Universidad de las Ciencias Informáticas

Facultad 3

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

"Desarrollo de las funcionalidades del proceso traspaso de gastos indirectos de Cedrux"

Autor:

Yusnely Montejo Vega.

Tutores:

MSc. Joisel Pérez Pérez.

Ing. Juan Alberto Caballero Portelles.

La Habana, Julio del 2013.

"Año 55 de la Revolución"

Declaración de autoría

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

Para que así conste se firma el presente a los ____ días del mes de _____ del año _____.

Yusnely Montejo Vega

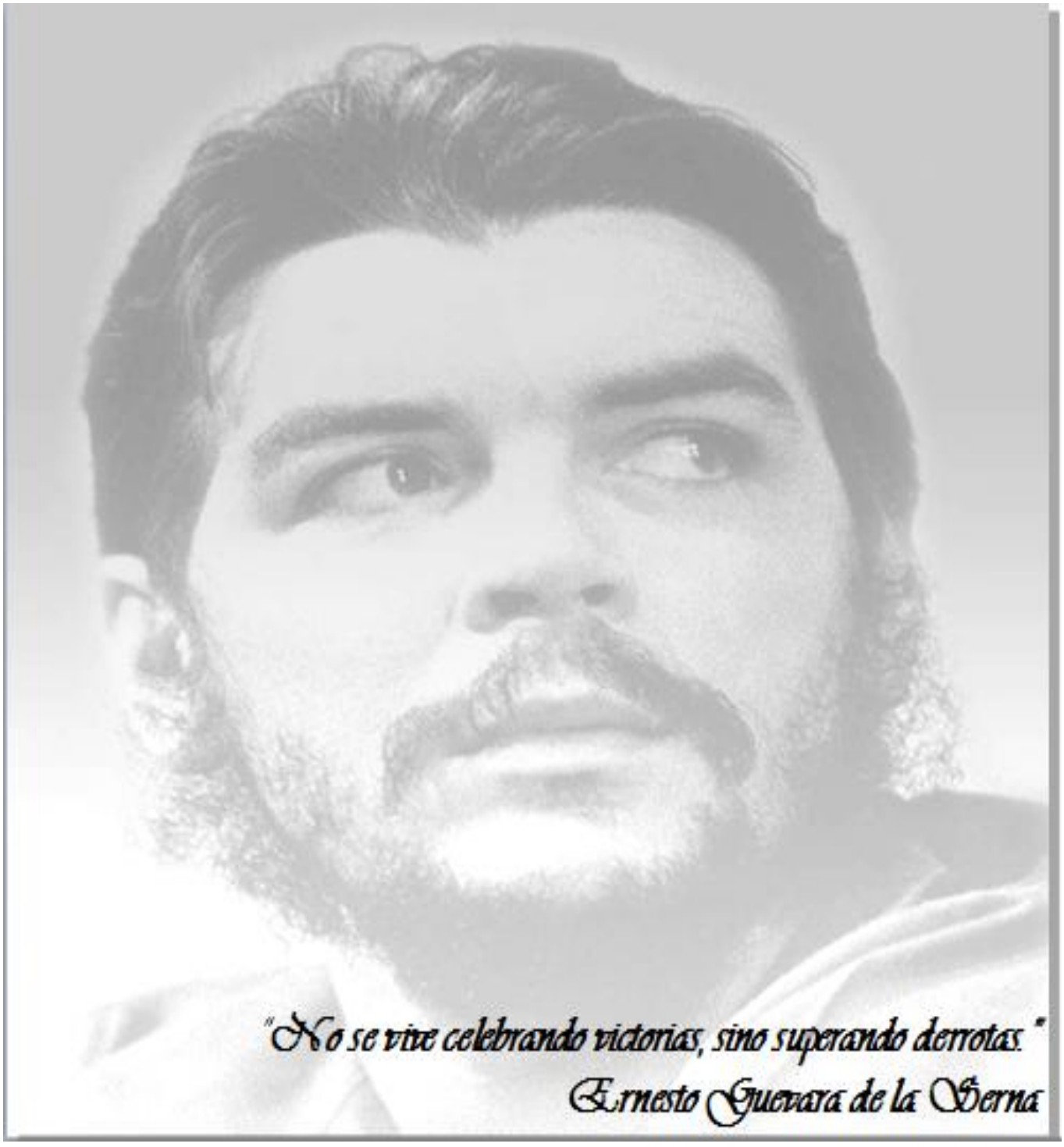
_____

Firma del Autor

_____ _____

MSc. Joisel Pérez Pérez **Ing. Juan Alberto Caballero Portelles**

Firma del Tutor Firma del Tutor


"No se vive celebrando victorias, sino superando derrotas."
Ernesto Guevara de la Serna

Dedicatoria

A mis padres….por confiar en mí y entregarme su infinito amor.

A mi novio….por darme las fuerzas para seguir adelante, por brindarme su apoyo y su amor incondicional.

A hermana… amiga fiel y sincera y por ser una razón para estar siempre alegre.

A toda mi familia por todo el apoyo y la confianza que siempre me han dado.

A a todo aquel que de una forma u otra aportó un granito de arena, que brindó su apoyo, su confianza y su esfuerzo para que este sueño se hiciese realidad.

Agradecimientos

A mis queridos padres María Eugenia y Joaquín, a quien les debo todo en la vida, por su dedicación, preocupación y amor, por depositar en mi toda su confianza, por estar siempre presente en los momentos buenos y malos, gracias ya que por su apoyo incondicional hoy soy una ingeniera y sé que están orgullosos de mí. Este triunfo también es de ustedes.

A mi querida hermanita Paita como le digo cariñosamente, gracias por estar presente cada vez que te he necesitado, gracias por tu apoyo, por ser quien eres e inspirarme confianza desde pequeña. Te quiero con la vida.

A mi novio Leonardo, mi rey, el gran amor de mi vida, por ayudarme en estos años de la carrera, por tu amor incondicional y por depositar en mí toda tu confianza, por estar siempre a mi lado y brindarme tu cariño y comprensión. Por ser mi motor impulsor y darme las fuerzas necesarias para seguir adelante en los momentos difíciles. Por darme tantas alegrías y compartir tu vida conmigo. Por ser mi guía y ejemplo a seguir, por tu preocupación, y dedicación a mí, por creer que soy una mejor persona y por lo importante que eres en mi vida. A ti te debo y dedico especialmente este sueño. Te amo mucho.

A mi primas Titi y Ani y a mis tías Api y Hai por ayudarme siempre, gracias por estar en todos los momentos de mi vida, las quiero mucho.

A mi suegros Silvia y Dago, por tratarme desde el primer momento como su hija, por acogerme en su casa y quererme. A ustedes también les dedico este logro.

A mi cuñado Reinier por ayudarme y estar en los momentos buenos y malos, por cuidar a mi hermanita y quererla mucho.

A Daliana, Lilian, Cielo,Celina, Mariana, Yesenia y a demás familiares que son como mi familia, gracias por todo y por preocuparse tanto por mí.

A mi familia, en especial a mis abuelos mami y abuelito como les decía cariñosamente, que aunque no están presente los llevo siempre en mi corazón, y a mis abuelos papi y abuelita, a mis tíos Maira, Guillermo, pipo, quienes siempre han estado presente en cada momento importante de mi vida brindándome su apoyo.

A mis amistades durante los cinco años en especial a Andrés, Zoima, Zoemi y Rodrigo, a todos muchas gracias por ayudarme y compartir conmigo estos años de universidad.

A Manuel y a Alberto por ayudarme en todos los momentos en que necesité de su ayuda y siempre darme el sí, muchas gracias a ustedes también.

A mis tutores Joisel y Juan Alberto por su ayuda y comprensión en la realización del trabajo. A todas las personas que he conocido en la universidad y a todas aquellas que propiciaron que pudiera cumplir este gran sueño. Gracias a todos.

Cedrux
SISTEMA INTEGRAL
DE GESTIÓN

Resumen

La Universidad de las Ciencias Informáticas (UCI) tiene la tarea de desarrollar un sistema denominado Cedrux. En el mismo, dentro del subsistema Costos y Procesos, el traspaso de los gastos no es más que traspasar los saldos de las cuentas de gastos indirectos a las cuentas de gastos directos.

Dicho proceso proporciona una vez ejecutado los comprobantes de operaciones, los cuales son utilizados en diferentes operaciones contables. Además complementa el ciclo de planificación, registro y control de los costos de producción. El sistema actualmente no permite realizar el traspaso de los gastos de un centro de costo a otro centro u otros centros. Además existen errores en los cálculos para efectuar el prorrateo[1] de acuerdo a los criterios de distribución por Subelementos y Bases combinadas. Debido a su importancia se necesita una herramienta capaz de realizar dicho proceso cumpliendo con las necesidades del cliente, siendo el objetivo del presente trabajo de diploma por las razones expuestas anteriormente.

Para un mejor entendimiento del proceso traspaso de gastos en el presente trabajo se realiza un estudio del arte acerca de algunas soluciones existentes a nivel nacional e internacional. Para la implementación del proceso fue utilizado el marco de trabajo Sauxe, como lenguaje de modelado UML y Visual Paradigm como herramienta de modelado. Con el desarrollo de este componente se logra realizar el proceso de gestión de traspasos de gastos atendiendo a las necesidades del cliente.

Palabras claves: cliente, comprobantes, costos, gastos, procesos, traspaso.

[1] Distribución en forma proporcional de un valor.

CedruX
SISTEMA INTEGRAL
DE GESTIÓN

Índice

Introducción

El desarrollo y evolución de los sistemas informáticos es uno de los grandes retos en la actualidad; el mismo exige mejoras significativas para apoyar la informatización de los procesos que se gestionan en las diferentes entidades, dentro de las cuales se destacan los sistemas de gestión contables con la adopción de tecnologías y herramientas existentes. En este contexto, las Tecnologías de la Información y las Comunicaciones (TIC) son esenciales para que las entidades empresariales puedan enfrentar dichos desafíos, con una sólida gestión de la información y de los procesos.

El entorno competitivo y exigente en el que se desenvuelven actualmente las empresas, ha obligado a mejorar la gestión de los costos y a facilitar la integración de las distintas áreas funcionales, con el objetivo de poder ofrecer un mejor servicio a los clientes, reducir los plazos de entrega y minimizar los inventarios de productos. Los Sistemas de Planificación de Recursos Empresariales (ERP según sus siglas en inglés) tienen el objetivo de facilitar la gestión de los recursos en la empresa, a través de la integración de la información. Estos son sistemas configurables y flexibles para la planeación de recursos empresariales, que permiten unir y consolidar la información a través de la empresa, constituyendo una solución viable para aquellas empresas que buscan integrar y manejar muchos de los negocios asociados con la producción de bienes y/o servicios (Brito, 2008).

La contabilidad de los costos constituye un eslabón fundamental para la organización y planificación de los recursos empresariales basándose en el registro de los hechos económicos de cada entidad (Zorrilla, 2010). Dentro de ella el traspaso de gastos al cierre de un período económico es un proceso laborioso y técnico donde cada producción recibe cuotas de gastos sobre determinadas bases de distribución, las cuales se utilizan teniendo en cuenta las características de los gastos que se distribuyen y de las producciones o servicios que los reciben. El traspaso es un proceso fundamental para la contabilidad de costos ya que proporciona una vez ejecutado, los comprobantes de operaciones, los cuales son utilizados en diferentes operaciones contables. Además constituye un proceso que complementa el ciclo de planificación, registro y control de los costos de producción (Meltom, 2010).

Llevar a cabo el traspaso de los gastos no es más que traspasar los saldos de las cuentas de gastos indirectos a las cuentas de gastos directos (Procesos, 2011). Este se crea por cinco criterios de distribución diferentes: Saldo, Volumen, Porciento, Subelementos y Bases combinadas. Los traspasos son

procedimientos para dejar una cuenta en cero, estos se hacen de un centro de costo a otro, pasando todo el saldo de la cuenta y utilizando un elemento de traspaso para registrar las operaciones contables. Siendo el orden en que deben ejecutarse los traspasos para que cada cuenta de gasto y los centros de costos asociados a ellas, reflejen los saldos que le corresponden en cada período del ejercicio económico y por tanto a cada producto o servicio.

Por estas razones en la Universidad de las Ciencias Informáticas (UCI) se está desarrollando un sistema denominado Cedrux, en el cual el subsistema Costos y Procesos es el encargado de informatizar la gestión de los costos de la entidad que lo utilice; pudiendo medir la suma de gastos que se aplica a una producción o servicio determinado en dicha entidad, con el objetivo de responder a las necesidades contables de las entidades cubanas (Procesos, 2011). En este sentido la realización del **proceso traspaso de gastos**, de forma manual, por su complejidad requeriría de un considerable tiempo y de personal calificado para realizarlo. Es por esto que el sistema debe posibilitar la realización de los traspasos de gastos, siendo este no solamente un instrumento para llegar a la generación de los comprobantes de operaciones a través del proceso de traspaso de gastos, sino que también debe permitir vincular los cálculos con la descripción de las actividades que caracterizan al proceso (Procesos, 2011).

Se desarrolló una primera versión de la solución donde presenta varias deficiencias que afectan la satisfacción de las necesidades de los clientes. Las mismas son:

- No permite realizar el traspaso de los gastos de un centro de costo a otro centro u otros centros.
- Los elementos de traspasos no se corresponden con el elemento asociado a la cuenta y al centro de costo seleccionado.
- Se dificulta la realización del traspaso ya que existen errores para efectuar el prorrateo de acuerdo a los criterios de distribución por Subelementos y Bases combinadas.
- Los cálculos se realizan de forma incorrecta ya que se aplica a la suma de los saldos de los centros de costos orígenes, el porciento definido para traspasar en la secuencia.
- Las interfaces de usuarios presentan dificultad de uso, aprendizaje y apreciación para el cliente, lo que genera un aumento de errores cometidos, menor rapidez en la realización del proceso y el aumento de las pérdidas de tiempo, provocando la disminución de la satisfacción y comodidad del cliente.

- Las actividades que se realizan en el sistema, pueden resultar engorrosas, a veces ineficientes y complicadas, comprometiendo el tiempo de respuesta a los clientes, así como la calidad de los comprobantes emitidos.
- El sistema no facilita la entrega de los comprobantes de operaciones atendiendo a las necesidades del cliente, ya que no cuenta con todos los datos válidos y necesarios.

Atendiendo a la situación anteriormente descrita se define entonces como **problema a resolver**: ¿Cómo gestionar la información referente al proceso traspaso de gastos en el sistema Cedrux para contribuir a satisfacer las necesidades de los clientes?

Se determina como **objeto de estudio:** la gestión de la información de los traspasos de los gastos. Tomando como **campo de acción:** el proceso de gestión de los traspasos de gastos en Cedrux. Definiendo como **objetivo general**: desarrollar un componente informático para la gestión del proceso traspaso de gastos para el sistema Cedrux que contribuya a satisfacer las necesidades de los clientes. Para darle cumplimiento a este objetivo se definen los siguientes **objetivos específicos**:

- Fundamentar la investigación mediante la elaboración del marco teórico para apoyar la propuesta de desarrollo del componente.
- Realizar el análisis, diseño e implementación del componente Traspaso de gasto para contribuir a satisfacer las necesidades de los clientes.
- Validar a través de pruebas de caja negra el componente a implementar.

Planteándose como **idea a defender:** El desarrollo de un componente informático para la gestión del proceso de traspasos de gastos para el sistema Cedrux contribuirá a satisfacer las necesidades de los clientes.

Durante la investigación se han utilizado diferentes métodos científicos para mejorar la calidad de la misma entre ellos se encuentran los métodos teóricos y los empíricos, los cuales se describen a continuación:

Métodos científicos de investigación:

La investigación estuvo guiada por el uso de diferentes métodos que la organizaron metodológicamente conllevando al cumplimiento del objetivo y al buen desarrollo del estudio planificado como las que se explican a continuación:

- **Métodos Teóricos:**

Analítico-sintético: Este método permite realizar un estudio de los sistemas de gestión contables tanto nacionales como extranjeros que más se utilizan en la actualidad, determinando qué particularidades presentan para establecer una comparación entre ellas y tomar los resultados arrojados por dicha comparación como datos de gran interés para la investigación.

Análisis histórico – lógico: Este método permite describir y explicar las características del objeto de estudio de la presente investigación y representar un nivel de la investigación. Además se utiliza para analizar las tecnologías, herramientas y lenguajes a utilizar en el desarrollo de la aplicación.

- **Métodos Empíricos:**

Entrevista: Este método se utiliza para obtener información acerca del proceso de gestión de traspaso de gastos. Para ello se realizan entrevistas que permiten profundizar el conocimiento sobre el negocio del sistema a implementar. Además de reunir y determinar la información básica y necesaria para la captura de requisitos del sistema.

Observación: Se detecta la situación existente en el componente Traspaso de gastos, en cuanto a las dificultades a la hora de llevar a cabo el proceso de traspaso de gastos del sistema, así como la necesidad de crear una herramienta que permita realizar dicho proceso atendiendo a las necesidades del cliente.

