Universidad de las Ciencias Informáticas

ERP-Cuba

Título: Implementación del Componente Nomencladores y Configuración del Sistema Integral de Gestión Cedrux.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor(es): José Antonio Linares Torres

Jorge Núñez Silveira

Tutor(es): Ing. Enrique Chaviano Gómez

DATOS DE CONTACTO

Síntesis del Tutor: Ing. Enrique Chaviano Gómez.
Profesión: Ingeniero en ciencias informáticas
Años de graduado: 1

RESUMEN

El país avanza con amplios pasos dentro del proceso de utilización ordenada y masiva de las tecnologías de la información y las comunicaciones. Factor decisivo para el desarrollo de las empresas, economía y de la sociedad cubana. Proceso que busca lograr más eficacia y eficiencia, satisfacer las necesidades de información y conocimiento de todas las personas y esferas de la sociedad.

La necesidad de mejorar los procesos de la Contabilidad de Costos, está impulsando a la adopción de sistemas de planificación de recursos empresariales, son sistemas de información gerenciales que integran y manejan muchos de los negocios asociados con las operaciones de producción y planificación estratégica, en base a esta, se pueden determinar los capitales destinado a los materiales necesarios para llevar a cabo las actividades empresariales, sean estas de producción industrial o de servicios.

Este trabajo abarca la implementación del componente Nomenclador y Configuración que permite mejorar la eficiencia y confiabilidad en la gestión de los procesos de la Contabilidad de Costo en cualquier entidad.

PALABRAS CLAVE

Centros de Costos, Elementos de Gastos, Cuentas de Gastos.

ÍNDICE

INTRODUCCIÓN

A lo largo de los años la contabilidad por su carácter social ha jugado un papel primordial en el desarrollo de las civilizaciones, específicamente, del comercio. La contabilidad de costos posee una gran relevancia en todas las empresas ya que forma parte importante durante la planificación estratégica de los negocios a concretar, en base a esta, se pueden determinar los capitales destinado a los materiales necesarios para llevar a cabo las actividades empresariales, sean estas de producción industrial o de servicios (López, 2008).

La Tecnología de la Información (TI) está cambiando constantemente en el registro de las operaciones contables que en las empresas se están llevando acabo, las personas que trabajan con el gobierno, en empresas privadas, que dirigen personal o que trabajan como profesional en cualquier campo, utilizan la TI cotidianamente mediante el uso de Internet, las tarjetas de crédito, el pago electrónico de la nómina, entre otras funciones; es por eso que la función de la TI en los procesos de la empresa como manufactura y ventas se han expandido grandemente.

La necesidad de mejorar los procesos de negocio está impulsando la adopción de Sistemas de Planificación de Recursos Empresariales (ERP), son sistemas de información gerenciales que integran y manejan muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía comprometida en la producción de bienes o servicios. Mejorar la eficacia de sus procesos de negocio y conseguir más transparencia y calidad de la información son los dos principales argumentos para adoptar un sistema ERP.

Los sistemas que controlan la Contabilidad de Costo en Cuba no permiten tomar decisiones administrativas con rapidez, además de coexistir varios sistemas que realizan las operaciones de manera diversa lo que impide la integración de los procesos contables. En algunas empresas estas actividades todavía se llevan de forma manual utilizando como soporte el papel, lo cual torna el trabajo engorroso, pude acarrear la pérdida de datos y baja confidencialidad de los mismos (Cardoso, 2008).

Analizado lo planteado anteriormente surge como **problema científico**: ¿Cómo mejorar la eficiencia y confiabilidad en la gestión de los procesos de la Contabilidad de Costo permitiendo mayor rapidez en la toma de decisiones en cualquier entidad?

Objeto de Estudio:

Los procesos de Nomenclador y Configuración de la Contabilidad de Costo.

Objetivos Generales:

Obtención del componente Nomenclador y Configuración mediante la implementación, permitiendo una mayor eficiencia, confiabilidad y rapidez en la toma de decisiones en la gestión de procesos de la Contabilidad de Costo en una entidad.

Campo de Acción:

Diseño de la Implementación de los procesos de Nomenclador y Configuración del módulo Costos y Procesos.

Idea a defender:

La implementación del componente Nomenclador y Configuración del Sistema Integral de Gestión Cedrux, contribuirá a una mayor eficiencia, confiabilidad y rapidez en la toma de decisiones en la gestión de procesos de la Contabilidad de Costo en una entidad.

Objetivos Específicos:

- Definir estándares de implementación para procesos Nomenclador y Configuración.
- Implementar la lógica del negocio de los componentes propuestos para el correcto funcionamiento del subsistema Costos y Procesos.
- Obtener componente Nomenclador y Configuración.
- Identificar e implementar las necesidades de integración con otros componentes dentro del subsistema Costos y Procesos.
- Analizar la Factibilidad del sistema a implementar.

Tareas de la Investigación:

- Estudiar la descripción de los procesos Nomenclador y Configuración del subsistema Costos y Procesos.

- Estudio el análisis, diseño y levantamiento de requisitos del subsistema Costos y Procesos del ERP Cubano.
- Estudio de la arquitectura general del proyecto ERP Cubano.
- Análisis de arquitectura de sistema y principales estándares definidos para la implementación del subsistema Costos y Procesos.
- Implementar la lógica del negocio de los componentes propuestos para el correcto funcionamiento del subsistema Costos y Procesos.
- Obtener componente Nomenclador y Configuración.
- Identificar e implementar las necesidades de integración con otros componentes dentro del subsistema Costos y Procesos.
- Analizar la Factibilidad del sistema a implementar.

Métodos investigativos:

Métodos Teóricos:

- Hipotético-deductivo
- Análisis Histórico-lógico.

Métodos empíricos

Observación

Para realizar una descripción detallada, este documento muestra el resultado de la investigación de la siguiente forma:

Capítulo 1. Fundamentación Teórica: Se expone el estado del arte, donde se muestran módulos de contabilidad de costo vinculados al campo de acción. Al mismo tiempo se describe el objeto de estudio, se explica el funcionamiento del componente Nomenclador y Configuración. También se describen tendencias y tecnologías actuales seleccionadas para el desarrollo del módulo y porque su utilización.

Capítulo 2 Descripción y análisis de la solución propuesta: Se elaboró una crítica al diseño propuesto por el analista, dado la importancia del mismo para la implementación. Se describe uno de los algoritmos más complejo con el que cuenta la aplicación explicando además la estructura de datos usada en el mismo, se describen las clases que fueron definidas, que sirven de documentación a los que den continuidad a la implementación del software y como se integran las mismas.

Capítulo 3. Validación de la solución propuesta: se valida la solución propuesta a través de la realización de pruebas, el mismo se encuentra dividido en dos partes la explicación detallada de las pruebas de caja blanca que fueron realizadas al código del software y las pruebas de caja negra que se realizaron a la interfaz del mismo.

CAPITULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Los avances tecnológicos con los que se conviven en la actualidad y ante la necesidad de crear aplicaciones que permitan lograr una mayor calidad en el trabajo que hoy enfrenta en disímiles esferas de la sociedad, se hace necesaria la creación de software. Mediante la implementación de los mismos se debe ser capaz de resolver todos los problemas planteados. Para lograr este objetivo, es necesaria la realización de un profundo análisis del negocio donde se desarrollan los procesos a automatizar, la investigación de sistemas que han sido implementados y el estudio de las principales tecnologías y herramientas que serán usadas para la implementación del módulo.

No solo en Cuba, sino en el mundo se han desarrollado software relacionados con la Contabilidad de Costo, cuyo estudio ha servido para tener en cuenta las ventajas y defectos que tienen. A menudo surgen dudas sobre cuál usar, debido a que actualmente aparecen nuevas herramientas con mejorías, que se van imponiendo para el desarrollo de software. Es por esto, que surge la necesidad de hacer un estudio de las herramientas a utilizar, buscando obtener la información más actual de los mismos.

1.2 Conceptos básicos asociados al dominio del problema

Los conceptos que se encuentran a continuación están asociados al dominio del problema y resultan necesarios para lograr un mayor entendimiento del mismo.

1.2.1 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.

1.2.2 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.

1.2.3 Nomencladores: Instrumento de clasificación para determinar la jerarquía de elementos.

1.2.4 Saldo Inicio de Año: Importe con que comienzan las cuentas de gastos definidas como de procesos en el periodo contable.

1.2.5 Cuenta de gasto: Es un presupuesto de gastos que está compuesto por gastos de funcionamiento, servicios de la deuda y gastos de inversión.

1.3 Sistemas existentes vinculados al campo de acción

Se investigó sobre las principales aplicaciones informáticas, similares al que se desea desarrollar, caracterizándolas y describiendo sus principales funcionalidades. A continuación, se analizan sistemas desarrollados para la Contabilidad de Costo en ámbito internacional y nacional.

1.3.1 Open Bravo:

Software desarrollado por la Universidad de Navarra en España, presenta el modulo **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 (OpenBravo, 2007).

Características:

- Implementado en el lenguaje Java.
- Aplicación completamente web que ha sido 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 desarrollado en Argentina, cuenta con un modulo **Gestión contable y financiera**, 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 (OpenERP, 2007).

Características:

- Implementado en Python.
- Tiene componentes separados en esquema Cliente-servidor, dispone de interfaces XML-RPC, y SOAP.

- Soporte para bases de datos PostgreSQL .
- Es multiplataforma, funciona sobre Linux y Windows.

1.3.4 SAP

Software desarrollado en la Ciudad de Mannheim, Alemania, por antiguos empleados de IBM. La corporación se ha desarrollado hasta convertirse en la quinta más grande compañía mundial de software. Cuenta con el modulo **Controlling (CO)**, que permite el control de los gastos generales, costes de producto, cuenta de resultados y centros de beneficio (Maestre, 2008).

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 .

1.3.5 Versat-Sarasola

Software desarrollado por la entidad cubana Desoft, sistema económico integrado constituido por 12 módulos entre ellos **Costos y Procesos**. 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 directo y el ajuste o costeo, según los volúmenes de producción o servicios y sus destinos (Rodríguez, 2006).

Características:

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

1.3.6 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 dato 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:

Teniendo en cuenta las características de los lenguajes de programación de los módulos estudiados se puede concluir que con el uso del lenguaje java en aplicaciones web el acceso a las páginas es más lento debido a la sobrecarga que genera la maquina virtual, se necesita mejoras de recursos de hardware, además de la imposibilidad de el país de acceder a la ultima versión de la maquina virtual por estar bloqueados. De .net podemos señalar que es un lenguaje propietario y lento para aplicaciones web. En el caso de Python se puede decir que no cuenta con IDEs de desarrollo para un proyecto de grandes dimensiones, además de ser un lenguaje poco conocido por lo desarrolladores. De Delphi se pude comentar que no esta diseñado para aplicaciones web, los IDE de desarrollo en su mayoría propietarios. Podemos concluir según lo analizado anteriormente que el lenguaje php es el que mas se ajusta a las necesidades del proyecto.

1.5 Tendencias y tecnologías actuales utilizadas

Después de haber realizado un análisis de las necesidades encontradas en la Contabilidad de Costo, se realizó un estudio de las tendencias y tecnologías actuales posibles a emplear para adoptar la tecnología que aporte mayores ventajas en la elaboración de la solución. Estas tecnologías serán descritas en este epígrafe.

Se desarrollará una **Aplicación Web,** es un sistema informático que los usuarios utilizan accediendo a un servidor Web a través de Internet o de una intranet y sus ventajas más significativas son:

Compatibilidad 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 y desde cualquier lugar: las aplicaciones basadas en tecnologías web no necesitan ser descargadas, instaladas y configuradas. Además pueden ser accedidas desde cualquier computadora conectada a la red en donde se accede a la aplicación.

Menos requerimientos de hardware: este tipo de aplicación no consume (o consume muy poco) espacio en disco y también es mínimo el consumo de memoria RAM en comparación con los programas instalados localmente. Tampoco es necesario disponer de computadoras con poderosos procesadores ya que la mayor parte del trabajo se realiza en el servidor en donde reside la aplicación.

