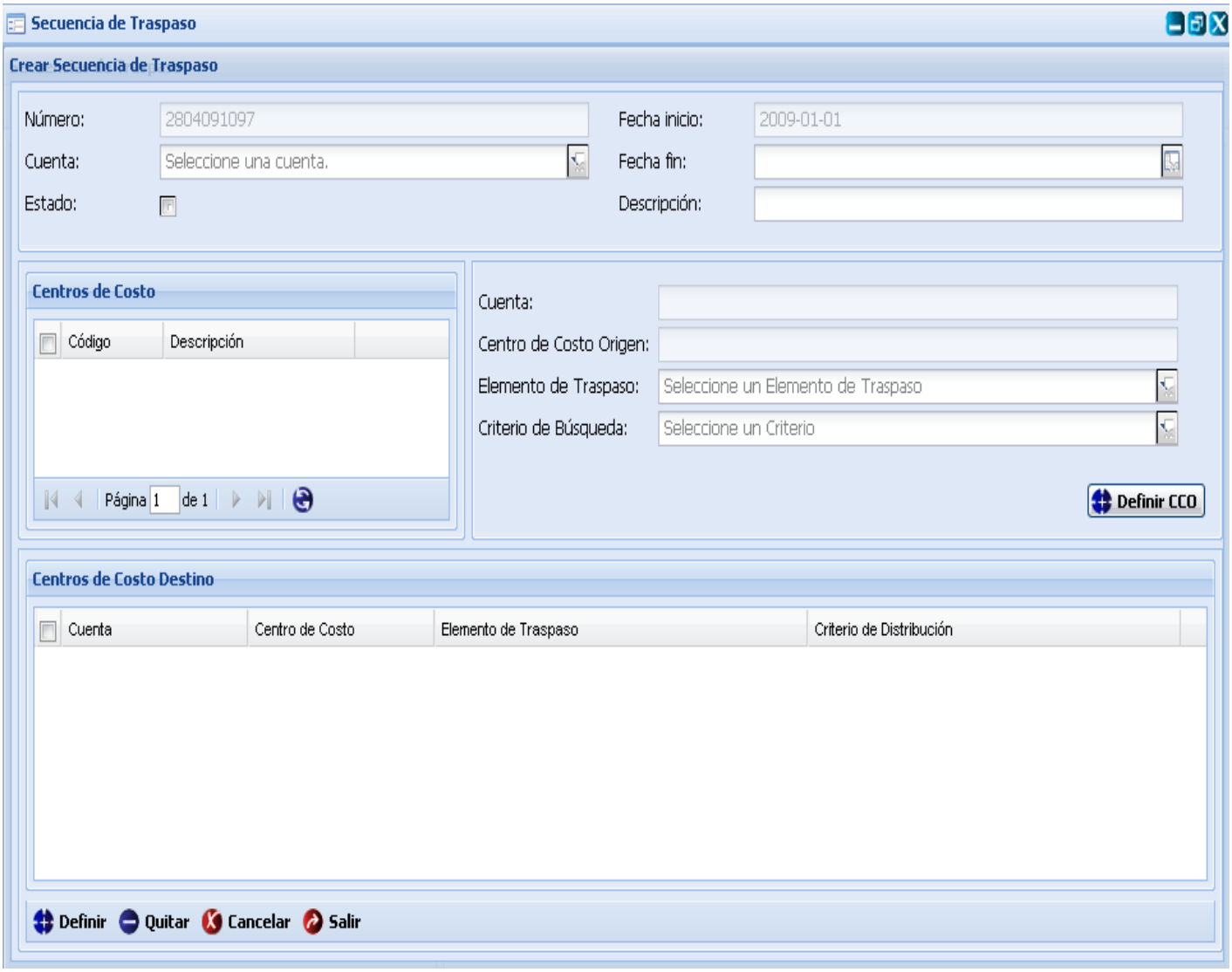

Figura # 19: Interfaz Crear secuencia de traspaso.