Con el fin de desarrollar un buen argumento que sustente esta investigación se dividirá la información en capítulos que engloben diferentes temas ajustados al objetivo general de la misma. La estructura será presentada como:

- Introducción.
- Capítulo 1. Fundamentación teórica.
- Capítulo 2. Análisis y diseño.
- Capítulo 3. Implementación y validación.

- Conclusiones.
- Recomendaciones.
- Bibliografía.
- Anexos

CAPÍTULO I: **Fundamentación Teórica**:

En este capítulo se abordan los fundamentos teóricos y conceptos que abarcan las temáticas expuestas en el documento, necesarias para ayudar a su mejor comprensión y sustentar la investigación. Se describen los principales aspectos relacionados con los sistemas informatizados vinculados a los procesos de traspasos de gastos. Además se realiza un estudio del modelo de desarrollo, las distintas herramientas y tecnologías que se utilizan en el diseño e implementación de las nuevas funcionalidades del componente Traspaso de gastos.

CAPÍTULO II: **Análisis y Diseño:**

En el capítulo se define un conjunto de artefactos generados en el análisis y diseño de la propuesta de solución, de gran valor para la fase de construcción como es el modelado del proceso de negocio y el modelo conceptual. Además se realiza la descripción de los requisitos, validándose estos a través de los prototipos de interfaz de usuario. También se crean los diagramas de clases del diseño, los diagramas de secuencia y el modelo de datos. Igualmente se especifica la utilización de un conjunto de patrones para el diseño del componente.

CAPÍTULO III: **Implementación y Validación:**

En este capítulo a partir de los resultados obtenidos del diseño se comienza la implementación y luego las pruebas del sistema. Se describe cómo realizar una implementación a partir del modelo del diseño en términos de componentes. Por último se valida el diseño propuesto a través de las métricas TOC y RC y se describen las pruebas de caja negra realizadas al software y los resultados obtenidos de estas.

Capítulo 1. Fundamentación teórica

1. Introducción

En el presente capítulo se realiza un análisis para obtener información de algunos sistemas contables que han sido implementados nacional e internacionalmente, así como las diferentes técnicas de captura y validación de requisitos. Se especifican las tecnologías, herramientas y los lenguajes de desarrollo a utilizar durante la implementación para facilitar el cumplimento de los requisitos tanto técnicos como funcionales.

1.2. Sistemas de gestión de costos

Debido al enorme desarrollo comercial interrelacionado con el auge de la producción y el consecuente desarrollo que ha alcanzado la informática, han surgido diversos sistemas de gestión contables, con el objetivo de gestionar la información en una entidad. Por el interés que existe en las empresas en cuanto a la informatización de los procesos contables gran número de estas se dedican a la creación de los mismos. La gestión de los costos es de gran interés para las empresas por lo que conlleva tener un manejo adecuado de la misma, permitiendo mejorar la creciente necesidad de competitividad (Vaquerano, 2009).

Para conocer mejor sobre los diferentes sistemas de costos, las funcionalidades que presentan y los métodos de costos, se realiza un estudio de cada uno de ellos de acuerdo a diferentes aspectos relacionados con la gestión de los traspasos de gastos, con el objetivo de adquirir experiencia de los mejores en esta esfera.

1.2.1. Sistemas extranjeros

En el mundo existen diferentes sistemas de gestión contables informatizados que hacen uso de los grandes avances en la informática. En general existe una gran diversidad de sistemas contables que se encuentran en explotación a nivel mundial. Los sistemas extranjeros seleccionados fueron Openbravo, SAP ERP y OpenERP por encontrarse entre los sistemas más utilizados internacionalmente.

- **Openbravo**

Es un producto de la empresa española Openbravo S.L, orientado a la gestión empresarial integrada dentro de pequeñas y medianas empresas. Se encuentra disponible en varios lenguajes entre ellos el español. Es un software libre y de código abierto, basado en la web con arquitectura cliente/servidor, implementado en Java. Necesita para su ejecución Apache Tomcat y posee soporte para bases de datos en PostgreSQL y Oracle; ha sido desarrollado siguiendo el estilo MVC (Modelo, Vista, Controlador) (Openbravo, 2010).

Openbravo ofrece importantes beneficios a sus clientes que les permiten la gestión financiera y contable, integra todas las actividades contables, incluyendo cuentas a pagar, cuentas a cobrar, contabilidad general, presupuestos, amortización de activos, y la gestión de los períodos fiscales. Además presenta el módulo Gestión de la producción el cual cubre la planificación de la producción, cálculo de los costes de producción, notificación de incidencias de trabajo y partes de mantenimiento (Openbravo, 2010). Al mismo tiempo presenta varias limitantes ya que no integra las funcionalidades necesarias para realizar el proceso de traspaso de gastos que pudieran ser útiles para el componente a implementar, tales como realizar el traspaso de un centro a otro y generar comprobantes de operaciones.

- **SAP ERP**

SAP es desarrollado por SAP AG compañía alemana de software para aplicaciones de negocios. Comprende cuatro soluciones que brindan soporte a procesos de negocio claves a través de su sistema ERP: SAP ERP Administración de Capital Humano, SAP ERP Operaciones, SAP ERP Servicios Generales, SAP ERP Planificación de la Producción y SAP ERP Finanzas. Dentro de este módulo se encuentra el submódulo de Contabilidad General (Hoy, 2010).

Una de las funcionalidades que cuenta SAP ERP es permitir realizar el traspaso de un centro de coste a otro. Esta función es una gran ventaja en el estudio realizado, ya que puede ser utilizada la forma en que realizan esta funcionalidad para la implementación del componte. La asignación de clases de coste a centros de coste de gastos generales permite controlar los costes y comparar los costes plan con los reales (Hoy, 2010). Presenta el inconveniente de ser privativo.

- **OpenERP**

Software de código abierto desarrollado en Bélgica, tiene componentes separados en esquema cliente-servidor, presenta soporte para bases de datos PostgreSQL. OpenERP es un sistema de gestión de empresas/organizaciones de licencia libre que cubre las necesidades de las áreas de contabilidad. Incorpora funcionalidades de gestión de documentos, conectores con otras aplicaciones, permite trabajar remotamente mediante una interfaz web o aplicación de escritorio multiplataforma (Windows, Linux y Mac) e incluye un entorno modular de programación/adaptación rápida de aplicaciones (OpenObject). Se basa en tecnología Python/XML. El software está publicado bajo la licencia AGPL (Ecured, 2011).

OpenERP cuenta con un módulo Gestión contable y financiera el cual provee contabilidad general, analítica y presupuestaria, y cuenta con todas las funcionalidades para llevar los libros contables, puede definir centros de costos. Al mismo tiempo presenta varias limitantes ya que no integra las funcionalidades necesarias para realizar el proceso de traspaso de gasto que pudieran ser de utilidad para la implementación del componente.

1.2.2. Sistemas nacionales

En Cuba han sido desarrollados y se encuentran en explotación a nivel nacional una gran diversidad de sistemas contables. Los sistemas seleccionados para su estudio fueron Versat Sarasola, Rodas XXI y Siscont5 atendiendo a las funcionalidades que estos integran y que pueden ser utilizadas en la gestión de los traspasos de gastos en Cedrux.

- **Versat Sarasola**

Es un sistema de gestión contable-financiero, representa un ejemplo de sustitución de importaciones en materia de aplicaciones informáticas. Desarrollado en 1998 por TEICO Villa Clara, empresa del Ministerio del Azúcar. Es un sistema económico integrado y multientidad netamente cubano. Diseñado para ser utilizado por el sector empresarial de Cuba, configurable, que se ajusta a las características de cada entidad (Ecured, 2011).

Versat Sarasola permite llevar el control y el registro contable individual de todos los hechos económicos que se originan en las estructuras internas de las entidades y obtener los estados financieros y análisis económicos y financieros en estos niveles. Es un producto 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 directos y el ajuste al costo, según los volúmenes de producción o servicios y sus destinos (Ecured, 2011). Constituyendo esta funcionalidad de gran utilidad, ya que pudiera ser de beneficio para el componente a implementar. Tiene la limitante de estar soportado sobre tecnología privativa, es compatible solamente con el sistema operativo Windows, utiliza el motor de base datos SQLServer 2000(Lenguaje Estructurado de Consultas). Es una aplicación de escritorio implementada en Delphi.

- **Rodas XXI**

Sistema Integral Económico Administrativo es un sistema multiempresa desarrollado por la empresa CITMATEL[2] que posibilita automatizar el funcionamiento de una institución autofinanciada o presupuestada. Cuenta actualmente con ocho módulos: Finanzas, Contabilidad, Activos Fijos, Nóminas, Inventario, Facturación, Recursos Humanos y Telecobranzas. Estos módulos pueden emplearse integrados en su totalidad o no, formando cualquier subconjunto entre ellos, o cada uno de forma independiente (CITMATEL, 2012).

En el proceso de cierre del año, el módulo de contabilidad brinda la posibilidad de informatizar mediante opción configurable la realización del comprobante de cancelación de las cuentas de gastos e ingresos trasfiriendo sus saldos a la cuenta resultado, quedando solamente por parte del usuario la realización del comprobante de traspaso de saldo de esta cuenta a las cuentas que correspondan antes de cerrar el año, (CITMATEL, 2012) constituyendo esta funcionalidad de gran utilidad, ya que pudiera ser de interés la forma en que realizan el comprobante. Presenta la limitante de estar desarrollado para plataformas privativas y de no realizar muchas de las funcionalidades para el proceso de traspaso que pudieran ser de beneficio para el componente a implementar

- **Siscont5**

Sistema cubano creado por la empresa Tecnomática, 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

[2] **CITMATEL:** Empresa de Tecnologías de la Información y Servicios Telemáticos Avanzados .

costo que afectan, la distribución y el cálculo de costos unitarios. También 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 (Tecnomática, 2008). Tiene la ventaja de permitir definir las bases de distribución de los costos indirectos de un objeto de costo a otros, definir el método de traspaso de gastos, que pueden ser útiles para la implementación del componente. Presenta la limitante de trabajar sobre el sistema operativo Windows.

1.3. Análisis de los sistemas estudiados

Se realizó un estudio de varios sistemas teniendo en cuenta las principales funcionalidades de la aplicación a desarrollar para llevar a cabo el proceso traspaso de gastos.

Funcionalidades	Openbravo	SAP ERP	OpenERP	Versat-Sarasola	Rodas XXI	Siscont5
Adicionar configuración de traspaso.		X		X	X	
Realizar el traspaso de un centro de costo a otro.		X		X		X
Realizar el traspaso de un periodo						X
Cálculos según criterios distribución.						X
Ejecutar traspasos de gastos.		X		X	X	X
Generación de comprobantes de operaciones.					X	
Licencia		X		X	X	X
Multiplataforma	X	X	X			
Tipo de Sofware	Libre	Propietario	Libre	Propietario	Propietario	Propietario

Tabla1. Comparación entre las funcionalidades y características de los sistemas

Analizando las funcionalidades que brindan los sistemas anteriormente mencionados, ya sea los nacionales como los extranjeros, se plantea que estos no satisfacen las necesidades del cliente, debido a que no incluyen muchas de las funcionalidades que deben estar presentes en el sistema a desarrollar.

Aunque en el caso de SAP ERP, Versat Sarasola y Siscont5 integran algunas de las funcionalidades que se corresponden con el proceso de traspaso de gastos entre las que se encuentran, realizar el traspaso de un centro a otro, permitir definir las bases de distribución y nuevos traspasos. Así mismo Rodas XXI permite realizar los comprobantes de operaciones siendo de gran importancia para el componente a implementar, ya que es el resultado del proceso. Dichas funcionalidades son facilidades para la propuesta del sistema, ya que pueden ser utilizadas, añadiendo otras que permitan un mejor funcionamiento del sistema.

Igualmente un conjunto de estos sistemas han sido desarrollados para darle solución a diferentes inconvenientes de las empresas, donde las herramientas sobre las que están soportadas y las licencias de dichos productos no son libres. El empleo de sistemas de carácter extranjero genera al país gran cantidad de gastos por pago de licencias y mantenimiento. En el caso de las aplicaciones nacionales, son creadas para sectores específicos de la economía, por lo que su uso no se puede generalizar a todas las entidades, ya que estas no gestionan los procesos de forma estándar.

Aunque estos sistemas están altamente calificados y reportan un gran número de ventajas para cualquier empresa del mundo, no son de utilidad para las especificidades que necesita el cliente. Por lo que se hace evidente la necesidad de crear un sistema que permita informatizar los nuevos requisitos para la gestión de los traspasos en Cedrux, con todos los beneficios y las funcionalidades propias que complementan y perfeccionan los traspasos de gastos, utilizando las ventajas brindadas por los sistemas estudiados.
Con el propósito de cumplir los objetivos que plantea la ingeniería de requerimientos, son aplicadas en el sistema diferentes actividades encaminadas a lograr mejores resultados.

1.4. Actividades y técnicas aplicadas en la ingeniería de requisitos

Como parte de las necesidades y exigencias del proyecto y con el propósito de obtener requisitos de software con calidad se decidieron utilizar tres de los pasos propuestos por Pressman: Elicitación, Especificación y Validación (Pressman, 2008).

Elicitación de requisitos

La elicitación de requisitos es la parte de la IR (Ingeniería de Requisitos) en la que se tiene contacto con los clientes y usuarios y donde debe quedar claro el dominio del problema, las necesidades reales de clientes y usuarios y la negociación de los requisitos con estos.

A continuación relacionamos algunas de las técnicas que fueron aplicadas durante la elicitación:

- **Revisión de documentos**

Esta técnica depende de la información almacenada por las entidades acerca de los procesos y términos que se manejan dentro de la misma. Las entidades guardan información referente a sus procesos, los modelos o informes necesarios para el desarrollo de la misma o para rendir cuenta a los organismos superiores.

- **Entrevistas**

Esta técnica es la más utilizada y aceptada para la obtención de la información de los clientes. Las entrevistas le permiten al analista familiarizarse con el problema por medio de una conversación, generalmente los encuestados son usuarios de los sistemas existentes o usuarios en potencia del sistema propuesto.

- **Tormenta de ideas**

Esta es una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas en un entorno libre de críticas o juicios. Como técnica de captura de requisitos es sencilla de usar y de aplicar. El éxito de esta técnica depende principalmente de la libertad para expresar las ideas.

- **Prototipos**

Los prototipos son útiles para comunicar, discutir y definir las ideas entre los diseñadores y las partes responsables; puede ser cualquier cosa, desde un trozo de papel con sencillos dibujos a un complejo software. Permiten llegar a niveles muy detallados en las especificaciones del futuro software concretando sus elementos, centrándose en la representación de aquellos aspectos del software que serán visibles para el cliente o usuario final.

- **Modelado del negocio**

Esta técnica se enfoca en la definición e identificación de los requerimientos funcionales.

Especificación de requisitos

La especificación permite documentar los requisitos negociados utilizando las notaciones que sean necesarias para que todos los participantes la entiendan. Obtener especificaciones de requerimientos con la calidad suficiente es fundamental para asegurar un software que corresponda con las necesidades del cliente.

Algunas de las técnicas utilizadas durante la especificación de requisitos son las siguientes:

- **Glosario de términos**

Esta técnica permite registrar el conocimiento que se va adquiriendo sobre el dominio del problema y compartirlo con todos los participantes en el proyecto, creando un vocabulario propio. Definen los conceptos principales y críticos para el sistema.

- **Plantillas o patrones**

El objetivo principal de esta técnica es describir los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos predefinidos que el equipo de desarrollo va construyendo, usando para ello el lenguaje del usuario.

Validación de requisitos

La validación de requisitos permite verificar que las funcionalidades que se necesitan para el sistema cumplen con las expectativas del usuario, que los requerimientos sean consistentes y que estén completos. Se examinan las especificaciones para garantizar que todos los requisitos del sistema han sido establecidos correctamente. Las dos técnicas más utilizadas durante el proceso de validación se describen a continuación:

- **Revisiones**

Esta técnica consiste en la lectura y corrección de la completa documentación o modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida.

- **Prototipo orientado a clientes y/o usuarios**

Es un modelo a escala reducida de la solución final que sirve para verificar que las especificaciones han sido construidas de acuerdo a los requisitos del sistema, es decir que le ofrece a los usuarios una idea más clara del producto que van a recibir.

1.5. Patrones de control de flujo

Para el modelado del negocio que se emplean diversos patrones de control de flujo que aportan flexibilidad y organización al trabajo, ya que para diversos problemas que se pueden hallar en el modelado, hay soluciones (patrones) para resolver diversas situaciones de una forma práctica y óptima.

Los patrones básicos de control de flujo: Son patrones que capturan aspectos elementales de control de procesos.

- **Secuencia:** Varias tareas de un proceso se ejecutan, uno tras otro. Una tarea es permitido después de la tarea anterior haya terminado y antes de continuar la tarea se ha iniciado
- **Elección exclusiva:** Una elección exclusiva es un punto distinto en un proceso de negocio, donde una rama se dividido en dos o más ramas que permite sólo una de estas ramas de varios.