Seguridad en los datos: los datos se alojan en servidores con sistemas de almacenamiento altamente fiables y se ven libres de problemas que comúnmente sufren los ordenadores de usuarios comunes como virus y roturas de disco.

1.6 Técnicas de programación

1.6.1 Programación estructurada

Este tipo de programación está basada en una secuencia de instrucciones modificando los datos globales en el transcurso de todo el programa. Donde los saltos y el fin de programa no siguen ninguna estructura. Los saltos pueden apuntar a cualquier punto del código, lo que ocasiona que el algoritmo termine siendo un ovillo indescifrable (Alvarez, 2006).

1.6.2 Programación procedimental

La programación procedimental es un tipo de programación estructurada en donde el código se divide en porciones llamadas "procedimientos" o "funciones" (Alvarez, 2006).

1.6.3 Programación modular

La programación modular está basada en la técnica de diseño descendente, consiste en dividir el problema original en diversos sub-problemas que se pueden resolver por separado, para después recomponer los resultados y obtener la solución al problema (Alvarez, 2006).

1.6.4 Programación orientada a objetos

La Programación Orientada a Objetos (POO u OOP según siglas en inglés) es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulo más fáciles de escribir, mantener y reutilizar. De esta forma, un objeto contiene toda la información, (los denominados atributos) que permite definirlo e identificarlo frente a otros objetos (Alvarez, 2006).

1.6.5 Programación orientada a aspectos

La Programación orientada a aspectos (POA o AOP según siglas en inglés) es un paradigma de programación relativamente reciente cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos. Gracias a la AOP se pueden encapsular los

diferentes conceptos que componen una aplicación en entidades bien definidas, eliminando las dependencias entre cada uno de los módulos. De esta forma se consigue razonar mejor sobre los conceptos, se elimina la dispersión del código y las implementaciones resultan más comprensibles, adaptables y reusables (Alvarez, 2006).

1.7 Lenguajes y plataformas de desarrollo

1.7.1 Lenguajes

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una computadora. 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.

1.7.2 Lenguajes del lado del servidor

Se les clasifica así a los lenguajes de programación en la tecnología cliente servidor que se ejecutan del lado del servidor y de los que los cuales los usuarios solo obtienen el beneficio del procesamiento de la información.

1.7.2.1 Lenguaje PHP

PHP Hypertext Pre-processor (Hipertexto Pre-processor) es software libre, puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, soporta la mayoría de servidores Web de hoy en día y ofrece soporte para unos 20 gestores de bases de datos. Su característica de ser software libre trae como consecuencia que implique menos costes y servidores más baratos que otras alternativas. Es además muy rápido, contiene una biblioteca nativa de funciones sumamente amplia e incluida, no requiere definición de tipos de variables ni manejo detallado del bajo nivel, presenta mejoras de rendimiento, es un lenguaje multiplataforma que funciona en todas las plataformas que soporten Apache.

No es un lenguaje de marcas y la meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas.PHP también soporta el uso de otros servicios que usen protocolos como IMAP, SNMP, NNTP, POP3, HTTP y derivados. También se pueden abrir sockets de red directos(raw sockets) e interactuar con otros protocolos (web, 2001).

Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas Web dinámicas: Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader), analizar código XML. Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.

1.7.2.2 Lenguaje C#

Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes (entre ellos Delphi) (URRIELLUnet, 2007).

Ventajas:

- Su código se puede tratar íntegramente como un objeto. Ahorramos 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 se derivan del uso de este lenguaje de programación son que en primer lugar se tiene que conseguir una versión reciente de Visual Studio .NET, por otra parte se tiene que tener algunos

requerimientos mínimos del sistema para poder trabajar adecuadamente tales como contar con Windows NT 4 o superior, tener alrededor de 4 gigas de espacio libre para la pura instalación.

1.7.2.3 Lenguaje Java

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 (Guanajuato, 2007).

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995.

Ventajas:

- Un lenguaje multiplataforma potente. Se puede descargar de forma gratuita. lo que más destaque 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.
- La integración con la red es asombrosa ya que sus posibilidades son muy altas.

Desventajas:

- El compilador es su principal desventaja pues sabes decir si un programa al compilarlo tiene errores pero no sabe decir con precisión el lugar donde esta el error. La Maquina virtual Java consume muchos recursos, así que debemos tener un ordenador con muy buenas propiedades sino notaremos mucho la recarga de memoria.

1.7.3 Lenguajes del lado del cliente

Un lenguaje del lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. Pero la página no se verá 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.

1.7.3.1 HTML

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

HTML también es usado para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (HtmlCastellano, 2009).

1.7.3.1 XML

Sigla en inglés de Extensible Markup Language (lenguaje de marcas ampliable), 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 no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable (Wikilibros, 2007).

1.7.3.4 JavaScript:

JavaScript es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de Herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas Web. Para interactuar con una página Web se provee al lenguaje

JavaScript de una implementación del DOM. JavaScript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página Web y en programas más grandes, orientados a objetos mucho más complejos. Con JavaScript se puede crear diferentes efectos e interactuar con los usuarios. JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros (territoriopc, 2001).

1.7.3.5 Tecnología AJAX:

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 tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting lenguaje) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML (AbartiaTeam, 2006).

1.7.4 Control de versiones

Una versión, revisión o edición de un producto, es el estado en el se encuentra en un momento dado en su desarrollo o modificación. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. Un sistema de control de versiones debe proporcionar 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. Es posible duplicar la última versión o cualquier versión almacenada en el historial.

1.7.4.1 Subversión

Subversión es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversión es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

1.8.5 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.

1.8.5.1 PostgreSQL:

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

1.9 Herramientas de desarrollo

1.9.1 Zend Studio para Eclipse

Zend Studio para Eclipse Combina la probada tecnología y desarrolladores de PHP de Eclipse Tools (PDT) proyecto para crear el más poderoso del mundo IDE para el desarrollo de ricas aplicaciones Web. Zend Studio para Eclipse está diseñado para profesionales que necesitan los desarrolladores de PHP para apoyar el ciclo de vida de toda la aplicación PHP, y quieren tomar ventaja de la sofisticación y la extensibilidad del marco de Eclipse y de los ecosistemas (Almada, 2008). Esta herramienta presenta entre otras las siguientes características:

- Código refactorización.
- Nueva generación de código del archivo y magos.
- Código de Cobertura.
- PHP Unit pruebas de apoyo.
- Mejora con PHP Editor avanzado de formato, las nuevas listas de tareas y problemas de vista.
- Mejora de soporte JavaScript.
- Mejora de apoyo, incluyendo HTML, Código de Folding, Drag & Drop y componentes más.
- Mejora de control de versiones con el apoyo de historia local.
- Mejora de depuración y perfilado con Sendero Cartografía.
- Mejora de apoyo con Zend marco nuevo marco del proyecto, plantillas de código, opinión y más MVC.
- Acceso al ecosistema de plug-ins de Eclipse.
- Apoyar el desarrollo de múltiples idiomas.
- Zend Studio 5.5 Herramientas de Migración.
- Mecanismo de actualización automática.

1.9.2 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 (Traduce, 2008).

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.

1.9.3 Mozilla Firefox

Mozilla Firefox es el nuevo e innovador navegador open source del que todo el mundo está hablando. Firefox ha sido creado por el proyecto Mozilla, un esfuerzo open source sin ánimo de lucro que incluye a miles de voluntarios alrededor del mundo. La misión del proyecto Mozilla es preservar la elección y la innovación en Internet. El apoyo organizativo del proyecto Mozilla es proporcionado por Mozilla Foundation (en los Estados Unidos de América), Mozilla Europe y Mozilla Japón (Mozilla, 2009).

¿Por qué Firefox y no Internet Explorer?

Por nombrar algunas de las posibilidades que ofrece Firefox y que no ofrece IE, están:

Es Software libre.

En Firefox no existen la cantidad de bugs que posee el catastrófico IE, inmediatamente se encuentra un bug en el producto es notificado al Proyecto Mozilla para que sea reparado el problema.

Navegación por tabs: Esta es una de las principales características que tiene Firefox.

También existen excelentes extensiones y skins de fácil instalación, estos mejoran la usabilidad y el aspecto del navegador, cosa que no se logra en IE el cual siempre permanece con los mismos colores.

1.10 Conclusiones

En este capítulo se han introducido conceptos indispensables para la comprensión. Es fácil comprender que la manera en que hoy día se desarrolla este proceso no es la más factible pues toda la información esta en soporte duro y es difícil a la hora de gestionar la información.. La no existencia de un software que contenga las funcionalidades que se requieren para gestionar la Contabilidad de Costo, hacen necesaria la elaboración de un sistema informático que su principal objetivo sea cumplir con las funcionalidades requeridas.

CAPITULO 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

2.1 Introducción

En este capítulo se hizo una profunda valoración del diseño propuesto por el analista del sistema exponiendo las principales ventajas que ofreció la obtención del mismo. Se pretende mostrar como está aprovechada la arquitectura y las posibilidades que proporciona el framework y las librerías utilizadas en la programación de aplicaciones web en el producto que se esta tratando, con el objetivo de facilitar la comprensión del funcionamiento del componente Nomenclador y Configuración.

2.2 Valoración del diseño propuesto por el analista

La obtención del diseño propuesto por el analista, resultó de gran importancia, pues permitió una mejor comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y tecnologías de interfaz de usuario. También posibilitó crear una entrada apropiada y un punto de partida para las actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.

El diseño propuesto fue creado siguiendo patrones, que d*e* manera general constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, basados en la experiencia y que se ha demostrado que funcionan, permitiendo llevar a cabo la implementación clara y limpia del modulo bajo patrones como los GRASP.

Los patrones GRASP son aplicados a las clases definidas en el diseño, distribuyendo responsabilidades entre las mismas de forma tal que no existan muchas relaciones, que no sea sobrecargada de métodos una clase en específico, pudiendo acomodarlos en otras, entre otras mejoras que brinda el uso de este grupo de patrones. 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 Frameworks

Zend Frameworks es un frameworks open source, que esta diseñado para php 5 y buenas capacidades de ampliación. Presenta entre otras las siguientes características (Leopoldo, 2005):

- 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.
- Proporciona los componentes que forma la infraestructura del patrón MVC.

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 que mantiene un máximo de flexibilidad sin requerir la duplicación del código innecesario. También permite 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 (Doctrine, 2008).

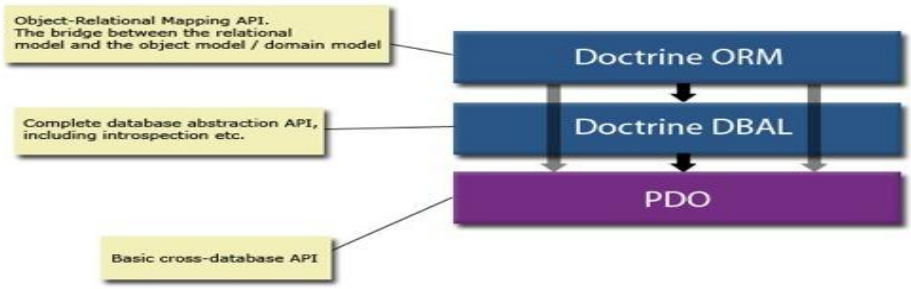

Figura 1 **Doctrine ORM.**

2.3.3 Extjs

Es un framework JavaScript del lado del cliente para el desarrollo de aplicaciones Web. Tiene un sistema dual de licencia: Comercial y Open Source. Este framework puede correr en cualquier plataforma que pueda procesar POST y devolver datos estructurados (PHP, Java, .NET y algunas otras) en tiempo de ejecución carga y crea todos los objetos HTML a través del uso intenso de DOM. Los datos son obtenidos mediante mensajes AJAX a través de XML y/o JSON (Corzo, 2009).

Funcionalidades:

Dispone de un conjunto de componentes (widgets) para incluir dentro de una aplicación web, como:

• Cuadros y áreas de texto.
• Campos para fechas.
• Campos numéricos.
• Combos.
• Radiobuttons y 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.
• Sliders.

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

• Cuadros de diálogo.
• 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 trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

- 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.