Anexo 4: Interfaz Ejecutar

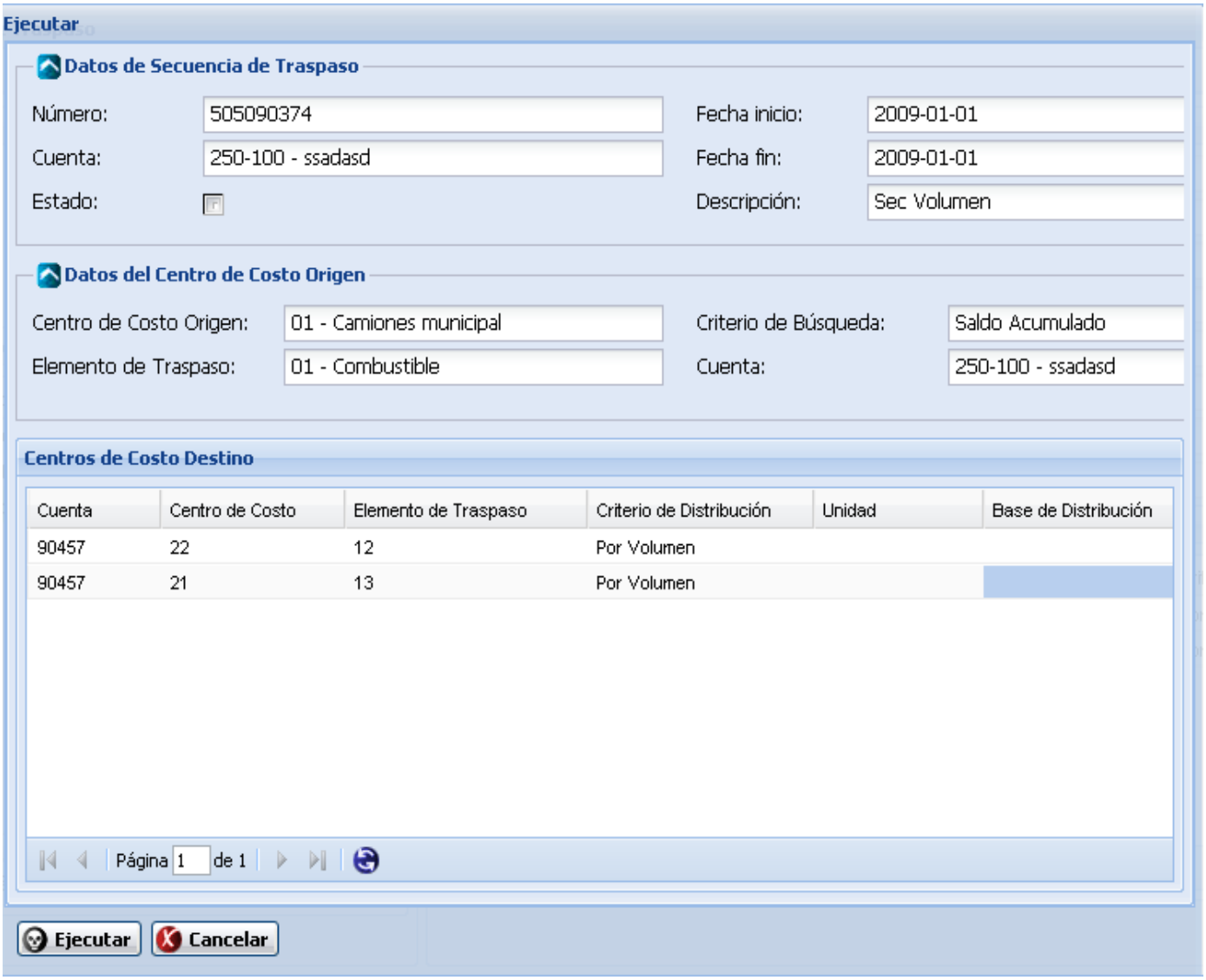

Figura # 20: Interfaz Ejecutar secuencia de traspaso.

