Universidad de las Ciencias Informáticas

Facultad 3

Título: Diseño de la versión 1.1 del componente Reglas del subsistema Configuración de Cedrux 1.0

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Yasmany Magdaleno Ramos

Tutor: Msc. César Lage Codorníu

Ciudad de la Habana, __ de _____ del 2011.

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

Yasmany Magdaleno Ramos

_____

Firma del Autor

Msc. Cesar Lage Codorniu

_____

Firma del Tutor

Agradecimientos:

A todas aquellas personas que han influido en mi formación como profesional.

En especial a mi tutor. (Sin que quepa duda es el mejor tutor del mundo, se los aseguro yo).

.A mis padres y familiares por haberme dado todo su apoyo a pesar de estar lejos de ellos.

A mi novia querida Liliana Ramírez Zayas que siempre estuvo apoyándome.

A todas los amigos y compañeros que me han apoyado y brindado su ayuda durante todos estos años.

Y en especial a la Revolución que sin la misma no me hubiese hecho ingeniero.

Dedicatoria:

Dedico el presente Trabajo de Diploma: A Mis padres, por la dedicación, amor, y educación que me han dado durante toda mi vida.

Especialmente a mi Madre querida por ser la mejor madre del mundo.

A mi hermana querida por darme todo su amor.

A mi novia querida Liliana Ramirez Zayas por apoyarme tanto en estos dos últimos años de mi carrera.

Y a todos los seres queridos que me rodean y yo siento tantos aprecios por ellos como son toda mi familia y mis amigos

Resumen

En la Universidad de las Ciencias Informáticas (UCI) se desarrolla un Sistema de Planificación de Recursos Empresariales (ERP), denominado Cedrux. En el mismo se creó un componente con el nombre de Reglas Contables cuyo objetivo radica en la generación automática de comprobantes de operaciones. Dicho componente permite la asociación de documentos primarios, hechos contables y cuentas para la definición de reglas de contabilización. La versión actual cumple con los objetivos previstos, sin embargo no cubre todos los escenarios identificados durante las pruebas piloto del sistema.

Para abordar la temática presentada; en este documento se expone un análisis del diseño de la solución actual, la revisión de algunos sistemas ERP existentes en el ámbito nacional e internacional y el estudio de los patrones de diseño como herramienta para la definición de una nueva solución. Se presenta la relación de los requisitos funcionales del componente con los cambios propuestos, el diseño de lógica de negocio y el modelo de datos de una nueva versión del componente. Finalmente, se valida el diseño propuesto mediante métricas de Relación entre Clases y Tamaño Operacional de Clases y se aplica un análisis del cumplimiento de los cambios en los requisitos funcionales.

Palabras claves:

Reglas Contables, comprobante de operaciones, diseño de lógica de negocio.

Tabla de Contenido

Tabla de Contenido

Índice de figuras

Índice de tablas

Introducción

Actualmente en la UCI se realiza el desarrollo de varios proyectos informáticos, en los cuales la dirección del país presenta cierto interés. En la Universidad existen varios centros de desarrollo, en los cuales se encuentran distribuidos los proyectos que se están desarrollando en la institución. El Centro para la Informatización de la Gestión de Entidades (CEIGE), es el encargado del desarrollo del sistema ERP-Cubano.

En CEIGE se desarrolla un sistema de gestión de entidades llamado Cedrux, el mismo está dividido por módulos como son, Inventario, Capital Humanos, Contabilidad y Finanzas y Configuración, etc.

El subsistema de Configuración, uno de los módulos anteriormente mencionados, tiene un componente denominado Reglas Contables. Este componente presenta dos funcionalidades, que a su vez determinan la existencia de sus dos componentes internos. El componente Gestión de Reglas permite la asociación de documentos primarios, hechos contables y cuentas para la definición de reglas de contabilización. El segundo componente con el nombre de Generación de Comprobantes Contables su objetivo radica en la generación automática de comprobantes de operaciones.Este componente constituye un elemento de alto uso dentro de Cedrux, ya que todo hecho económico registrado en el mismo debe generar un comprobante de operaciones. Por tanto, su calidad es primordial en el buen desempeño del sistema.

Durante el período de pruebas piloto a que fue sometido el sistema Cedrux se detectaron problemáticas asociadas a las capacidades funcionales del sistema las cuales son:

- No se pueden asociar directamente objetos y documentos, esta asociación es necesaria para la restricción del uso de los documentos según la funcionalidad que el usuario desee.

- No se pueden crear operaciones que tengan más de un cuenta, donde esta última depende de determinadas particularidades de la operación.

- Los conceptos para clasificar operaciones se repiten innecesariamente y tienden a confundir al usuario.

- Se permiten errores en la asociación de subsistemas y objetos.

- El algoritmo de generación de comprobante esta engorroso por su mal diseño.

Adicionalmente se pudieron constatar dificultades en la capacidad de mantenimiento y soporte del componente las cuales son:

- Dependencia entre los componentes Gestión de Reglas y Generación de Comprobantes.
- Poca claridad en el diseño del componente.
- Mala codificación en las clases.
- Ausencia de los desarrolladores originales.

Problema de la Investigación:

La actual versión del componente Reglas del sistema Cedrux no cubre todos los escenarios funcionales identificados durante las pruebas piloto y presenta dificultades en el mantenimiento y soporte del mismo.

El objeto de estudio:

Arquitectura y diseño de componentes de sistemas de información empresarial

Campo de acción:

Arquitectura y diseño del componente Reglas del subsistema Configuración de Cedrux

Objetivos:

Objetivo general: Proponer el diseño para una nueva versión del componente Reglas del subsistema Configuración de Cedrux que cubra todos los escenarios funcionales identificados y permita facilidades para el mantenimiento y soporte del mismo.

Objetivos específicos:

- Elaborar el marco teórico de la investigación sobre la base de la solución existente, las propuestas de otros sistemas ERP y el uso de patrones de diseño.
- Proponer ajustes en los requisitos funcionales del componente Reglas.
- Proponer el diseño de la lógica de negocio y del modelo de datos para la nueva versión del componente Reglas.
- Evaluar los resultados obtenidos.

Se plantea como ***idea a defender*** que el desarrollo del diseño de la versión 1.1 del componente Reglas del subsistema de Configuración cubra todos los escenarios funcionales identificados y permita facilidades para el mantenimiento y soporte.

Capítulo 1: Fundamentación Teórica.

En este capítulo se abordan definiciones y argumentos usados en la investigación, así como principales ERP existentes en el mundo y cómo los mismos desarrollan la contabilidad que es la tarea fundamental del componente. También se fundamenta de la situación actualmente existente en la Universidad respecto al componente y sobre el estudio de algunos patrones de diseño.

Capítulo 2: Características del Sistema

Se explica la integración de la solución propuesta a las reglas contables. Se desarrollan los diagramas de clases del diseño así como su respectivo diagrama de secuencias. También se da a conocer el modelo de entidad relación y el diagrama de componentes.

Capítulo 3: Validaciones

En este capítulo se valida todo lo propuesto en el diseño de la solución mediante una serie de métricas de software se hace un análisis de los requisitos funcionales y ventajas que los mismo pueden traer.

Capítulo 1: Fundamentación teórica

Introducción

En este capítulo se hace referencia a la fundamentación teórica del trabajo, donde se lleva a cabo la realización de un estudio tanto conceptual como del estado del arte, con el objetivo principal de detectar los diferentes sistemas y aplicaciones que tengan algunas funcionalidades semejantes a las del componente . Se realiza un estudio de las herramientas y tecnologías informáticas que están definidas para el desarrollo de aplicaciones web, las herramientas cases, los controladores de versiones de proyectos informáticos y navegadores web. También se realiza un minucioso estudio de las metodologías empleadas por algunos patrones de diseño con el objetivo de darle un nuevo diseño a la aplicación.

1.1 Reglas contables

1.1.1 Reglas de negocio

No existe una definición única para el término regla de negocio (Business Rule).

El Business Rule Group (1) la define como:

“Una declaración que define o restringe algún aspecto del negocio. Su cometido es declarar una estructura relevante para el negocio o controlar e influir sobre el comportamiento del mismo”

Gartner (2) las define como:
“Políticas de negocio implícitas y explícitas que definen y describen una acción de negocio, donde las reglas implícitas son aquellas que se encuentran embebidas dentro de las aplicaciones”

El enfoque de reglas de negocio es una metodología y seguramente un conjunto de tecnologías en la cual se captura, publica, automatiza y modifican las reglas desde un punto de vista estratégico. El resultado es un sistema de reglas de negocio en el cual las reglas se encuentran separadas lógica y hasta físicamente de otros aspectos del sistema y son compartidas entre diferentes aplicaciones.

Existen muchas ventajas en que un sistema aplique reglas en el negocio y ellas son:

Mayor modularidad. Separar las reglas de negocio permite su reutilización desde cualquier aplicación. Se aseguran servicios modulares y reutilizables al utilizar un motor de evaluación de reglas. Dado que las reglas y por lo tanto la lógica del negocio se encuentra aislada, las responsabilidades en el desarrollo e integración se encuentran claramente definidas.

Mayor consistencia. Todas las reglas que se relacionan en forma lógica pueden organizarse y mantenerse en una ubicación centralizada, minimizando la duplicación de código y aumentando la consistencia.
Simplicidad. Se captura la regla de negocio en un formato que el analista puede entender. Se pueden definir diferentes formas de representación de reglas, como por ejemplo árboles y tablas de decisión.

Auto-descriptiva y Entendible. Una tabla de precios en una base de datos nunca podrá comunicar el escenario completo tan bien como una tabla de decisión. Se aumenta la colaboración entre las áreas de negocio y el departamento de sistemas al lograr un lenguaje común de comunicación. Test independiente. Dado que las reglas se testean en forma separada, además de asegurar la consistencia minimiza el esfuerzo a realizar cuando se pone en producción la aplicación.

Uno de los retos más importantes en el uso del enfoque de reglas es encontrar aquellas políticas empresariales y restricciones que rigen el negocio (3).

Este documento está más centrado en 1er concepto de reglas. Según las características antes mencionadas las que más se asemeja a Reglas contables son Mayor Consistencia y Mayor modularidad eso implica que no se van a usar las reglas en toda su amplitud teórica.

1.1.2 Solución particular de reglas contables en Cedrux

1.1.2.1 Conceptos de Regla

El primer paso en el estudio de la presente investigación lo constituye el análisis de la solución actual, para lo cual es imprescindible revisar los conceptos fundamentales que maneja.

Objeto

Se entiende por objeto al concepto fundamental que maneja un sistema de gestión o la entidad que constituye el núcleo del negocio que se modela. El sistema de gestión en cuestión constantemente afecta su estado, actuando sobre este en la mayoría de las operaciones que se realizan. En la mayoría de los casos el objeto se puede respaldar mediante una cuenta en el nomenclador de cuentas de la entidad. (4)

El objeto es aquel activo que identifique cualquier sistema, ejemplo de esto para el sistema de Capital Humano el objeto lo constituye la entidad Persona, donde a este objeto se le puede realizar varias operaciones como se puede dar baja, alta, pagarle, sancionarle, entre otros y es ahí donde entra en función otro concepto.

También puede darse el caso de que un sistema de gestión cuente con más de un objeto o que se decide por interés del desarrollo de la aplicación desagregar el objeto en otros más pequeños. Un sistema de Cobros y Pagos puede decirse que tiene varios objetos: Derecho de Cobro y Obligaciones de Pago. Estos objetos a su vez pueden desagregados en cuatro objetos: Derecho de Cobro a Cliente, Derecho de Cobro Fiscal, Obligación de Pago a Cliente, Obligación de Pago Fiscal; división que se realiza atendiendo a la diferenciación del Estado como un cliente especial.

Operación.

La operación tiene mucha relación con el uso que habitualmente se le otorga al término, igual que en el caso de los objetos está enmarcada en los sistemas de gestión y representa todas las transacciones que se realizan en un área de negocio de terminada. Una operación es el flujo de actividades que afecta el estado de un objeto y por tanto en muchos casos tiene expresión en la contabilidad de una empresa. En una aplicación estas operaciones también se pueden conocer como funcionalidades o movimientos. En el entorno empresarial se puede entender como el hecho económico que genera registros contables. (4)

Documento.

Un documento primario contiene toda la información básica que se genera en cada uno de los hechos económicos que ocurren en una empresa. En un sistema informático los documentos

primarios se mantienen como entidad que almacenan todos los detalles de las operaciones que se realizan sobre un objeto determinado, entidad que se trata como documento. (4)

Regla Contable.

Después de conocer estos conceptos se puede decir entonces que una Regla Contable es la asignación de una operación a un objeto donde la relación de cada una de estas operaciones queda plasmada en un documento, para luego lograr generar un comprobante de operaciones. Para una mejor visualización del flujo de la creación de una regla contable ver Figura 1. (4)

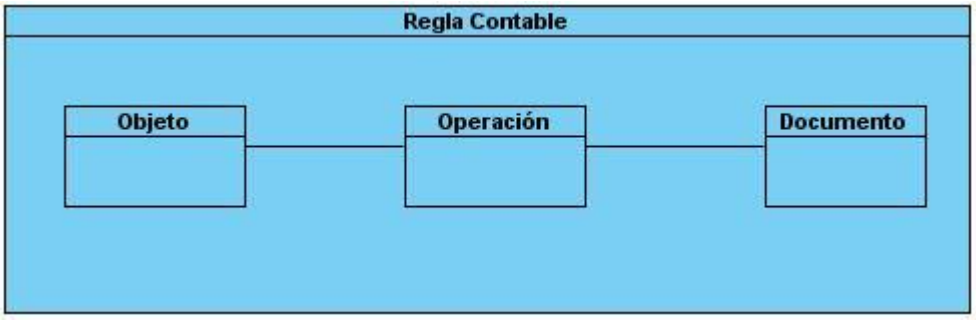

Figura 1(Regla contable).

1.1.2.2 Solución General

En la Figura 2 se puede ver a simple vista cómo está relacionado cada uno de los componentes con los que está fusionado el paquete de Reglas Contables. Uno de los problemas a resolver por su alto nivel de complejidad de mantenimiento es cómo está diseñado el modelo de componentes. En la relación entre los mismo existen 5 componentes. El componente Subsistema representa el Subsistema de Configuración pues si se está analizando un componente que se encuentra dentro del mismo no es necesario representarlo. El componente Generación de Comprobantes se encuentra separado del componente Gestión de Reglas el cual uno depende del otro y es ahí uno de los principales problemas de complejidad a la hora de dar mantenimiento a los mismos.