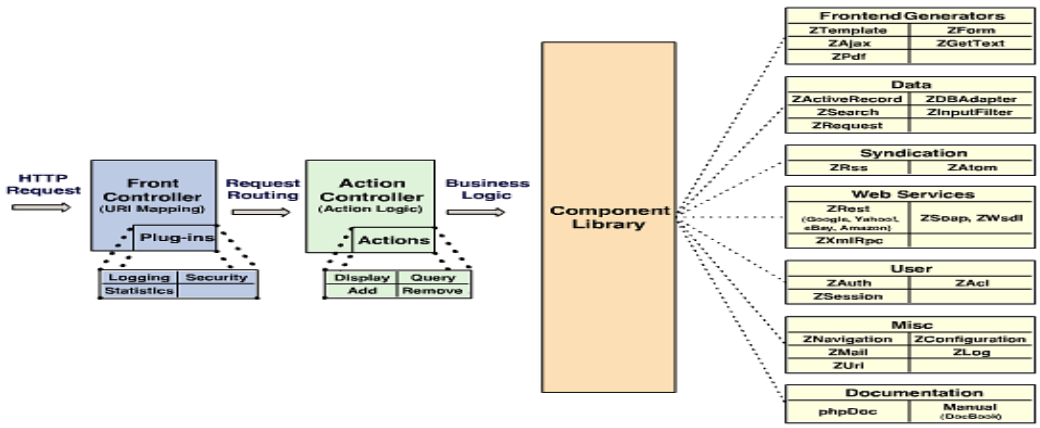

Figura 2 **Funcionamiento del Zend Framework.**

2.4.1 Estilo Basado en Capas

El modelo (n−capas) de informática distribuida ha emergido como la arquitectura predominante para la construcción de aplicaciones multiplataforma en la mayor parte de las empresas. Como tecnología, las

arquitecturas de n−capas proporcionan una gran cantidad de beneficios para las empresas que necesitan soluciones flexibles y fiables para resolver complejos problemas inmersos en cambios constantes.

Ventajas del modelo

- 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 nueva funcionalidad)
- 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.

2.4.2 Modelo-Vista –Controlador

MVC es un patrón de diseño de arquitectura de software usado principalmente en aplicación que maneja gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de los conceptos para que el desarrollo este estructurado de una mejor manera, facilitando la programación en diferentes capas de manera paralela e independiente. MVC sugiere la separación del software en 3 estratos

Modelo:
Es la representación de la información que maneja la aplicación. El modelo en si son los datos puros que puestos en un contexto del sistema proveen de información al usuario o a la aplicación misma.

Vista:
Es la representación del modelo en forma grafica disponible para la interacción con el usuario. En el caso de una aplicación web la "Vista " es la pagina HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.

Controlador:

Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el Modelo en caso de ser necesario.

2.4.3 Estilo híbrido basado en Capas y MVC

Descripción general del estilo:

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:

- 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 nueva funcionalidad)
- 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.

2.5 Estrategias de integración

En los componentes se aplica la integración vertical que consiste en el flujo de 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 encontramos entre la vista y el controlador, el que está entre la el controlador y el modelo, el que vincula el modelo con el framework doctrine y el que se encuentra entre el doctrine y la base de datos. Todo el código dentro un mismo componente utiliza llamadas a métodos o eventos de forma directa. La comunicación entre diferentes módulos y componentes se realiza mediantes llamadas a la inversión de control (Inversion of Control en inglés, IoC). El IoC especifica respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que otro modulo o componente lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir.

Cada componente tiene su registro de los datos de los módulos en un fichero xml que será mapeado por el framework para el funcionamiento del mismo, dicho fichero tiene por nombre IoC y registra las funcionalidades que ofrecen los métodos de las clases control de los componentes del sistema. La base de datos es accedida de forma directa mediante controladoras y los componentes rehusados son integrados mediantes interfaces sencillas.

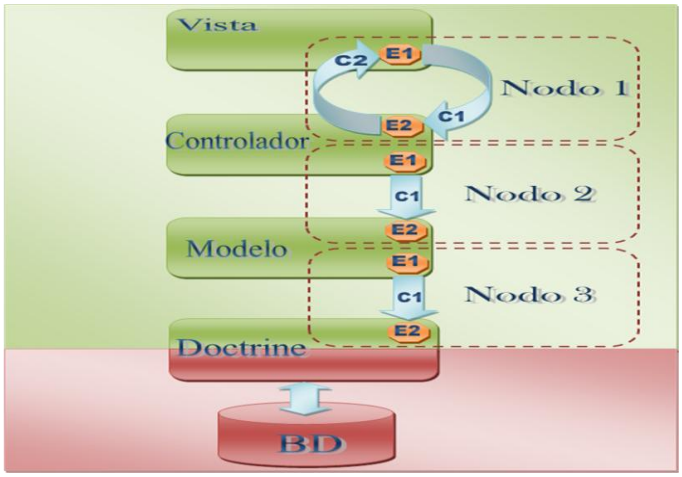

Figura 3 **Nodos involucrados en la integración.**

2.6 Estándares de código

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Los estándares de codificación permiten una mejor integración entre las líneas de producción y establece pautas que conlleven a lograr un código más legible y reutilizable, de tal forma que se pueda aumenta su mantenibilidad a lo largo del tiempo.

2.6.1 Notación Camello

Se usa para denotar variables, parámetros y funciones. En esta notación, si el identificador es una palabra simple se escribe todo con minúscula, pero si es compuesta, la primera letra de todas las palabras que vienen a continuación de la primera comienza con mayúscula. La primera palabra debe ser un sustantivo que describa claramente al identificador y las otras palabras a continuación deberán ser adjetivos.

Ejemplo de función: insertarMonedaAction.
Ejemplo de variable: arrMoneda.

2.6.2 Notación Pascal

Se usa para denotar clases. En esta notación, si el identificador es simple, el primer carácter se escribe con mayúscula y el resto con minúscula, si el identificador es una palabra compuesta, la segunda palabra debe comenzar con mayúscula también.

Ejemplo: GestionarUsuario.

2.7 Descripción de los algoritmos no triviales a implementar

Uno de los procesos fundamentales que ocurren en la Contabilidad de Costo, es la asociación de cuentas de gastos con centros de costo y elementos de gastos ya sean estos patrimoniales o presupuestados. Este proceso se inicia luego de seleccionar un tipo de cuenta y luego una cuenta de gasto para ser asociada, si dicha cuenta posee gastos por subelementos podrá ser asociada a centros y a elementos que no posean otros centros o elementos que dependan de él; de no poseer esta cuenta gastos por subelementos solo podrá ser asociada a centros de costo.

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. **(Ver Anexo I)**

```
public function asociarCuentasCentroElementoAction()
  {
    $cuentas =json_decode( stripslashes ($this->_request->getPost('idcuentas'))); 1
    $centros =json_decode( stripslashes ($this->_request->getPost('idcentrocostos'))); 1
```

```
$elementos =json_decode( stripslashes ( $this->_request->getPost('idelementogasto'))); 1
if($this->_request->getPost('tipocuenta')==0)2
{
   foreach ($cuentas as $cuenta) 3
      foreach ($centros as $centro) 4
         foreach ($elementos as $elemento) { 5
            $asociacion = new ConfCuentascentrosegpat(); 6
            $asociacion->idnomcuenta=$cuenta[0]; 6
            $asociacion->idcentrocostopatrimonial=$centro; 6
            $asociacion->idelementogastopatrimonial=$elemento; 6
            OpasociacionModel::asociar($asociacion); 6
         }6
   echo('{success: 1}');7
}7
else if($this->_request->getPost('tipocuenta')==1)8
{
   foreach ($cuentas as $cuenta) {9
      foreach ($centros as $centro) {10
         foreach ($elementos as $elemento) {11
            $asociacion =new ConfCuentascentrosegpres();12
            $asociacion->idnomcuenta=$cuenta[0];12
            $asociacion->idcentrocostopresupuestario=$centro;12
            $asociacion->idelementogastopresupuestario=$elemento;12
            OpasociacionModel::asociar($asociacion);12
         }12
    echo('{success: 1}');13
}13
```

```
    else 14
    echo('{success: 1}');15
}16
```

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.

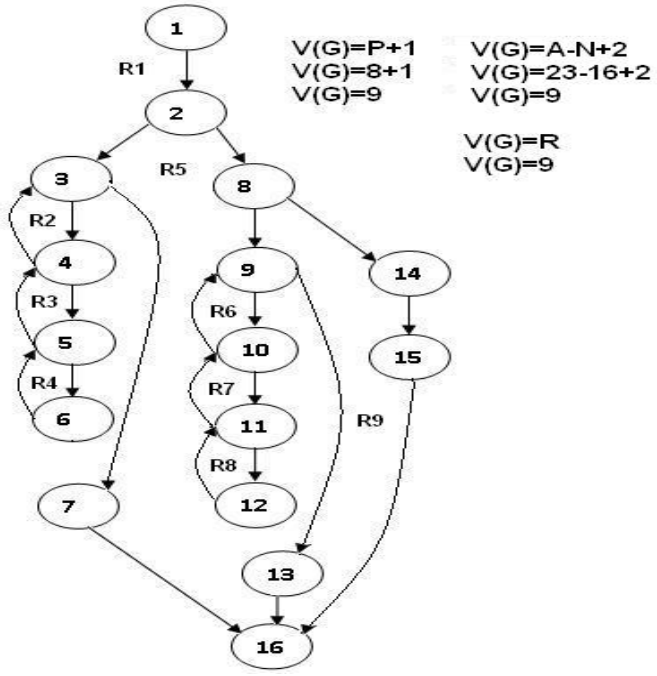

Figura 4 **Grafo de flujo del algoritmo.**

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) = (23 – 16) + 2
V (G) = 9
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) = 8 +1
V (G) =9
V (G) = R
Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del calculo.
V (G) = 9
Realizado el cálculo por las 3 vías necesarias se llega a la conclusión que el algoritmo presentado anteriormente tiene un complejidad ciclomática de 9 que da la visión de que existen a lo sumo nueve caminos lógicos por donde recorrer el algoritmo.

2.8 Estructura de datos a usar para implementar el algoritmo

En programación, una estructura de datos es una forma de organizar un conjunto de datos elementales con el objetivo de facilitar su manipulación. Un dato elemental es la mínima información que se tiene en un sistema. Cada estructura ofrece ventajas y desventajas en relación a la simplicidad y eficiencia para la realización de cada operación. De esta forma, la elección de la estructura de datos apropiada para cada problema depende de factores como la frecuencia y el orden en que se realiza cada operación sobre los datos.
El array es una estructura de datos que permite almacenar información en distintas casillas. En lugar de las variables normales que son capaces de almacenar un dato, el array es capaz de almacenar una

colección de datos, al disponer de varias casillas donde almacenar un dato distinto en cada una de ellas. Los arrays en PHP se utilizan muy habitualmente y se dispone de una serie de funciones muy amplia para trabajar con ellos. Los tipos de array que existen en PHP son los de índices numéricos y arrays con índices asociativos. Además poseen una serie de funciones creadas para ordenarlos por orden alfabético directo o inverso, por claves, contar el número de elementos que componen el array además de poder movernos por dentro de él hacia delante o atrás; en fin muchas son las funciones propuestas por PHP para el tratamiento de arrays.