Anexo 5: Interfaz Modificar

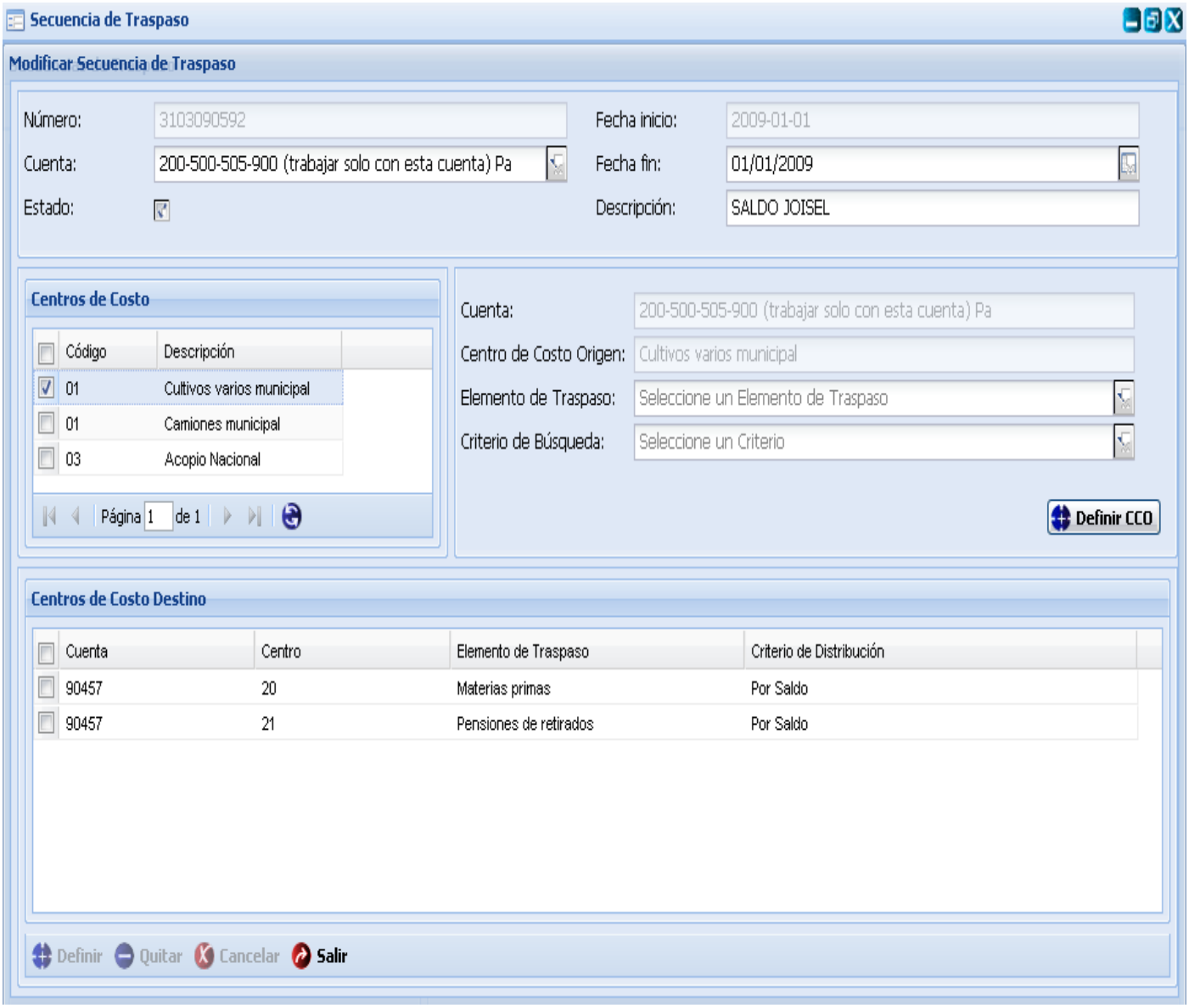

Figura # 21: Interfaz Crear secuencia de traspaso.