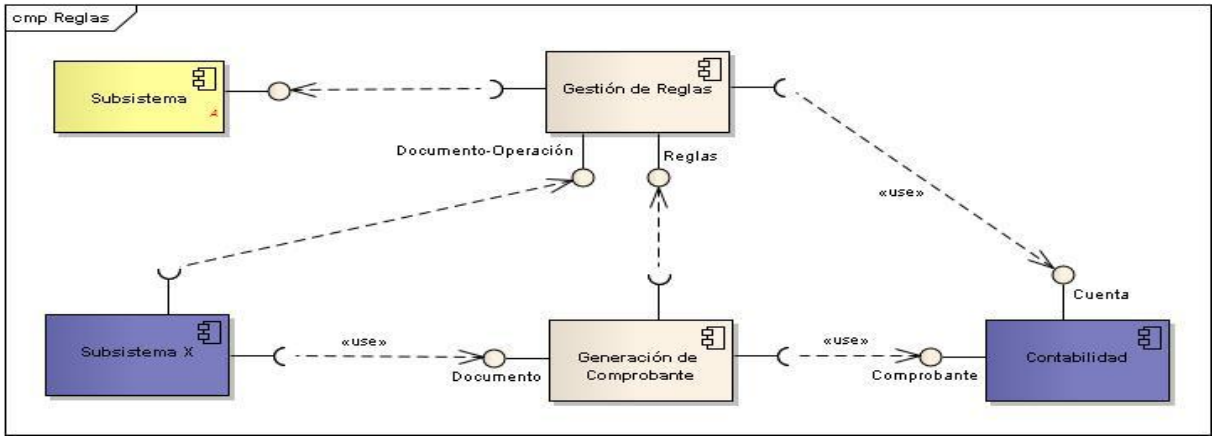

Figura 2(Modelo de componentes actualmente)

1.1.2.3 Componente Gestión de Reglas

El diseño existente trae como consecuencia errores en la asociación de subsistemas y objetos. Ya que el nomenclador **NomObjeto** debería traer consigo un atributo que le permitiera al objeto saber a qué subsistema pertenece. Otro de los problemas que trae consigo es que no se pueden asociar directamente objetos y documentos, donde esta asociación es necesaria para la restricción del uso de los documentos según la funcionalidad que el usuario desee. También se detectó que los conceptos para clasificar operaciones se repiten innecesariamente y tienden a confundir al usuario. Para poder visualizar lo dicho anteriormente ver Figura 3.

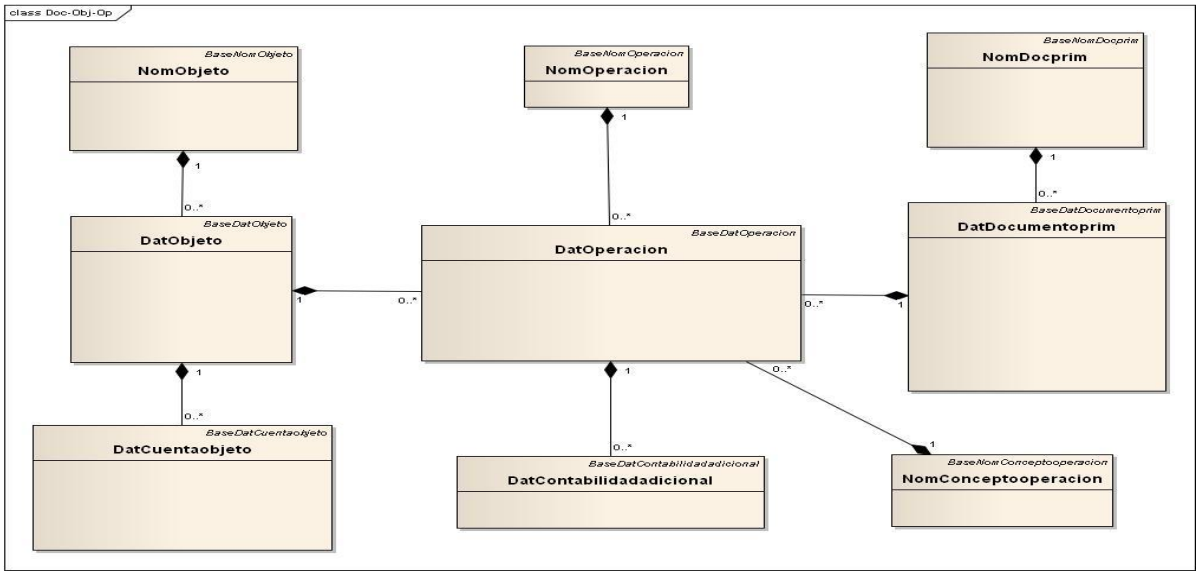

Figura 3(Modelo de dominio del componente Reglas)

1.1.2.4 Componente Generación de Comprobantes

La generación de comprobante es que a partir de un documento primario, generar un comprobante de operaciones. Para ellos se identifica el subsistema origen, el objeto sobre el que se trabaja, el documento portador de la información y las operaciones que se realizan. Luego busca, mediante un servicio del componente Gestión de Reglas, cuáles son los detalles de contabilización de esa combinación de conceptos. Con toda la información recopilada, ejecuta un algoritmo que le permite convertir el documento en comprobante de operaciones, partiendo del principio que conoce la estructura de un comprobante. Ahora ese algoritmo dado que su diseño fue algo engorroso no está dado por ningún tipo de patrón conocido hace que su complejidad de mantenimiento aumente y como dicho anteriormente al estar separado del componente que gestiona las reglas y que de él depende muchos servicios se hace el trabajo un poco más complejo.

Al igual que en el componente anterior, en este caso se muestra el diagrama de clases del negocio que implementa la generación de comprobantes de operaciones actualmente existente.

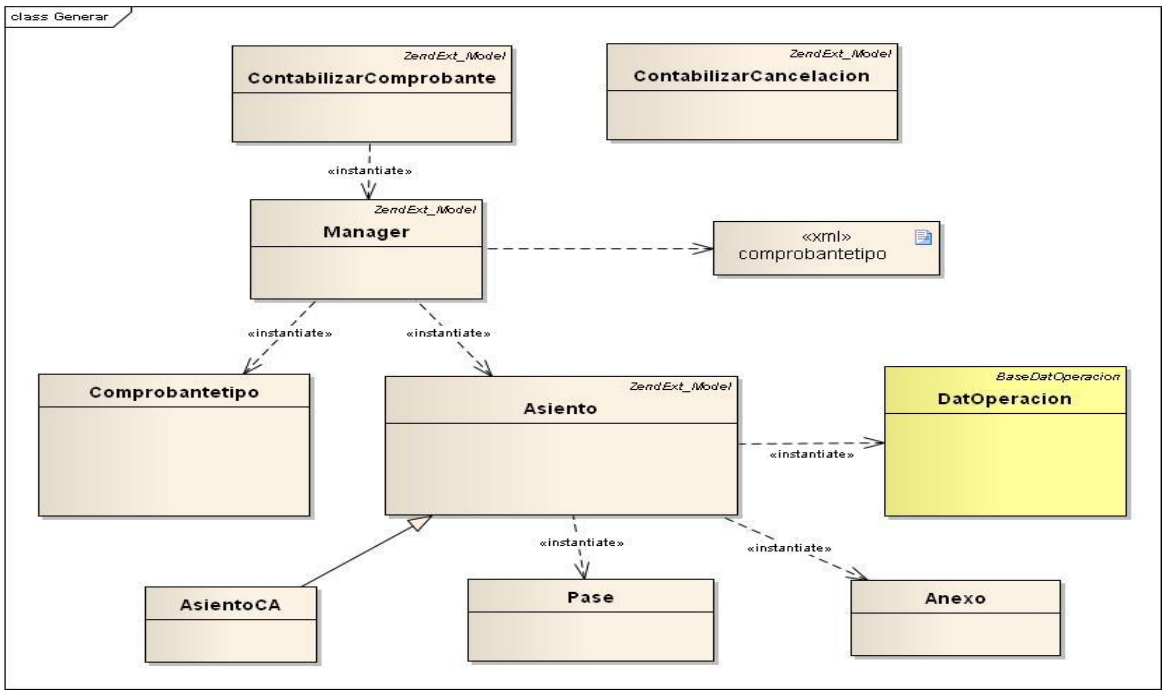

Figura 4(Diagrama de clases del negocio del componente Reglas)

Las dos primeras clases que aparecen implementan los procesos macro de contabilización de comprobantes. La clase *ContabilizarCancelacion* no necesita colaborar con otra clase del componente, ya que esta solicita un comprobante desde Contabilidad y genera otro al revés logrando la cancelación del original. Para la clase *ContabilizarComprobante* sí existen un grupo de clases con las que se debe colaborar para obtener un comprobante a partir de un documento primario.

Primero instancia la clase *Manager* que es la responsable de orquestar la creación del comprobante, luego esta lee de un fichero de configuración XML el formato que debe tener el documento según el subsistema y entonces puede interpretar el documento. La existencia del resto de las clases responde a que cada uno de ellos conoce cómo crear un objeto de su tipo y por tanto cada cual crea a quien conoce y luego la clase que lo instancia agrupa los objetos creados quedando conformado el comprobante.

Un comprobante se expresa en los siguientes conceptos: Comprobante, Asiento, Pase, Anexo; donde cada uno es subconjunto del anterior. Se puede ver además, la clase *DatOperacion*, que pertenece al modelo de domino y es donde se accede a la información necesaria de las reglas. Debido a lo descrito de anteriormente se puede decir que una mejora del diseño ayudaría en gran parte al algoritmo generador de comprobantes

1.2 Estudio de algunos sistemas ERP

Después de un estudio minucioso de cada unos de los conceptos de Reglas Contables y un análisis detallado por parte de cada una de las problemáticas asociadas a las capacidades funcionales del sistema. Se decidió la revisión de algunos sistemas ERP existentes en el ámbito nacional e internacional.

1.2.1 OpenBravo

La solución de contabilidad proporcionada por Openbravo ERP está diseñada para minimizar la introducción manual de datos por parte del usuario, liberándole así de tareas pesadas y rutinarias y permitiendo, por tanto, que pueda focalizarse en otras de mayor valor añadido. Este incremento de productividad es debido a que el área financiera actúa como un recolector de todos los hechos relevantes que se van generando desde el resto de áreas de gestión, de manera que éstos tienen

un reflejo automático en la contabilidad general, en las cuentas a cobrar y en las cuentas a pagar en cuanto se producen.

El módulo de Gestión Contable permite conocer:

- La situación de la empresa en el aspecto cuantitativo, económico, financiero (Inventarios y balances)
- Resultados obtenidos (cuanta de explotación)
- Causas de estos resultados

Openbravo es una aplicación de gestión con un repositorio contable automático. Es decir, los asientos contables son reflejo fiel de las operaciones realizadas en gestión. De este modo si es necesario se puede borrar la contabilidad y volver a generarla automáticamente.

El primer paso de la configuración del módulo es la creación del Plan General Contable para cada organización, pudiendo llevar contabilidad diferenciada para cada una de las organizaciones.

El Plan General Contable es un archivo en el formato .csv que hay que importar a la aplicación.

La aplicación también permite llevar contabilidades paralelas utilizando distintos esquemas contables con distintos planes contables. (5)

1.2.2 SAP

Contabilidad financiera e interna, ofrece control e integración de toda la información financiera y contable esencial para la toma de decisiones estratégicas y operativas. Esta herramienta mejora la gestión de controles internos, auditoria, simplifica el análisis financiero y le permite llevar a cabo procesos de negocio adaptables a cambios en el mundo de las finanzas. Todo esto le permitirá transformar sus controles financieros de una simple función administrativa, en una parte de su empresa centrada en la estrategia, el valor y la rentabilidad del capital. Rentabilidad y crecimiento sostenibles metas alcanzables con esta solución financiera.

Permite mantener el control y la responsabilidad de las finanzas y contabilidad al mismo tiempo que capacita a su empresa para conseguir altos niveles de crecimiento sostenible. Esta solución financiera le facilita, acelerar el proceso de cierre mediante la automatización de procesos, flujo de trabajo y colaboración interna y externa; mejorar la eficacia de sus esfuerzos de conformidad

mediante auditorías completas, informes más exhaustivos y gestión de controles internos, mejorar el análisis de negocio y el soporte para la toma de decisiones, implantando herramientas de gestión que analizan toda la empresa y sus recursos y maximiza el flujo de caja mediante la mejora de la facturación, las cuentas de deudores, los cobros y la gestión de tesorería. (6)

1.2.3 OpenErp

El módulo contable de OpenERP provee contabilidad general, analítica y presupuestaria, y cuenta con todas las funcionalidades para llevar los libros contables en forma rigurosa. Puede ser usado como un programa independiente o completamente integrado con los otros módulos de OpenERP para desarrollar su máximo potencial. OpenERP dispone de contabilidad automática de doble entrada, que se combina con una gran cantidad de herramientas de informes y análisis totalmente integradas. Este módulo permite la generación de presupuestos, informes, etc. Con este módulo la tesorería puede gestionar los flujos de caja y el efectivo con un alto nivel de trazabilidad. La mayoría de los asientos y las funciones están completamente automatizadas. Los procesos automatizados le permiten disminuir las tareas relacionadas con el ingreso de datos. (7)

1.2.4 Versat

El módulo de Contabilidad General del VERSAT Sarasola constituye el corazón del sistema, rector de todo el VERSAT, al resumir las necesidades y requerimientos del registro contable. Hacia este módulo tributan todas las operaciones contables de los restantes subsistemas. En este subsistema, los comprobantes de operaciones a procesar manualmente son mínimos, pues la casi totalidad de los mismos serán generados por los restantes módulos. El módulo posee cinco funciones básicas: comprobantes de operaciones, clasificador de cuentas, análisis de las cuentas, costos y procesos y estados financieros. (8)

Luego del estudio de algunos de los ERP existente solo sirvió de aporte en parte Openbravo por sus clasificaciones de tipos de reglas lo que esto puede servir de ayuda para así poderle dar un tipo de especificación a las Reglas. SAP como software privativo y además de que sus funcionalidades en la contabilidad no se asemejan, se puede decir que no queda como candidato a seguir al igual que OpenERP pues no satisfacen las necesidades del producto nacional. De Versat se puede decir que es el promotor de Reglas Contables ya que fue la idea principal para su nacimiento por lo que no sirve de mucha utilidad a la investigación pues dado que actualmente el componente cuenta con más funcionalidades que el propio Versat .Próximo paso a seguir es el

estudio de patrones de diseño como herramienta para la definición de una nueva solución a Reglas Contables.