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: OpasociacionccgpController	
Tipo de clase: Controladora	
Atributo	**Tipo**
Idestructura	
Para cada responsabilidad:	
Nombre:	Descripción:
CargarareresponsabilidadAction	Carga las áreas de responsabilidad
CargarcentroscostosAction	Asocia las áreas de responsabilidad con los centros de costo
CargarasociacionAction	Carga los centros de costo pertenecientes a una estructura dada
CentrosasociadosAction	Devuelve los centros asociados a un área
EliminarasociacionAction	Elimina un asociación
opnomencladorAction	Muestra la vista asociada al controlador

Tabla 1 Clase controladora "**OpasociacionccgpController**"

Nombre: OpasociacionController	
Tipo de clase: Controladora	
Atributo	**Tipo**
Idestructura	
Para cada responsabilidad:	
Nombre:	**Descripción:**
MostrarreporteAction	Muestra un reporte dado un ID
MostrarreportesusuarioAction	Muestra reportes a nivel de módulos
CargarcuentasAction	Carga las cuentas existentes en la BD según la especificación
CargarcentroscostosAction	Devuelve los centros de costos asociados al tipo de cuenta(Patrimonial o presupuestada)
CargarelementogastoAction	Devuelve los elementos de gasto asociados al tipo de cuenta(Patrimonial o presupuestada)
CargarasociacionAction	Devuelve Grupo de Cuentas, Centros de Costos y Elementos de Gastos asociados a una cuenta
CargarasociacionelementoAction	Carga un elemento asociado según un tipo de cuenta
AsociarcuentascentroAction	Asocia cuentas con centros de costo
EliminarasociacionAction	Elimina una asociación según un ID
EliminarcentroselementosAction	Elimina centros y elementos
ModificarasociacionAction	Modifica una asociación según un ID
OpasociacionAction	Muestra la vista asociada al controlador

Tabla 2 Clase controladora "**OpasociacionController**"

Nombre: OpcuentasgastosController	
Tipo de clase: Controladora	
Atributo	**Tipo**
Model	
idestructura	
Para cada responsabilidad:	

Nombre:	Descripción:
OpcuentasgastosAction	Si existen cuentas de gastos muestra la vista asociada
MostrarreporteAction	Muestra un reporte según un ID dado
MostrarreportesusuarioAction	Muestra reportes a nivel de módulos
CargarcuentasgastosAction	Devuelve las cuentas de gastos según una estructura
CargarcuentasAction	Cargar las cuentas patrimoniales y presupuestadas que no han sido definidas de gastos
AdicionarcuentasAction	Adiciona las cuentas que son de gasto
ActualizarcuentasgastosAction	Actualiza las cuentas de gastos
EliminarcuentagastoAction	Elimina las cuentas de gastos
CargarcuentasinicioanoAction	Carga cuentas de gastos que sean de inicio de año
BuscarcuentasAction	Busca las cuentas dado un patrón y estructura

Tabla 3 Clase controladora "**OpcuentasgastosController**"

Nombre: OpnomencladorController	
Tipo de clase: Controladora	
Atributo	**Tipo**
nomenclador	
Idestructura	
Para cada responsabilidad:	
Nombre:	Descripción:
OpnomencladorAction	Si existen formatos muestra la vista asociada
MostrarreporteAction	Muestra un reporte según un ID
CargarparteformatoAction	Muestra reportes a nivel de módulos
CargarformatosAction	Crea un lista con las partes del formato
CargarnomencladoresAction	Devuelve los nomencladores que existen en la BD
PadreAction	Busca los padres
BuscarPadre_blAction	Busca los padres que coinciden con el criterio de búsqueda
ImportarAction	Importa un nomenclador
InserterAction	Insertar una denominación en los nomencladores

ObtenercodigoAction	Valida que no haya un nomenclador con el mismo código en el mismo nivel
ModificarAction	Actualiza una denominación en los nomencladores y todos sus hijos
ListaHijosAction	Devuelve la lista de hijos de un nodo pasado por parámetro
EliminarAction	Eliminar una denominación en los nomencladores
VerificaEliminarHijosAction	Verifica que no haya asociaciones
ExporterAction	Exporta en un fichero todos los elementos asociados al nomenclador seleccionado
FormarcaposAction	Construye una lista de campos diferentes para los nomencladores
ArbolNomencladorAction	Carga el árbol de Centros de costo y Elementos de gasto dado el nomenclador

Tabla 4 Clase controladora "**OpnomencladorController**"

2.9.2 Clases Auxiliares

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

Nombre: OpasociacionModel	
Tipo de clase: Auxiliar	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
Asociar($asociacion)	Guarda una asociación
Eliminarasociacion($espresupuestada,$cuentas,$subelemento)	Elimina una asociación dado una cuenta
Eliminarcentroselementos($espresupuestada,$subelemento,$cuenta,$centros, $elementos=array())	Elimina Centros de costos y Elementos de gasto

Tabla 5 Clase Auxiliar "**OpasociacionModel**"

Nombre: OpcuentasgastosModel	
Tipo de clase: Auxiliar	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
Adicionarcuentas($array)	Adiciona las cuentas dadas
Actualizarcuentasgastos($id,$version)	Actualiza las cuentas según un ID
Eliminarcuentagasto($array)	Elimina las cuentas de gasto dadas
Cargarcuentainicioano	Carga las cuentas que son Inicio de Año

Tipo de clase: Auxiliar	
Nombre: OpasociacionareaModel	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
Asociararea($idarea, $idcentro, $tipo)	Asocia un área con un centro
Eliminararea($idarea,$tipo)	Elimina las áreas asociadas a un centro
Eliminarasociacion($idarea,$centros,$tipo)	Elimina las asociaciones de los centros dados

Tabla 6 Clase Auxiliar "**OpcuentasgastosModel**"

Tabla 7 Clase Auxiliar "**OpcuentasgastosModel**"

Tipo de clase: Auxiliar	
Nombre: NomencladoresService	
Atributo	**Tipo**
nomenclador	
estructura	
Para cada responsabilidad:	

Nombre:	Descripción:
ObtenerElementosGastos($tabla, $node, $est,$id)	Devuelve los elementos de gastos del nomenclador
ObtenerElementosNomenclador($tabla,$est, $pkelementos,$id)	Devuelve los elementos especificados de un nomenclador
ObtenerElementosNom($tabla,$est,$pkelementos,$id)	Devuelve los elementos especificados de un nomenclador, especializado para Secuencia de Traspasos
ObtenerCentrosCostos($tabla, $node, $est,$id)	Devuelve los centros de costos del nomenclador
obtenercuentaspatrimoniales($estructurax)	Obtiene las cuentas patrimoniales dada una estructura
ObtenerCuentasGasto($filtro,$inicio,$limite,$idestructura)	

Tabla 8 Clase Auxiliar "**NomencladoresService**"

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: ConfCuentasgasto	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
Cargarcuentasgastos($node,$idestructura)	Carga las cuentas de gasto indicadas
Cuentagastopadres	Busca las cuentas de gasto que son padres
Cargarcuentasnogastos	Carga las cuentas que no son de gasto
Cargarcuentainicioano	Carga las cuentas que son Inicio de Año
Cantidadcuentasgasto($filtro,$inicio,$limite)	Cantidad de cuentas que son de gasto
Cargarcuentas($filtro,$inicio,$limite,$idestructura)	Carga todas las cuentas que son indicadas
Buscarcuenta ($descripcion)	Busca una cuenta según las una descripción dada
Devolverdescripcioncuenta($idcuenta)	Devuelve la descripción de una cuenta según un ID
Devolvercuentaspatrimonial($estructura)	Devuelve cuentas patrimoniales según una estructura

Devolvercentroscuenta($idcuenta)	Devuelve los centros asociados a una cuenta dada
Cargarcentrosorigen($idcuenta)	Devuelve centros origen según un ID
Devolverdatoscuenta($idcuenta)	Devuelve una cuenta según un ID
VerificaCentro($idcentro)	Verifica si un centro dado esta asociado
VerificaElemento($idelemento)	Verifica si un elemento dado esta asociado
Devolverelementosscuenta($idcuenta, $idcentrocosto)	Devuelve los elementos asociados a una cuenta y un centro dado
Devolverelementoscentro($idcentrocosto)	Devuelve los elementos asociados a una centro dado
Devolvercentroelementocuenta($idcuenta)	Devuelve un lista con todos los centros y los elementos asociados a una cuenta de gasto entrada
Esinicioano($idcuenta)	Busca si una cuenta es inicio de año

Tabla 9 Clase Entidad "**ConfCuentasgasto**"

Nombre: ConfCuentascentrospresupuestario	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
Mostrarcentros($id)	Busca todos los centros presupuestados dado un ID

Tabla 10 Clase Entidad "**ConfCuentascentrospresupuestario**"

Nombre: ConfCuentascentrospatrimonial	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
Mostrarcentros($id)	Busca todos los centros patrimoniales dado un ID

Tabla 11 Clase Entidad "**ConfCuentascentrospatrimonial**"

Nombre: ConfCuentascentrosegpres	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
Mostrarelementosgastos($idcentro,$idcuenta)	Busca todos los elementos gastos presupuestado asociado a una cuenta y un centro dado
Mostrarcentros($id)	Busca los centros presupuestado asociados a una cuenta dada

Tabla 12 Clase Entidad "**ConfCuentascentrosegpres**"

Nombre: ConfCuentascentrosegpat	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
Mostrarelementosgastos($idcentro,$idcuenta)	Busca todos los elementos gastos patrimoniales asociado a una cuenta y un centro dado
Mostrarcentros($id)	Busca los centros patrimoniales asociados a una cuenta dada

Tabla 13 Clase Entidad "**ConfCuentascentrosegpat**"

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

Nombre:	Descripción:
Devolverasociacion	Devuelve todas las asociaciones con centros de costo presupuestario existentes en la BD
Devolverasociado($idarea)	Devuelve una asociación dado un ID de área
Devolvercentrosasociados($idarea)	Devuelve una centro costo presupuestario asociado a un área dada

Tabla 14 Clase Entidad "**ConfAreascentrospres**"

Nombre: ConfAreascentrospat	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
Devolverasociacion	Devuelve todas las asociaciones con centros de costo patrimonial existentes en la BD
Devolverasociado($idarea)	Devuelve una asociación dado un ID de área
Devolvercentrosasociados($idarea)	Devuelve una centro costo patrimonial asociado a un área dada

Tabla 15 Clase Entidad "**ConfAreascentrospat**"

Nombre: BaseConfAreascentrospres	
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 "**BaseConfAreascentrospres**"

Nombre: BaseConfCuentascentrosegpres	
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 "**BaseConfCuentascentrosegpres**"

Nombre: BaseConfCuentascentrospresupuestario	
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 "**BaseConfCuentascentrospresupuestario**"

Nombre: BaseConfAreascentrospat	
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 "**BaseConfAreascentrospat**"

Nombre: BaseConfCuentascentrosegpat	
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 "**BaseConfCuentascentrosegpat**"

Nombre: BaseConfCuentascentrospatrimonial	
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 "**BaseConfCuentascentrospatrimonial**"

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

Tabla 22 Clase Entidad "**BaseConfCuentasgasto**"

CAPITULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción

En el desarrollo del software, las posibilidades de error son innumerables. Pueden darse por una mala especificación de los requisitos funcionales, uso indebido de las estructuras de datos, errores al enlazar módulos, etc. El desarrollo del software ha de ir acompañado de alguna actividad que garantice la calidad; la prueba es un elemento crítico para la garantía de calidad del software.

La prueba y validación de los resultados no es un proceso que se realiza una vez desarrollado el software, sino que debe efectuarse en cada una de las etapas de desarrollo. Es fundamental medir la cobertura de las pruebas, es decir, la determinación de cuando se han realizado las suficientes pruebas. Si se siguen encontrando errores cada vez que se procesa el programa, las pruebas deben continuar.

Durante el mantenimiento debe de existir documentación de pruebas que incluya casos de prueba y resultados esperados. Si se producen modificaciones en el programa, habrá que probar de nuevo todas las partes del programa afectadas por las modificaciones.

Por lo dicho anteriormente, se desarrolló este capítulo, con el objetivo de especificar las pruebas que serán hechas al software y describir todo lo encontrado en las mismas.

3.2 Descripción de las pruebas

Los “test de unidades” son pruebas llevadas a cabo por los implementadores sobre las unidades mínimas desarrolladas por ellos, estas unidades pueden ser clases, métodos, propiedades, componentes, etc. Estas unidades se prueban separadas unas de otras y básicamente se hacen durante la implementación del software. Para realizar estas pruebas es necesario establecer una serie de reglas que sirven como objetivos de estas pruebas:

- La prueba es un proceso de ejecución de un programa con la intención de descubrir errores.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Los “test de unidades” son orientados casi siempre a las pruebas de “caja blanca” aunque para realizar uno de estos test es necesario probar el flujo de datos desde la interfaz del componente. Si los datos no entran correctamente, todas las demás pruebas no tienen sentido. Estas pruebas son aplicadas a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que estos funcionen como se espera.