Anexo 6: Interfaz Mostrar secuencias de traspasos ejecutadas

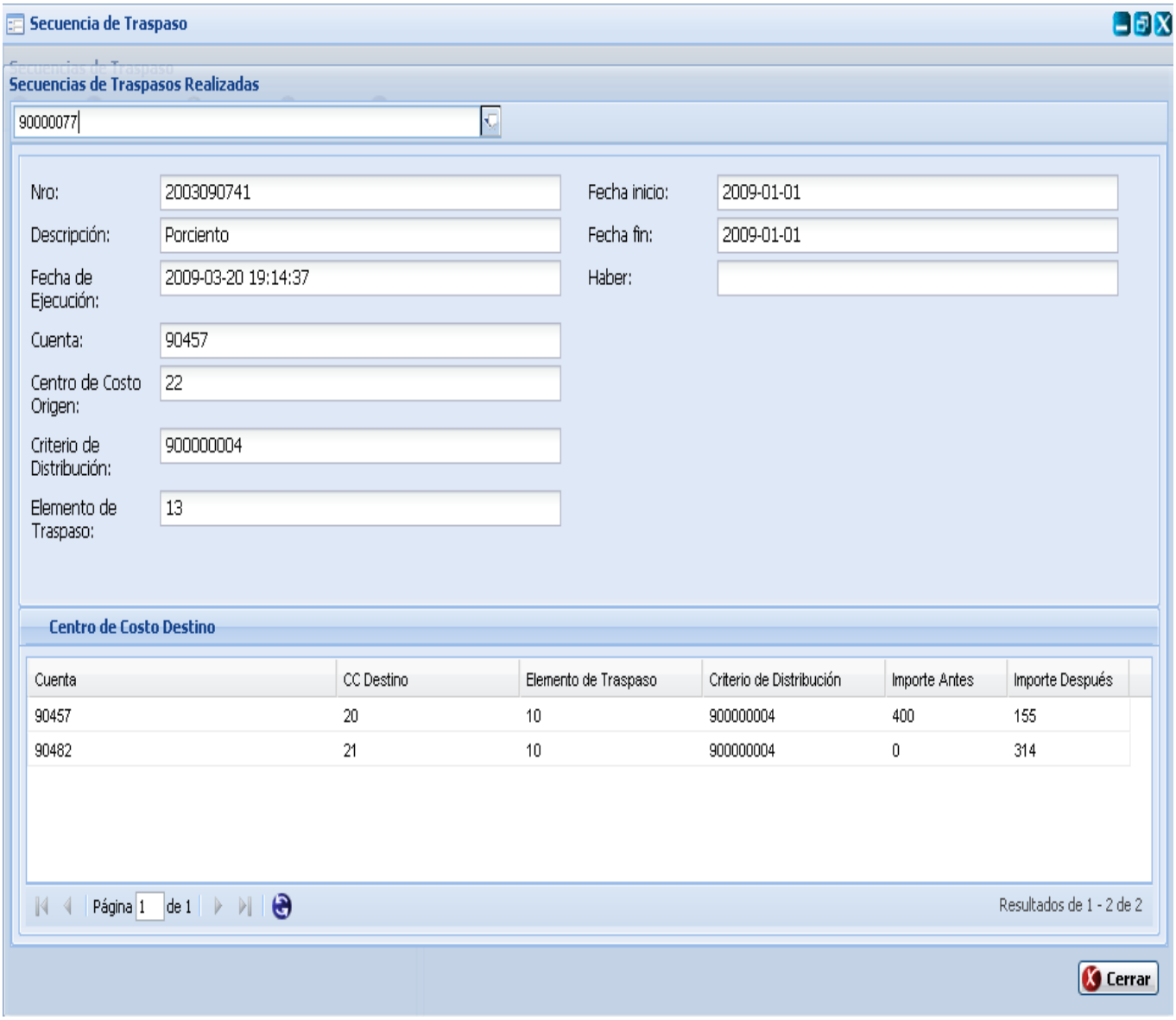

Figura # 22: Interfaz Mostrar secuencias de traspasos ejecutadas.