Una vez completadas las actividades de la ingeniería de requisitos y aplicadas efectivamente cada una de las técnicas antes expuestas, es necesario adoptar una guía que indique qué se debe hacer en cada momento y cómo hacerlo, siendo esta la función principal del modelo de desarrollo de software.

1.6. Modelo de desarrollo, lenguaje de modelado y herramientas

La definición de las tecnologías para el desarrollo de software en las empresas es la base para la ejecución de cualquier sistema, originando la utilización de metodologías, técnicas y lenguajes que soporten el proceso de desarrollo de software en correspondencia con los intereses del cliente. Por lo que es muy importante conocer las herramientas y tecnologías definidas por el grupo de arquitectura del proyecto Cedrux, ya que estas influyen considerablemente en el desarrollo y la calidad del producto.
A continuación se realiza una breve caracterización de las mismas, mostrando así los beneficios que aportan en el trabajo de análisis, diseño e implementación del componente Traspaso de gastos.

1.6.1. Modelo de desarrollo

Para el desarrollo del sistema Traspaso de gastos se asume el modelo de desarrollo propuesto por el Departamento de Tecnología del Centro de Informatización y Gestión de Entidades (CEIGE), que propone las siguientes fases:

Inicio o estudio preliminar: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel, la evaluación de la factibilidad del proyecto y el registro de este. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, y decidir si se ejecuta o no el proyecto (Obregón, 2012).

Desarrollo: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se refinan los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. En esta fase se ejecutan las disciplinas Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas y Pruebas de liberación (Obregón, 2012).

Dicho modelo está basado en componentes. Una de las principales ventajas que proporciona la utilización de este modelo es que permite la reutilización de los componentes existentes o desarrollados desde un inicio, lo que trae consigo que los ciclos de desarrollo de un proyecto en la universidad se efectúen en menos tiempo. Las iteraciones son orientadas por el nivel de significación arquitectónica de los componentes, que constituyen abstracciones de los procesos de negocio y requisitos asociados a modelar; establecen al componente como unidad de medición y ordenamiento de las iteraciones. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados. Cuando existe un débil acoplamiento entre componentes, se puede actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema. Dado que un componente puede ser construido y luego mejorado continuamente, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo (Obregón, 2012).

El sistema será realizado utilizando los estándares, tecnologías y herramientas propuestas por CEIGE para el trabajo en proyecto Cedrux.

1.6.2. Lenguaje y herramienta para el modelado

Lenguaje Unificado de Modelado (UML 2.1)

Para modelar el sistema se utiliza el Lenguaje Unificado de Modelado (Unified Modeling Lenguaje, UML por sus siglas en inglés) debido a que este ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como los procesos de negocios y funciones del sistema, aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Es un lenguaje basado en una notación gráfica la cual permite: especificar, construir, visualizar y documentar los objetos de un sistema programado. Es aplicable para tratar asuntos inherentes a sistemas complejos, de misión crítica, tiempo real y cliente/servidor (Romero, 2011).

Para realizar el modelado de los diagramas se utilizan las herramientas CASE (Ingeniería de Software asistida por ordenador) ya que estas presentan ventajas que permiten desarrollar el software con mayor rapidez y pueden ser usadas en todo el ciclo de vida del mismo.

CASE:

La herramienta Visual Paradigm For UML que cumple con la estructura de CASE, por lo que esta puede ser aprovechada en todo el ciclo de desarrollo del software.

Visual Paradigm for UML 6.4

Es una herramienta que emplea UML y permite representar el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Constituye una herramienta de software libre de gran utilidad para el analista. Dentro de sus características se aprecia que soporta BPMN (Paradimg, 2008).

1.6.3. Herramientas

NetBeans IDE 7.0.1

Es un entorno de desarrollo integrado disponible para varios Sistemas Operativos como, Windows, Mac, Linux y Solaris, de código abierto, escrito completamente en Java. Es una plataforma de aplicaciones que

permite a los desarrolladores crear aplicaciones del tipo web, de escritorio, empresariales, y móviles, utilizando la plataforma Java, así como un número importante de módulos para extenderlo a otros lenguajes (Oracle, 2011).

Apache 2.0x

Es un servidor web, configurable y de diseño modular. Es gratuito, de código fuente abierto, el cual brinda transparencia al software de manera que se pueda conocer lo que se está instalando como servidor. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto. Posee configurabilidad en la creación y gestión de logs[3], permitiendo la creación de estos ficheros a la medida de las necesidades del administrador, de este modo se puede tener un mayor control sobre lo que sucede al servidor (Ciberaula, 2011).

Subversion 1.6.6

Es un controlador de versiones empleado en la administración de archivos utilizados en el desarrollo de software que permite seguir la historia de los archivos y directorios a través de copias y renombrados, incluyendo cambios a varios archivos. Presenta integración con Apache y existen varias interfaces de subversion, ya sean programas individuales o interfaces que lo integran en entornos de desarrollo (CENDITEL, 2009).

PostgreSQL 8.3

Es un sistema de gestión de bases de datos, objeto-relacional (Object Relational Database Management System, ORDBMS por sus siglas en inglés), basado en el proyecto POSTGRES de la Universidad de Berkeley. Publicado bajo la licencia BSD[4]. PostgreSQL como Sistema Gestor de Base de Datos garantiza la integridad de la información porque es escalable, además es libre y multiplataforma. Presenta características orientadas a objetos, herencia entre tablas (Martinez, 2010).

PgAdmin III

[3] **Logs:** Registro de errores.

[4] **BSD**: Berkeley Software Distribution

PgAdmin III es una aplicación gráfica para manejar el gestor de bases de datos PostgreSQL, con licencia Open Source[5]. Incluye un editor SQL (Structured Query Language por sus siglas en inglés) con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, soporte para el motor de replicación Slony-I[6]. Está escrita en C++, lo que permite que se pueda usar en varios sistemas operativos como Linux, FreeBSD, Solaris, Mac OS X y Windows. La conexión al servidor puede hacerse mediante conexión TCP/IP (Protocolo de control de transmisión/Protocolo de Internet), y puede encriptarse mediante SSL (Protocolo de Capa de Conexión Segura) para mayor seguridad (Martínez, 2010).

Mozilla Firefox 3.6.24x

Es un navegador libre y de código abierto. Mozilla Firefox es multiplataforma, permite la integración con los antivirus, realiza la navegación por pestañas, presenta compatibilidad para múltiples extensiones y utiliza el sistema SSL para proteger la comunicación con los servidores web, utilizando fuerte criptografía cuando se utiliza el protocolo HTTPS (Protocolo de transferencia de hipertexto seguro). Es usado para visualizar páginas web. Incluye corrector ortográfico, búsqueda progresiva y marcadores dinámicos. Además se pueden añadir funciones a través de complementos desarrollados por terceros (Firefox, 2010).

Marco de trabajo Sauxe 1.5.4

El marco de trabajo Sauxe 1.5.4 se estructura de manera tal que facilita la reutilización de los diferentes componentes y es de fácil entendimiento para quienes desarrollen sobre el mismo. Sauxe utiliza ExtJS para implementar la capa de presentación, siendo una librería construida con JavaScript que proporciona una interfaz cuya potencia radica en la colección de componentes para el diseño de interfaces del lado del cliente. Tiene incluidos muchos controles de los formularios web incluyendo tablas para mostrar datos y elementos semejantes a la programación de escritorio como los formularios, paneles, barras de herramientas, menús y muchos otros. Dentro de su biblioteca de componentes incluye el manejo de datos,

[5] **Open source:** Código abierto es el término con el que se conoce al software distribuido y desarrollado libremente.

[6] **Slony-I:** Es un software que permite hacer replicaciones maestro/esclavo asíncrono, realizando actualizaciones en cascada.

lectura de XML (Lenguaje Etiquetado Extensible), lectura de datos JSON[7] e implementaciones basadas en AJAX[8]. Presenta el uso de JavaScript con una programación orientada a objetos (Extend, 2010).
Se apoya en Zend Framework para el desarrollo de la lógica del negocio, el mismo es framework para el desarrollo de aplicaciones web y servicios web con PHP. Proporciona los componentes que forman la infraestructura del patrón Modelo-Vista-Controlador, además de una capa de acceso a la base de datos, construida sobre PDO[9]. También proporciona mecanismos de filtrado y validación de entrada de datos y cuenta con módulos para manejar documentos en formato portable (Portable Document Format, PDF por sus siglas en inglés) y Web Services (Eslomas, 2011).

Para la gestión de los datos utiliza Doctrine. Doctrine es un ORM (Mapeador de relación de objetos) para PHP 5.2.3 y posterior. Uno de sus puntos fuertes es su lenguaje DQL (Doctrine Query Language) inspirado en el HQL[10] de Hibernate, lenguaje para consultas a la base de datos. Un ORM es una técnica de programación que permite convertir datos del tipo utilizado en un lenguaje de programación orientado a objetos y es utilizado en una base de datos relacional, es decir, las tablas de la base de datos se convierten en clases y los registros en objetos, que se pueden manejar con facilidad (Doctrine, 2011).

El marco de trabajo Sauxe tiene como propósito insertar la programación orientada a aspectos así como la inversión de controles. El framework en su nueva configuración define que la conexión a la base de datos será configurada en un archivo de tipo XML almacenado en la carpeta de recursos comunes del proyecto. Es válido aclarar que este cuenta con un componente de transacciones mediante el cual serán salvados automáticamente los datos de modificaciones e inserciones. Es decir, ya no será necesaria la implementación por parte del programador de las consultas de inserción, este solamente deberá programar la obtención de los campos de la presentación y dentro de Sauxe el Doctrine es el responsable de que los mismos sean guardados en la base de datos (Hernández, 2011).

---

[7] **JSON:** acrónimo de JavaScript Object Notation es un formato ligero de intercambio de datos

[8] **AJAX**: acrónimo de Asynchronous JavaScript And XML (JavaScript y XML asíncronos), es una técnica de desarrollo web para crear aplicaciones interactivas.

[9] **PDO:(**PHP Data Objects por sus siglas en inglés) Capa de abstracción de acceso a datos para PHP. con lo cual se consigue hacer uso de las mismas funciones para hacer consultas y obtener datos de distintos manejadores de bases de datos.

[10] **HQL:** Lenguajes de consultas para Hibernate (Hibernate Query Language, HQL por sus siglas inglés). Es un ORM para el lenguaje JAVA.

1.6.4. Lenguajes y tecnologías

Lenguaje de Etiquetado Extensible (XML 1.1)

El Lenguaje de Etiquetado Extensible (Extensible Markup Language, XML por sus siglas en inglés) es muy simple, pero estricto y juega un papel fundamental en el intercambio de una gran variedad de datos. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones. Es utilizado para estructurar, almacenar e intercambiar información. A diferencia del HTML su función principal es describir datos y no mostrarlos (Editorial Colectiva, 2011).

Notación de Objetos de JavaScript (JSON)

JSON (acrónimo de JavaScript Object Notation en inglés) es un formato de intercambio de datos. Está basado en un subconjunto del lenguaje de programación JavaScript. JSON es un formato de texto independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidas por los programadores de la familia de lenguajes C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. JSON se ha convertido en un estándar en el desarrollo de aplicaciones web. En los servicios web es empleado JSON en vez de XML para permitir la integración de servicios en el navegador del usuario en vez del servidor (JSON, 2010).

JavaScript 1.6

JavaScript es un lenguaje de programación interpretado, que puede usarse sin necesidad de adquirir una licencia. Permite a los desarrolladores añadir y crear interactividad en el desarrollo y diseño de sitios web. Por otra parte permite validar datos. No requiere de compilación ya que el lenguaje funciona del lado del cliente y los navegadores son los encargados de interpretar estos códigos (W3Schools, 2010).

Lenguaje de Marcas de Hipertexto (HTML)

El Lenguaje de Marcas de Hipertexto (HyperText Markup Language, HTML por sus siglas en inglés), es un conjunto de etiquetas o comandos, complementados en la mayoría de los casos por extensiones que permiten dar formato a un archivo, con el objetivo básico de crear un documento que pueda ser visualizado por navegadores en forma de página web y que además, pueda, por medio de dichas etiquetas, tener la estructura o forma deseada por quien la diseñó (Lanzillotta, 2008).

JavaScript y XML asíncronos (AJAX)

AJAX (acrónimo de Asynchronous JavaScript And XML en inglés), es una técnica de desarrollo web para crear aplicaciones interactivas. Se ejecuta en el navegador del usuario, y mantiene comunicación asíncrona con el servidor. De esta forma, es posible realizar cambios sobre la misma página sin necesidad de recargarla, aumentando la interactividad, velocidad y usabilidad. AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de tecnologías que trabajan conjuntamente (W3Schools, 2010).

Pre-procesador de hipertexto (PHP)

PHP está diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente para la interpretación del lado del servidor pero actualmente puede ser utilizado desde una interfaz de línea de comandos, incluyendo aplicaciones con interfaz gráfica, usando distintas bibliotecas. El Pre-procesador de hipertextos (Hypertext Pre-processor, PHP por sus siglas en inglés) inicialmente se llamó PHP Tools, siendo publicado bajo licencia de software libre (Group, 2009).

PL/pgSQL

Lenguaje de procedimientos almacenados para Postgres (Procedural Language/Postgres Structured Query Language en inglés) es un lenguaje provisto por el gestor de base de datos PostgreSQL. Permite ejecutar comandos SQL mediante un lenguaje de sentencias imperativas y uso de funciones. PL/pgSQL tiene como objetivo ser usado para crear funciones, añadir estructuras de control al lenguaje SQL, realizar cálculos complejos, además de heredar todos los tipos, funciones y operadores definidos por el usuario (Martínez, 2010).

1.7. Arquitectura

Se puede decir que una arquitectura de software, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema (Clements, 2009).

Estilo arquitectónico Modelo- Vista- Controlador (MVC)

MVC es un estilo arquitectónico usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de los conceptos para que el desarrollo esté estructurado de una mejor manera. Esto facilita la programación en diferentes capas de forma paralela e independiente. MVC sugiere la separación del software en 3 capas (ASP.Net, 2010).

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

El Modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos.
- Definir las reglas de negocio.
- Si se encuentra ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo.

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

Las Vistas son responsables de:

- Recibir datos del modelo y mostrarlos al usuario.
- Tener un registro de su controlador asociado (normalmente porque además lo instancia).

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

El Controlador se encarga de:

- Recibir los eventos de entrada.
- Contiene reglas de gestión de eventos. Estas acciones pueden suponer peticiones al modelo o a las vistas.

1.8. Conclusiones

Una vez finalizado el presente capítulo se pudo arribar a las siguientes conclusiones:

- El estudio de los sistemas informáticos vinculados a la gestión de costos demostró que no existe un sistema contable que cumpla con la totalidad de las necesidades contables y soberanía tecnológica de las entidades cubanas, evidenciándose la necesidad de desarrollar la versión 1.1 del componente Traspasos de gastos.
- El ambiente de desarrollo integrado por el modelo utilizado, y arquitectura seleccionadas, además de los lenguajes, tecnologías y herramientas definidas, posibilitaron el desarrollo de una solución que apoya el cumplimiento de los principios de soberanía e independencia tecnológica.

Capítulo 2: Análisis y diseño

2.1. Introducción

En el presente capítulo se modelan los artefactos correspondientes al análisis de la propuesta de solución para el componente a implementar, quedando definido el listado de requisitos y la descripción de los mismos. Como resultado se obtienen los artefactos que son de gran valor para la fase de construcción como: el diagrama de clases del diseño, el modelo de datos y por último los prototipos de interfaz de usuario. Además se especifica la utilización de un conjunto de patrones dentro del diseño del componente.

2.2. Propuesta del sistema

El sistema debe permitir al usuario realizar el proceso de gestionar los traspasos de gastos atendiendo a las necesidades del cliente. Para ello el usuario debe autenticarse en la aplicación. Luego selecciona el centro de costo origen, el o los centros de costos destinos, especificando la cuenta, el centro y el elemento de traspaso, donde además define el criterio de distribución, los cuales pueden ser por Saldo, por Subelemento, por Bases combinadas, por Porciento o por Volumen. Posteriormente las configuraciones realizadas, el sistema permite ejecutarlas en el período que se encuentra la entidad económica, obteniendo como resultado la generación de los comprobantes de operaciones en el que se registra el traspaso de los saldos.

A continuación se realiza una breve descripción del proceso de negocio, patrones de diseño utilizados, algunos artefactos que propone el modelo de desarrollo establecido y los diagramas de clases del diseño.

2.3. Modelación de los procesos de negocio

La modelación se realiza mediante los artefactos tales como el diagrama de proceso de negocio, la descripción de los mismos y el modelo conceptual.

2.3.1. Diagrama del proceso de negocio

Un proceso de negocio es un conjunto de tareas relacionadas lógicamente que se llevan a cabo para lograr como resultado un negocio definido y tienen como objetivo fundamental satisfacer al cliente. Para llevar a cabo el diagrama de proceso de negocio **Realizar traspaso de gasto** se tuvieron en cuenta varios patrones de control de flujo tales como el patrón secuencia que muestra que una tarea se ejecuta después

que la anterior haya terminado y el patrón elección exclusiva ya existe una rama que se divide en cinco ramas que permite sólo la ejecución de una de estas ramas.