1.3 Diseño

Después del estudio de la solución particular existente en Reglas Contables y de ver como algunos ERP se desenvuelven en la contabilidad, se dio la necesidad de estudiar algunos patrones de diseño como herramienta para la definición de una nueva solución.

1.3.1 Patrones de asignación de responsabilidades

Un patrón es una descripción de un problema y su solución (pareja problema/solución), con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas.
Los Patrones Generales de Software para Asignar Responsabilidades (GRASP) describe los principios de la asignación de responsabilidades de un objeto, ejemplos de estos son:

- Experto
- Creador
- Alta cohesión
- Bajo acoplamiento
- Controlador

Cada uno de estos tiene sus particularidades y funciones que lo diferencia, como el creador que asigna la tarea de crear el objeto a la clase que tiene más información de lo que se quiere realizar.
Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación. (9)

El grupo de GoF clasificaron los patrones en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

Creacionales: Patrones creacionales tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

Estructurales: Los patrones estructurales describen cómo las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.

Comportamiento: Los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos. (10)

Dentro de los patrones GoF el de comportamiento es el más indicado para darle solución a la problemática existente, pues con un buen empleo de este patrón se puede lograr un bajo acoplamiento entre las clases del sistema, con lo que se lograría menor complejidad de mantenimiento. De los patrones de comportamiento existe Cadena de Responsabilidades patrón estudiado con el objetivo de mejorar el diseño actual.

Cadena de Responsabilidades

La definición formal de cadena de responsabilidad dada por "GoF" es:

Evitar el acoplamiento del emisor del requerimiento y el receptor, al obtener más de un objeto con posibilidad de tratar el requerimiento. Es la cadena de objetos que reciben y pasan la petición a lo largo de la cadena hasta que un objeto la manipule.

En el ejemplo previo muestra cómo utilizar una estrategia sin que importe el problema de aplicar una localización específica de tarifas de impuesto de ventas en tiempo de ejecución. Sin embargo, como se ha visto, para implementar una estrategia de algunos objetos, en algún lugar tiene que decidir, en tiempo de ejecución, cuáles son las posibles subclases a ser implementadas. Esto puede no ser siempre deseado, o incluso, posible.

En una cadena de responsabilidad cada objeto sabe cómo evaluar la petición para una acción y, si no la puede controlar por sí mismo, sabe solamente cómo pasarla a otro objeto, por ello la "cadena". La consecuencia de esto es que el cliente, (que inicia la petición de la acción) ahora sólo necesita conocer sobre el primer objeto en la cadena. Además, cada objeto en la cadena, sólo necesita conocer también sobre un objeto, el siguiente en la cadena. La cadena de responsabilidad puede ser implementada utilizando una cadena predefinida o estática, o las cadenas pueden ser

dinámicas, construidas en tiempo de ejecución teniendo cada objeto su propio sucesor cuando sea necesario.

Componentes de una cadena de responsabilidad.

Una cadena de responsabilidad puede ser implementada al crear una clase “manipuladora” (handler) abstracta (para especificar la interfaz y funcionalidad genérica) y crear subclases concretas para definir varias posibles implementaciones. Sin embargo, no existen requerimientos absolutos para todos los miembros de la cadena de responsabilidad, por descender a partir de la misma clase, proporcionando que todos ellos soporten la interfaz necesaria para integrar con otros miembros de la cadena.

Los objetos clientes necesitan una referencia a la subclase específica, la cual es su punto de entrada individual a la cadena. (Observe que no todos los clientes necesitan utilizar el mismo punto de entrada). (11)

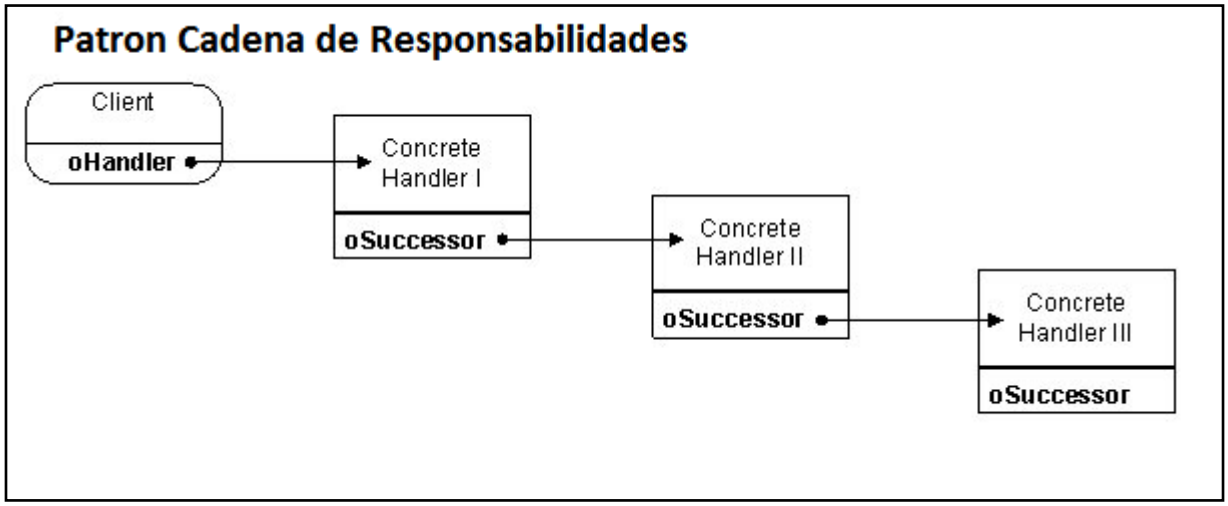

Figura 5(Patrón Cadena de Responsabilidades)

1.3.2 Arquitectura de Cedrux

Dada la arquitectura empleada en Cedrux se decidió estudiar el patrón modelo vista controladora para así poder tener un mejor punto de vista en el desarrollo del diseño, patrón utilizado por la misma.

Patrón MVC (Modelo-Vista-Controlador).

Para el diseño de aplicaciones con sofisticados interfaces se utiliza el patrón de diseño Modelo-Vista-Controlador. La lógica de un interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio. Si se realiza un diseño ofuscado, es decir, un pastiche que mezcle los componentes de interfaz y de negocio, entonces la consecuencia será que, cuando se necesita cambiar el interfaz, se tiene que modificar trabajosamente los componentes de negocio. Mayor trabajo y más riesgo de error.

Se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

Elementos del patrón:

- Modelo: datos y reglas de negocio
- Vista: muestra la información del modelo al usuario
- Controlador: gestiona las entradas del usuario

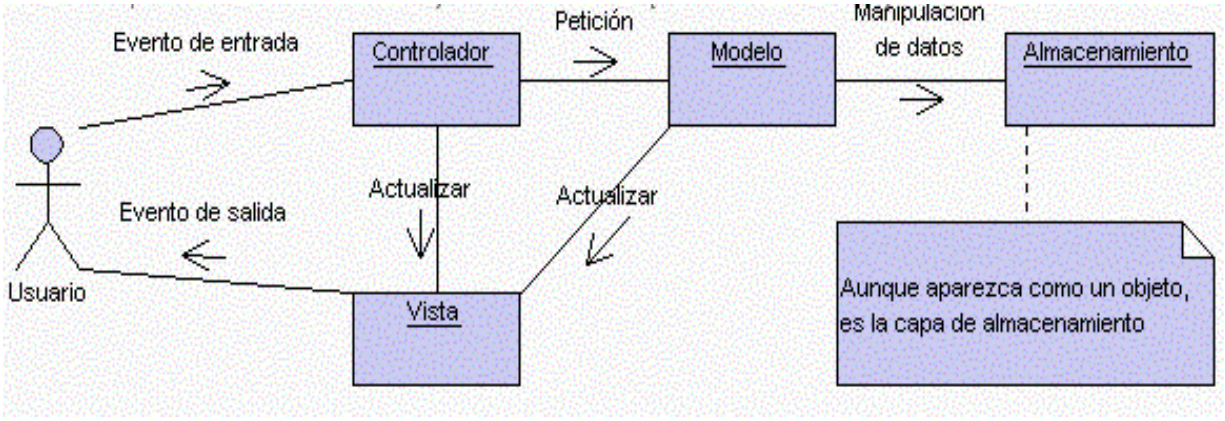

Figura 6(Modelo vista controlador)1

Un modelo puede tener diversas vistas, cada una con su correspondiente controlador. Un ejemplo clásico es el de la información de una base de datos, que se puede presentar de diversas formas: diagrama de tarta, de barras, tabular, etc. Se ve en cada componente:

1. El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
- Lleva un registro de las vistas y controladores del sistema.
- Si se está ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero bath que actualiza los datos, un temporizador que desencadena una inserción, etc).

2. El controlador es responsable de:

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar ()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega (nueva_orden_de_venta)".

3. Las vistas son responsables de:

- Recibir datos del modelo y la muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de "Actualización ()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Un ejemplo de MVC con un modelo pasivo (aquel que no notifica cambios en los datos) es la navegación web, que responde a las entradas del usuario, pero no detecta los cambios en datos del servidor.

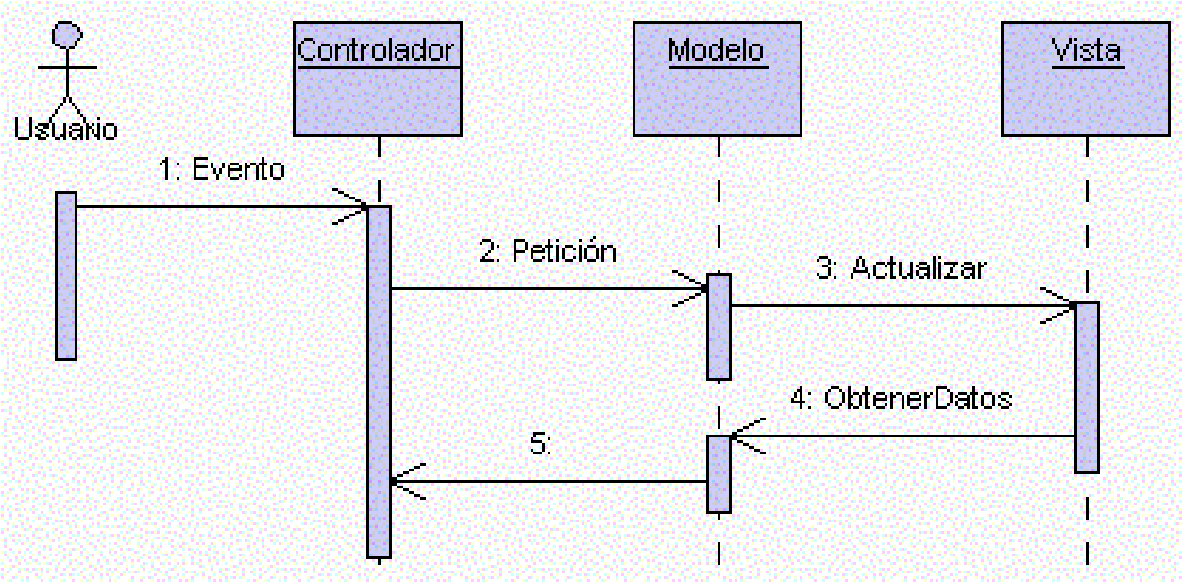

Figura 7(Modelo vista controlador)2

Pasos:

1. El usuario introduce el evento.
2. El Controlador recibe el evento y lo traduce en una petición al Modelo (aunque también puede llamar directamente a la vista).
3. El modelo (si es necesario) llama a la vista para su actualización.
4. Para cumplir con la actualización la Vista puede solicitar datos al Modelo.
5. El Controlador recibe el control. (12)

1.3 Herramientas

Para el desarrollo de la solución se emplean las siguientes herramientas:

1.3.1 Herramientas CASE[1]

[1] Por sus siglas en ingles(Computer Aided Software Engineering) en español Ingeniería de Software Asistida por Computadora

Visual Paradigm

Visual Paradigm para UML (Lenguaje de Modelado) es una herramienta que emplea UML como lenguaje de modelado, soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Se usó para el modelado del sistema propuesto debido a que entre sus utilidades permite construir aplicaciones con mejor calidad, además posibilita entre sus funciones realizar diagramas de clases, código inverso, así como generar código desde diagramas y generar documentación.

UML El Lenguaje Unificado de Modelado, en sus siglas en inglés (UML), es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir. UML un lenguaje de propósito general para el modelado orientado a objetos. UML es también un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes. Objetivos del UML:

UML es un lenguaje de modelado que puede ser usado por todos los modeladores. Incluye todos los conceptos que se consideran necesarios para utilizar un proceso moderno iterativo, basado en construir una sólida arquitectura para resolver requisitos dirigidos por casos de uso. Debe ser un lenguaje universal, como cualquier lenguaje de propósito general. (13)

1.3.2 Navegadores

Un navegador o navegador web (del inglés, web browser) es un programa que permite visualizar la información que contiene una página web (ya esté alojada en un servidor dentro de la World Wide Web o en uno local). El navegador interpreta el código, HTML generalmente, en el que está escrita la página web y lo presenta en pantalla permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares de la red mediante enlaces o hipervínculos.

La funcionalidad básica de un navegador web es permitir la visualización de documentos de texto, posiblemente con recursos multimedia incrustados. **Mozilla Firefox** Mozilla Firefox es un navegador de software libre. Entre sus características se encuentran que presenta una forma rápida y eficiente de navegar por la web, que le permite abrir varias páginas en una misma ventana mediante el empleo de pestañas separadas. Mozilla Firefox además protege al usuario de la publicidad de ventanas emergentes no solicitadas. (14)

Firefox utiliza además a Firebug que es una extensión creada y diseñada especialmente para desarrolladores y programadores web. Es un paquete de utilidades con el que se puede analizar (revisar velocidad de carga, estructura DOM)2, editar, monitorizar y depurar el código fuente, CSS, HTML y JavaScript de una página web de manera instantánea y online.