Anexo 7: Servicios internos del Módulo de Costos y Procesos

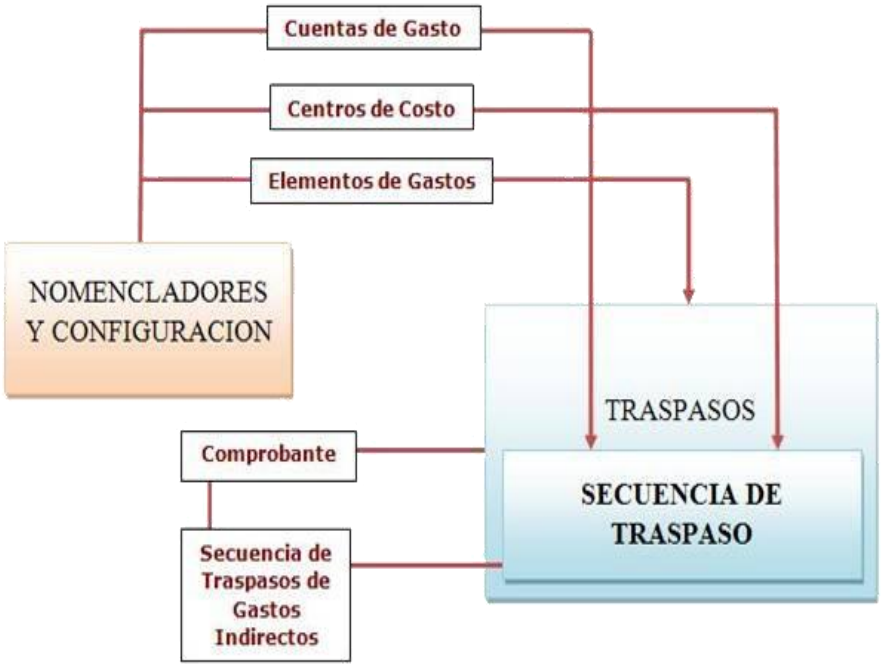

Figura # 23: Servicios internos del Módulo de Costos y Procesos.

Anexo 8: Servicios compartidos entre Subsistemas relacionados con la secuencia de traspaso

Figura # 24: Servicios compartidos entre Subsistemas relacionados con la secuencia de traspaso.

Anexo 9: Instrumento de medición de la métrica Tamaño operacional de clase (TOC)

	Categoria	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Complejidad implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	<= Prom.

Tabla # 28: Rango de valores de para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados con la métrica TOC.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
OpsecuenciatraspasoModel	16	Alta	Alta	Baja
ContabilizarSecuenciaTraspaso	3	Baja	Baja	Alta
ConfBaseejecucion	2	Baja	Baja	Alta
ConfCentrosdestino	2	Baja	Baja	Alta
ConfCentrosorigen	2	Baja	Baja	Alta
ConfCriteriobusqueda	2	Baja	Baja	Alta
ConfCriteriodistribucion	2	Baja	Baja	Alta
ConfElementos	2	Baja	Baja	Alta
ConfSecuenciasdistribucion	11	Alta	Alta	Baja
ConfUnidades	0	Baja	Baja	Alta
DatEjecucionsecuencia	7	Media	Media	Media

Tabla # 29 Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).

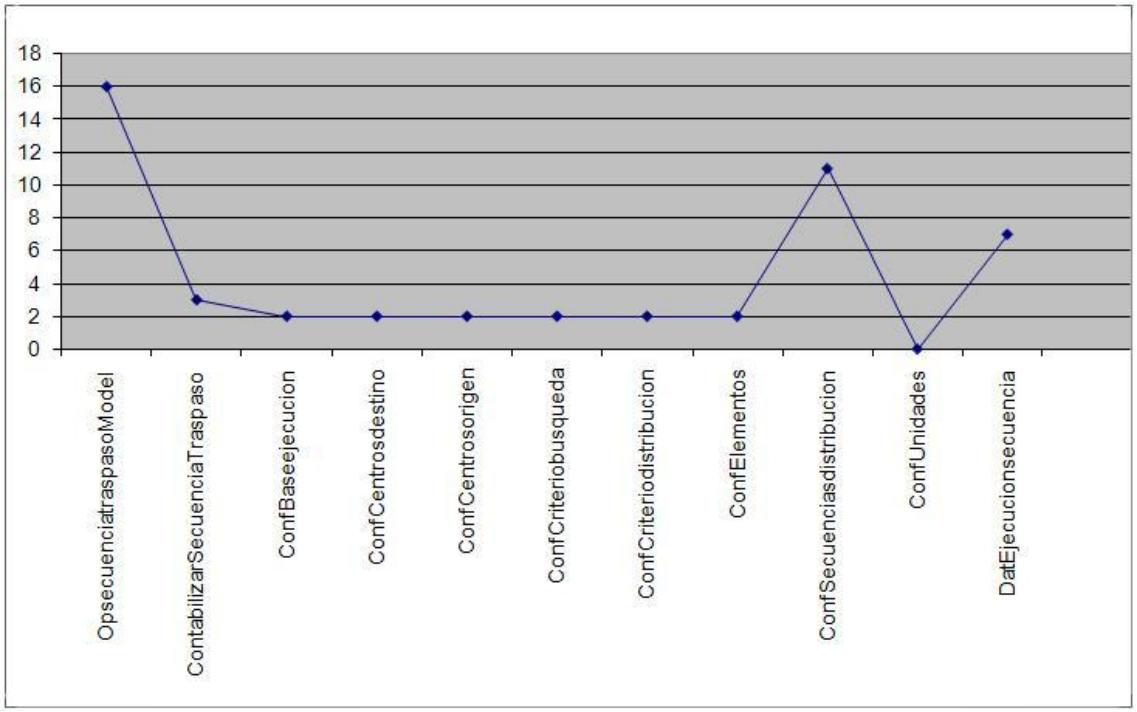

Figura # 25: Gráfica de los resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).

Anexo 10: Instrumento de medición de la métrica Relaciones entre clases (RC)

	Categoria	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2

	Categoria	Criterio
Complejidad Mantenimiento	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

	Categoria	Criterio
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.

	Categoria	Criterio
Cantidad de Pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Tabla # 30: Rango de valores de para la evaluación técnica de los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas) relacionados con la métrica RC.

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mantenimiento	Reutilización	Cantidad de Pruebas
OpsecuenciatraspasoModel	6	Alto	Alta	Baja	Alta
ContabilizarSecuenciaTraspaso	2	Medio	Baja	Alta	Baja
ConfBaseejecucion	0	Ninguno	Baja	Alta	Baja
ConfCentrosdestino	1	Bajo	Baja	Alta	Baja
ConfCentrosorigen	1	Bajo	Baja	Alta	Baja
ConfCriteriobusqueda	0	Ninguno	Baja	Alta	Baja
ConfCriteriodistribucion	0	Ninguno	Baja	Alta	Baja
ConfElementos	0	Ninguno	Baja	Alta	Baja
ConfSecuenciasdistribucion	3	Alto	Media	Media	Media
ConfUnidades	0	Ninguno	Baja	Alta	Baja
DatEjecucionsecuencia	2	Medio	Baja	Alta	Baja

Tabla # 31: Resultados de la evaluación de la métrica RC y su influencia en los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas).

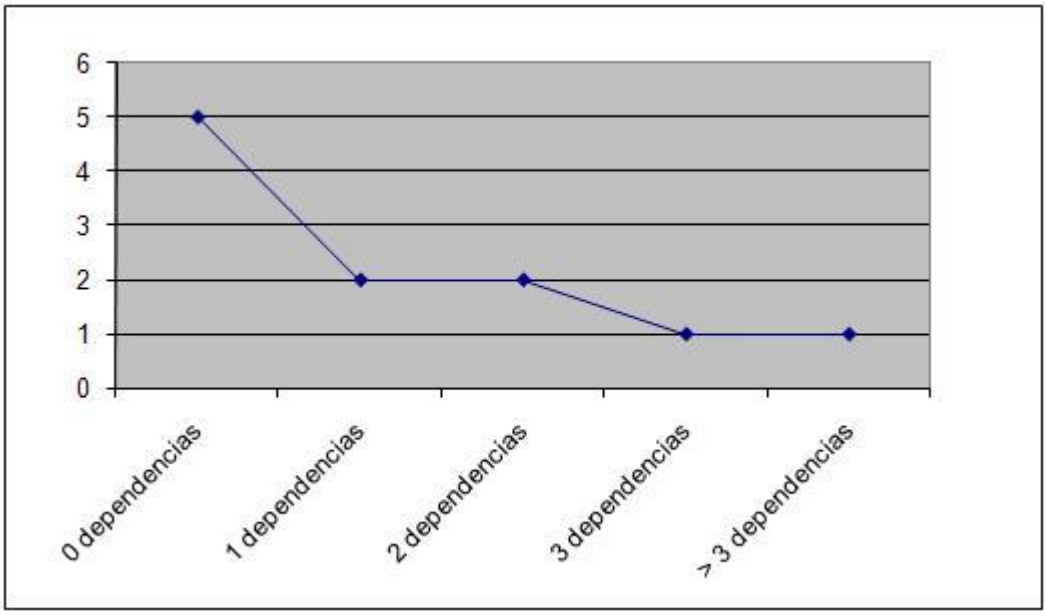

Figura # 26: Gráfica de los resultados de la evaluación de la métrica RC agrupados por la tendencia de los valores.