A continuación se muestra cómo se realiza el proceso **traspaso de gasto** en Cedrux. Para establecer el procedimiento el usuario selecciona el centro de costo origen y luego el o los destinos. De cada centro selecciona una cuenta, un elemento de traspaso y el criterio que desea seleccionar para dicho proceso. Luego el mismo ejecuta la secuencia y se obtiene un comprobante de operaciones. Ver **Figura 2.**

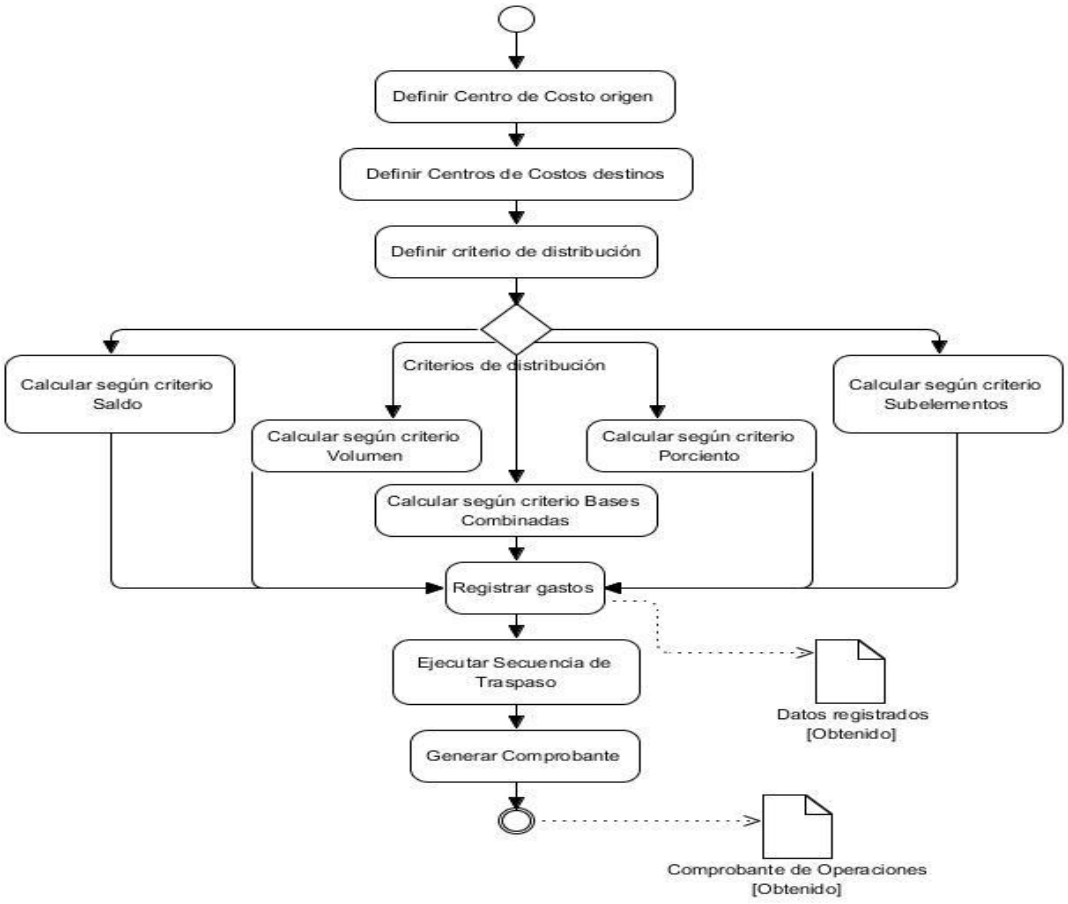


Figura 2: Proceso de negocio: Realizar traspaso de gastos

Descripción del proceso de negocio

A continuación se presenta un breve resumen del proceso de negocio Realizar traspasos de gastos con el propósito de describir en detalles cómo es el método de realización del mismo, el resto de los procesos se detalla en los anexos correspondientes.

Descripción del proceso Realizar traspasos de gastos

Objetivo	Realizar traspasos de gastos indirectos.
Evento(s) que lo genera(n)	N/A.
Pre condiciones	Deben estar definidos los centros de costos y los elementos de gastos asociados a sus cuentas de control. Deben haberse efectuado las asociaciones entre cuentas de gastos, centros de costos y elementos de gastos patrimoniales.
Entradas	Cuentas de gastos, centros de costo y elementos de gastos.
Flujo de eventos	
Flujo básico Traspasos de gastos	

1. **Definir centro de costo origen:** El contador define el centro de costo que se va a establecer como centro de costo origen. Del cual se determina la cuenta a la que desea realizar el traspaso, se define el elemento que va a tomarse como elemento de traspaso.
2. **Definir centros de costos destinos:** El contador define el o los centros de costos destinos que se encuentren asociados a la cuenta de gasto seleccionada. Por cada uno de los centros de costos destinos especificados, se define el elemento que va a tomarse como elemento de traspaso. Se define el Criterio de distribución(los mismos pueden ser: por Saldo, por Subelemento, Por Bases Combinadas, por Porciento o por Volumen).
3. **Realizar traspaso:** El contador realiza el traspaso con todos los elementos antes seleccionados.

Pos-condiciones
1. Se obtiene el traspaso realizado.

Salidas

1. Comprobante de operaciones.

Flujos alternos 2.a La base de distribución seleccionada es por Saldo

Sumar los saldos de los centros de costo destinos (SD). Dividir el saldo total del centro de costo origen (SO) entre el saldo total de los centros de costos destinos (SO/SD) obteniendo un coeficiente denominado Base de Distribución. Multiplicar la base de distribución por los saldos de cada uno de los centros de costos destinos, esto da como resultado el importe que le corresponde a cada uno de los centros de costos destinos.

Pos-condiciones

N/A

Flujos alternos 2.b La base de distribución seleccionada es por Volumen

Sumar todos los valores de los volúmenes asignados a los centros de costos destinos. Dividir el valor del saldo total del elemento de traspaso del centro de costo origen entre la suma total de los volúmenes, obteniendo un coeficiente. Multiplicar el coeficiente por cada uno los volúmenes de los centros de costo destinos, obteniéndose como resultado el importe que le corresponde a cada uno de los elementos de traspaso de los centros de costo destino.

Pos-condiciones

N/A

Flujos alternos 2.c La base de distribución seleccionada es por Subelemento

Sumar el saldo de los subelementos definidos por cada uno de los elementos de traspaso de los centros de costos destinos (SD). Dividir el valor del saldo del elemento de traspaso origen entre SO, obteniendo un coeficiente. Multiplicar el coeficiente por cada uno de los saldos de los elementos de traspaso definidos para cada centro de costo destino, obteniéndose como resultado el importe que le corresponde sumarle a cada uno.

Pos-condiciones

N/A

Flujos alternos 2.d La base de distribución seleccionada es por Bases combinadas
Dividir el valor del saldo del elemento de traspaso origen entre la ∑SD, obteniéndose un coeficiente.
Sumar el saldo de los subelementos definidos por cada uno de los elementos de traspaso de los centros de costo destino (SD). Obtener la sumatoria de los saldos de los todos los subelementos definidos por cada centro de costo destino o la sumatoria de los subtotales o sea: ∑SD. Multiplicar el coeficiente por cada total de saldos correspondiente a un centro de costo destino (SD), obteniéndose como resultado el importe que le corresponde a cada elemento de traspaso definido para ese centro de costo.
Pos-condiciones

N/A.

2.3.2. Modelo conceptual

La Figura 3 muestra el modelo conceptual correspondiente a este sistema, en él se representan los principales conceptos que convergen a la descripción del proceso traspaso de gastos, así como sus relaciones y objetivos fundamentales. El estudio y análisis de este diagrama puede ser muy útil para entender el funcionamiento del sistema descrito, partiendo de las principales funcionalidades que dispone, hasta llegar a los detalles más básicos de su implementación. Este constituye la base teórica en la cual se desarrolla la investigación. Ver **Figura 3.**

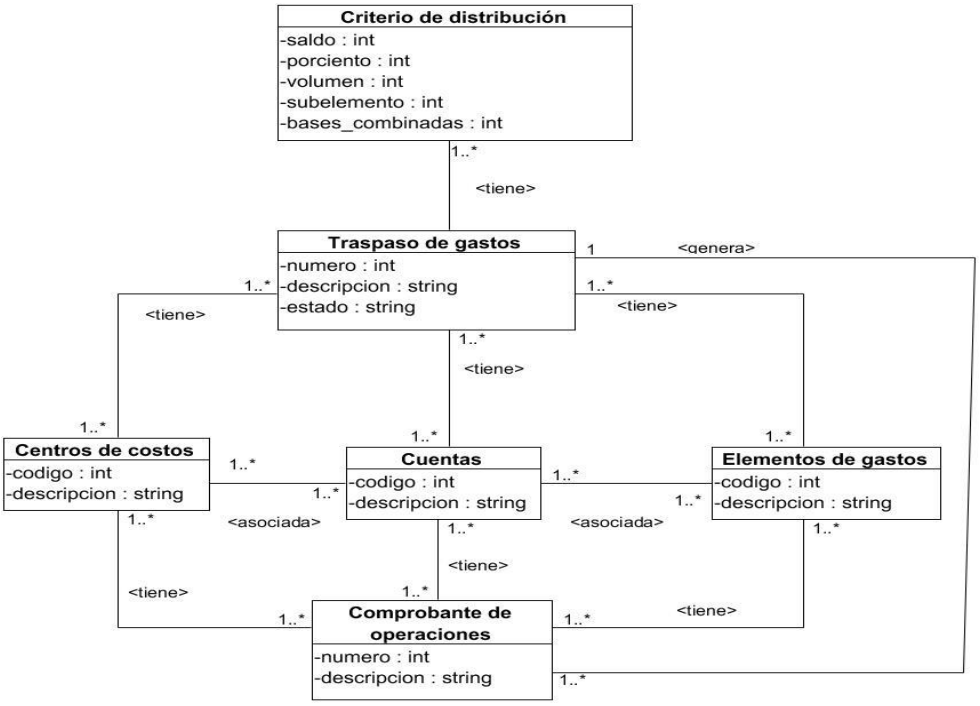


Figura 3: Modelo conceptual

2.3.3. Actividades y técnicas aplicadas en la ingeniería de requisitos

Como parte de las necesidades y exigencias del proyecto y con el propósito de obtener requisitos de software con calidad se utilizaron tres de los pasos propuestos por Pressman: Elicitación, Especificación y Validación (Pressman, 2008).

Se definieron los requisitos funcionales de los procesos del sistema, donde fue necesario emplear técnicas que facilitaran su extracción, es por ello que en el componente Traspaso de gastos se aplicaron las técnicas de captura de requisitos: entrevistas y tormenta de ideas; resultando favorables a las necesidades del cliente. A continuación se relacionan algunas de las técnicas que fueron aplicadas:

Elicitación

- **Entrevistas**

Se realizaron varias entrevistas al director de políticas contables del Ministerio de Finanzas y Precios Jose Carlos de Toro Ríos que permitieron obtener la mayor cantidad de información y realizar con calidad la captura requisitos.

- **Tormenta de ideas**

Se realizaron reuniones con la jefa de línea Ing. Yanay Hernández Sosa, con el arquitecto de datos MSc. Joisel Pérez Pérez y con la analista de Costo y Procesos Ing. Ileana Centelles, donde se plantearon las ideas de importancia para el componente y así sentar las bases para la especificación de los requisitos.

Especificación de requisitos

La especificación permitió documentar los requisitos del componente Traspaso de gastos con la calidad suficiente para asegurar un software que corresponda con las necesidades del cliente.

Algunas de las técnicas utilizadas durante la especificación de requisitos son las siguientes:

- **Glosario de términos**

Se registraron los principales conceptos involucrados en el componente a implementar, creando un vocabulario propio con los conocimientos obtenidos sobre el dominio del problema.

- **Plantillas o patrones**

Se utilizaron las planillas definidas por el proyecto para describir los requisitos mediante el lenguaje natural de una forma estructurada.

Validación de Requisitos

La validación de los requisitos permitió verificar que las funcionalidades que se necesitan para el sistema cumplen con las expectativas del cliente. Se examinaron las especificaciones para garantizar que todos los requisitos del sistema fueron establecidos correctamente. Las dos técnicas utilizadas durante el proceso de validación se describen a continuación:

- **Revisiones**

Esta técnica consistió en la lectura y corrección de la completa documentación o modelado de la definición de requisitos, revisada por la analista de Costo y Procesos Ing. Ileana Centelles.

- **Prototipo orientado a clientes y/o usuarios**

Fueron realizados para verificar que las especificaciones construidas están en correspondencia con los requisitos del sistema, ofreciendo al cliente una idea más clara del producto que va a recibir.

2.4. Definición de los requisitos de software

Se realizó la definición de los requisitos con el propósito de lograr especificar las condiciones o capacidades que el sistema debe cumplir y las restricciones bajo las cuales debe operar. Se definió todo el proceso de negocio y quedó documentado, donde también se tuvieron en cuenta las características o propiedades haciendo posible la obtención de los requisitos a cumplir por el sistema (Tecnológico, 2010). Los mismos se clasificaron en Requisitos funcionales y Requisitos no funcionales.

Requisitos funcionales

Partiendo del resultado de la elicitación de requisitos que se obtuvo previamente en el trabajo de diploma que antecede a este, donde se obtuvieron los requisitos funcionales que debe cumplir el sistema para la gestión de los traspasos de gastos en Cedrux, es necesario un cambio en los mismos a petición del cliente, ya que fue modificada toda la lógica de negocio, añadiendo un nuevo requisito, el mismo tiene como nombre Cambiar estado de traspaso de gastos. A continuación se listan dichos requisitos:

Gestionar configuración de traspaso de gastos

R1. Adicionar configuración de traspaso de gastos.
R2. Modificar configuración de traspaso de gastos.
R3. Eliminar configuración de traspaso de gastos.
R4. Consultar configuración de traspaso de gastos.
R5. Listar traspasos de gastos configurados.
R6. Buscar traspasos de gastos configurados.
R7. Cambiar estado de traspaso de gastos.

Gestionar ejecución de traspaso de gastos

R1. Ejecutar traspaso de gastos.

R2. Visualizar los datos correspondientes de la ejecución de los traspasos de gastos.

Requisitos no funcionales

De los requisitos no funcionales los que se tuvieron en cuenta fueron:

Usabilidad: El sistema está enfocado al cliente, con el mismo se mantuvo un estrecho intercambio en el transcurso del desarrollo del sistema posibilitando el fácil aprendizaje y uso del mismo.

Interfaz: La interfaz de la aplicación es sencilla para reducir el tiempo de capacitación de los usuarios. Además, por el uso diario y constante del software, la interfaz es agradable, favorece el estado de ánimo del cliente y combina correctamente los colores, tipo de letra, tamaño y los iconos están en correspondencia con lo que representan. Se utilizan plantillas con el mismo estilo.

2.4.1. Descripción de los requisitos funcionales

A continuación se muestra la descripción del requisito funcional Adicionar configuración de traspaso de gastos, el resto de las descripciones de los requisitos funcionales pueden consultarse en el Anexo 1.

Descripción del requisito Adicionar configuración de traspaso de gastos

Precondiciones	El usuario debe tener permisos para llevar a cabo esta operación. Deben estar definidos los centros de costos y los elementos de gastos asociados a sus cuentas de control. Deben haberse efectuado las asociaciones entre cuentas de gastos, centros de costos y elementos de gastos patrimoniales.
Flujo de eventos	
Flujo básico Adicionar configuración de traspaso de gastos.	
1. Se especifica que se desea adicionar una nueva configuración de traspaso de gasto.	
2. El sistema valida que se pueda proceder a la inserción de una configuración de traspaso de gasto (ver validación 1)	
3. El sistema habilita una ventana para la inserción de los datos correspondientes a la	

	configuración, en la que el campo Número se llena de manera automática.
4.	Se introducen los datos de la configuración de traspaso: Descripción del traspaso. Criterio de distribución (los mismos pueden ser: por Saldo, por Subelemento, por Bases combinadas, por Porciento o por Volumen). Cuenta de gasto, en este campo se cargarán las cuentas de gastos que se encuentren vigentes en el sistema cuyas asociaciones se encuentren activas.
5.	El sistema valida si de acuerdo al criterio de distribución escogido es necesario especificar nuevos datos (ver validación 3).
6.	Se selecciona el centro de costo que se va a definir como centro de costo origen. Del mismo debe definirse el elemento de traspaso.
7.	El sistema valida (ver validación 2) los datos introducidos.
8.	El sistema permite especificar los centros de costos destinos, para ello primeramente se debe seleccionar la cuenta asociada a estos para acceder a los mismos.
9.	Por cada uno de los centros de costos destinos especificados, se debe definir : Elemento de traspaso.
	Se procede a definir la configuración del traspaso.
10.	El sistema emite un mensaje de confirmación para obtener aprobación por parte del usuario.
11.	Si el usuario confirma, el sistema procede a validar los datos introducidos (ver validación 2, 4 y 5)
12.	Si los datos son correctos el sistema los registra.
13.	El sistema confirma el registro de los datos.
14.	Concluye el requisito.
Pos-condiciones	
2.	Se registró en el sistema una configuración de traspaso de gasto.
Flujos alternativos	
Flujo alternativo 7.a Información incompleta	
1	El sistema señala los datos vacíos y permite corregirlos.
2	El usuario corrige los datos.