1.3.3 Control de versiones

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

Subversion

Subversion es un sistema de control de versiones libre y de código fuente abierto, es decir, maneja ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. Entre las características que tienen estos sistemas están:

Entendimiento con los lenguajes de programación.

Suministro de herramientas para la construcción de software.

Subversion es un sistema general que puede ser usado para administrar cualquier conjunto de ficheros. (15)

TortoiseSVN

Es un cliente de código abierto para el sistema de control de versiones Subversion. Maneja ficheros y directorios a lo largo del tiempo. Es fácil de usar, permite ver el estado de los archivos desde en el explorador de Windows y permite también crear gráficos de todas las revisiones asignadas.

Conclusiones

Después del estudio de algunos patrones de diseños y analizando los problemas existentes en el componente se tomó como decisión.

- Expresar la información de objetos, documento y operaciones en un formato único donde estas formaran una nueva entidad que se le nombrara regla.
- Adaptar el diseño de clases del negocio al cumplimiento de las funcionalidades requeridas.
- Redefinir la integración con los demás componentes del sistema, sin varia los contratos entre componentes.
- Diseñar una nueva versión del componente Reglas contables.

Capítulo 2: Características del sistema

Introducción

En este capítulo se tratan varios aspectos relacionados en el diseño del componente. Se comienza realizando un profundo análisis de los requisitos funcionales para así poder modificar el flujo de aquellos que los necesiten. Se pretende mostrar cómo está aprovechada la arquitectura y las posibilidades que proporcionan los patrones estudiados en la fundamentación teórica La representación de los diagramas de clases del diseño, con sus respectivos diagramas de secuencia, diagrama de componente y modelado de la base de datos con el objetivo de facilitar la comprensión del funcionamiento del componente a implementar.

2.1 Requisitos funcionales del componente Regla

El componente Reglas Contables cuenta con 5 grupos de requisitos que están desglosados por distintas especificaciones de requisitos. De los mismo solo se modifican el flujo del requisito Adicionar operación contable y adicionar objeto a subsistema ver Tabla 1 y Tabla2.Estos requisitos funcionales necesitan de cambios en el flujo básico porque para los mismo se cambió la forma en que los mismos actuaban. No se modifica otro requisito funcional pues dado que para darle solución a los mismos solo necesitan de cambios en el diseño y no en el flujo de los mismos.

Tabla 1 (Especificación del requisito Adicionar objeto a subsistema)

Precondiciones	El usuario se ha identificado y autenticado ante el sistema y tiene los permisos para realizar esta acción. Se han creado cuentas contables.
Flujo de eventos	
Flujo básico	
1	El sistema permite seleccionar un subsistema.
2	El sistema debe cargar los objetos según el subsistema seleccionado.
3	El sistema permite seleccionar un objeto.

4	El sistema permite seleccionar las cuentas contables en las que se registrará el valor de este objeto para el subsistema.	
5	El sistema valida los datos según la validación 1.	
6	Si los datos son correctos el sistema los registra.	
7	El sistema confirma el registro de los datos.	
8	Concluye el requisito.	
Pos-condiciones		
1	Se creó una asociación entre objeto y subsistema.	
Flujos alternativos		
Flujo alternativo 4.a Información incompleta o errónea		
1	El sistema señala los datos erróneos y permite corregirlos.	
2	Volver al paso 3 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	Se debe asociar al menos una cuenta a la relación subsistema-objeto.	
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	**Objeto**	Visibles en la interfaz: Denominación Utilizados internamente: N/A
	Subsistema	Visibles en la interfaz: Nombre Utilizados internamente:

		N/A
	Asociación subsistema-objeto	Visibles en la interfaz: Cuenta contable Utilizados internamente: N/A
	Cuenta contable	Visibles en la interfaz: Denominación Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

En el requisito anterior en el flujo básico hubo cambios en el primero y segundo paso. El usuario primero debes escoger con que subsistema desea trabajar, luego según el subsistema que el usuario haya seleccionado, el sistema debe mostrar en un campo de combo box todos los objetos que pertenecen al dicho sistema.

Tabla 2(Especificación del requisito Adicionar operación contable)

Precondiciones	El usuario se ha identificado y autenticado ante el sistema y tiene los permisos para realizar esta acción. Se han creado cuentas contables. Se han seleccionado un subsistema y un objeto asociado a ese subsistema.
Flujo de eventos	
Flujo básico	
1	El sistema permite seleccionar el documento primario mediante el cual se registra la

	operación, y el concepto de operación. Además, permite indicar si la operación contabiliza o no.
2	Si la operación no contabiliza el sistema registra los datos.
3	El sistema confirma el registro de los datos.
4	Concluye el requisito.
Pos-condiciones	
1	Se creó una nueva operación contable.
2	Se asoció la operación contable a un documento primario y una operación.
Flujos alternativos	
Flujo alternativo 2.a La operación contabiliza	
1	El sistema permite seleccionar más de una cuenta contable en la que se contabiliza la operación y si debita o acredita. Además, permite indicar si se requiere contabilización adicional.
2	Si no se requiere contabilización adicional el sistema registra los datos tanto como cuentas contables haya seleccionado.
3	El sistema confirma el registro de los datos.
4	Concluye el requisito.
Pos-condiciones	
1	Se creó una nueva operación contable.
2	Se asoció la operación contable a un documento primario y una operación.
3	Se creó una relación entre la operación contable y la cuenta en la que se contabilizará.
Flujo alternativo 2.a.2.a La operación contabiliza y requiere contabilización adicional	
1	El sistema permite seleccionar todas las cuentas que serán afectadas por la contabilización adicional. Para cada cuenta se registra si se debita o acredita y el porciento de la operación contable que se debe acreditar o debitar en dicha cuenta.
2	El sistema valida los datos según la Validación 1.
3	Si los datos son correctos el sistema los registra.
4	El sistema confirma el registro de los datos.
5	Concluye el requisito.
Pos-condiciones	

1	Se creó una nueva operación contable.	
2	Se asoció la operación contable a un documento primario y una operación.	
3	Se creó una relación entre la operación contable y la cuenta en la que se contabilizará.	
4	Se creó una regla para la contabilización adicional.	
5	Se asociaron a la regla las cuentas contables afectadas y se registró el porciento de la operación original que afectará a cada cuenta.	
Flujo alternativo 4.a Información incompleta o errónea		
1	El sistema señala los datos erróneos y permite corregirlos.	
2	Volver al paso 2 del flujo alternativo 2.a.2.a	
Pos-condiciones		
	N/A	
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Pos-condiciones		
1	No se registran los datos.	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual CIG-ERP-N-CON-i2201.	
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	**Operación contable**	Visibles en la interfaz: N/A Utilizados internamente: N/A
	Contabilización	Visibles en la interfaz: naturaleza Utilizados internamente: N/A
	Contabilización adicional	Visibles en la interfaz: Naturaleza

		Porciento Utilizados internamente: N/A
	Asociación subsistema-objeto	Visibles en la interfaz: Cuenta contable Utilizados internamente: N/A
	Cuenta contable	Visibles en la interfaz: Denominación Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

En la especificación de requisitos anteriormente vista se puede ver como se gestiona una operación contable. El flujo 2.a fue modificado, pues para realizar una operación se necesita más de una cuenta para poder contabilizarla. Al contabilizar una operación esta se guarda como distintas operaciones tantas veces se le haya asignado una cuenta, es ahí el cambio en el flujo 2.a segundo paso. Para gestionar una operación contable se necesita también cambiar en el componente el concepto de operación pero ya esto viene mas dado en el diseño.

2.2 Descripción de la solución

La solución a nivel de componente está dada en la creación de un nuevo diseño del componente. El componente Generación de Comprobante y Gestión de Reglas se unen para así formar un nuevo denominado Reglas Contables. También se modifica los paquetes de clases de dichos componente.

Para iniciar con la nueva versión lo primero que se desea hacer es la unión de los dos paquetes de clases en uno solo. Como se puede ver en las figuras siguientes como estaba diseñado físicamente el paquete de clases ***services*** y las ***models*** que pertenecían al paquete ***contrato***, paquete el cual

estaba creado para el componente Generación de Comprobantes pasaron a formar parte de un solo paquete junto al de Gestionar Reglas Contables ver figuras 8 y 9.

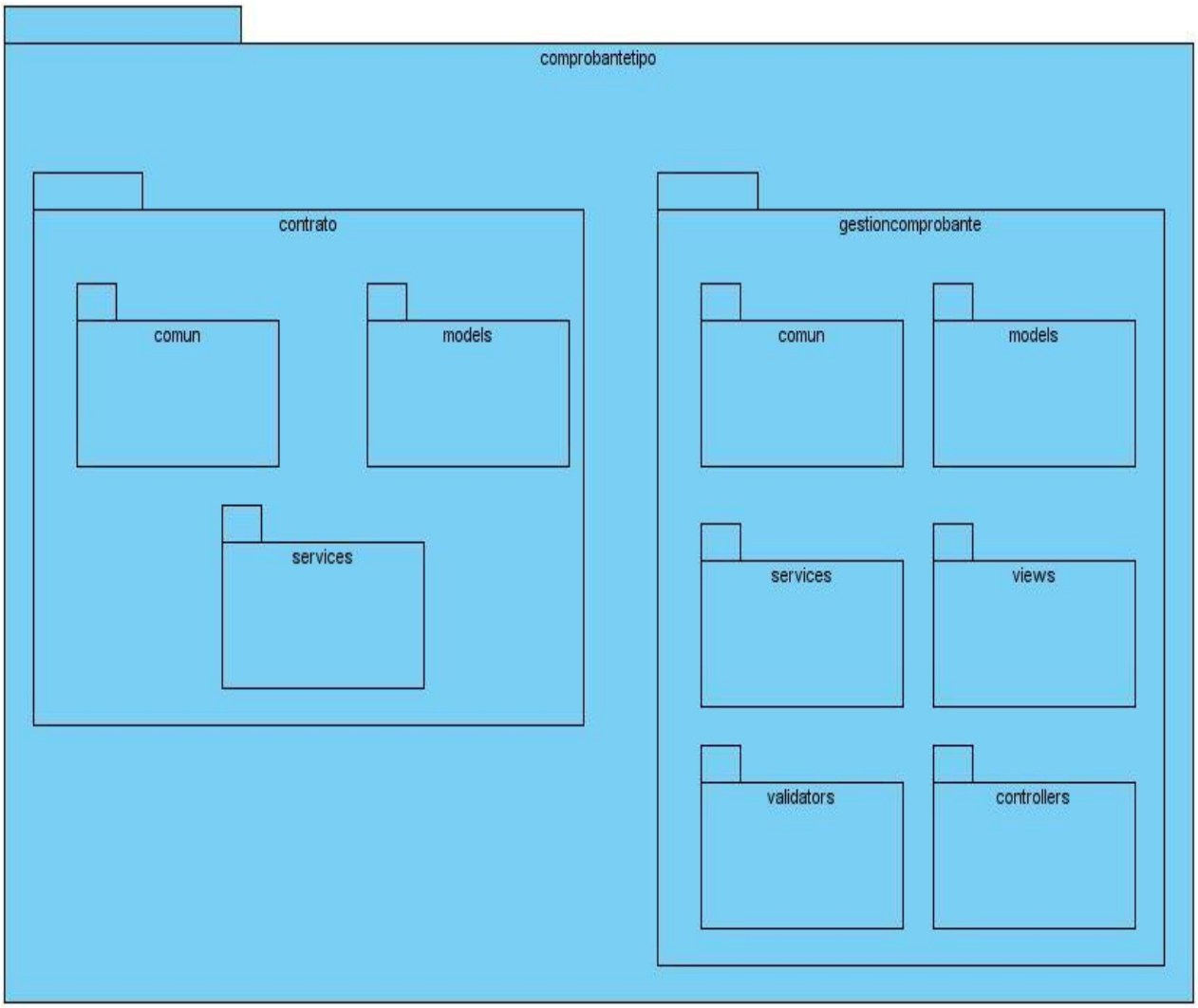

Figura 8(Estado de paquetes antes de la nueva versión)

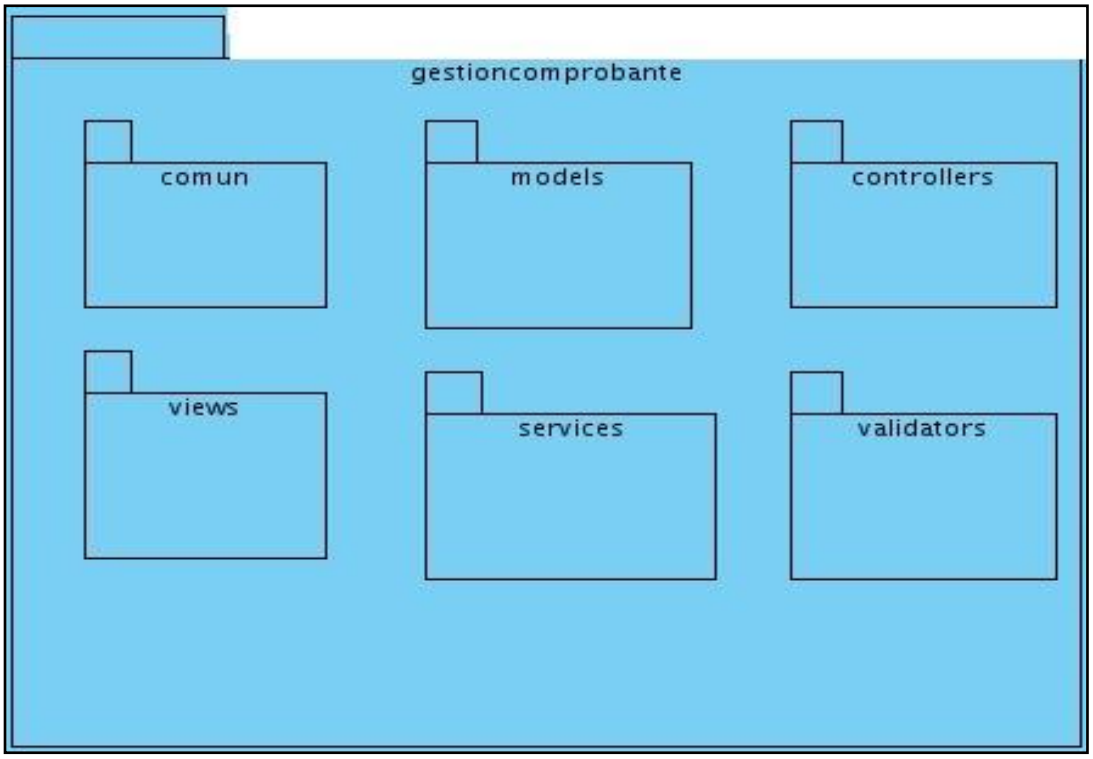

Figura 9(Estado de paquetes en la nueva versión)

En el paquete de las clases controladoras se suprimieron dos clases controladoras que no ejercían ningún tipo de función dentro del componente las cuales fueron ***IndexController*** y ***ObjetosController***. Al paquete de las clases ***models*** pasan todas las clases del paquete contrato que pertenecía al componente Generación de Comprobantes que al surgir la unión, estas pertenecen al paquete ***Comprobantetipo***, paquete actualmente unificado donde se realizan todas las operaciones pertinentes respecto a las reglas contables. También se modelan 4 escenarios donde se puede ver cada uno de los diseños de clases por separados dando así un mejor entendimiento. Se modela el diagrama entidad relación de base de datos donde se puede ver modificaciones en varias entidades, así como también una serie de diagramas de secuencias para ver paso a paso cómo se trabaja con dicho componente.