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 lógica 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”.

Otra forma de probar el código es mediante las pruebas de “caja negra”. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas 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 del procedimiento al cual se le va a aplicar la prueba. A continuación se enumera las sentencias de código del procedimiento. **(Ver Anexo II)**

```
public function actualizarCuentasGastosAction()
  {
      $this->model = new OpcuentasgastosModel(); 1
       $padres=ConfCuentasgasto::cargarcuentasgastos($this->_request->getPost('idpadre'),$this-
>idestructura);1
                  $saldoinicio=true; 1
                   if($padres) 2
                        foreach ($padres as $k => $v) 3
```

```
                {
                        if ($v['saldo']==1) 4
                        $saldoinicio=false; 5
                } 6
               if($saldoinicio) 7
               {
               if($this->model->actualizarcuentasgastos($this->_request->getPost('idcuenta'),$this-
>_request->getPost('version'))) 8
                        echo '{success: 1}';  9
                        else
              echo '{success: 2}'; 10
               }
               else
                echo '{success: 3}';  11
   } 12
```

Seguidamente se construye el grafo de flujo asociado al código anterior.

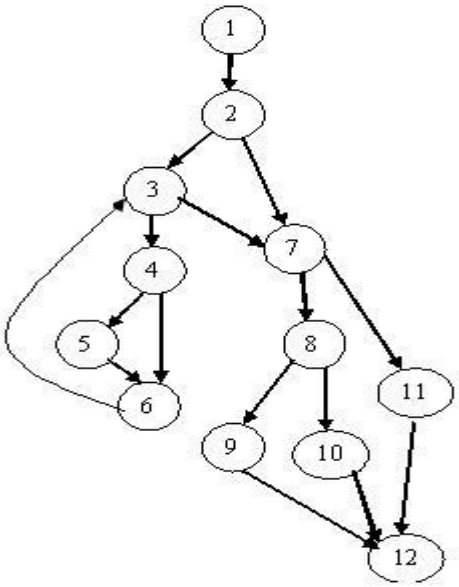

Figura 5 **Grafo de flujo del algoritmo.**

Luego de haber construido el grafo, se realiza el cálculo de la complejidad ciclomática, mediante las tres fórmulas descritas en el capítulo 2, epígrafe 2.5.1 “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 5 se obtienen los siguientes resultados:

Calculando mediante la fórmula 1:

V (G) = (16 -12) + 2

V (G) = 6.

Calculando mediante la fórmula 2:

V (G) = 5 + 1

V (G) = 6.

Calculando mediante la fórmula 3:

V(G) = 6.

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de 4, lo que significa que existen cuatro posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo. En estas representaciones se subrayan los elementos de cada camino que los hacen independientes a los demás.

Camino básico #1:

1 – 2 – 3 – 4 – 5 – 6 – 3 – 7 – 8 – 9 - 12.

Camino básico #2:

1 – 2 – 3 – 4 – 6 – 3 – 7 – 8 – 10 - 12.

Camino básico #3:
1 – 2 – 3 – 7 – 8 – 9 - 12
Camino básico #4:
1 – 2 – 3 – 7 – 8 – 10 - 12
Camino básico #5:
1 – 2 – 3 – 7 – 11 – 12
Camino básico #6:
1 – 2 – 3 – 11 – 12

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este 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.

Caso de prueba para el camino básico # 1.
Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El id del padre, id de la estructura, id de la cuenta y la versión son un numero entero.
Condición de ejecución: El id de padre será igual 90415, el id de la estructura será igual 100000024, el id de la cuenta será igual a 90419 y la versión será igual a 0.
Entrada: idpadre=90415, idestructura=100000024, idcuenta=90419, versión=0.
Resultados esperados: Se espera que la cuenta de gasto sea actualizada correctamente.
La cuenta fue actualizada correctamente.

Caso de prueba para el camino básico # 2.

Descripción: La misma descripción que el caso de prueba para el camino básico # 1.
Condición de ejecución: El id de padre será igual 90415, el id de la estructura será igual 100000024, el id de la cuenta será igual a 90420 y la versión será igual a 3.
Entrada: idpadre=90415, idestructura=100000024, idcuenta=90420, versión=3.
Resultados esperados: Se espera que la cuenta de gasto sea actualizada correctamente.
La cuenta no fue actualizada porque no era saldo inicio de año y versión era incorrecta.

Caso de prueba para el camino básico # 3.
Descripción: La misma descripción que el caso de prueba para el camino básico # 1.
Condición de ejecución: El id de padre será igual 90415, el id de la estructura será igual 100000024, el id de la cuenta será igual a 90415 y la versión será igual a 0.
Entrada: idpadre=90415, idestructura=100000024, idcuenta=90415, versión=0.
Resultados esperados: Se espera que la cuenta de gasto sea actualizada correctamente.
La cuenta fue actualizada correctamente.

Caso de prueba para el camino básico # 4.
Descripción: La misma descripción que el caso de prueba para el camino básico # 1.
Condición de ejecución: El id de padre será igual 90425, el id de la estructura será igual 100000024, el id de la cuenta será igual a 90425 y la versión será igual a 0.
Entrada: idpadre=90425, idestructura=100000024, idcuenta=90425, versión=0.
Resultados esperados: Se espera que la cuenta de gasto sea actualizada correctamente.
La cuenta no se actualizada correctamente no fue encontrada en la base de datos.

Caso de prueba para el camino básico # 5.
Descripción: La misma descripción que el caso de prueba para el camino básico # 1.
Condición de ejecución: El id de padre será igual 90428, el id de la estructura será igual 100000024, el id de la cuenta será igual a 90428 y la versión será igual a 0.
Entrada: idpadre=90428, idestructura=100000024, idcuenta=90428, versión=0.
Resultados esperados: Se espera que la cuenta de gasto sea actualizada correctamente.
La cuenta no se actualizada correctamente debido a que no era de proceso.

Caso de prueba para el camino básico # 6.

Descripción: La misma descripción que el caso de prueba para el camino básico # 1.

Condición de ejecución: El id de padre será igual 90424, el id de la estructura será igual 100000024, el id de la cuenta será igual a 90421 y la versión será igual a 0.

Entrada: idpadre=90424, idestructura=100000024, idcuenta=90421, versión=0.

Resultados esperados: Se espera que la cuenta de gasto sea actualizada correctamente.

La cuenta no se actualizada correctamente debido a que no existían cuentas padres en la base de datos de contabilidad.

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

3.4 Aplicación de pruebas de caja negra

Para la aplicación de este tipo de prueba se usará el requisito Asociación entre Cuenta de Gasto Patrimonial o Presupuestada, Centro de Costo o Grupo Presupuestario y Elemento del Gasto u Objeto del Gasto. El objetivo de este requisito es relacionar las cuentas de gastos ya sean patrimoniales o presupuestadas que fueron definidas con antelación, con centros de costo ya sean patrimoniales o presupuestados y con elementos de gasto ya sean patrimoniales o presupuestados.

3.4.1 Requisito a probar

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario

Asociación entre Cuenta de Gasto Patrimonial o Presupuestada, Centro de Costo o Grupo Presupuestario y Elemento del Gasto u Objeto del Gasto.	Se selecciona el tipo de Cuenta de Gasto, automáticamente se cargan en la interfaz todas las cuentas de gastos del tipo seleccionado que han sido definidas hasta ese momento. Luego se selecciona la cuenta que se quiere asociar siempre que no esté subrayada, pues en caso de estarlo indica que ya ha sido asociada. Luego se presiona el botón **Definir**, inmediatamente en dependencia del tipo de cuenta seleccionado se mostrarán en otra interfaz las Entidades con todos los Centros de Costo o Grupos Presupuestarios y en caso de que la cuenta tenga Análisis por subelemento o partida presupuestaria se mostrarán también las entidades con los Elementos del Gasto u Objetos del Gasto que están definidos hasta ese momento. Se selecciona de las entidades existentes el(os) Centro(s) de Costo o Grupo Presupuestario y si la cuenta tiene Análisis por subelemento o partida	Asociar correctamente la Cuenta de Gasto Patrimonial o Presupuestada, el(os) Centro(s) de Costo Patrimonial o Presupuestario y el Elemento(s) del Gasto u Objeto(s) del Gasto.	-Se selecciona una cuenta de las que no aparecen resaltadas. Se presiona el botón **Definir**. - Se selecciona de las Entidades el(os) Centro(s) de Costo o Grupo(s) Presupuestario que se quiere(n) asociar. -Se debe tener en cuenta si la cuenta seleccionada tiene Análisis por subelemento o partida presupuestaria, en caso de tener seleccionar el(s) Elemento(s) de Gasto u Objeto(s) del Gasto que se quiere asociar. -Se selecciona de la tabla el(os) Centro(s) de Costo o Grupo(s) Presupuestario y el(os) Elemento(s) del Gasto u Objeto(s) del Gasto. -Se presiona el botón **Asociar.** Se muestra un mensaje para verificar si el usuario realmente quiere realizar la operación. -Se presiona el botón **Aceptar.** -Se muestra un mensaje notificando que la operación se realizó correctamente. -Se presiona el botón **Aceptar**. - Se notifica que se realizo la operación correctamente.

		Eliminar la asociación de una cuenta ya asociada.	-Se selecciona de las cuentas que están subrayadas, la que se quiere eliminar. -Se presiona el botón **Eliminar.** -Se muestra un mensaje para confirmar que se desea realizar la operación. -Se presiona el botón **Aceptar**. -Se muestra un mensaje notificando que la operación se realizó correctamente. -Se presiona el botón **Aceptar**.

		Presionar el botón **Eliminar** sin haber seleccionado una cuenta.	-Se presiona el botón **Eliminar.** -Se muestra un mensaje notificando que debe seleccionar una cuenta para poder eliminar su asociación. – Se presiona el botón **Aceptar**.
		Seleccionar una Cuenta de Gasto que ya ha sido asociada con anterioridad.	-Se selecciona una cuenta de las que aparecen resaltadas. -Se presiona el botón **Definir**. -Se muestra un mensaje notificando que la cuenta seleccionada no puede ser asociada nuevamente. -Se presiona el botón **Aceptar.**
		Modificar la asociación realizada a una cuenta.	-Se selecciona de las cuentas que están subrayadas, la que se quiere modificar. -Se presiona el botón **Modificar**. -Se selecciona de las Entidades el(os) Centro(s) de Costo o Grupo(s) Presupuestario que se quiere(n) asociar. -Se debe tener en cuenta si la cuenta seleccionada tiene análisis por subelemento o partida presupuestaria, en caso de tener seleccionar el(s) Elemento(s) de Gasto u Objeto(s) del Gasto que se quiere asociar. -Se selecciona de la tabla la cuenta, el(os) Centro(s) de Costo o Grupo(s) Presupuestario y el(os) Elemento(s) del Gasto u Objeto(s) del Gasto.