3	Volver al paso 3 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 7.b Información errónea	
1	El sistema señala los datos erróneos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 3 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 10.a La base de distribución seleccionada es por Bases combinadas	
1	El sistema habilita una ventana con todos los elementos que están asociados al centro de costo destino especificado, para que se proceda a escoger los elementos que se quieren traspasar, de los mismos se muestran el código y la descripción.
2	Se especifican los elementos que se desean traspasar.
3	El sistema valida el registro de los datos (ver validación 5)
4	Si se seleccionó al menos uno de los elementos, volver al paso 4 del flujo básico.
Pos-condiciones	
1	N/A.
Flujo alternativo 10.a La base de distribución seleccionada es por Subelementos	
1	El sistema habilita una ventana con todos los elementos que están asociados al centro de costo destino especificado, para que se proceda a escoger el elemento que se quiere traspasar, de estos se muestra el código y descripción.
2	El sistema valida el registro de los datos (ver validación 5)
3	Si se seleccionó el elemento de traspaso, volver al paso 4 del flujo básico.
Pos-condiciones	
1	N/A.
Flujo alternativo 13.a Se han especificado centros de costo destinos con diferentes criterios de distribución.	
1	El sistema emite un mensaje notificando al usuario la existencia de errores en la definición de

	los criterios de distribución de los centros de costos destinos.
2	El usuario corrige los datos.
3	Volver al paso 8 del flujo básico.
Pos-condiciones	
1	N/A.
Flujo alternativo *.a El usuario cancela la acción	
1	Concluye el requisito.
Pos-condiciones	
1	No se registran los datos.
Validaciones	
1	El sistema valida que existan cuentas de gastos vigentes que sean analizadas por subelementos cuyas asociaciones se encuentren activas.
2	El sistema verifica el criterio de distribución seleccionado, en caso de que sea por Bases combinadas, habilitará una ventana para que se especifiquen los elementos que se desean involucrar en el traspaso, para el caso en el que se haya seleccionado un criterio de distribución por Subelementos, se mostrará la misma ventana pero sólo se dará la opción de seleccionar un elemento para el traspaso. Para el caso de los criterios de distribución por Saldo, Volumen y Porciento, no se procede a especificar ningún otro dato en la configuración; el valor del porciento y del volumen se especifica en la ejecución.
3	El sistema verifica que para todos los centros de costos de destinos se hayan especificado el mismo criterio de distribución.
4	El sistema valida que quede definido completamente al menos un centro de costo destino para realizar el traspaso, con todos los datos relacionados al mismo, dígase elemento de traspaso y criterio de distribución.
5	El sistema valida que se haya especificado al menos un elemento de los que se quieren traspasar para cuando el criterio de distribución es por Bases combinadas o por Subelementos.
Extensiones	N/A.

Conceptos	**Traspaso de gastos**	Visibles en la interfaz: número, descripción. Utilizados internamente: N/A.
	Cuenta	Visibles en la interfaz: código, descripción.
	Centro de costo	Visibles en la interfaz: código, descripción.
	Elemento de gasto	Visibles en la interfaz: código, descripción.
	Criterios de distribución	Saldo, Subelementos, Bases combinadas, Volumen y Porciento
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	

A continuación se muestra la validación del requisito funcional Adicionar configuración de traspaso de gastos. Ver Figura 4.

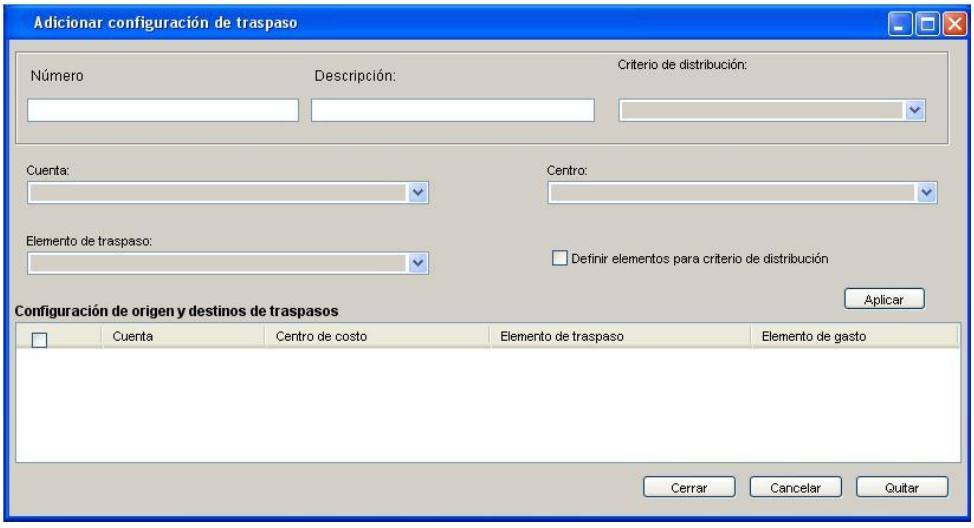


Figura 4: Prototipo de interfaz de usuario: Adicionar configuración de traspasos de gastos

La interfaz muestra los datos que son necesarios que el usuario ingrese en el sistema para crear un traspaso, además de las diferentes operaciones que puede realizar, como son: Aplicar, Cerrar y Cancelar.

2.5. Patrones de diseños utilizados

El diseño de la solución fue elaborado siguiendo patrones que constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. En este caso se emplearon los patrones generales de software para asignación de responsabilidades (**General Responsability Assignment Software Patterns**, **GRASP** acrónimo en inglés), los que describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman, 2009).

Dentro de los patrones GRASP se utilizaron:

Experto: Se emplea al trabajar con el marco de trabajo Sauxe y un ejemplo de ello es la inclusión de Doctrine Generator para mapear la base de datos, el mismo es el encargado de generar las clases para la gestión de las tablas en dicha base de datos con las responsabilidades debidamente asignadas. Cada clase cuenta con un grupo de funcionalidades que la convierte en experta de la información de la tabla a la que representa. Esto se puede observar en la clase (class ConfSecuenciasdistribucion extends BaseConfSecuenciasdistribucion) que es la encargada de realizar las acciones directamente a la BD.

Creador: Este patrón se evidencia en la clase (class OpsecuenciatraspasoController extends ZendExt_Controller_Secure), la misma contiene las acciones y es la encargada de ejecutarlas. Dentro de esta existen varias funcionalidades que crean varios objetos o instancias de las clases que representan cada una de las tablas existentes en la base de datos. Este patrón es adaptable a las clases del paquete Domain, quienes son las encargadas de crear los objetos de tipo Doctrine_Query, para permitir el acceso a la información almacenada a nivel de datos.

Alta cohesión: Sauxe presenta entre sus principales características la organización del trabajo en cuanto a estructura y responsabilidades bien definidas, esto permite que se trabaje con las clases con una alta cohesión. Un ejemplo de esto es la clase (class OpsecuenciatraspasoController extends ZendExt_Controller_Secure), la misma delega funciones específicas a otras clases, encargándose de definir las acciones a realizar.

Bajo Acoplamiento: Este patrón se evidencia en la aplicación ya que el sistema presenta poca dependencia entre las clases. Es en el modelo donde únicamente se encuentran algunas relaciones de asociación entre las clases, pero no representa una gran jerarquía. La característica principal de este patrón es mantener las clases más independientes entre sí y con la menor cantidad de relaciones; la misma posibilita que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las clases, asignándoles una responsabilidad para mantener un bajo acoplamiento.

Controlador: Este patrón asigna la responsabilidad del manejo de un mensaje de los eventos del sistema a una clase, los mismos pueden ser de alto nivel generado por un actor externo; es decir, un evento de entrada externa. La clase controladora definida denominada (class OpsecuenciatraspasoController extends ZendExt_Controller_Secure) es un ejemplo de la aplicación de este patrón, teniendo a su cargo la responsabilidad de manejar los eventos dentro del componente.

Dentro de los patrones GOF (Gang of Four en inglés) se utilizaron:

Cadena de Responsabilidad: Está concebido que ante la ocurrencia de un error al realizarse una determinada consulta a la base de datos el mismo sea manejado por el Modelo, creando una nueva excepción de tipo ZendExt_Exception. Dicha excepción debe ser propagada al Controlador, el mismo es el encargado de capturarla y enviarla a la Vista ya traducida, esta última por su parte debe mostrar un mensaje al usuario en un lenguaje entendible, notificando el error y sin especificar detalles del mismo. De esta manera se distribuyen las responsabilidades entre los diferentes componentes, evidenciándose por lo tanto el empleo de este patrón.

También es utilizado el patrón de diseño mostrado a continuación:

Inversión de Control (IOC): Este es un patrón de diseño pensado para permitir un menor acoplamiento entre los componentes de una aplicación y fomentar así el uso de los mismos. La utilización de este patrón en el sistema está reflejada en la creación y empleo de la clase IOC, ya que por la necesidad de utilizar varios servicios desde diferentes componentes, era necesaria la implementación de una clase donde son publicados los mismos, facilitando su interacción (Guillerón, 2012).

2.6. Diagrama de clases del diseño

El diagrama de clases permite visualizar las relaciones estructurales (herencia, agregación, asociación, etc.) entre las clases que involucran el sistema (Universidad de Computación de Chile, 2011). Los diagramas del diseño tienen como objetivo crear los modelos que permitan un entendimiento visual de los requerimientos del sistema y preparar el ambiente para su implementación y prueba, además su lenguaje es mucho más técnico y cercano a la programación ya que son precisamente los programadores los usuarios finales de los mismos. (Project, 2011)**.** A continuación se muestra el diagrama de clases del requisito funcional **Adicionar configuración de traspasos de gastos**, el resto de los diagramas de clases se encuentran en el Anexo 2.

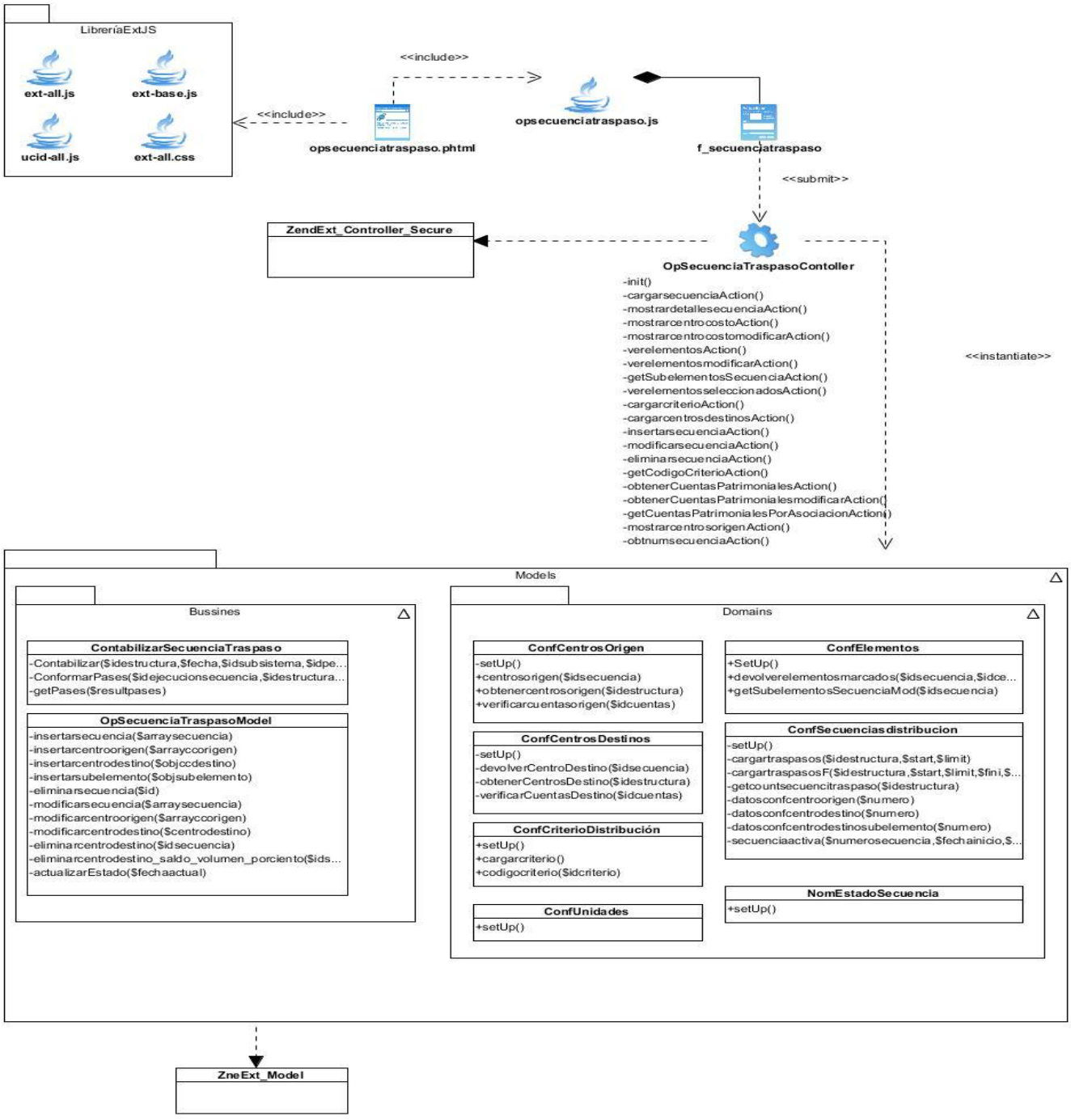


Figura 5: Diagrama de clases del diseño

Descripción del diseño de clases del proceso

Clases	Descripción
Librería ExtJS	Contiene los componentes generados a través de la librería Java Script ExtJS.
opsecuenciatraspaso.phtml	Responsable de visualizar a través de los js la información necesaria del proceso traspaso de gastos.
opsecuenciatraspaso.js	Debe generar de forma dinámica a través del DOM y utilizando la librería ExtJS componentes a través de los cuales se adicionen y se modifiquen los traspasos. Responsable de enviar y recibir los datos de la controladora utilizando tecnología AJAX.
OpSecuenciaTraspasoContoller	Clase controladora que sirve de puente entre la vista y el modelo. Contiene los métodos necesarios para adicionar, modificar, eliminar o ejecutar los traspasos de gastos.
ZendExt_Controller_Secure	Encargada de gestionar acciones personalizadas y está integrada a la seguridad.
Paquete Models	Encargado de manejar los datos persistentes dentro del componente. Contiene el Bussines y el Domain.
ZendExt_Model	Modelo gestor de negocio que permite entre otras funcionalidades iniciar la conexión a la base de datos

Tabla2: Descripción de las clases del diseño

2.7. Modelo de datos

Un modelo de datos es la descripción de una base de datos. Típicamente un modelo de datos permite describir las estructuras de datos de la base, su tipo, descripción y la forma en que se relacionan, restricciones de integridad, entre otros. En este sentido es factible pensar que un modelo de datos permite describir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan esos elementos entre sí (Chronotech, 2011).