2.3 Modelo de componente

Un modelo de componentes representa los componentes, sus interfaces y las relaciones de los componentes con las interfaces que utilizan.

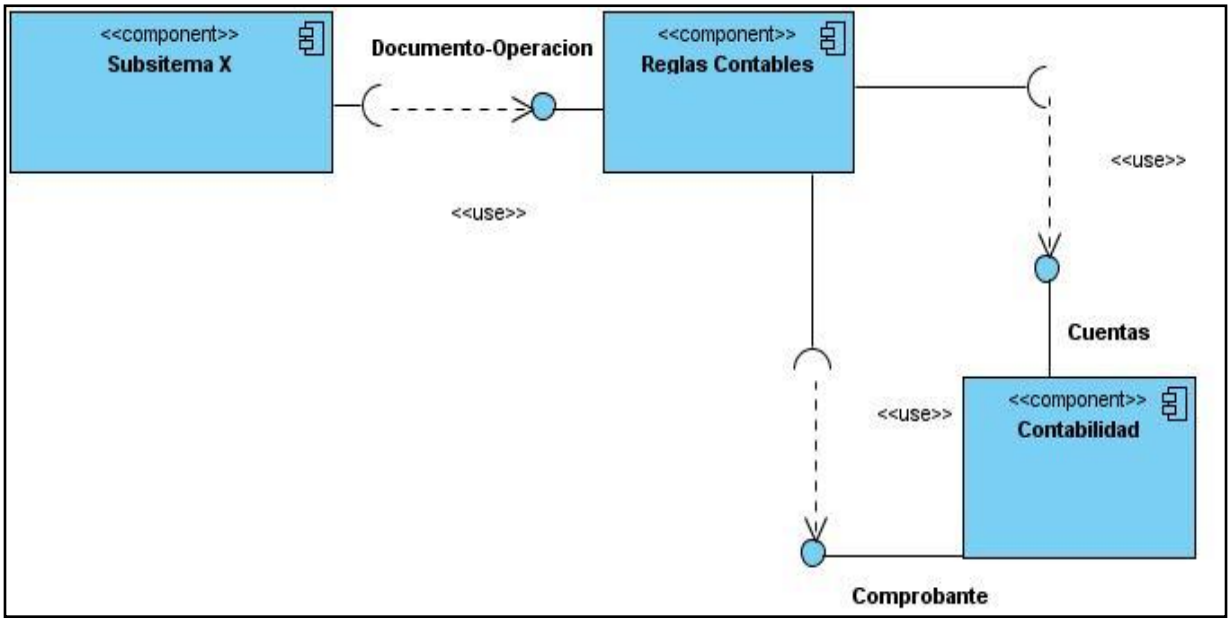

Figura 10(Modelo de componentes para la nueva versión)

En la figura anterior se muestra la relación entre componentes donde se puede ver la unión del componente Generación de Comprobantes con el de Gestión de Reglas para así darle surgimiento a Reglas Contables donde este se interrelacionará con el componente Subsistema ya que las reglas contables se le aplican al mismo, como también tendrá un estrecha relación con el componente Contabilidad ya que para realizar una operación contable se necesitan nomencladores básicos como son las cuentas que se le asignan a los objetos de los distintos Subsistemas .

Dada la situación, el mecanismo a desarrollar en Reglas Contables no es más que la introducción de un subsistemas con sus distintos objetos a dicho componente y este con su nomencladores básicos que son objeto, documento y operación es el encargado de dictar una regla u operación contable de dicho subsistema donde también se podrá generar un comprobante con todas las operaciones realizadas al mismo.

2.4 Diagrama de clases de negocio

Los diagramas de clases según la clasificación UML son diagramas de estructura estática donde la representación de los requerimientos se lleva a cabo a través de las clases del sistema y sus interrelaciones. Representan una abstracción del dominio de modo que es formalizado el análisis

de conceptos y constituyen el pilar básico del modelado, mostrando en términos generales qué debe hacer el sistema.

Específicamente los diagramas de clases de diseño son muy útiles porque muestran a través de atributos y métodos la estructura de las clases que después serán escritas en algún lenguaje de programación (PHP en este caso).

En este caso hay tres diagrama de clases cada uno representa un escenario por distinto de la funcionalidad completa del componente.

Diagrama 1: En ese diagrama está toda la relación principal de las operaciones objetos y documentos donde se realiza el mayor peso del componente, así como la generación del comprobante ahora en la nueva versión. En este diagrama están representadas las clases que interactúan completamente con todo lo relacionado al negocio en Reglas Contables. Se ve representado la clase ***GestionaroperacionespentidadController*** clase con mayor peso dentro del componente ya que la misma interactúa con las clases ***model*** y las clases de interfaces.

Diagrama 2: Diagrama encargado del diseño del trabajo con documentos. En dicho diagrama se ve como clase principal la clase ***DocumentosController*** es la encargada de gestionar el trabajo de documentos ya se crearlos eliminarlos o modificarlos

Diagrama 3: Diagrama encargado del diseño del trabajo con operaciones. Este diagrama al igual que el de documento es el encargado del trabajo con todo lo relacionado con las operaciones ya se crear una operación modificarla o eliminarla y su clase principal es la de ***OperacionesController.***

Diagrama 4: Diagrama encargado de explicar la lógica de negocio del generador de comprobantes. En este diagrama se puede ver lo relevante de aplicar un patrón de diseño conocido como es el patrón de comportamiento cadena de responsabilidades. En dicho diagrama el mayor peso de las funcionalidades de la generación de un comprobante está dado en la clase Manager

A continuación para poder visualizar lo dicho anteriormente se puede ver el diseño de clases de los 4 diagrama:

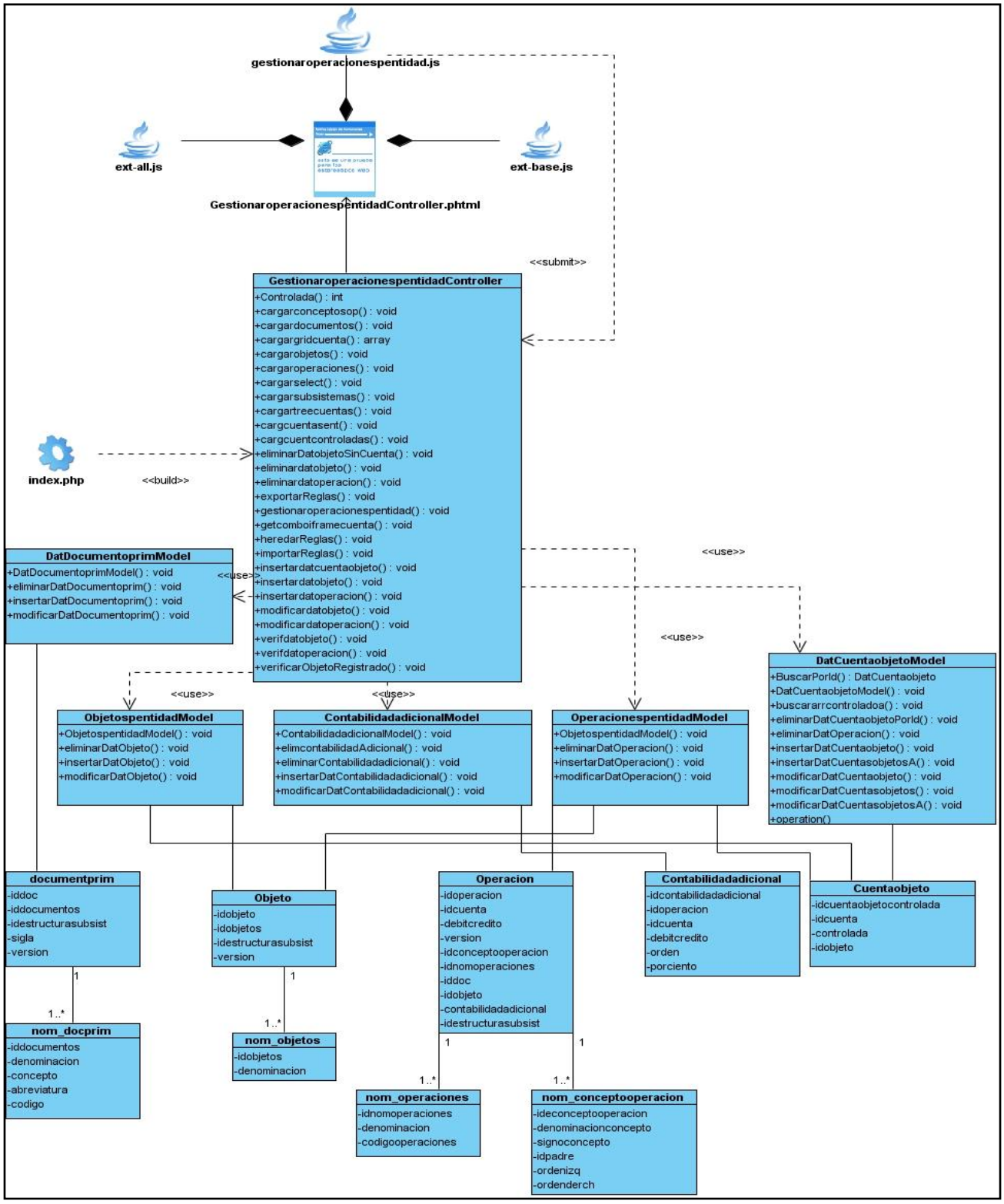

Figura 11(Diagrama de clases del escenario de reglas)

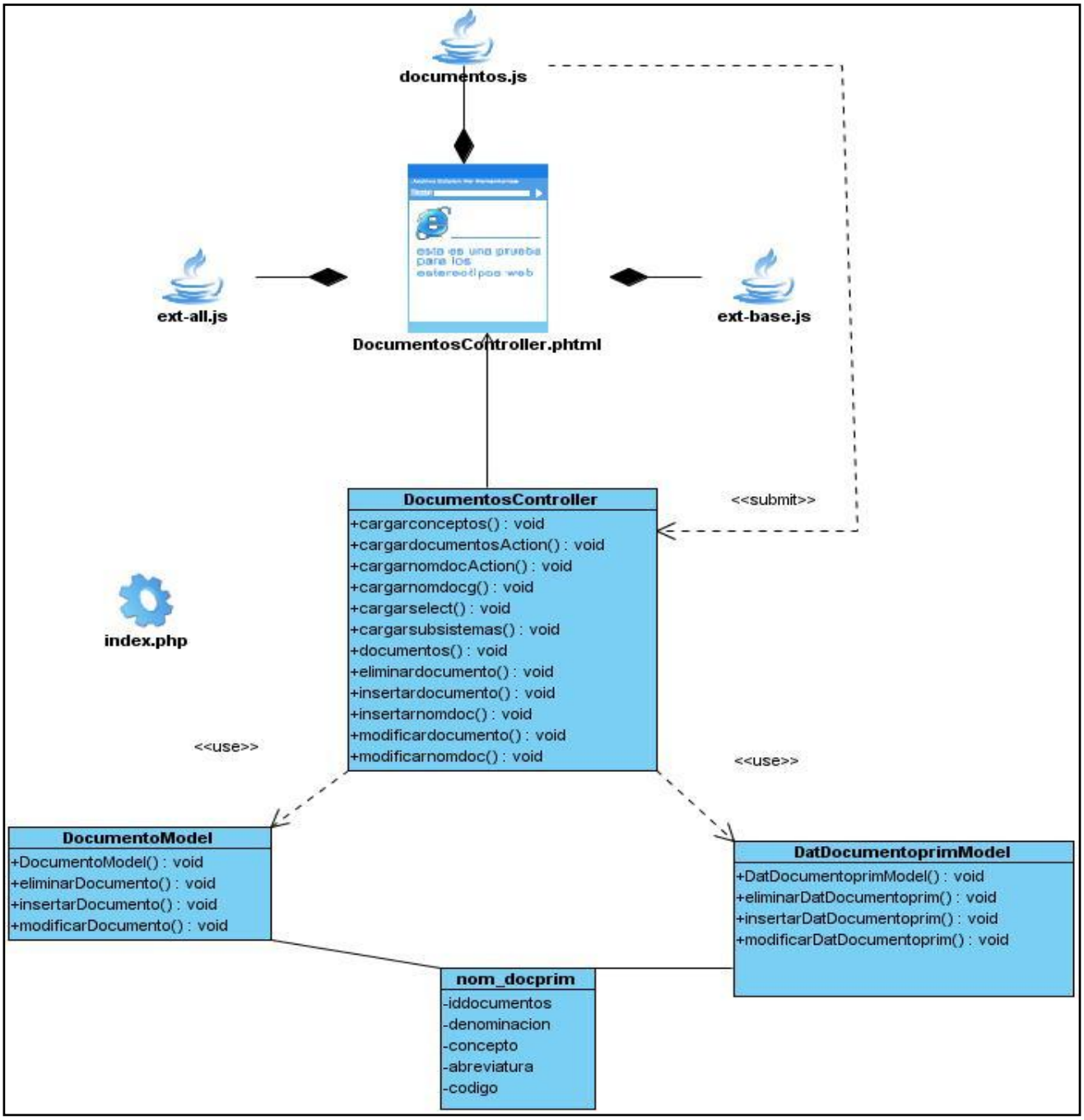

Figura 12(Diagrama de clases para el escenario de documentos)

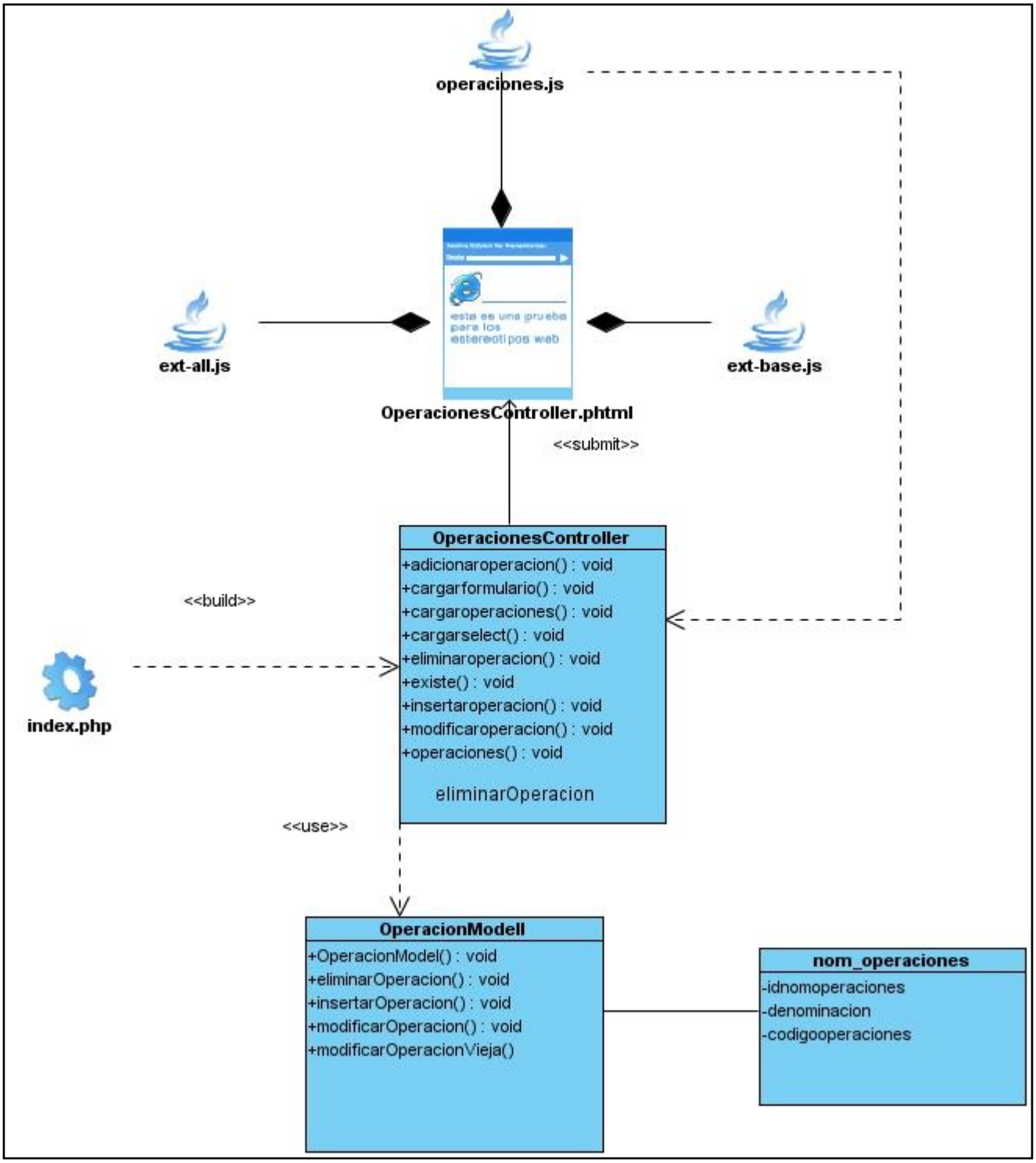

Figura 13(Diagrama de clases para el escenario de operaciones).

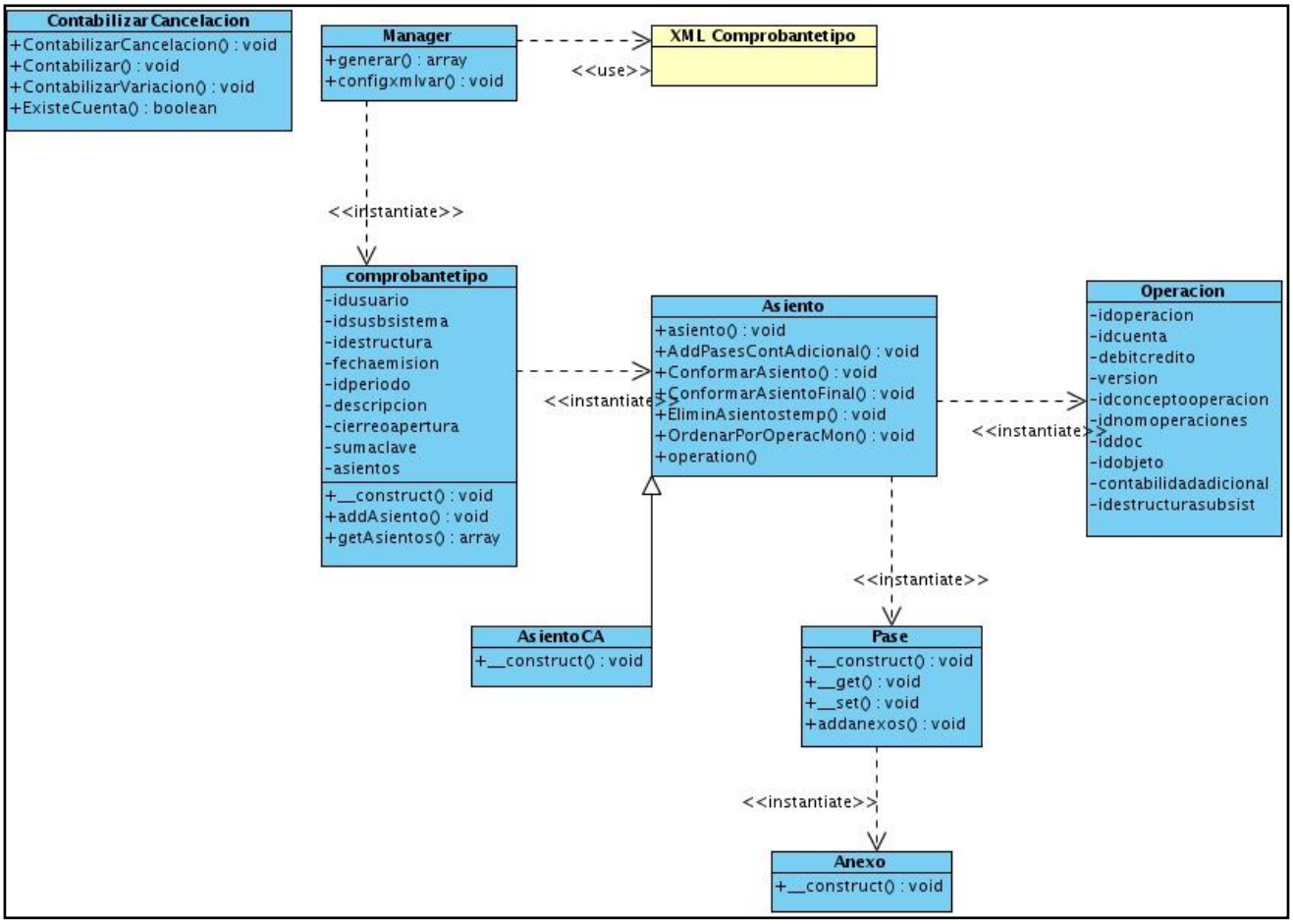

Figura 14(Diagrama de clases de negocio)

En este diagrama se ve reflejado cómo está aplicado el patrón de diseño Cadena de Responsabilidades. La clase Manager no es más que la clase **Handler**(Controladora) de la misma que viene siendo la clase que interactúa con el cliente, le siguen las clases como es **comprobantetipo**, **Asiento**, **Pase**, **Operación** y **Anexo** que no son más que los **Successors** (Sucesores). Esto le permite al algoritmo de Generación de Comprobantes una mayor facilidad a la hora de generar el mismo ya que la clase controladora no hace más que una llamada al eslabón siguiente de la cadena que es el responsable de lo que ella no pueda facilitarle al cliente. Ejemplo de lo dicho anteriormente es cómo se genera un Comprobante, pues este está dado por la clase **comprobantetipo** donde mediante él se accede a la Clase **Asiento** ya que el mismo depende de Asientos para su creación y Asiento depende de varias operaciones realizadas y de Pases y a su vez **Pase** depende de una series de Anexos y así poco a poco se va desglosando la cadena hasta llegar a su objetivo que es la creación de un comprobante de operaciones.

2.5 Diagrama de secuencia

En un diagrama de secuencia se muestran varias de las clases u objetos que forman parte del programa y se muestra qué llamadas van haciendo unos a otros para realizar una operación dada.

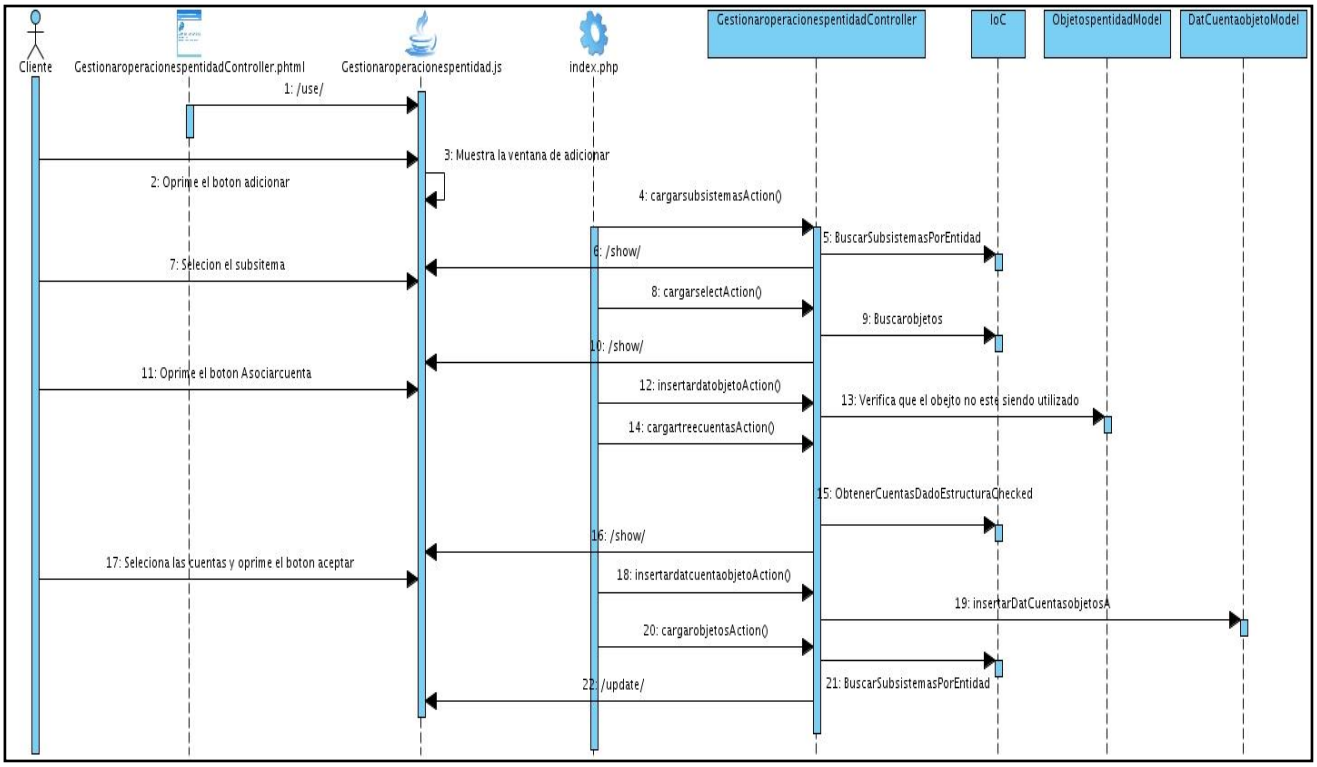

Figura 15(Diagrama de secuencia para Asociar una o varias cuentas a un objeto)

En este diagrama se puede ver el cambio de flujo a la hora de asociar una o más cuenta a un objeto. El cambio viene dado ya que en la versión actual el mismo carga todos los objetos y se los muestra al usuario donde no era necesario. En la versión nueva el usuario debe elegir el subsistema y en dependencia del subsistema el componente le mostrará solo los objetos que pertenecen al mismo. Esto garantiza que no se permitan errores en la asociación de subsistemas y objetos.

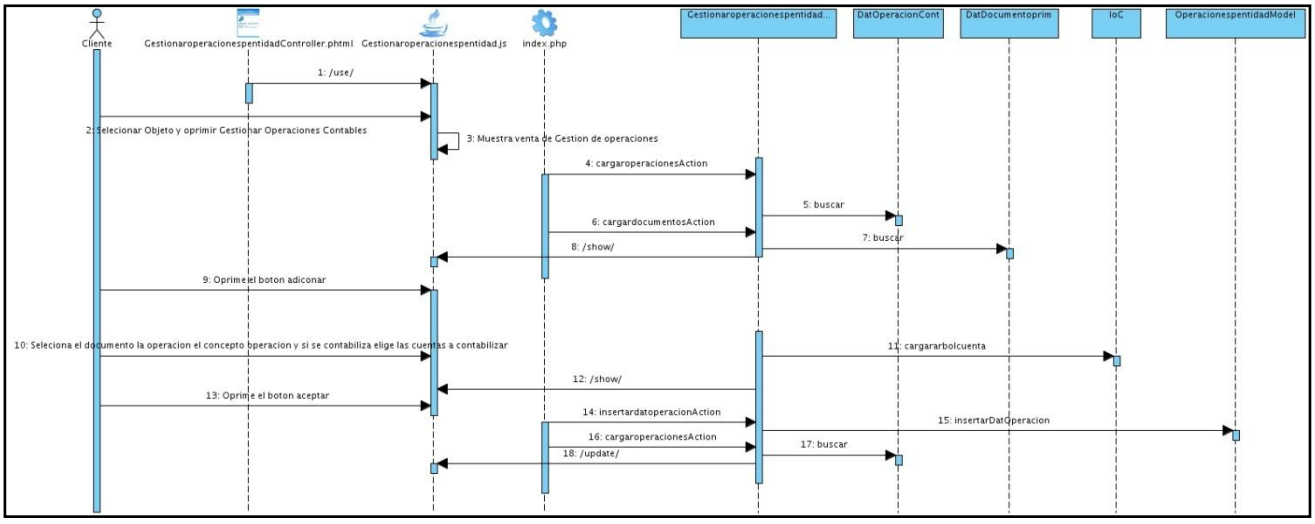

Figura 16(Diagrama de secuencia para gestionar una operación)

Para gestionar una operación se debe primero elegir a qué objeto se le desea realizar la operación. En este diagrama de secuencia se ve paso a paso cómo realizar una operación y los requisitos que conlleva la misma los cuales son; asignarle un documento, asignarle una o más cuenta, un concepto de operación. En la versión actual el cliente no podía elegir más de una cuenta, ahora el cliente puede elegir más de una cuenta para realizar una operación pero ya eso se guardaría como varias operaciones con la misma denominación al objeto. Donde se cubre que se puedan crear operaciones que tengan más de un cuenta, donde esta última dependa de determinadas particularidades de la operación.

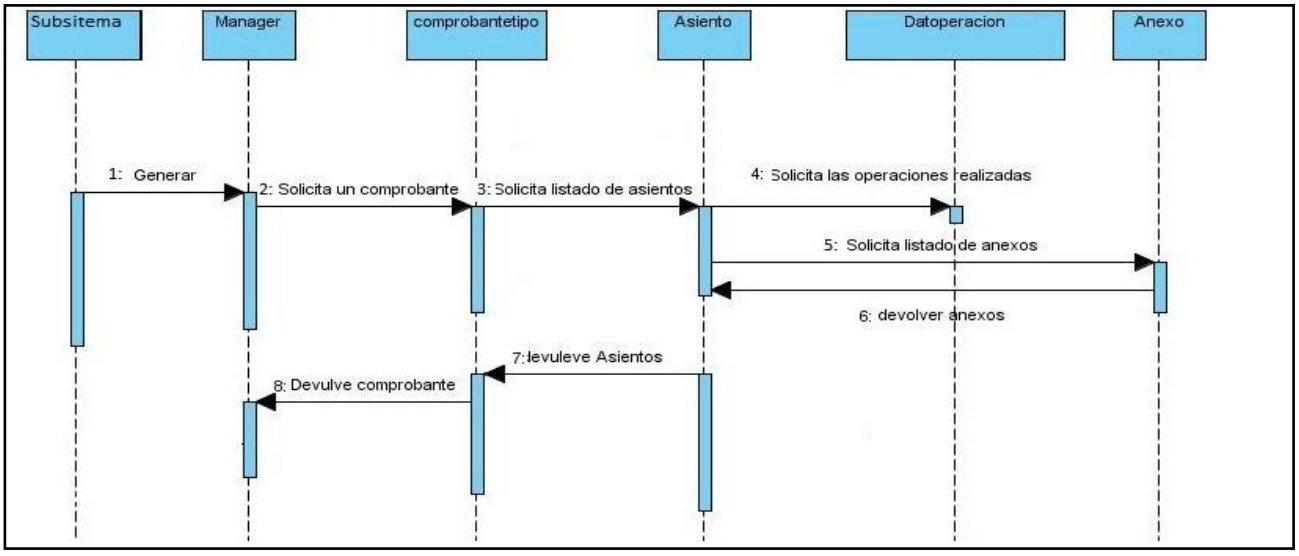

Figura 17(Generación de un comprobante)

En el diagrama anterior se puede ver paso por paso cómo se genera un comprobante de operaciones utilizando ya el nuevo patrón de diseño Cadena de responsabilidades.

2.6 Modelo de datos

Los modelos de datos aportan la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos, así como la base formal para las herramientas y técnicas empleadas en el desarrollo y uso de sistemas de información.

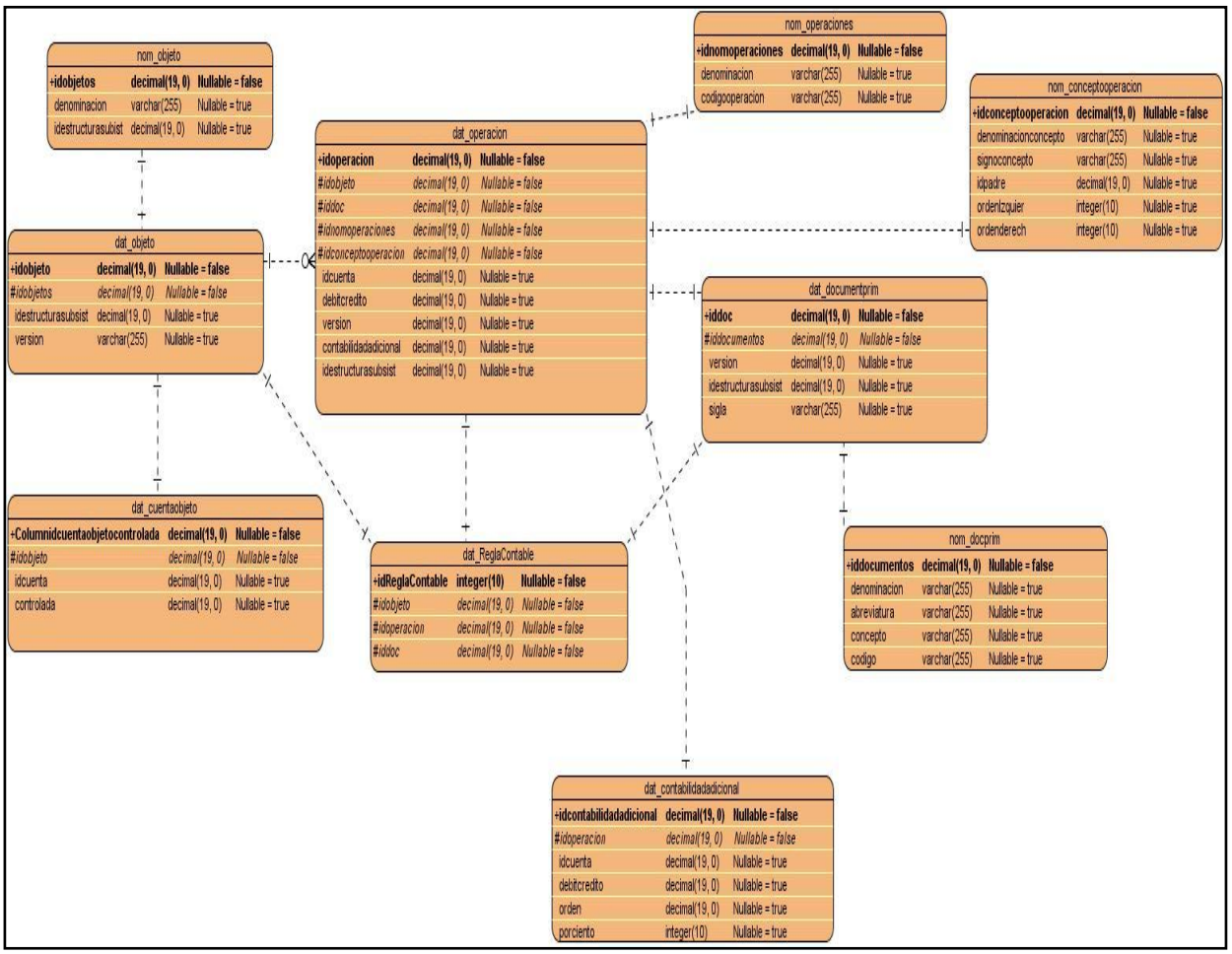

Figura 18(Modelo de datos para la nueva versión).

En el diagrama de entidad relación se puede ver cómo surge una nueva entidad a la que se nombró regla contable para así poder tener una comunicación entre objeto, documento y operación ya que esta no disponía de la misma. También se renovaron varias entidades como es **nom_objeto** ya que la misma no tenía como identificarse de que subsistema era, para cuando se realizara en la

aplicación una selección de los objetos de un subsistema saliera nada más los objetos que pertenecen a ese subsistema.

Otra de las actualizaciones que se le hizo a las entidades fue a la entidad ***nom_conceptooperacion*** que a la misma para una mejor operatividad en reglas contables pase a ser una entidad arbórea, mediante un patrón de entidades en forma de árbol donde la misma debe tener el id del padre un orden izquierdo y un orden derecho. Esta renovación permite que los conceptos para clasificar operaciones no se repitan innecesariamente y tiendan a confundir al usuario

Conclusiones

En el siguiente capítulo se llego a la conclusión de:

- Los cambios propuestos en los requisitos fueron mínimos ya que el componente tenía mayor problema de diseño que funcional.
- El patrón Cadena de Responsabilidades es factible para el algoritmo de generación de comprobante.
- Los mayores problemas fueron mala asignación entre clases y una mala codificación.
- Las clases entidades no estaban bien diseñadas para las funcionalidades de cada escenario.

Capítulo 3: Análisis de resultados

Introducción.

En este capítulo luego de diseñar la nueva versión del componente Reglas Contables se llevara a cabo mediantes métricas de software unas series de validaciones donde se puede ver en manifiesto si se cumplió con los objetivos trazados en la problemática inicial y si cumple con los estándares de calidad.

3.1 Métricas de software

Las métricas de software son las que en gran medida permite a los desarrolladores tener una visión de la eficacia en los procesos de software desarrollados. Es donde se reúnen los datos básicos de productividad y calidad, los cuales son analizados y comparados con resultados obtenidos anteriormente y evaluados para determinar las mejoras en la calidad y la productividad.

Un aspecto importante a tener en cuenta en la fase de evaluación de la calidad del diseño ha sido la creación de métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objeto, teniendo en cuenta que este estudio brinda un esquema sencillo de implementar y que a la vez cubre los principales atributos de calidad de software.

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

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

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:

Tamaño operacional de clase (TOC): Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Tabla 3(Atributos)

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

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Tabla 4(Promedio de atributos)

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Relaciones entre clases (RC): Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Tabla 5(Modo en que lo afecta)

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

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Tabla 6(Criterios y categoría).

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2

Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

3.1.1 Resultados obtenidos de la aplicación de las métricas TOC

Después de realizar la aplicación de las métricas TOC se llega a la conclusión de que el diseño propuesto cumple con lo requerido inicialmente ya que el mismo dio como resultado un grado bajo de complejidad representando el 15 % y un grado alto de clases reutilizables donde esto ayuda en la eliminación de clases repetidas.

Tabla 7(Instrumento para la evaluación del TOC)

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
GestionaroperacionController	27	Alta	Alta	Baja
DatDocumentprimModel	4	Media	Media	Media
ObjetospentidadModel	4	Media	Media	Media
ContabilidadadiconalModel	5	Media	Media	Media
OperacionespentidadModel	4	Media	Media	Media

DatCuentaobjetoModel	11	Alta	Alta	Baja
DocumentosController	12	Alta	Alta	Baja
DocumentoModel	4	Media	Media	Media
OperacionesController	9	Alta	Alta	Baja
OperacionModel	5	Media	Media	Media
DatDocumenprim	0	Baja	Baja	Alta
NomDocprim	0	Baja	Baja	Alta
DatObjeto	0	Baja	Baja	Alta
NomObjeto	0	Baja	Baja	Alta
DatOperacion	0	Baja	Baja	Alta
NomOperaciones	0	Baja	Baja	Alta
NomConceptooperacon	0	Baja	Baja	Alta
DatContabilidadadicional	0	Baja	Baja	Alta
DatCuentaobjeto	0	Baja	Baja	Alta
ContabilizarCancelacion	4	Media	Media	Media
Manager	2	Baja	Baja	Alta
Comprobantetipo	3	Baja	Baja	Alta
Asiento	7	Media	Media	Media
AsientoCA	1	Baja	Baja	Alta
Pase	4	Media	Media	Media
Anexo	1	Baja	Baja	Alta
DatReglacontable	0	Baja	Baja	Alta

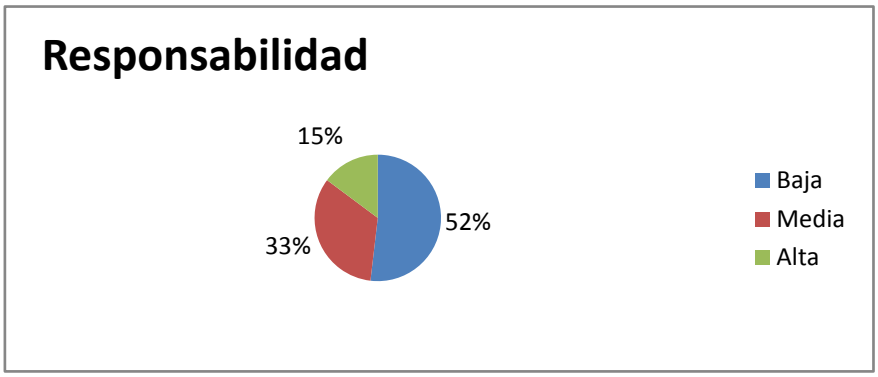

Figura 19(Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad.)

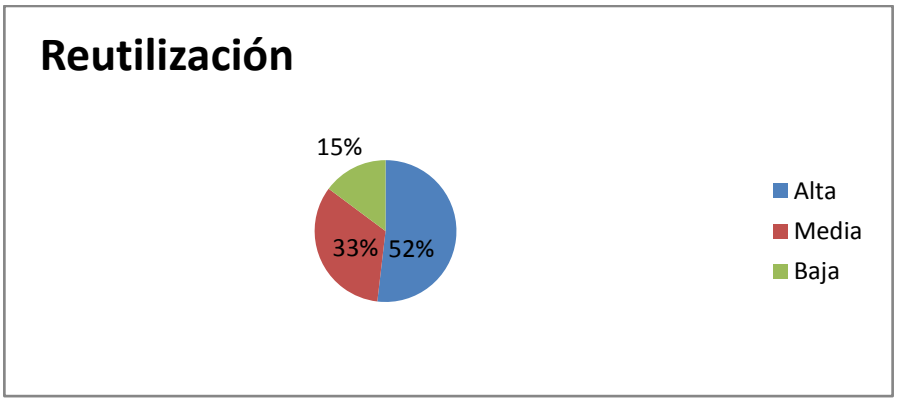

Figura 20(Resultados de la evaluación de la métrica TOC para el atributo Reutilización)

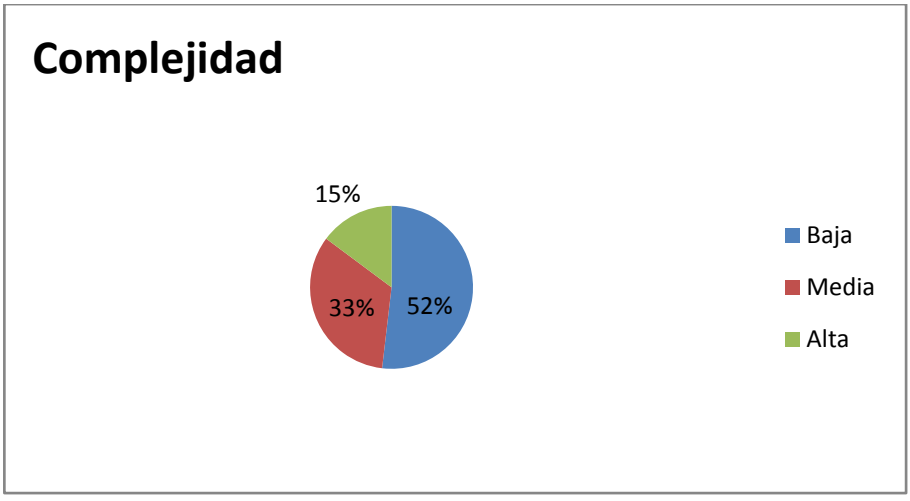

Figura 21(Resultados de la evaluación de la métrica TOC para el atributo Complejidad)

3.1.2 Resultados obtenidos de la aplicación de la métrica RC

Obtenidos los resultados de la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 63% de las clases empleadas posee menos de 2 dependencias de otras clases lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización). A continuación se muestran los resultados obtenidos.

Tabla 8 (Instrumento de evaluación de la métrica RC).

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mmto	Reutilización	Cantidad de Pruebas
GestionaroperacionController	5	Alto	Alta	Baja	Alta
DatDocumentprimModel	1	Bajo	Baja	Alta	Baja
ObjetospentidadModel	2	Medio	Media	Media	Media
ContabilidadadiconalModel	1	Bajo	Baja	Alta	Baja
OperacionespentidadModel	2	Medio	Media	Media	Media
DatCuentaobjetoModel	1	Bajo	Baja	Alta	Baja
DocumentosController	2	Medio	Media	Media	Media
DocumentoModel	1	Bajo	Baja	Alta	Baja
OperacionesController	1	Bajo	Baja	Alta	Baja
OperacionModel	1	Bajo	Baja	Alta	Baja
DatDocumenprim	1	Bajo	Baja	Alta	Baja

NomDocprim	1	Bajo	Baja	Alta	Baja
DatObjeto	1	Bajo	Baja	Alta	Baja
NomObjeto	0	Ninguno	Baja	Alta	Baja
DatOperacion	2	Medio	Alta	Baja	Alta
NomOperaciones	0	Ninguno	Baja	Alta	Baja
NomConceptooperacon	0	Ninguno	Baja	Alta	Baja
DatContabilidadadicional	0	Ninguno	Baja	Alta	Baja
DatCuentaobjeto	0	Ninguno	Baja	Alta	Baja
ContabilizarCancelacion	0	Ninguno	Baja	Alta	Baja
Manager	1	Bajo	Alta	Baja	Alta
Comprobantetipo	1	Bajo	Alta	Baja	Alta
Asiento	1	Bajo	Alta	Baja	Alta
AsientoCA	0	Ninguno	Baja	Alta	Baja
Pase	1	Bajo	Alta	Baja	Alta
Anexo	0	Ninguno	Baja	Alta	Baja
DatReglacontable	3	Alto	Alta	Baja	Alta

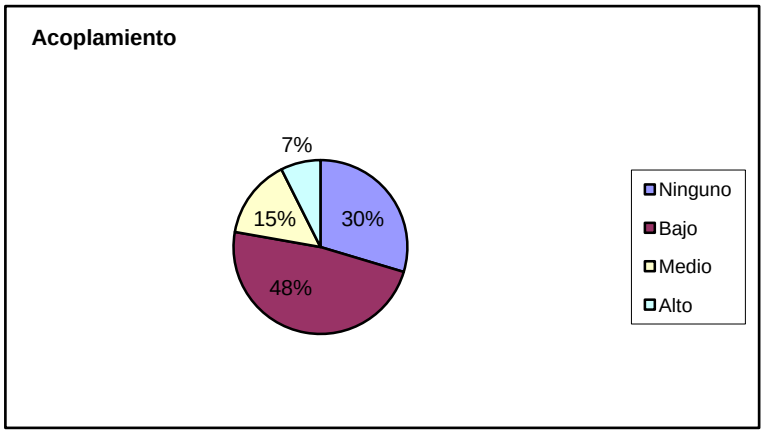

Figura 22(Resultados de la evaluación de la métrica RC para el atributo Acoplamiento)

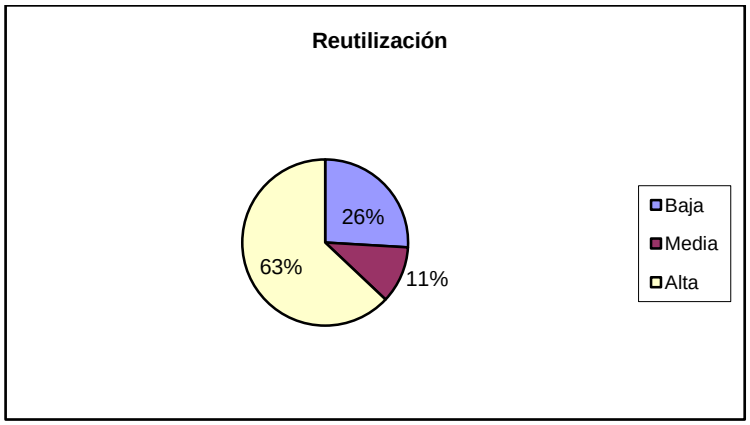

Figura 23(Resultados de la evaluación de la métrica RC para el atributo Reutilización)

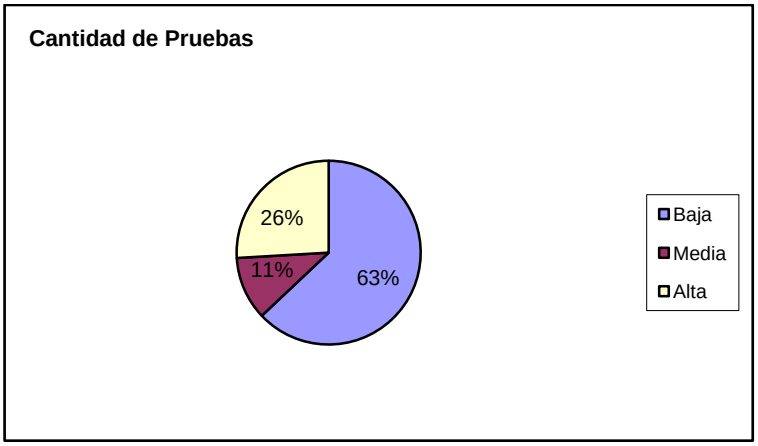

Figura 24(Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas)

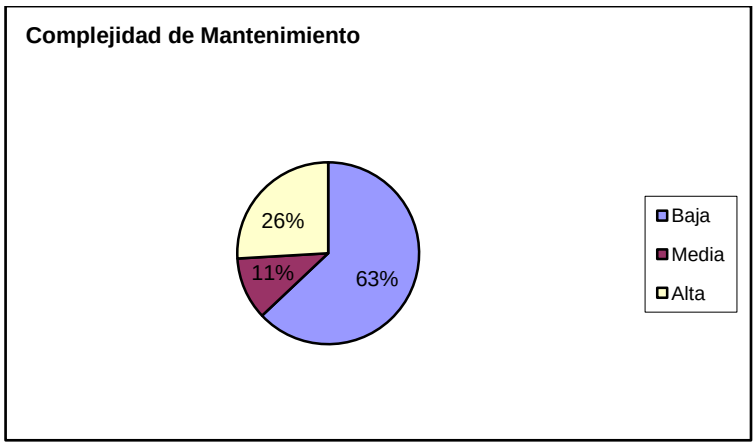

Figura 25(Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento)

3.1.3 Matriz de inferencia de indicadores de calidad

La matriz inferencia de indicadores de calidad, también conocida como matriz de cubrimiento, es una representación estructurada de los atributos de calidad y métricas utilizadas para evaluar la calidad del diseño propuesto. La misma permite conocer si los resultados obtenidos de las relaciones atributo/métrica son positivos o no, llevando estos resultados a una escalabilidad numérica donde, si los resultados son positivos se le asigna el valor de 1, si son negativos toma valor 0 y si no existe relación es considerada como nula y se representa con un guión simple (-). Luego se puede obtener un resultado general para cada atributo promediando todas sus relaciones no nulas.

A continuación se muestran los resultados obtenidos.

Tabla 9 (Resultados de la evaluación de la relación atributo/métrica).

Atributos\Métricas	TOC	RC	Promedio
Responsabilidad	1	(-)	1
Complejidad de Implementación	1	(-)	1
Reutilización	1	1	1

Acoplamiento	(-)	1	1
Complejidad de Mantenimiento	(-)	1	1
Cantidad de pruebas	(-)	1	1

Tabla 10 (Rango de valores para la evaluación de la relación atributo/métrica).

Categoría	Rango de valores
Malo	<=0.4
Regular	>0.4 y <0.7
Bueno	>=0.7

Luego de haber aplicado las métricas se puede ver en la tabla anterior si el diseño propuesto es Bueno Regular o Malo y según los resultados de la matriz atributo/métricas se puede decir que el resultado es lo esperado y es que el diseño tiene buena calidad.

3.2 Resultados de la propuesta en los escenarios funcionales de las Reglas

Luego de analizar mediantes métricas se software el diseño propuesto se decidió hacer una análisis de cada escenario funcional para ver si el mismo cumplió con lo planteado en la problemática inicial. Se puede ver cada uno de los escenarios funcionales como tuvo su solución ene l diseño y que ventajas puede traer los mismos.

Tabla 11(Resultados de la propuesta en los escenarios funcionales de las Reglas.)

Escenarios funcionales	Solución en el diseño	Ventajas
Asociación documento - objeto	Se creó una nueva entidad denominada Regla Contable para tener una relación entre objetos-documentos.	Usabilidad
Generación de Comprobantes	Se le aplicó el patrón de diseño Cadena de Responsabilidades. .	Baja complejidad de mantenimiento y soporte
Clasificación de operaciones	En el modelo de datos la entidad concepto operación se hizo arbórea.	Usabilidad Comodidad Eliminación de repeticiones
Asociación de subsistemas y objetos.	Se modificó la entidad objeto agregándole el concepto subsistema	Elimina ambigüedad Usabilidad

Conclusiones

Luego de realizar las pruebas de TOC y RC se da como concluido que el diseño propuesto luego de resolver cada una de las problemáticas inicial, también cumple con los requisitos de calidad por lo que fue cumplido cada uno de los objetivos trazados inicialmente. Se puede ver mediante el análisis de los resultados que cada requisito funcional cubre con las problemáticas asociadas a las capacidades funcionales.

Conclusiones generales:

La realización del presente trabajo ha posibilitado cumplir con los objetivos propuestos, por lo que se pueden plantear las siguientes conclusiones:

1. Los ERP estudiados no se asemejaban a lo buscado en la investigación, solamente Openbravo que tenia clasificaciones de Reglas.
2. El diseño propuesto cubre con los requisitos funcionales identificados.
3. En las validaciones de diseño y el análisis de los resultados se obtuvieron los resultados esperados.

Recomendaciones

Luego de realizada la investigación y de haber obtenido los resultados esperados de la misma se recomienda:

- Implementar la nueva versión del componente Reglas Contables apoyándose completamente en el desarrollo de este documento.

Bibliografía

1. *http://www.businessrulesgroup.org/.* **Group, the Business Rule.** 2011.

2. *http://www.gartner.com/.* **Insight, Gartner Technology Business Research.** 2011.

3. **Silveira, Luciano.** *Tesis de Maestría.* 2010.

4. **Codorniu, Msc Cesar Lage.** *Solución arquitectónica de la Configuración General del sistema para la parametrización.* La Habana : s.n., 2011.

5. **OpenBravo.** *Manual de Usuario.* 2006.

6. *http://www.sap.com/spain/solutions/business-suite/erp/financials.epx.* **SAP.** 2011.

7. **OpenERP.** *http://www.openerpspain.com/gestion-contable-y-financiera.* 2011.

8. **Versat-Sarasola.** *Manual de usuario y de explotacion Versat-Sarasola.*

9. patrone grasp. *http://jorgesaavedra.wordpress.com/2007/05/08/patrones-grasp-patrones-de-software-para-la-asignacion-general-de-responsabilidadparte-ii/.* [En línea]

10. patrones gof. *http://geektheplanet.net/5462/patrones-gof.xhtml.* [En línea]

11.cadena de responsabilidades.
http://www.portalfox.com/index.php?name=Sections&req=viewarticle&artid=157&page=1. [En línea]

12. mvc. *http://www.proactiva-calidad.com/java/patrones/mvc.html.* [En línea]

13. *http://www.uml.org/..* **UML.** 2010.

14. *http://jorgesaavedra.wordpress.com...* **Informático, El Mundo.**

15. *http://www.ineter.gob.ni/sig/docs/Intro%20IDE%20-%20IDEE.pdf...* **IDEE, El proyecto.**

16. **Gof, Patrones.** http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php. [En línea] 2011.