			-Se presiona el botón **Asociar.** -Se muestra un mensaje para verificar si el usuario realmente quiere realizar la operación. -Se presiona el botón **Aceptar.** -Se muestra un mensaje notificando que la operación se realizó correctamente. -Se presiona el botón **Aceptar**.
		Presionar el botón **Modificar** sin haber seleccionado la cuenta que se quiere modificar.	-Se presiona el botón **Modificar**. -Se muestra un mensaje notificando que debe seleccionar una Cuenta de Gasto para poder modificar su asociación. -Se presiona el botón **Aceptar.**
		Mostrar una asociación ya realizada.	-Se selecciona la Cuenta de Gasto asociada. -Se presiona el botón **Ver.** -Se presiona el botón **Cerrar** o **Imprimir**.
		Presionar el botón **Ver** sin haber seleccionado una Cuenta de Gasto.	-Se presiona el botón **Ver**. -Se muestra un mensaje notificando que debe seleccionar una Cuenta de Gasto para poder ver su asociación. -Se presiona el botón **Aceptar.**
		Presionar el botón **Ver** seleccionando una cuenta que no se le haya definido asociación	-Se presiona el botón **Ver**. -Se muestra un mensaje notificando que debe seleccionar una cuenta que se le haya definido una asociación. -Se presiona el botón **Aceptar.**

		Presionar el botón **Definir** sin haber seleccionado una cuenta.	-Se presiona el botón **Definir**. -Se muestra un mensaje indicando que debe seleccionar una Cuenta de Gasto. -Se presiona el botón **Aceptar**.
		Seleccionar para la modificación una cuenta que no tiene asociación.	Se selecciona una cuenta de las que no aparecen resaltadas. -Se presiona el botón **Modificar.** -Se muestra un mensaje indicando que se debe seleccionar una cuenta que tenga asociación para poder hacer la modificación. -Se presiona el botón **Aceptar.**
		Presionar el botón **Asociar** sin haber seleccionado el Elemento del Gasto.	-Se selecciona una cuenta de las que no aparecen resaltadas. -Se presionar el botón **Definir**. -Se selecciona de las Entidades el(os) Centro(s) de Costo o Grupo(s) Presupuestario que se quiere(n) asociar. -Se presiona el botón **Asociar**. -Se muestra un mensaje indicando que debe seleccionar el elemento que desea asociar. -Se presiona el botón **Aceptar**.
		Presionar el botón **Asociar** sin haber seleccionado el Centro de Costo.	-Se selecciona una cuenta de las que no aparecen resaltadas. -Se presionar el botón **Definir**. -Se selecciona de las Entidades el(os) Centro(s) de Costo o Grupo(s) Presupuestario que se quiere(n) asociar.

			-Se presiona el botón **Asociar**. -Se muestra un mensaje indicando que debe seleccionar el elemento que desea asociar. -Se presiona el botón **Aceptar**.

3.4.2 Descripción de las variables

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1	Cuenta de gasto patrimonial	RadioButton	SI	Se debe seleccionar como tipo la Cuenta de Gastos Patrimonial se quiere realizar la asociación.
2	Cuentas de gasto presupuestado	RadioButton	SI	Se debe seleccionar como tipo la Cuenta de Gastos Gubernamental se quiere realizar la asociación.
3	Cuentas de gasto	CheckBox	NO	Las Cuentas de Gastos estarán previamente definidas y se debe seleccionar en este caso la que se quiera asociar.
4	Entidades(Centros de Costo)	CheckBox	SI	Los Centros de Costo estarán previamente definidos y se debe seleccionar en este caso la que se quiera asociar.
5	Entidades(Elementos del Gasto)	CheckBox	SI	Los Elementos del Gasto estarán previamente definidos y se debe seleccionar en este caso la que se quiera asociar.

3.5 Validación del modelo de diseño propuesto

El desarrollo de software es algo muy complejo y la medida de su calidad real no es automatizable. Por esta razón la aplicación de métricas de calidad para evaluar el diseño orientado a objeto posee gran importancia. A continuación presentaremos un estudio que brinda 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 (PRESSMAN, 1998).

Atributos de calidad que se abarcan:

1. Responsabilidad. Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

2. Complejidad del diseño. Consiste en la complejidad que posee una estructura de diseño de clases.

3. Complejidad de implementación. Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.

4. Reutilización. Consiste en el grado de reutilización de presente en una clase o estructura de clase, dentro de un diseño de software.

5. Acoplamiento. Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, esta 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.

7. Cantidad de pruebas. Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, modulo, clase, conjunto de clases, etc.) diseñado.

8. Nivel de Cohesión. Consiste en el grado de especialización de las clases concebidas para modelar un dominio o concepto específico.

Tamaño operacional de clase (TOC): Está dado por el número de métodos asignados 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 23 **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 24 **Relaciones entre clases (RC).**

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

Ver instrumentos y tabla de resultados en (**Anexo III** Instrumento de medición de la métrica Tamaño operacional de clase (TOC)).

Figura 6 **Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.**

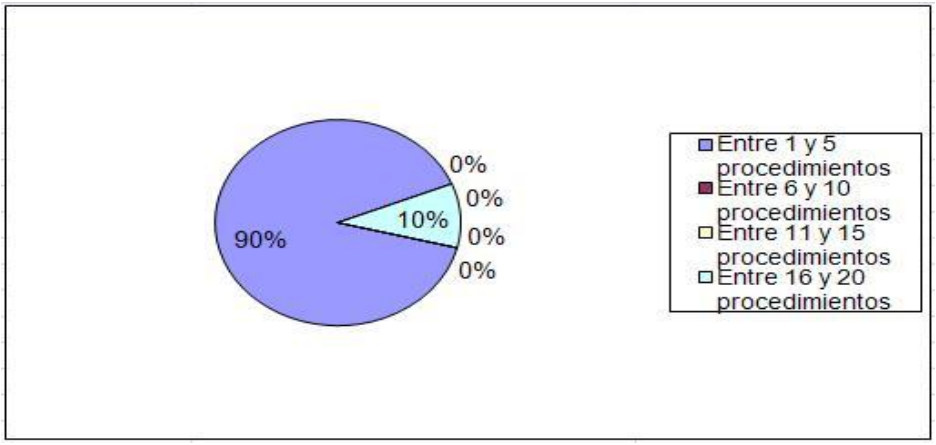

Figura 7 **Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.**

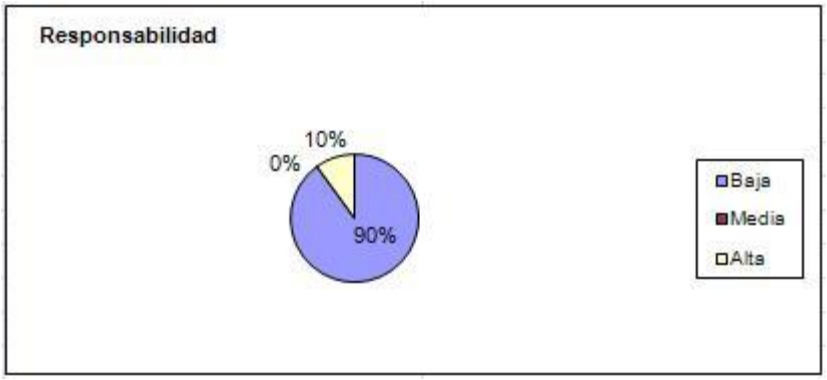

Figura 8 **Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.**

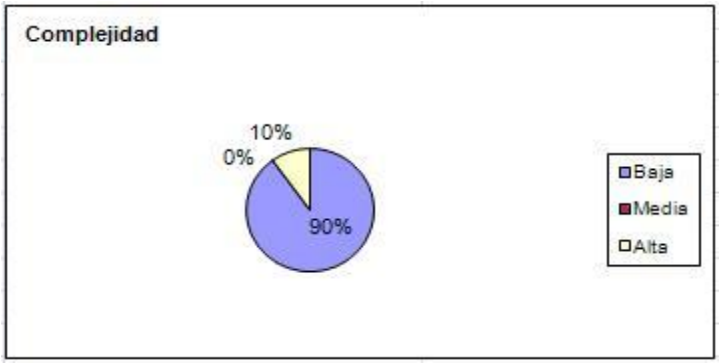

Figura 9 **Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.**

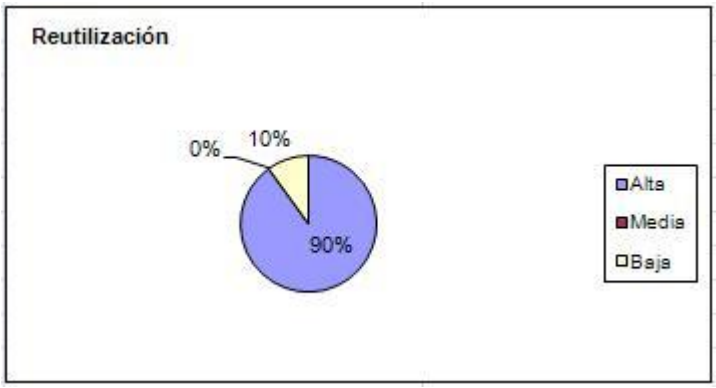

Figura 10 **Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.**

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 componente Nomenclador y Configuración tienen una calidad buena pudiéndose observar que el 90% de las clases posee menos cantidad de operaciones que la media registrada en las mediciones. Además el 90% de las clases posee evaluaciones positivas en los atributos (Responsabilidad, Complejidad de Implementación y Reutilización).

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

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

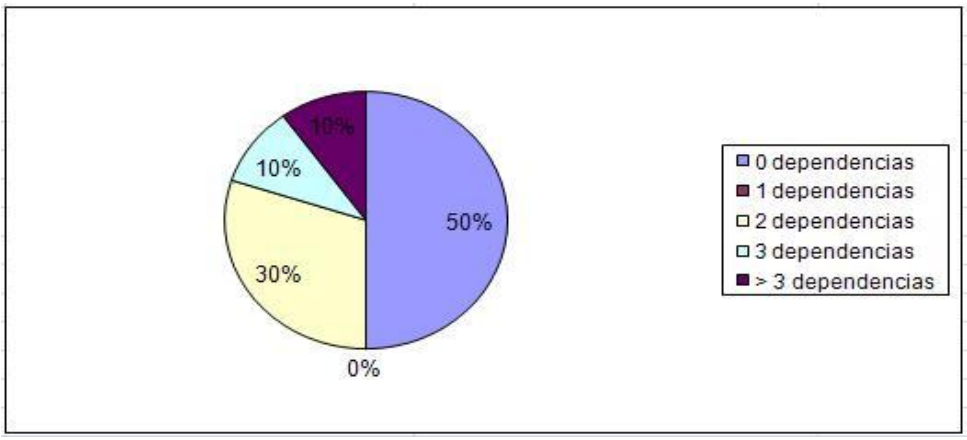

Figura 11 **Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.**

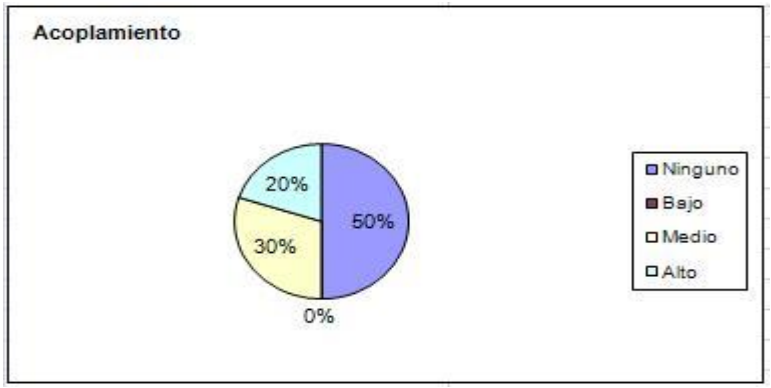

Figura 12 **Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.**

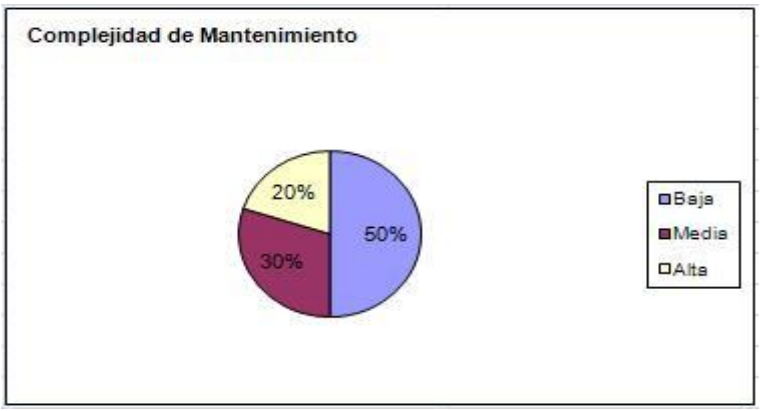

Figura 13 **Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.**

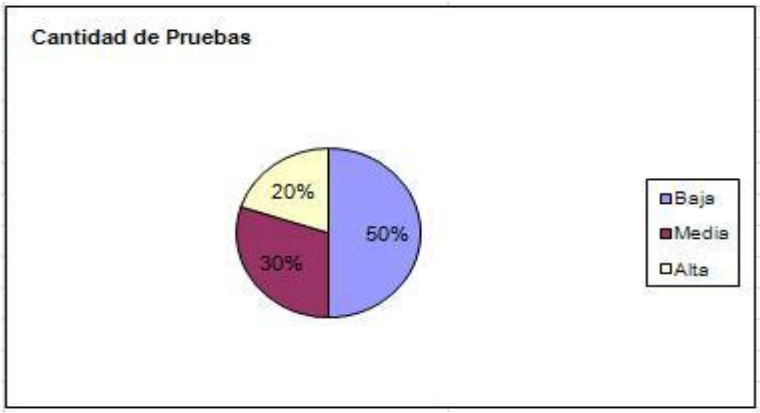

Figura 14 **Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.**

Figura 15 **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 de el componente Nomenclador y Configuración tienen una calidad buena pudiéndose observar que el 90% de las clases posee menos de 3 dependencias de otras clases. Además el 50% de las clases no poseen acoplamiento con otras y el 40% 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 80% de las clases.

A manera de resumen se han tabulado los resultados obtenidos en la siguiente tabla:

<=0	M
0 < a >= 0.5	R
0.5 < a >= 1	B

Tabla 26 **Umbrales.**

	TOC	PH	RC	ND	NOR	Total
Complejidad Implementación	1	-	-	-	-	1
Reutilización	1	-	1	-	-	1
Acoplamiento		-	1	-	-	1
Complejidad Mantenimiento		-	0.5	-	-	0.5
Cantidad de Pruebas		-	0.5	-	-	0.5
Responsabilidad	1	-	-	-	-	1
Total		-	-	-	-	1

Tabla 27 **Resumen de los resultados.**

3.6 Conclusiones

Una de las últimas fases del ciclo de vida antes de entregar un programa para su explotación, es la fase de pruebas. Esta fase del desarrollo de un software es la que mayor cantidad de tiempo y de esfuerzo requiere, se estima que la mitad del esfuerzo de desarrollo de un programa tanto en tiempo como en gastos se invierte en esta. Esta fase añade valor al producto, todos los programas poseen errores y la fase de pruebas los descubre, siendo este el valor que le añade. Mientras más errores se encuentren al software en esta fase mejor. Una prueba del software es un conjunto de actividades que se lleva a cabo sistemáticamente, que puede planificarse por adelantado y ejecutar una vez construido el código para la revisión final de las especificaciones, del diseño y de la codificación del software.

En este capítulo se realiza un estudio de los principales niveles, métodos y tipo de prueba que se llevan a cabo durante el ciclo de vida del software. Se describen los test de caja negra y los valores utilizados, así como, la ejecución de los mismos y los resultados obtenidos.

CONCLUSIONES

Con la implementación del componente nomenclador y configuración se logró una mayor eficiencia, confiabilidad y rapidez en la toma de decisiones en la gestión de procesos de la Contabilidad de Costo en cualquier entidad.

El desarrollo de este componente constituye un aporte práctico muy importante, debido a que es una novedad tecnológica que marca un proceso de avance en el desarrollo del software interoperable dentro de la entidad que favorecerá el incremento en la eficiencia de la gestión de la información de los proceso.

Al finalizar el presente trabajo de diploma se dan por cumplidos los objetivos planteados en sus inicios pues se ha logrado una realización eficiente de los procesos involucrados, obteniendo la implementación de un componente en el que se aplican los resultados de la investigación llevada a cabo.

RECOMENDACIONES

- Realizar mejoras de diseño de interfaces de modo que su uso le sea más sencilla y amigable al usuario.
- Optimizar los algoritmos de métodos que manejen grandes volúmenes de datos.

BIBLIOGRAFÍA

AbartiaTeam. 2006. Abartia Team. [En línea] 2006. http://www.abartiateam.com/desarrollo-web/200602_uso-de-la-tecnologia-ajax-en-el-desarrollo-web.

Almada, Federico. 2008. techtear. [En línea] 22 de Enero de 2008. [Citado el: 20 de Enero de 2009.] http://www.techtear.com/2008/01/22/zend-studio-for-eclipse-desarrollo-profesional-en-php.

Alvarez, Sara. 2006. Desarrollo Web. [En línea] 18 de mayo de 2006. http://www.desarrolloweb.com/articulos/2477.php.

Cardoso, Pedro Bermejo. 2008. BWT SIME. [En línea] 2008. [Citado el: 11 de Diciembre de 2008.] http://www.betsime.disaic.cu/secciones/fin_ja_05.htm.

Corzo, Giancarlo. 2009. Desarrollo en Web. [En línea] 2009. http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

Doctrine. 2008. Doctrine. [En línea] 2008. [Citado el: 26 de Enero de 2009.] http://www.doctrine-project.org.

Guanajuato, León. 2007. [En línea] 2007. [Citado el: 20 de Enero de 2009.] http://www.cad.com.mx/historia_del_lenguaje_java.htm.

HtmlCastellano. 2009. HTML Castellano. [En línea] 2009. http://www.programacion.com/html/.

Leopoldo, Carlos. 2005. [En línea] 2005. [Citado el: 25 de Enero de 2009.] http://www.carlosleopoldo.com/post/zend-framework-una-introduccion.

linares, jose. 2008. wikipedia. [En línea] 2008. http://www.wikipwedia.com.

López, Carlos. 2008. Gestion Polis. [En línea] 2008. [Citado el: 12 de Diciembre de 2008.] http://www.gestiopolis.com/dirgp/fin/costos.htm.

Maestre, David. 2008. [En línea] 2008. [Citado el: 17 de Diciembre de 2008.] http://davidmaestre.com/2008/01/modulos-de-sap-r3.html.

Mozilla Firefox. 2009. Mozilla Firefox. [En línea] 2009. [Citado el: 19 de Enero de 2009.] http://www.getfirefox.es/firefox-features.

OpenBravo. 2007. OpenBravo. [En línea] 2007. [Citado el: 10 de Enero de 2009.] http://www.openbravo.com/es/product/erp/features.

OpenERP. 2007. openERP. [En línea] 2007. [Citado el: 10 de Enero de 2009.] http://www.openerpsite.com/?page_id=35.

PRESSMAN, R. S. 1998. *Ingenieria de software. Un enfoque practico.* 1998.

Rodríguez, Agnerys y José Alejandro. 2006. El eco del contador. [En línea] 2006. [Citado el: 14 de Enero de 2009.] http://elecodelcontador.blogspot.com/2008/10/el-versat-sarasola-sistema-cubano-de.html.

territoriopc. 2001. territoriopc. [En línea] 2001. http://www.territoriopc.com/javascript/tutorial_javascript_introduccion.php.

Traduce gratis. 2008. Traduce gratis. [En línea] 2008. [Citado el: 21 de Enero de 2009.] http://descargar.traducegratis.com/es_soft_v_iegjci/EMS-SQL-MANAGER-FOR-POSTGRESQL-LITE.htm.

URRIELLUnet. 2007. URRIELLUnet. [En línea] 2007. [Citado el: 20 de Enero de 2009.] http://urriellu.net/es/articles-software/csharp-advantages.html.

web, Desarrollo. 2001. Desarrollo web. [En línea] 2001. [Citado el: 20 de Enero de 2009.] http://www.desarrolloweb.com/articulos/392.php.

Wikilibros. 2007. Wikilibros. [En línea] 2007. http://es.wikibooks.org/wiki/Lenguaje_XHTML/Ventajas_del_XML.

ANEXOS

Anexo I. Asociación de Cuentas de Gastos a Centros de Costos y Elementos de Gasto.

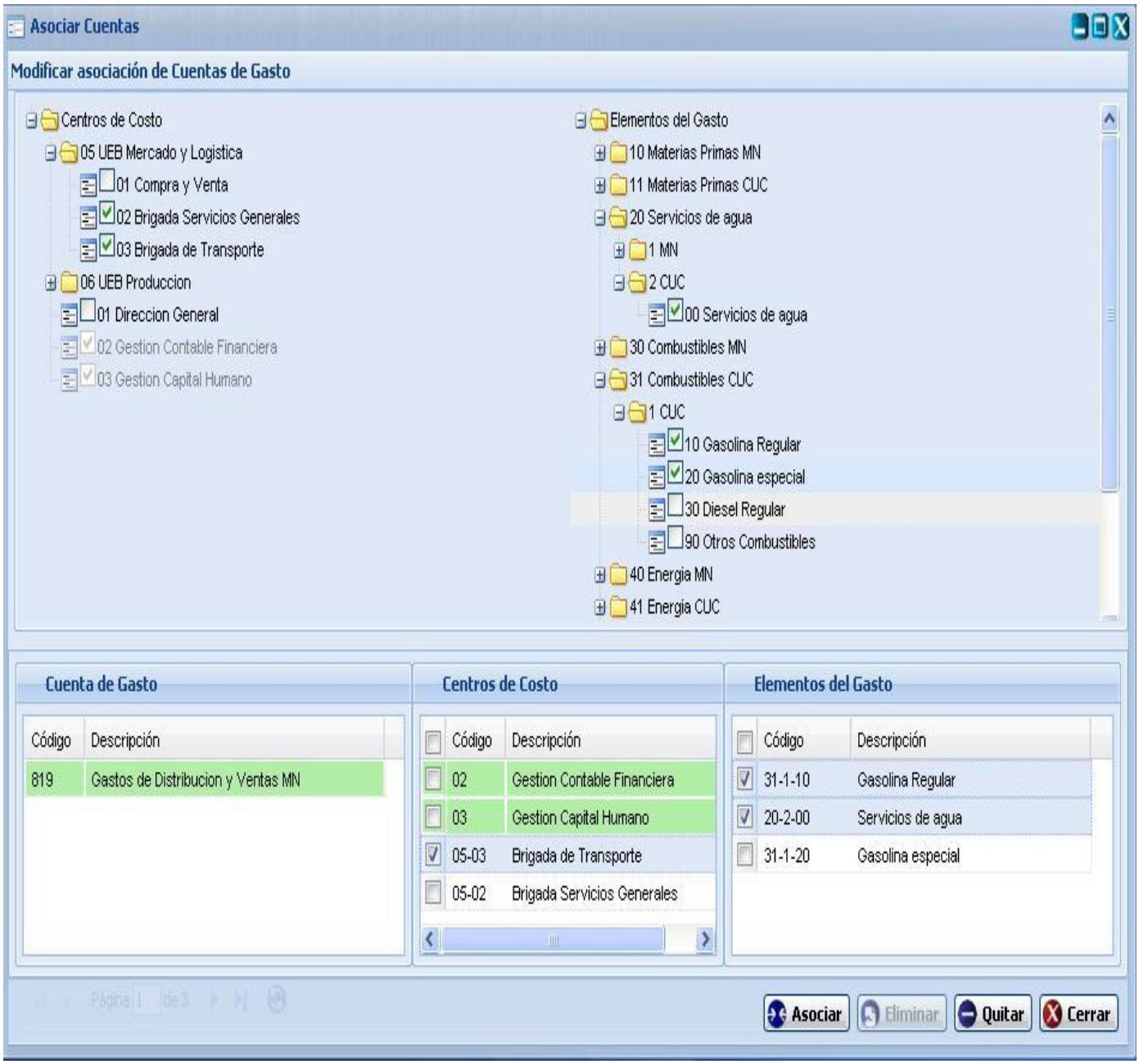

Figura 16 **Asociación de Cuentas de Gastos a Centros de Costos y Elementos de Gasto.**

Anexo II. Definir una Cuenta de Gasto como saldo inicio de año.

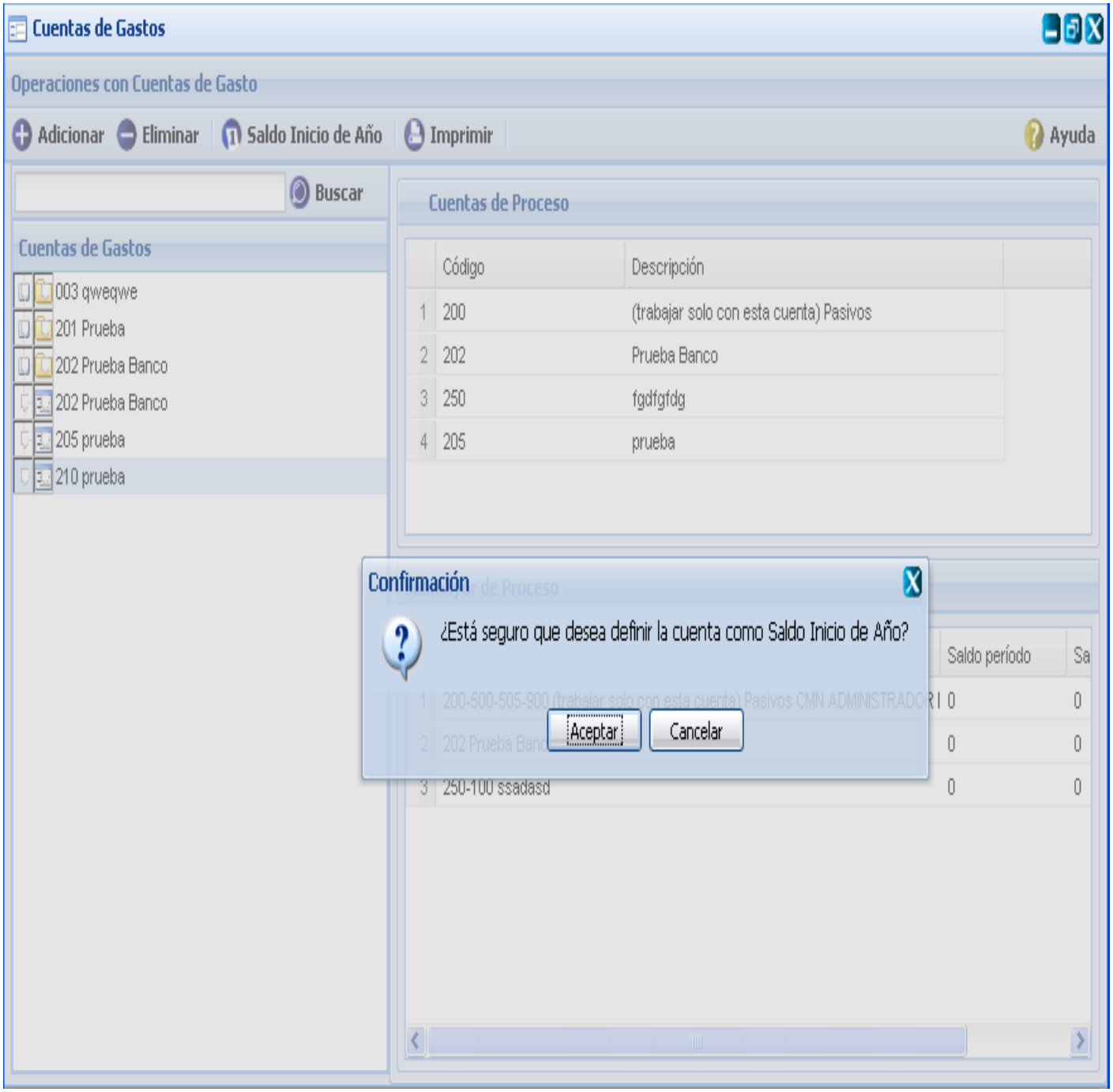

Figura 16 **Definir una Cuenta de Gasto como saldo inicio de año.**

Anexo III. 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.**

No	Subsistema	Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
1	Nomencladores y Configuración	OpasociacionModel	3	Baja	Baja	Alta
2	Nomencladores y Configuración	OpcuentasgastosModel	4	Baja	Baja	Alta
3	Nomencladores y Configuración	OpasociacionareaModel	4	Baja	Baja	Alta
4	Nomencladores y Configuración	NomencladoresService	4	Baja	Baja	Alta
5	Nomencladores y Configuración	ConfCuentasgasto	18	Alta	Alta	Baja
6	Nomencladores y Configuración	ConfCuentascentrospresupuestario	1	Baja	Baja	Alta
7	Nomencladores y Configuración	ConfCuentascentrospatrimonial	1	Baja	Baja	Alta
8	Nomencladores y Configuración	ConfCuentascentrosegpres	2	Baja	Baja	Alta
9	Nomencladores y Configuración	ConfCuentascentrosegpat	2	Baja	Baja	Alta
10	Nomencladores y Configuración	ConfAreascentrospres	3	Baja	Baja	Alta

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

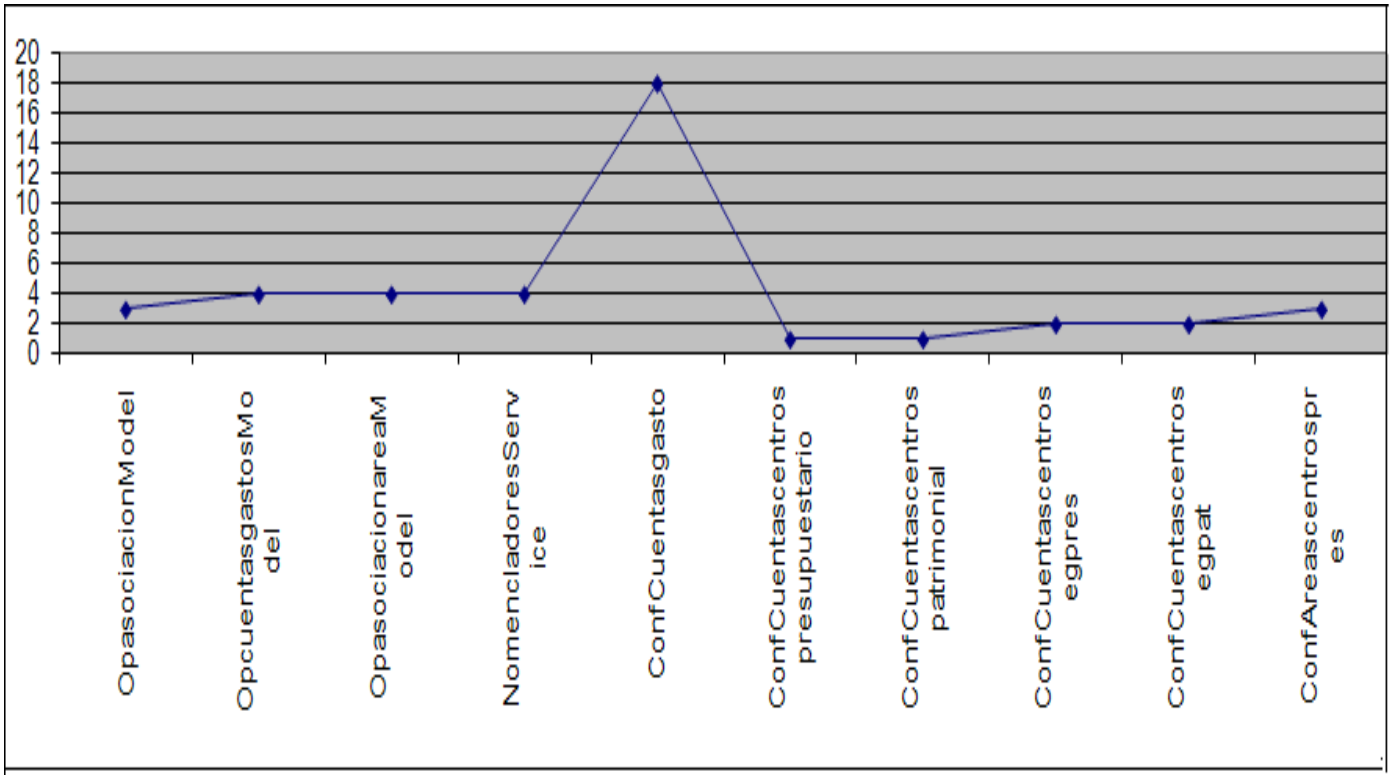

Figura 17 **Gráfica de los resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).**

Anexo IV. Instrumento de medición de la métrica Relaciones entre clases (RC)

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

	Categoria	Criterio
Complejidad Mant.	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.**

No	Subsistema	Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
1	Nomencladores y Configuración	OpasociacionModel	5	Alto	Alta	Baja	Alta
2	Nomencladores y Configuración	OpcuentasgastosModel	2	Medio	Media	Media	Media
3	Nomencladores y Configuración	OpasociacionareaModel	2	Medio	Media	Media	Media
4	Nomencladores y Configuración	NomencladoresService	3	Alto	Alta	Baja	Alta
5	Nomencladores y Configuración	ConfCuentasgasto	2	Medio	Media	Media	Media
6	Nomencladores y Configuración	ConfCuentascentrospresupuestario	0	Ninguno	Baja	Alta	Baja
7	Nomencladores y Configuración	ConfCuentascentrospatrimonial	0	Ninguno	Baja	Alta	Baja

8	Nomencladores y Configuración	ConfCuentascentrosegpres	0	Ninguno	Baja	Alta	Baja
9	Nomencladores y Configuración	ConfCuentascentrosegpat	0	Ninguno	Baja	Alta	Baja
10	Nomencladores y Configuración	ConfAreascentrospres	0	Ninguno	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).**

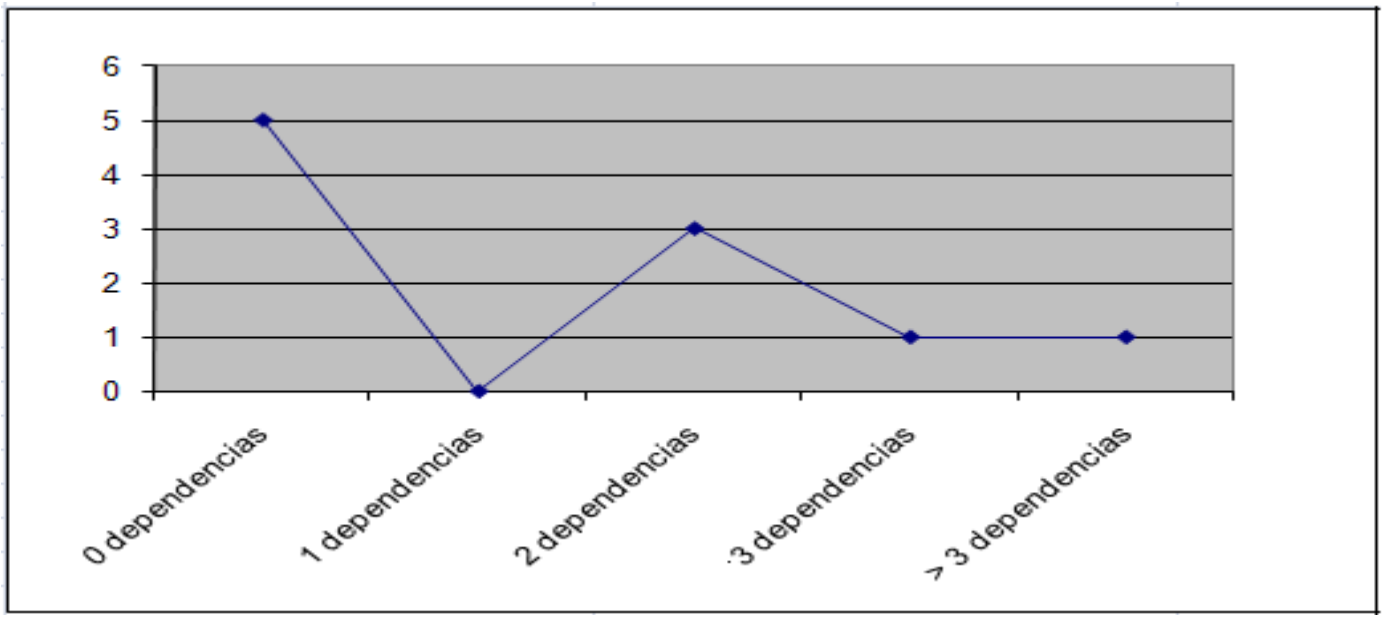

Figura 18 **Gráfica de los resultados de la evaluación de la métrica RC agrupados por la tendencia de los Valores.**

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.

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 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 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,

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 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, 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 del 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 Standard Generalized Markup Language 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 Rich Internet Applications (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.

GLOSARIO DE TÉRMINOS

1