El modelo de datos propuesto en la solución cuenta con un total de 9 tablas, dicho modelo tuvo poca modificación ya que se utilizaron todas las tablas definidas en la anterior solución, agregándose la tabla nom_estadosecuencia, la misma almacena el estado en que se encuentran las secuencias registradas. Dicho modelo está en 3era forma normal. Del análisis se obtuvo finalmente el siguiente modelo de datos:

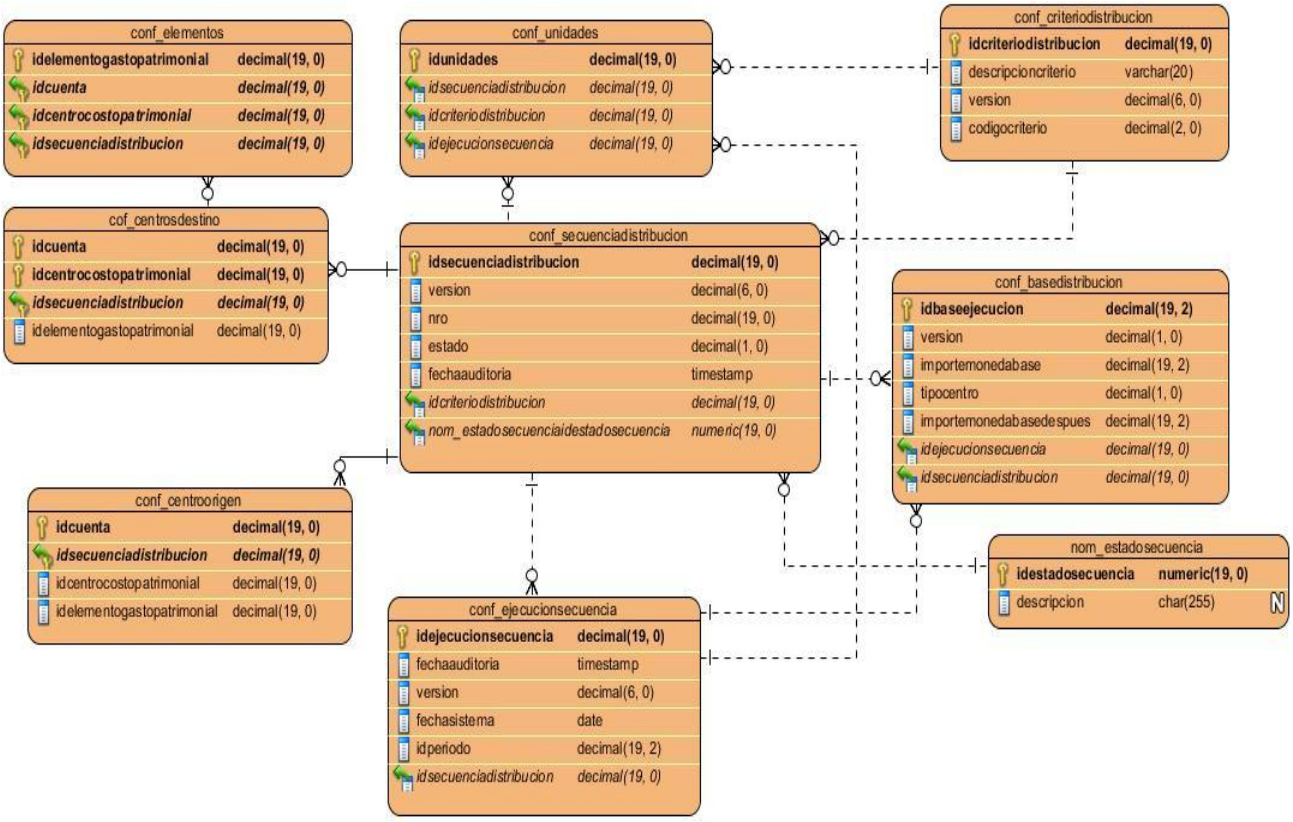


Figura 17: Modelo de datos

2.7.1. Descripción de las tablas

Posteriormente se muestran las descripciones de las tablas del modelo de datos del sistema. El resto de las tablas se encuentran en el Anexo 3.

Nombre: "Conf_centrosdestino"		
Tipo: Entidad		
Descripción: Almacena los datos de los centros de costos destinos.		
Atributo	**Tipo**	**Descripción**
idcuenta	decimal	Identificador de la tabla.
idcentrocostopatrimonial	decimal	Identificador de la tabla.

idsecuenciasdistribucion	decimal	Identificador de la tabla "Conf_elementos" (Llave foránea).
Idelementogastopatrimonial	decimal	Elementos de gastos patrimoniales.

Tabla 3: Descripción de la tabla "Conf_centrosdestino"

2.8. Prototipo de interfaz de usuario funcional

A continuación se muestra el prototipo de interfaz de usuario funcional del requisito Adicionar configuración de traspaso que se obtuvo como resultado del diseño del sistema. El resto de los prototipos de interfaz de usuario se encuentran en el Anexo 4.

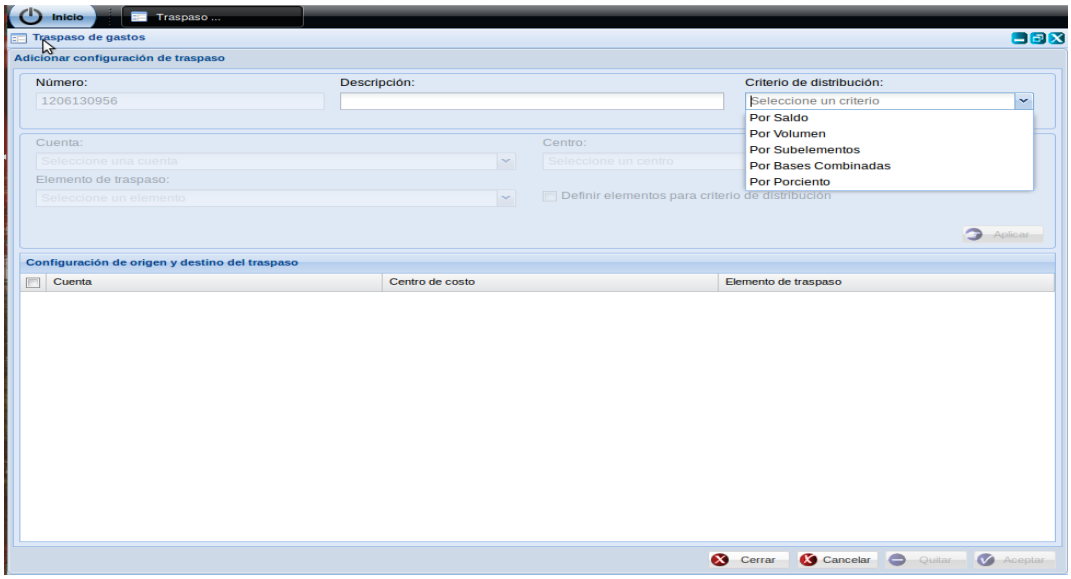


Figura 18: Prototipo de interfaz de usuario funcional

2.9. Conclusiones

A partir del estudio realizado en el capítulo se obtuvieron los siguientes resultados:

- Quedó definida la propuesta de sistema, se obtuvo el diagrama de proceso de negocio y la descripción del mismo.
- Quedaron definidos los requisitos funcionales y no funcionales además de sus respectivas descripciones.

- Quedó diseñado el modelo físico de datos, donde se almacenaron los datos que se gestionan en el componente.
- Como parte de la validación de los requisitos, se definieron los prototipos de interfaz de usuario para el componente que será implementado.
- Se logró fundamentar la base para la realización del presente capítulo donde fue posible llevar a cabo el análisis y diseño de dicho componente, permitiendo sentar las bases para la implementación.

Capítulo 3: Implementación y validación de la solución propuesta

3.1. Introducción

El presente capítulo tiene como objetivo principal validar la solución en cuestión, para esto se utilizan algunas métricas que se aplican en la actualidad para validar la calidad en el diseño de software, tales como TOC y RC. Las mismas ayudan a comprender todo el proceso técnico que se utiliza para desarrollar un producto, así como el propio producto. Se realiza una breve descripción de los estándares a utilizar durante la implementación del sistema. Se describen las pruebas de caja negra realizadas al software y los resultados obtenidos de estas, brindando una valoración de la solución del componente.

3.2. Diagrama de componentes

Un diagrama de componentes representa los componentes, sus interfaces y las relaciones de los componentes con las interfaces que utilizan (Brito, 2008)

A continuación se muestra el diagrama de componentes del sistema:

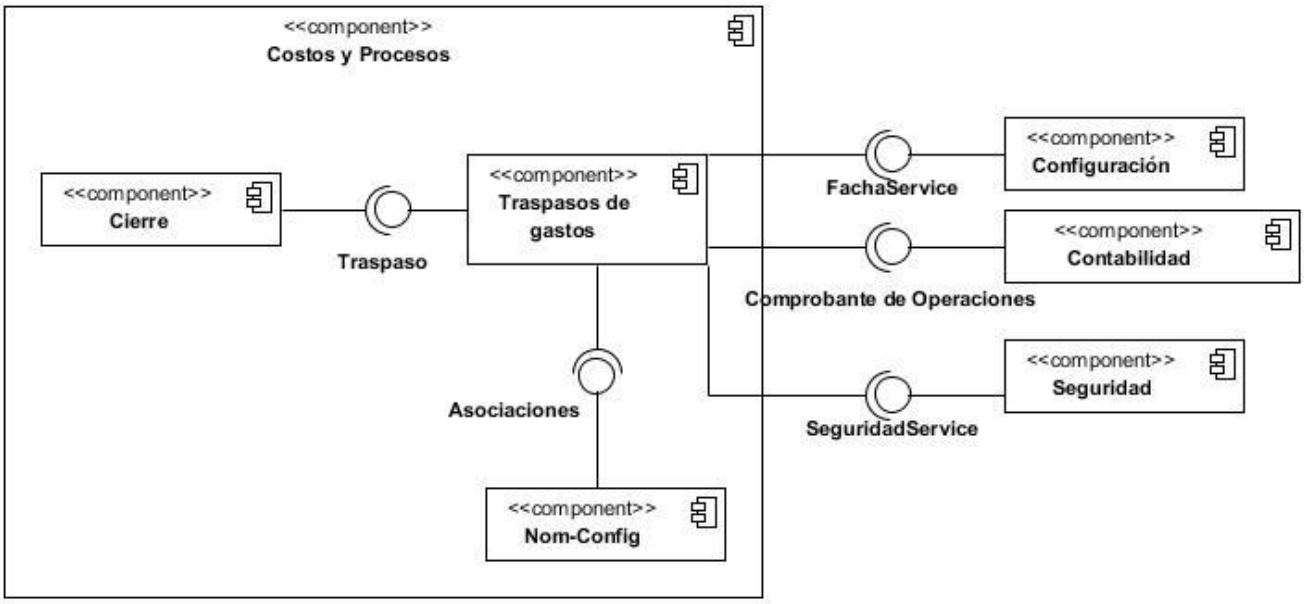


Figura 21: Diagrama de componentes: Traspasos de gastos

A continuación se explica la forma de interacción del componente Traspaso de gastos con los componentes Cierre, Nom-Config, Contabilidad, Configuración y Seguridad.

Traspaso de gastos: Es el encargado de la gestión de los traspasos de gastos, el cual recibe las cuentas asociadas a los centros de costos y a los elementos de gastos del Nomenclador de cuentas. Además al Cierre envía los traspasos en estado ejecutado y de Contabilidad consume el servicio de los comprobantes de operaciones. Por otra parte recibe el período y el ejercicio contable del componente de Configuración general. Además de consumir los servicios de autenticación del componente Seguridad.

3.3. Estándares de codificación

Los estándares de codificación son modelos de programación enfocadas a su estructura para facilitar la lectura, comprensión y mantenimiento del código. Se definieron normas de codificación con el fin de obtener un estándar en la implementación por el equipo de desarrollo que permitiera asegurar la calidad del software, obteniendo un código más legible y reutilizable.
A continuación se describen algunos de estos estándares empleados durante la implementación:

PascalCasing

El estándar PascalCasing establece que los identificadores, nombres de clases, variables, métodos o funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.
La nomenclatura de las clases de Costos y Procesos se realizó sobre la base de este estándar, usando palabras compuestas sugerentes acordes al propósito de la misma.

Nomenclatura de clases según su tipo

- **Controllers**: clases controladoras del negocio.

El nombre de la clase controladora debe estar estructurado por el nombre propio de la misma en mayúsculas seguido por la palabra Controller y heredar siempre de la súper clase del framework ZendExt_Controller_Secure. Ejemplo: OpSecuenciaTraspasoContoller

Clases de los modelos

Business: Clases modelo del negocio.
Las clases modelo tendrán por identificador el nombre de la tabla en la que trabajan seguido por la palabra Model y heredarán de la súper clase del framework ZendExt_Model. Ejemplo: OpSecuenciaTraspasoModel.

Domain: clases entidades del dominio.

Los archivos situados en el domain tienen el mismo nombre de las tablas que representan, definiéndolas como clases php, pueden incluir los prefijos Dat o Conf para diferenciar entre sus usos. Ejemplo: ConfCentrosOrigen

Generated: Clases bases del dominio

Las clases que se encuentran dentro de Generated comienza su nombre con la palabra: “Base”, seguido del nombre de la tabla en la Base de Datos. Ejemplo: BaseConfCentrosorigen

CamelCasing

El estándar CamelCasing es parecido al PascalCasing con la particularidad de que la letra inicial del identificador no comienza con mayúscula. Esta notación se utilizó para el nombre de funciones y atributos.

Nomenclatura de las funciones

El identificativo a emplear para las funciones o métodos se escribe con la primera palabra en minúscula utilizando la notación CamelCasing y nombres que deduzcan su propósito. Ejemplo: obtenercentrosorigen

Los denominadores de las acciones de las clases controladoras tienen la peculiaridad de ir seguidos por la palabra “Action”. Ejemplo: mostrardetallessecuenciaAction.

Nomenclatura de los atributos Las variables se nombran convenientemente de acuerdo con el estándar CamelCasing. Ejemplo: $idestructura, $idsecuencia.

3.4. Descripción de clases y funcionalidades del componente

A continuación se describen las clases y sus operaciones más importantes por componentes agrupándolas de acuerdo a la clasificación siguiente, el resto de las descripciones de las clases pueden consultarse en el Anexo 5.

Clases Controladoras: Las clases controladoras son responsables de la lógica de negocio, gestionando todo el flujo de datos y operaciones entre la vista y demás clases del negocio.

Nombre: OpSecuenciaTraspasoContoller
Tipo de clase: Controladora

Atributo	Tipo
idestructura	int
idperiodo	int
model	object
service	object
comprobante	object
idsubsistema	int
fechadelsubsistema	date
contabilizartraspaso	object

Para cada responsabilidad	
Nombre	**Descripción**
cargarsecuenciaAction()	Carga todas las secuencias ejecutadas como no ejecutada.
obtenerDatosCCDModificarAction()	Obtiene los centros destinos para ser modificados.
mostrardetallesecuenciaAction()	Muestra en la interfaz base, en la Crear y en la Modificar los centros destinos.
mostrarcentrocostoAction()	Muestra los datos del centro de costo.
verelementosAction()	Carga los elementos asociados a un centro de costo seleccionado.
getSubelementosSecuenciaAction()	Devuelve los subelementos de una secuencia.
verelementosseleccionadosAction()	Carga los elementos asociados a un centro de costo seleccionado en el modificar.
cargarcriterioAction()	Carga el criterio de distribución.
cargarcentrosdestinosAction()	Carga los centros destinos.
insertarsecuenciaAction()	Inserta una secuencia por cualquiera de los criterios de distribución.
modificarsecuenciaAction()	Modifica la secuencia por cualquiera de los criterios.

eliminarsecuenciaAction()	Elimina la secuencia siempre que no esté ejecutada.
getCodigoCriterioAction()	Devuelve el código del criterio.
ejecutarsecuenciaAction()	Ejecuta la secuencia.
insertardatejecSecuenciaAction($datocco, $datosccd)	Inserta en las tablas DatEjecucionsecuencia y ConfBaseejecucion los datos de la ejecución.
obtenersaldoAction($datosccd)	Se encarga de obtener los saldos de las cuentas, centros y elementos.
obtenersaldosubelementoAction($datosccd)	Obtiene el saldo del subelemento.
calcularingresoAction($datosccd, $saldocuentaorigen)	Se encarga de calcular el ingreso que se debe traspasar al centro destino.
calcularingresosubelementoAction($datosc cd, $saldocuentaorigen)	Calcula el ingreso del subelemento.
obtenerCuentasPatrimonialesAction()	Devuelve las cuentas de gastos patrimoniales.
mostrarcentrosorigenAction()	Muestra todos los datos de los centros origen.
obtnumsecuenciaAction()	Es generado un único número para cada secuencia.
obtenerfechactualAction()	Obtiene la fecha del servidor.
getVerEjecucionAction()	Obtiene todos los datos de la secuencia ejecutada.
getDatosComprobanteAction()	Devuelve el id de la ejecución de la secuencia.
getLlenGridComprobAction()	Obtiene los datos necesarios de los centros destinos para ver su ejecución.
getsaldo($cuenta, $centro)	Devuelve el saldo.
getbases($arraycuentacentro)	Devuelve las bases.
suma($array)	Suma las bases.
porciento($valor, $total)	Calcula el porciento que representa valor de total.
importe($porciento, $importe)	Devuelve el resultado del importe multiplicado por le porciento.

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

Tabla 4: Clase controladora OpSecuenciaTraspasoContoller.

3.5. Validación del diseño propuesto

Las métricas de software son una medida cuantitativa que permite a los desarrolladores tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software (Roger S.Pressman, 2009).
Las métricas escogidas como instrumento para evaluar la calidad del diseño descrito en el capítulo anterior y su relación con los atributos de calidad son las siguientes:

La métrica Tamaño operacional de clase (TOC): posibilita medir los siguientes atributos de calidad

Responsabilidad: Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.

Complejidad de implementación: Grado de dificultad que tiene implementar un diseño de clases determinado.

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

En la métrica TOC los atributos de calidad responsabilidad y complejidad de implementación son inversamente proporcionales a la reutilización, lo que puede ser traducido como, mientras mayor sea la responsabilidad y complejidad de implementación de una clase, menor será su nivel de reutilización.

Resultados de la evaluación de la métrica TOC

Como resultado de la evaluación de la métrica TOC se obtuvo lo siguiente:

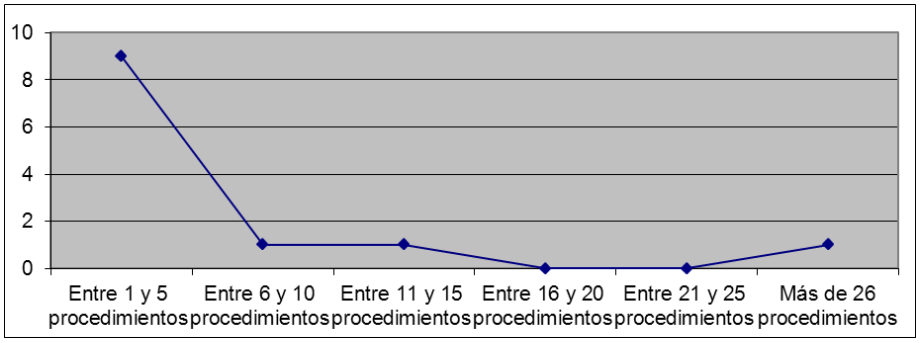


Figura 22: Representación de la cantidad de clases y el número de procedimientos que contienen

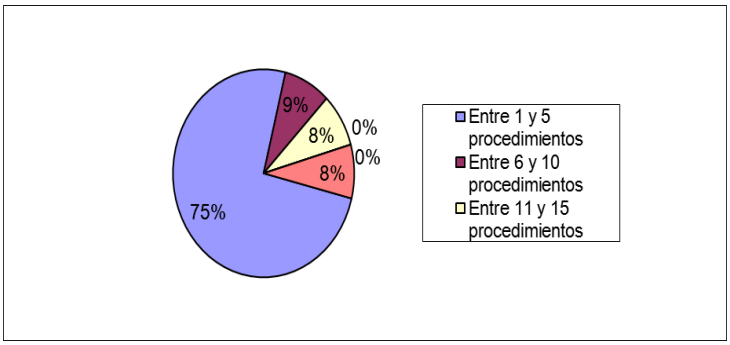


Figura 23: Representación en % de la cantidad de clases y el número procedimientos que contienen

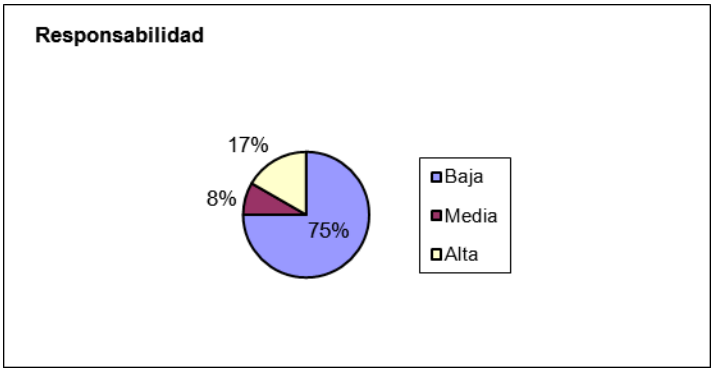


Figura 24: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo responsabilidad

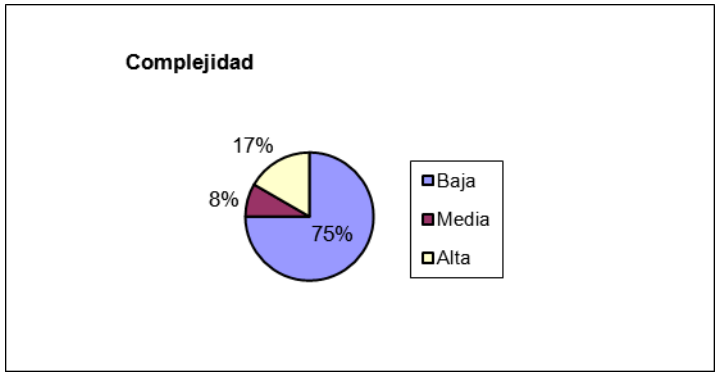


Figura 25: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo complejidad de implementación

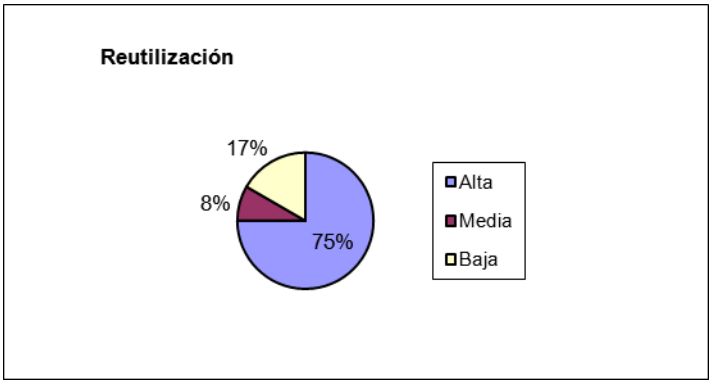


Figura 26: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo reutilización

Análisis de los resultados obtenidos en la evaluación de la métrica TOC

Luego de aplicar la métrica TOC, se puede concluir que el diseño propuesto para el componente Traspasos de gastos está entre los límites aceptables de calidad. Los atributos de calidad se encuentran en un nivel satisfactorio en la mayoría de las clases; de manera que se puede observar cómo se promueve la reutilización (elemento clave en el proceso de desarrollo de software) y cómo están reducidas en menor grado la responsabilidad y la complejidad de implementación, lo que promueve el bajo acoplamiento entre las clases y facilita la implementación y la comprobación al programador.

Métrica Relaciones entre clases (RC):

Se refiere al número de relaciones de uso de una clase. En la métrica RC los atributos acoplamiento, complejidad de mantenimiento y cantidad de pruebas son inversamente proporcionales a la reutilización,

es decir mientras mayor sea el acoplamiento, complejidad de mantenimiento de una clase y cantidad de pruebas, menor será su nivel de reutilización.

La métrica RC posibilita medir los siguientes atributos de calidad:

- **Acoplamiento:** Dependencia o interconexión de una clase o estructura de clase respecto a otras.
- **Complejidad del mantenimiento:** Nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto. Puede influir significativamente en los costes y la planificación del proyecto.
- **Reutilización:** Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.
- **Cantidad de pruebas:** Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto (componente) diseñado.

Resultados de la evaluación de la métrica RC

Como resultado de la evaluación de la métrica RC se obtuvo lo siguiente:

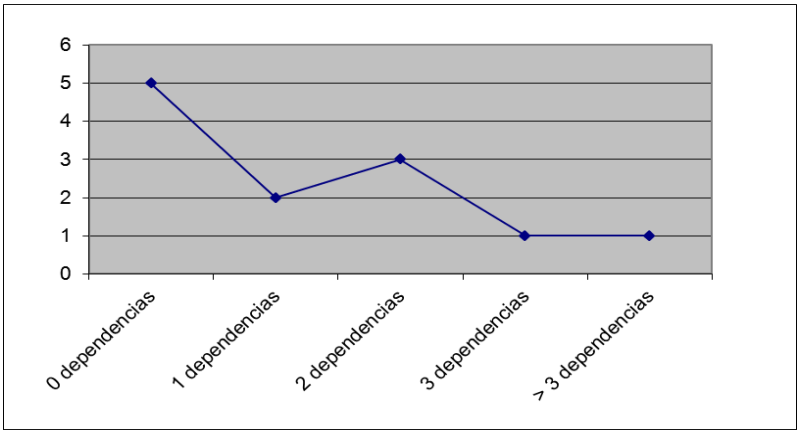


Figura 27: Representación de las asociaciones de uso por cantidad de clases.

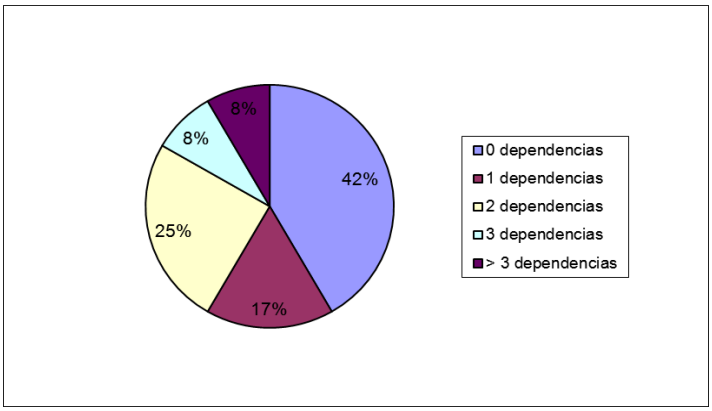


Figura 28: Representación en % de las asociaciones de uso por cantidad de clases

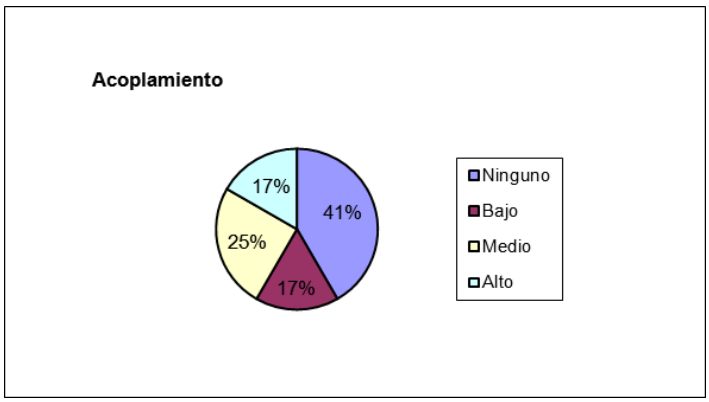


Figura 29: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo acoplamiento

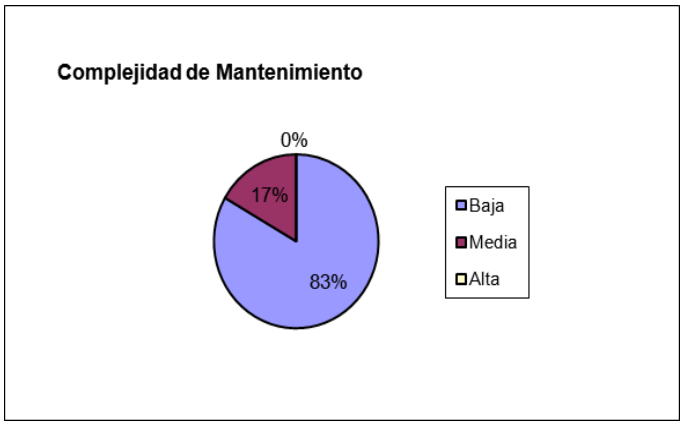


Figura 30: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo complejidad de mantenimiento

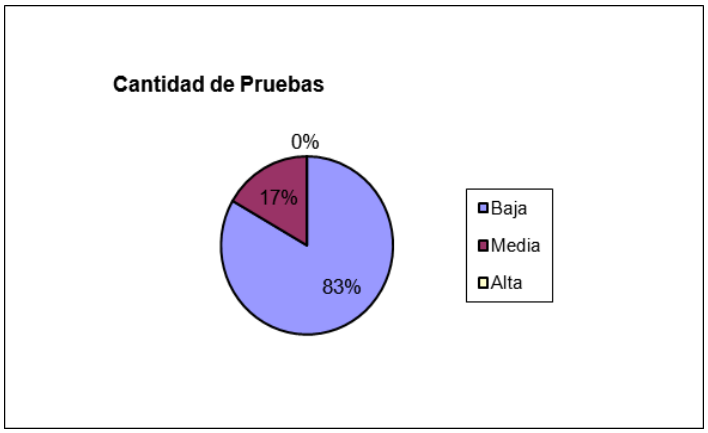


Figura 31: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo cantidad de pruebas

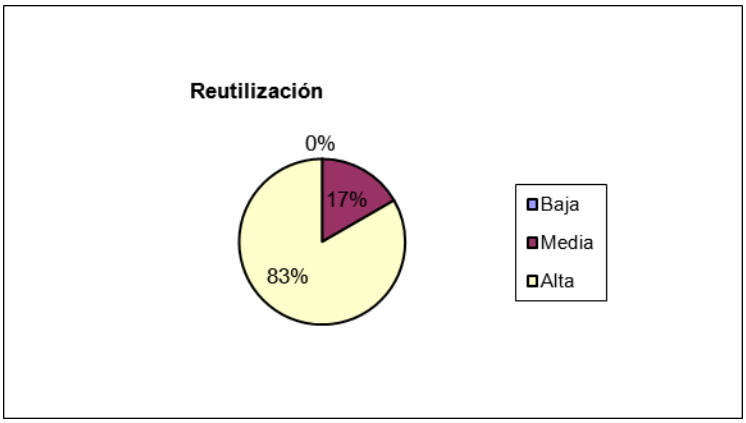


Figura 32: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo reutilización

Análisis de los resultados obtenidos en la evaluación de la métrica RC

Mediante la evaluación de la métrica RC se puede deducir que el diseño del componente tiene una calidad aceptable teniendo en cuenta que el 83% de las clases presentes en el diseño cuentan con una baja cantidad de relaciones de uso. Como se puede observar en las gráficas anteriores las clases promueven un bajo nivel de acoplamiento, además la complejidad de mantenimiento y la cantidad de pruebas que se realizarán serán pocas. En consecuencia a esto el grado de reusabilidad es mayor. De forma general el resultado de la aplicación de estas métricas demuestra que las clases no se encuentran muy sobrecargadas en responsabilidad, existe además bajo acoplamiento entre las mismas y presentan un alto

nivel de reutilización. Indican que el diseño no es complejo, pues la complejidad de mantenimiento es baja, así como la complejidad en las pruebas.

3.6. Validación de la implementación

Durante el proceso de desarrollo de software las posibilidades de errores son múltiples por lo que se hace necesario detectar a tiempo las imperfecciones e irregularidades del mismo, además de proporcionar una visión objetiva de la madurez y calidad de los procesos asociados.
Las pruebas de software son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador, probando el comportamiento del mismo, que determinan en cierto grado la calidad de un producto. Las pruebas permiten verificar si el sistema cumple con los requisitos propuestos por el cliente. El aspecto fundamental que rige esta etapa es determinar mediante las pruebas, cómo y en qué sentido el componente Traspasos de gastos cumple con las expectativas del cliente, a partir de los requisitos establecidos (Desarrollo, 2009). En ese ámbito se trazan un conjunto de objetivos dentro de los que se sitúan:

- Verificar la implementación del componente Traspaso de gastos.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar los errores y asegurar que estos sean corregidos de la mejor manera.

3.6.1. Métodos de Prueba:

Son un enfoque sistemático, independiente del nivel en que se enmarque la prueba, que ayuda a encontrar buenos conjuntos de casos de prueba para detectar diferentes tipos de errores. Existen dos enfoques alternativos, descritos como:

Caja Negra: Se comprueban las funcionalidades sin tener en cuenta la estructura interna.

Caja Blanca: Se comprueban los componentes internos.

Al componente fue aplicado el método de caja negra, descrito en el siguiente epígrafe.

Pruebas de caja negra

Este tipo de prueba se centra en lo que se espera del software, es decir, los casos de prueba pretenden demostrar que las funciones del sistema son operativas, que los valores de entradas se aceptan adecuadamente y que se produce una salida correcta, sin preocuparse de lo que pueda estar haciendo el software internamente, es decir, todas las pruebas se realizan sobre la interfaz de usuario.
Las pruebas que se realizan al componente Traspaso de gastos tienen el objetivo de verificar que las funciones son operativas a través de la interfaz del software, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, manteniendo la integridad de la información externa (Zorrilla, 2010). En esencia estas pruebas permiten encontrar:

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

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa funcione correctamente.

Para esta prueba existen varias técnicas (Pressman, 2008):

1. Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

2. Técnica del Análisis de Valores Límites: esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

3. **Grafos de causa-efecto:** Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Para la validación del sistema se hace uso de la técnica Partición de Equivalencia ya que permite examinar los valores válidos e inválidos de las entradas existentes en el sistema, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar. Para cada uno de los requisitos funcionales fueron diseñados los casos de prueba, los cuales son los siguientes. El resto de los casos de pruebas se encuentran en el Anexo 6.

Caso de prueba para el requisito Adicionar configuración de traspaso de gastos:

Condiciones de ejecución:

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar el subsistema de Costo y Procesos.
- Se debe seleccionar la opción **Traspaso de gastos /Traspaso de gastos.**
- Deben haberse efectuado las asociaciones entre cuentas de gastos, centros de costos y elementos de gastos patrimoniales y las mismas deben encontrarse activas, o sea, los centros y/o elementos asociados a estas cuentas no deben encontrarse inactivados en el sistema.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar configuración de traspasos de gastos.	El sistema permite adicionar una nueva configuración de traspasos de gastos, para ello se precisan todos los datos necesarios para su definición, dentro de los que se declaran	EP 1.1: Adicionar una nueva configuración de traspasos de gastos introduciendo los datos válidos.	➢ Se presiona el botón **Adicionar.** ➢ Se inserta la descripción, el criterio de distribución, la cuenta a partir de la cual se seleccionará el centro de costo con el que se efectuará el traspaso, el centro de costo y el elemento de traspaso. ➢ Si el criterio seleccionado es por Subelemento y Bases Combinadas se

	los centros de costo de origen, los centros de costo de destino, el elemento de traspaso y el criterio de distribución por el cual se realizará la distribución de los saldos de los centros de origen a los centros de destino.	debe especificar los subelementos de traspasos. En el caso de ser por Subelementos se adiciona además de los datos de la secuencia un único Subelemento de gasto y en caso de ser por Bases Combinadas se adicionan tantos como el usuario desee. ➢ Se presiona el botón **Aplicar.** ➢ Se muestra un mensaje "Está seguro que desea realizar la operación"**.** ➢ Se presiona el botón Aceptar del mensaje, se muestra el centro de costo origen en la interfaz. ➢ Se selecciona nuevamente una cuenta para especificar los centros de costos destinos. ➢ Se seleccionan, los centros de costos asociados a la cuenta seleccionada. ➢ Se selecciona el elemento de traspaso. ➢ Se presiona el botón **Aplicar.** ➢ Se muestra un mensaje "Está seguro que desea realizar la operación"**.** ➢ Se presiona el botón Aceptar del mensaje, se muestra el centro de costo destino en la interfaz. ➢ Se presiona el botón **Aceptar.** ➢ Se muestra un mensaje "Está seguro que desea realizar la operación"**.** ➢ Se presiona el botón Aceptar del mensaje ➢ Se muestra un mensaje "La operación se realizó correctamente"**.** ➢ Se presiona el botón Aceptar del

		mensaje.
	EP1.2: Adicionar una nueva configuración de traspasos de gastos, dejando campos requeridos en blanco.	➢ Se introducen los datos dejando campos requeridos en blanco. ➢ Se presiona el botón **Aplicar** de la ventana ➢ Se muestra ➢ un mensaje "Por favor verifique nuevamente porque existen campo(s) vacío(s)"**.** ➢ Se presiona el botón **Aceptar** del mensaje de error.
	EP 1.3:Cerrar	➢ Se presiona el botón **Adicionar.** ➢ Se procede o no a definir la configuración de traspaso. ➢ Se presiona el botón **Cerrar.** ➢ El sistema muestra el mensaje ¿¨Está seguro que desea salir de la configuración¨? ➢ Se presiona el botón Aceptar del mensaje.
	EP 1.4:Cancelar	➢ Se presiona el botón **Adicionar.** ➢ Se procede o no a definir los la configuración de traspaso. ➢ Se presiona el botón **Cancelar.** ➢ El sistema muestra el mensaje ¿¨Está seguro que desea cancelar la operación¨? ➢ Se presiona el botón **Aceptar** del mensaje.
	EP 1.5:Quitar	➢ Se presiona el botón **Adicionar.** ➢ Se inserta la descripción, el criterio de distribución, la cuenta a partir de la cual

se seleccionará el centro de costo con el que se efectuará el traspaso, y el elemento de traspaso.

- Si el criterio seleccionado es por Subelemento y Bases Combinadas se debe especificar los subelementos de traspasos. En el caso de ser por Subelementos se adiciona además de los datos de la secuencia un único Subelemento de gasto un único subelemento de gasto y en caso de ser por Bases Combinadas se adicionan tantos como el usuario desee.
- Se presiona el botón **Aplicar.**
- Se muestra un mensaje ¿"Está seguro que desea realizar la operación"**?**
- Se presiona el botón Aceptar del mensaje, se muestra el centro de costo origen en la interfaz.
- Se selecciona nuevamente una cuenta para especificar los centros de costos destinos.
- Se seleccionan, los centros de costos asociados a la cuenta seleccionada.
- Se muestra(n) el(los) centro(s) de costo(s) destino(s) en la interfaz.
- Se selecciona el centro de costo destino que se desee quitar
- Se presiona el botón **Quitar**
- El sistema muestra el mensaje ¨Está seguro que desea realizar esta operación.¨
- Se presiona el botón **Aceptar** del

	mensaje. ➢ Se elimina el centro de costo seleccionado.

3.6.2. Resultado de las pruebas:

De esta forma se realizaron las pruebas de funcionalidad a los requisitos del sistema. Se realizó un total de 9 casos de prueba, los cuales se encuentran especificados en los documentos de casos de pruebas para el sistema Cedrux correspondiente a cada uno de los requisitos funcionales.

El sistema fue sometido a dos iteraciones de pruebas de caja negra. En la primera iteración se detectaron 16 no conformidades, de las cuales 9 fueron de aplicación y 7 de documentación. Dentro de las no conformidades referentes a las de aplicación se encuentran 4 de validación, 2 de interfaz y 3 de funcionalidad. En cuanto a las de documentación se encontraron 4 no correspondencia del DCP con la aplicación, 1 de taxonomía y 2 de otras. Todas estas no conformidades fueron resueltas seguidamente de haberlas detectado, lo que permitió que el sistema pasara a una segunda iteración en la que no se detectaron no conformidades, obteniendo resultados satisfactorios.

La aplicación desarrollada fue presentada ante el cliente, el mismo luego de haberla probado estuvo satisfecho con el producto entregado. Prueba de esto lo constituye el acta de aceptación emitida por este, en el que consta que el sistema realizado cumple con las condiciones de calidad necesarias y satisface las necesidades plasmadas por el cliente, además contribuye a mejorar la gestión de las problemáticas de carácter científico de la facultad 3.

3.7. Conclusiones

A partir del estudio realizado en el capítulo se obtuvieron los siguientes resultados:

- Con la realización de este capítulo se aprovecharon al máximo en la implementación las funcionalidades de los marcos de trabajos utilizados para el desarrollo de la aplicación.
- Se validó el diseño utilizando algunas métricas para validar la calidad en el diseño de software, tales como TOC y RC las cuales arrojaron resultados satisfactorios sobre el diseño realizado, demostrando que este era simple y los atributos de calidad alcanzaban niveles favorables.

- Se realizaron al sistema las pruebas de caja negra a la solución desarrollada, permitiendo encontrar no conformidades, que se corrigieron a tiempo para obtener una aplicación que responde completamente a los requisitos.
- En el capítulo se pudo demostrar la correcta realización de un software con las características definidas en un inicio, permitiendo demostrar que el objetivo propuesto ha sido cumplido.

Conclusiones

Una vez concluido el presente trabajo de diploma se puede afirmar que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos:

- Se realizó el estudio de los diferentes sistemas existentes en el mundo que realizan procesos de contabilidad de costos, determinando las ventajas y deficiencias de estos, donde se tuvieron en cuenta las distintas funcionalidades que estos tienen integrados y que pudieran ser de beneficio para el componente a implementar, dando lugar al cumplimiento de las necesidades que el cliente desea. Evidenciándose de esta manera la necesidad de desarrollar una solución informática capaz de ejecutar dichas funcionalidades y que se ajusten a las necesidades del cliente.

- Fue realizado un estudio de las herramientas, tecnologías y el Modelo de desarrollo propuesto por la dirección de arquitectura del proyecto permitiendo sentar las bases para el desarrollo del trabajo de diploma.

- Quedaron definidos los requerimientos a informatizar donde fueron validados a través de la confección de los prototipos de interfaz de usuario.

- Las pruebas de caja negra permitieron detectar y corregir los errores no detectados durante la implementación posibilitando cumplir con las especificaciones requeridas y la validación del componente implementado.

- La implementación del componente se llevó a cabo con buena calidad y así lo demostraron los resultados arrojados por las métricas aplicadas y las pruebas al sistema.

Recomendaciones

Tomando como punto de partida los resultados obtenidos en la investigación se recomienda:

- Identificar nuevos requisitos para actualizar el sistema desarrollado y continuar realizando pruebas para garantizar una mayor funcionabilidad y aceptación del cliente.

- Validar la propuesta en ambientes de pruebas más amplios y complejos.

Bibliografía

Agencia de Control y Supervisión. 2010. Ministerio de la Informática y las Comunicaciones. . [Online] 2010. [Cited: diciembre 2012, 2012.] http://www.acs-mic.cu/sw_certificados.htm..

Ciberaula APACHE. 2011. Una Introducción a. Ciberaula. . [Online] 2011. [Cited: Diciembre 13, 2012.] http://linux.ciberaula.com/articulo/linux_apache_intro.

CITMATEL. Rodas XXI. 2012. Sistema Integral Económico. [Online] 2012. [Cited:] http://www.rodasxxi.cu/rodasxxi.php.

Editorial Colectiva. 2011. W3C Consortium. [Online] 2011. [Cited: Diciembre 5, 2012.] http://www.w3c.es/divulgacion/guiasbreves/tecnologiasxml.

Grupo de Autores . 2009. HTML.net. [Online] 2009. [Cited: Diciembre 13, 2011.] http://es.html.net/tutorials/html/lesson2.php.

Liannis Soria Barreda. 2010. ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE. 2010.

The PHP Group. 2009. . PHP: Hipertext Preprocesor. [Online] 2009. [Cited: Noviembre 29, 2012.] http://php.net/manual/es/intro-whatis.php.

Visual Paradigm. 2011. freedownloadmanager.org. [Online] 2011. [Cited: Diciembre 14, 2012.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p.

ASP.Net. 2010. Micrsoft. ASP.Net. [Online] 2010. [Cited: Enero 12, 2013.] http://www.asp.net/mvc.

Brito, Ing. Henry Raúl González and Ríos, Dr. José Carlos del Toro. 2008. Documento Visión ERP Cuba. La Habana : s.n., 2008.

Carnegie Mellon University. 2010. Software Engineering Institute. . [Online] 2010. [Cited: Noviembre 20 , 2012.] http://www.sei.cmu.edu/architecture/start/glossary/community.cfm.

CENDITEL SVN. 2009. Plataforma de Desarrollo de Software Libre (PDSL). Manual del Usuario del Sistema de Control de Versiones. [Online] 2009. [Cited: Noviembre 13, 2012.] http://plataforma.cenditel.gob.ve/wiki/ManualUsuarioSvn.

Chronotech, el futuro está en nuestras manos. Modelos De Datos . 2011. [Online] 2011. [Cited: Febrero 7, 2013.] http://sites.google.com/site/jalexiscv/modelosdedatos.

Clements, Linda Northrop y Paul. 2009. Software architecture: An executive overview. [Online] 2009. [Cited: Noviembre 2012, 20.] http://www.sei.cmu.edu/library/abstracts/reports/96tr003.cfm. Technical Report CMU/SEI-96-TR-003.

Desarrollo, Unidad de Compatibilización Integración y. 2009. Proceso de Desarrollo y Gestión de Proyectos de Software. Ciudad de la Habana : s.n., 2009.

Doctrine. 2011. Doctrine-Project.org. . [Online] 2011. [Cited: Noviembre 13, 2012.] http://www.doctrine-project.org.

E.U. Ingeniería Técnica Informática de Oviedo. 2010. Sitio web de la E.U.I.T.I.O. [Online] 2010. [Cited: Noviembre 5, 2012.] http://www.petra.euitio.uniovi.es%2F~i1667065%2FHD%2Fdocumentos%2FEntornos%2520de%2520Desarrollo%2520Integrado.pdf&ei=GannTrWxL8Pq.

Ecured Enciclopedia Cubana. 2011. Ecured Enciclopedia Cubana. [Online] 2011. [Cited: Enero 2012, 14.] http://www.ecured.cu..

ENTIDADES, CENTRO DE INFORMATIZACION DE LA GESTION DE. 2012. Modelo de desarrollo del centro CEIGE. s.l. : Universidad de las Ciencias Informáticas , 2012.

Eslomas, Patxi. 2011. Frameworks de Zend para el desarrollo de aplicaciones PHP. [Online] 2011. [Cited: Diciembre 2012, 2.] http://www.eslomas.com/index.php/archives/2007/07/03/framework-de-zend-para-el-desarrollo-de-aplicaciones-php.

Extend Js. 2010. [Online] 2010. [Cited: Diciembre 2012, 13.] http://extjses.com.

Firefox, Mozilla. 2010. GetFirefox. [Online] 2010. [Cited: Noviembre 22, 2012.] http://www.getfirefox.es/firefox-features.

Guillerón, Gastón. 2012. GLN Ingeniería & Consultoría. . [Online] 2012. http://glnconsultora.com/blog/?p=3.

Hoy, Informática. 2010. [Online] 2010. [Cited: Noviembre 30, 2012.]

JSON, Colectivo de. 2010. [Online] 2010. [Cited: Noviembre 23, 2012.] http://www.json.org/json-es.html.

Larman, Craig. 2009. UML y Patrones Introducción al análisis y diseño orientado a objetos. México: Prentice Hall : ISBN 0-13-748880-7 : s.n., 2009.

Martinez, Rafael. 2010. PostgreSQL-es. [Online] 2010. http://www.postgresql.org.es/node/297.

Meltom, Technologies. 2010. Gerencia y Negocios en HispanoAmérica. [Online] 2010. [Cited: Noviembre 12, 2012.]

Obregón, William González. 2012. MODELO DE DESARROLLO DE SOFTWARE. La Habana : s.n., 2012.

Openbravo. 2010. Manual de usuario v1.1. [Online] Openbravo S.L, 2010. [Cited: Noviembre 30, 2012.]

Oracle. 2011. NetBeans Corporation. [Online] 2011. [Cited: Noviembre 18, 2012.] http://netbeans.org/community/releases/70/.

Pressman, Roger. 2009. Ingeniería del Software: Un enfoque práctico. 2009.

Procesos, Costos y. 2011. Manual de Usuario Versat Sarasola V2. 2011.

Project, KDE. 2011. Elementos de UML. KDE Documentation. [Online] 2011. [Cited: Febrero 20, 2013.] http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html.

Proyecto SAUXE. 2012. 0120_Arquitectura de Software. Vista de integración. . La Habana : s.n., 2012.

Rolando Alfredo Hernández León, Sayda CoelloGonzález. 2011. El proceso de la investigación científica. 2011.

Romero, Omar Casasola. 2011. Programacion en Castellano. La Habana : s.n., 2011.

Scribd. 2009. [Online] 2009. [Cited: Diciembre 13, 2012.] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

2007. Sparx Systems. [Online] Sparx Systems Pty Ltd, 2007. [Cited: mayo 5, 2011.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

Tecnológico. 2010. Tecnologico . [Online] 2010. [Cited: Febrero 12, 2013.]

Tecnomática. 2008. [Online] 2008. http://tecnomatica.minbas.cu/Sistcont.html.

Universidad de Computación de Chile. 2011. Departamento de Ciencias de la. Sitio Web DCC. [Online] 2011. [Cited: Febrero 24, 2013.] http://www.dcc.uchile.cl/~psalinas/uml/modelo.html.

Vaquerano, Luisa. 2010. Gestion de los Costos. [Online] 2010. http://www.zanzivar.com/zzc/index.php..

W3Schools. 2010. W3Schools Online Web Tutorials. [Online] 2010. [Cited: Diciembre 2012, 2012.] http://www.w3schools.com/js/js_intro.asp..

Zorrilla, Mairelys Fernández and Rivera González, Osley. 2010. Diseño e implementacion del componente Ajuste al Costo del Subsistema Costo y Procesos del Sistema Integral. La Habana : s.n., 2010.

Glosario de términos

Costo: Es la expresión en términos monetarios de las cuantías asignados a la elaboración de un producto, a la prestación de un servicio, o los valores invertidos en las compras de productos con destino a su comercialización.

Gasto: Expresan el monto total en términos monetarios de los recursos materiales, laborales y monetarios utilizados en la producción de bienes materiales, la prestación de servicios o el desarrollo de cualquier actividad interna de una entidad en un período determinado. Se registran por elementos, los cuales permiten la cuantificación de los gastos y expresan el origen de los mismos.

Entidad: Es toda colectividad que puede considerarse como una unidad.

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.

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.

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

Gastos indirectos de producción: Comprenden los importes de los gastos en los que se incurren en las actividades asociadas a la producción, no identificables con un producto o servicio determinado. Incluyen los gastos de las actividades de mantenimiento, reparaciones corrientes y explotación de equipos, dirección de la producción, control de calidad, depreciación de activos fijos tangibles de producción y servicios auxiliares a ésta, entre otros.

Gastos directos de producción: Comprenden los importes en los que se incurren en las actividades asociadas a la producción, identificados con un producto o servicio determinado, ya sea en materia prima o mano de obra.

Cedrux: Denominación otorgada al ERP Cubano desarrollado por la UCI, su nombre se origina de la combinación de las primeras cuatro letras de Cedro y las dos últimas letras de Linux. El Cedro por ser una

planta de tronco duro, persistente, robusto y Linux por las nuevas concepciones de la migración a software libre en el territorio nacional para lograr la soberanía e independencia tecnológica.

DOM: Una página HTML normal no es más que una sucesión de caracteres, por lo que es un formato muy difícil de manipular, por ello los navegadores web se encargan de transformar las etiquetas en nodo.

ORM: Acrónimo de Object Relation Mapper es una técnica de programación que permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, es decir, las tablas de la base de datos pasan a ser clases y los registros objetos que se pueden manejar con facilidad.

Framework: La palabra inglesa framework define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

Métricas: las métricas nos ayudan a entender tanto el proceso técnico que se utiliza para desarrollar un producto, como el propio producto. El proceso para intentar mejorarlo, el producto se mide para intentar aumentar su calidad.

Web Services: Un servicio web es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet.