Universidad de las Ciencias Informáticas

FACULTAD 1

"Vistas de arquitectura de sistema y datos de los Subsistemas Contabilidad y Costos y Procesos del proyecto ERP Cuba."

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Martha Elena Hevia Polanco

Tutor: Ing. Erich M. Gómez Pérez

Co- tutor: Ing. Joisel Pérez Pérez

Ciudad de la Habana, Cuba

Junio, 2010

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro para la Informatización de Gestión de Entidades de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Martha Elena Hevia Polanco.

_____

"No vayas por donde el camino te lleve, ve por donde no haya camino y deja un sendero"

Ralph Waldo Emerson

DATOS DE CONTACTO

- **Ing. Erich Mario Gómez Pérez**

Graduado de Ingeniero en Ciencias Informáticas en la UCI en el año 2008. Actualmente es instructor recién graduado, posee un año de experiencia como Arquitecto principal de datos del proyecto ERP-Cuba y figura como jefe del proyecto Mantenimiento.

- **Ing. Joisel Pérez Pérez**

Graduado de Ingeniero en Ciencias Informáticas en la UCI en el año 2008. A partir de ese momento se incorpora como Arquitecto de datos en el proyecto ERP Cuba. Actualmente se desempeña como coordinador del subsistema Costos y Procesos del proyecto ERP Cuba. En su trabajo de diploma investigó sobre las arquitecturas de software y su relación con los atributos de calidad haciendo una propuesta de un modelo para seleccionar los estilos y patrones más adecuados a la hora de potenciar o inhibir dichos atributos. Ha tutorado una tesis y ha sido oponente de dos, ha impartido cursos de pregrado de Datawarehouse.

Agradecimientos

Quiero agradecer en primer lugar a mi mamá y a mi abuela, por todo lo que me han dado y enseñado, porque siempre confiaron en mí, por quererme tanto. A mi prima Nixys que cada vez que se me antoja algo ahí está. A mi papá. A Camilo por soportar muchas de mis malacrianzas. A mi familia por su apoyo. A Diana por la ayuda que medio en el último momento. A todas mis amigas y amigos, los que en su momento me demostraron que ahí estaban para lo que hiciera falta. A todas las personas que de una forma u otra me ayudaron y facilitaron las cosas. A mis tutores por su atención. A todos los que en estos cinco años hicieron que la vida fuera fácil y también a los que la pusieron difícil. En fin a todos con los que pude contar algún día.

¡GRACIAS!

Dedicatoria

En especial a mi abuela y a mi madre, las personas principales que han estado conmigo en todo momento. A mi familia en general. A todos los que de una forma u otra ocupan un lugar importante para mí.

RESUMEN

La base para la construcción de cualquier sistema, incluidos los Sistemas de Planificación de Recursos Empresariales, es la arquitectura. El objetivo esencial del presente trabajo de diploma es diseñar la arquitectura de los subsistemas Contabilidad y Costos y Procesos del proyecto ERP Cuba, específicamente las vistas arquitectónicas de sistema y datos de dichos subsistemas. En la vista arquitectónica de sistema se generan varios artefactos siendo el resultado más importante el Documento Línea Base (el cual incluye: la complejidad y criticidad de los componentes, su especificación y la matriz de integración) y en la vista arquitectónica de datos: el modelo de datos, el diccionario de datos, la matriz de trazabilidad y las pruebas de conceptos, teniendo como objetivo fundamental controlar, representar y describir todo lo relacionado con la Arquitectura de sistema y la Arquitectura de datos de los subsistemas Contabilidad y Costos y Procesos. De esa manera se facilita el entendimiento de la solución y se apoya el soporte y desarrollo de nuevas versiones del sistema.

PALABRAS CLAVE

Vistas arquitectónicas de sistema y datos, Contabilidad, Costos y Procesos.

TABLA DE CONTENIDOS

ÍNDICE DE ANEXOS

INTRODUCCIÓN

El andar inevitable del tiempo ha llevado al dinámico y creciente desarrollo de productos informáticos que satisfagan las continuas necesidades de la humanidad. Hoy más que nunca las empresas necesitan centralizar y controlar su información, así como garantizar que la misma sea íntegra y confiable para asegurar el cumplimiento de objetivos estratégicos y una mejor toma de decisiones.

Una de las alternativas para garantizar lo antes mencionado son los Sistemas de Planificación de Recursos Empresariales (ERP según sus siglas en inglés); estos son sistemas integrales, adaptables y modulares. Se caracterizan por estar compuestos por diferentes partes integradas en una única aplicación, llamada módulo.

Cuba no está exenta a esta tendencia, y es por ello que se encuentra inmersa en el desarrollo de un proyecto dedicado a la construcción de un ERP propio denominado Cedrux, coordinado por el Ministerio de Finanzas y Precios, compuesto por especialistas de diversos organismos, estudiantes y profesores de la Universidad de las Ciencias Informáticas.

Dos de los subsistemas dentro del Proyecto ERP Cuba son Contabilidad y Costos y Procesos, ambos llevan un año en desarrollo sin embargo no cuentan con la total descripción arquitectónica de las vistas de la arquitectura de sistema y datos. La falta en dichos subsistemas de datos relacionados con los componentes como por ejemplo: la complejidad y la criticidad, la especificación de los servicios brindados, la integración entre estos mediante servicios, la descripción de las clases controladoras y del modelo de negocio, además de los modelos de datos, la descripción de tablas, atributos y relaciones, entre otros, dificulta acciones como:

- El desarrollo de nuevas aplicaciones en otros proyectos paralelamente al desarrollo del subsistema.
- La reutilización de la solución en nuevos componentes.
- El soporte y mantenimiento por un equipo de desarrollo nuevo.

Según la situación antes descrita el presente trabajo de diploma define el siguiente **problema a resolver:** el estado actual de la descripción arquitectónica de sistema y datos de los subsistemas Contabilidad y Costos y Procesos, afecta la capacidad de entendimiento de los involucrados y por consiguiente el soporte y desarrollo de nuevas versiones del sistema.

Después de lo antes mencionado se establece como **objeto de estudio** la arquitectura de sistema y datos, para el cual se propone el siguiente **campo de acción**: arquitectura de sistema y datos de los subsistemas Contabilidad y Costos y Procesos.

Para dar respuesta al problema planteado se trazó el siguiente **objetivo general**: Construir las vistas de arquitectura de sistema y datos de los subsistemas Contabilidad y Costos y Procesos para facilitar su entendimiento por los involucrados y apoyar el soporte y desarrollo de nuevas versiones del sistema.

Objetivos específicos:

1. Realizar marco teórico de la investigación.
2. Realizar la vista de arquitectura de sistema de los subsistemas Contabilidad y Costos y Procesos.
3. Realizar la vista de arquitectura de datos de los subsistemas Contabilidad y Costos y Procesos.

A raíz de lo anteriormente expuesto se define la siguiente **idea a defender:** Si se construyen las vistas de arquitectura de sistema y datos de los subsistemas Contabilidad y Costos y Procesos se facilitará su entendimiento por los involucrados y apoyará el soporte y desarrollo de nuevas versiones del sistema.

Tareas para cumplir los objetivos:

1. Estudio de los artefactos arquitectónicos.
2. Estudio y análisis de la solución que ya existe para poder comprenderla y llenar los artefactos de arquitectura.
3. Estudio de los estándares de documentación utilizados por el proyecto.
4. Estudio de los artefactos de la vista de arquitectura.
5. Realización del documento complejidad y criticidad.
6. Realización del documento especificación de componentes.
7. Realización de la matriz de integración.
8. Realización del documento Línea Base.
9. Descripción del modelo de datos.
10. Realización del diccionario de datos.
11. Realización de la matriz de trazabilidad.
12. Realización de las pruebas de conceptos.

El documento se divide en tres capítulos:

Capítulo 1. Fundamentación Teórica: A través de este capítulo se analizan conceptos relacionados con el dominio del problema entre los que se destacan: Arquitectura de software, Vistas arquitectónicas,

Arquitectura de sistema y Arquitectura de datos. Se realiza un estudio de las soluciones arquitectónicas empleadas por otros sistemas ERP hasta centrar la investigación en el ERP Cuba.

Capítulo 2. Vista de la arquitectura de sistema de los subsistemas Contabilidad y Costos y Procesos: En este capítulo se construyen los artefactos que conforman la vista de sistema, los cuales son: complejidad y criticidad, especificación de componentes, matriz de integración y documento Línea Base, también se analiza la prioridad de los componentes de ambos subsistemas y el diagrama de componentes.

Capítulo 3. Vista de la arquitectura de datos de los subsistemas Contabilidad y Costos y Procesos: En este capítulo se construyen los artefactos que conforman la vista de datos, los cuales son: modelos de datos, el diccionario de datos, la matriz de trazabilidad y las pruebas de concepto.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

El presente capítulo brinda conceptos de suma importancia para la comprensión y desarrollo de dicho trabajo de diploma, cuenta con definiciones relacionadas con el objeto de estudio definido, así como diversos criterios acerca de la arquitectura de software, vistas arquitectónicas, beneficios de la arquitectura de software, tipos de arquitecturas existentes, arquitectura de sistema y datos, además se realiza un estudio acerca de las diversas soluciones arquitectónicas de sistemas ERP hasta centrarse en el ERP Cuba.

1.2. Conceptos relacionados con la Arquitectura de Software

Arquitectura de Software

La arquitectura de software es la portadora primaria de las cualidades del sistema tales como el rendimiento, la modificabilidad, la seguridad, entre otras, ninguna de las cuales se lograría sin una acertada visión arquitectónica. La arquitectura de software es un artefacto para el análisis temprano con el objetivo de asegurar un rendimiento aceptable, es la clave para la comprensión del sistema.

Una definición reconocida es la de Clements: “La arquitectura de software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones.” (Clements, 1996)

Entre las definiciones del campo de la arquitectura de software, existe un general acuerdo de que ella se refiere a la estructura a grandes rasgos del sistema, estructura consistente en componentes y relaciones entre ellos (Bass, 2003). Estas cuestiones estructurales se vinculan con el diseño, pues la arquitectura de software es después de todo, una forma de diseño de software que se manifiesta tempranamente en el proceso de creación de un sistema; pero este diseño ocurre a un nivel más abstracto que el de los algoritmos y las estructuras de datos. En el que muchos consideran un ensayo seminal de la disciplina, Mary Shaw y David Garlan, sugieren que dichas cuestiones estructurales incluyen organización a grandes

rasgos y estructura global de control, protocolos para la comunicación, la sincronización y el acceso a datos, la asignación de funcionalidad a elementos del diseño, la distribución física, la composición de los elementos de diseño, escalabilidad, rendimiento y selección entre alternativas de diseño (Garlan, 1994).

Después de analizar los conceptos anteriormente mencionados se decide que la definición que guiará la presente investigación es la que brinda el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) en el documento Std 1471-2000, la cual ha sido adoptada también por Microsoft y reza así: “La arquitectura de software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución”.

De manera general sobre la definición que plantea el IEEE se puede decir que la arquitectura de software es la encargada de evitar el fracaso de una aplicación, siempre y cuando se definan las prácticas y elementos más convenientes para su construcción.

Arquitectura de Sistema

Dentro de la arquitectura de software se encuentra la arquitectura de sistema y al igual que la arquitectura de software no posee un único concepto, existen diferentes definiciones de esta. A continuación se brindan algunos elementos importantes para definir una arquitectura de sistema.

La arquitectura de sistema no es más que el diseño conceptual que define la estructura y/o el comportamiento de un sistema. Define la organización principal de un sistema basada en sus componentes y sus relaciones. Se considera además que la arquitectura de sistema se puede subdividir en tres etapas como se muestra en la figura 1:

- Elección del estilo arquitectónico.
- Selección de los patrones de diseño.
- Diseño de componentes.

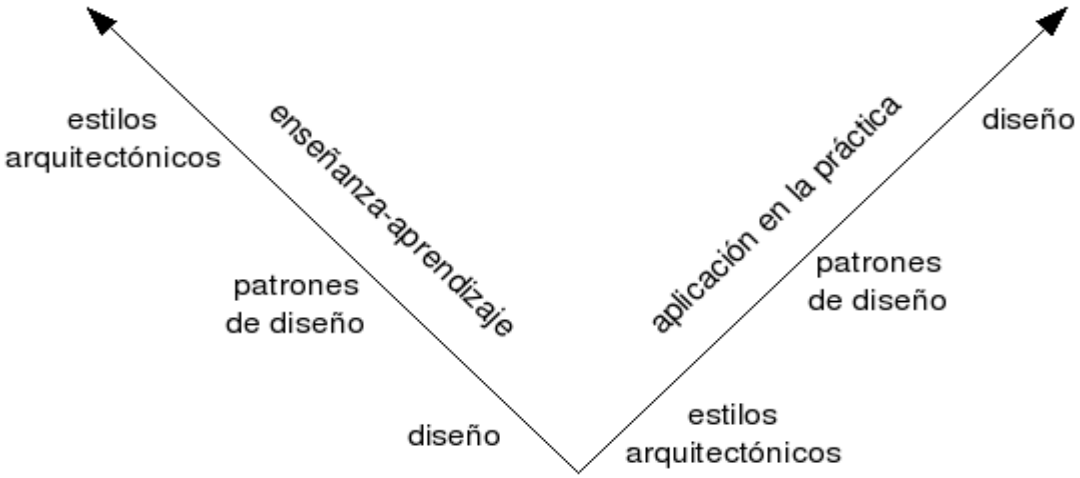


Figura 1: Las etapas de la fase Arquitectura del Sistema. (Cristiá, 2007).

Notar, por otra parte, que el proceso de enseñanza-aprendizaje de todo lo relativo a la fase arquitectura del sistema discurre en el sentido opuesto al que se aplica en la práctica como se intenta graficar en la Figura 1. En otras palabras, se enseña/aprende primero a diseñar componentes, luego se aprende/enseña a utilizar patrones de diseño y finalmente se aprende/enseña lo relativo a estilos arquitectónicos, aunque en la práctica deba seguirse el camino opuesto. (Cristiá, 2007)

Arquitectura de Datos

La arquitectura de datos identifica y define las clases de datos que apoyan las funciones del negocio, definidas en el modelo de negocio, es una de las primeras arquitecturas a ser definidas porque la calidad de los datos es el producto básico. Dicha arquitectura consta de entidades de datos, cada una de las cuales tiene atributos y relaciones con otras entidades de datos, establece las directrices comunes para las operaciones de datos que permitan predecir y controlar el flujo de datos en el sistema, describe los grupos de datos y sus elementos así como asigna los artefactos de datos. (Wiggins, Andrew y Richard, Johnston, 2003).

Beneficios de la Arquitectura de Software

Algunos de los beneficios que brinda la arquitectura de software son (Kontio, 2005), (Boehm, 1995), (IEEE 1471, 2000):

- Decisiones tempranas.

- Predecir y lograr la calidad de la conducta de control de atributos y hacer transacciones prácticas tempranas.
- Análisis de consistencia antes de elaborar el diseño (y escribir el código).
- Sistematización de la experiencia.
- Herramientas para la evolución.
- Reduce considerablemente las tasas de fracaso del proyecto.
- Permite la comunicación con las partes interesadas.
- Permite la gestión de cambios.
- Logra los costos más precisos y el calendario esperado.
- Crea prototipos evolutivos.
- Previene y mitiga los riesgos.
- Proporciona una idea de cómo interactúan los adjetivos de calidad.
- Permite planificar las necesidades personales.
- Re-utilización.

Después de todo lo antes expuesto es de suma importancia mencionar que la arquitectura consta de múltiples vistas asociadas a diferentes perspectivas o dimensiones de un sistema.

Vistas arquitectónicas

La arquitectura está representada por un número de vistas arquitectónicas. Estas vistas capturan las mayores decisiones de diseño estructural. En esencia, las vistas arquitectónicas son abstracciones o simplificaciones del diseño entero, en las cuales las características importantes se hacen más visibles dejando los detalles a un lado.

La vista arquitectónica, aporta el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones. (Bass, 1998)

Una vista arquitectónica es la representación coherente de elementos arquitectónicos y sus relaciones. Todo el trabajo que se lleva a cabo para estructurar una arquitectura, es necesario tenerlo en cuenta para separar las vistas, tratando de no dejar ningún contenido fuera de ellas. Como es lógico no es nada sencillo de hacer, se derivan muchas dudas acerca de donde ubicar un contenido, adicionándolo a alguna vista ya definida o creando una nueva sólo para él. (Mena, 2009).

1.2.1. Clasificaciones de las vistas

- Libro: “Software Systems Architecture” (Rozanski y otros, 2005):
  - ✓ Vista funcional.
  - ✓ Vista de información.
  - ✓ Vista de concurrencia.
  - ✓ Vista de desarrollo.
  - ✓ Vista de despliegue.
- SEI: Racionalización de Vistas.

  El SEI emplea el término en inglés Viewtypes, en español consiste en la categorización de las vistas.
  - ✓ ViewType Modular: ¿Cómo está estructurado el sistema como conjunto de unidades de implementación?
  - ✓ ViewType Componente y Conector: Como conjunto de elementos que tienen comportamiento e interacción en tiempo de ejecución.
  - ✓ ViewType Asignación: ¿Cómo se relaciona el software con elementos que no son software?
- Propuestas de Arquitectura 4+1 Vistas de RUP (Kruchten, 2003):
  - ✓ Vista lógica: describe el modelo de objetos.
  - ✓ Vista de procesos: muestra la concurrencia y sincronía de los procesos.
  - ✓ Vista física: muestra la ubicación del software en el hardware.
  - ✓ Vista de desarrollo: describe la organización del entorno de desarrollo.

Existe una quinta vista que consiste en la selección de casos de usos o de escenarios (Vista de casos de uso) que los arquitectos pueden elaborar a partir de las cuatro vistas anteriores. Ver Figuras 2 y 3.

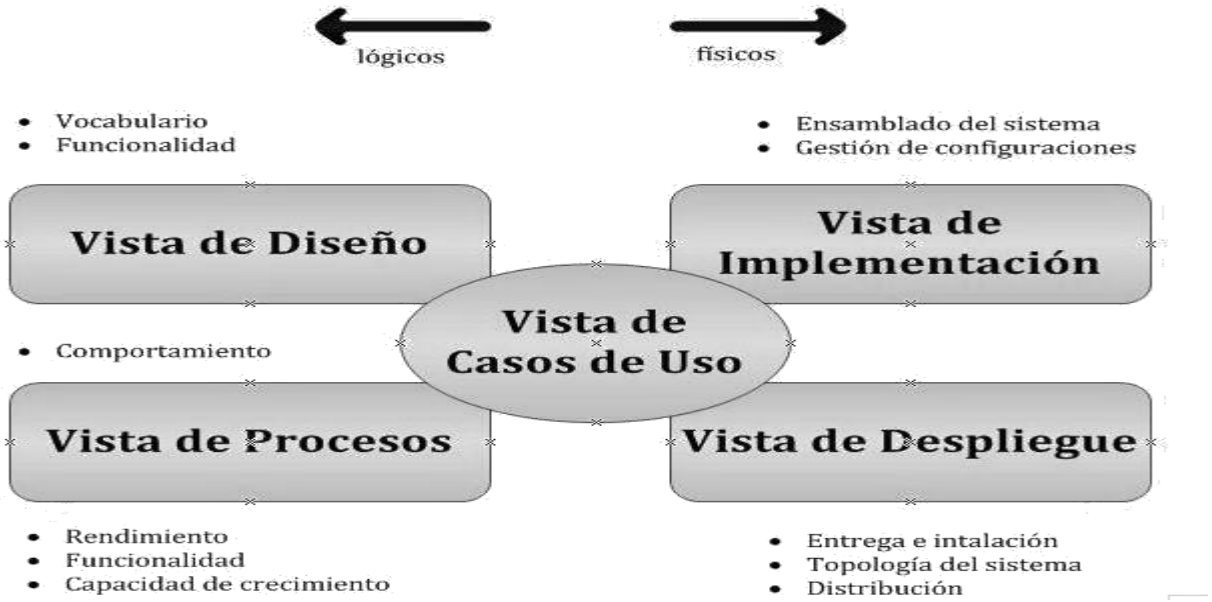


Figura 2: 4+1 vistas de RUP. (Kruchten, 2003)

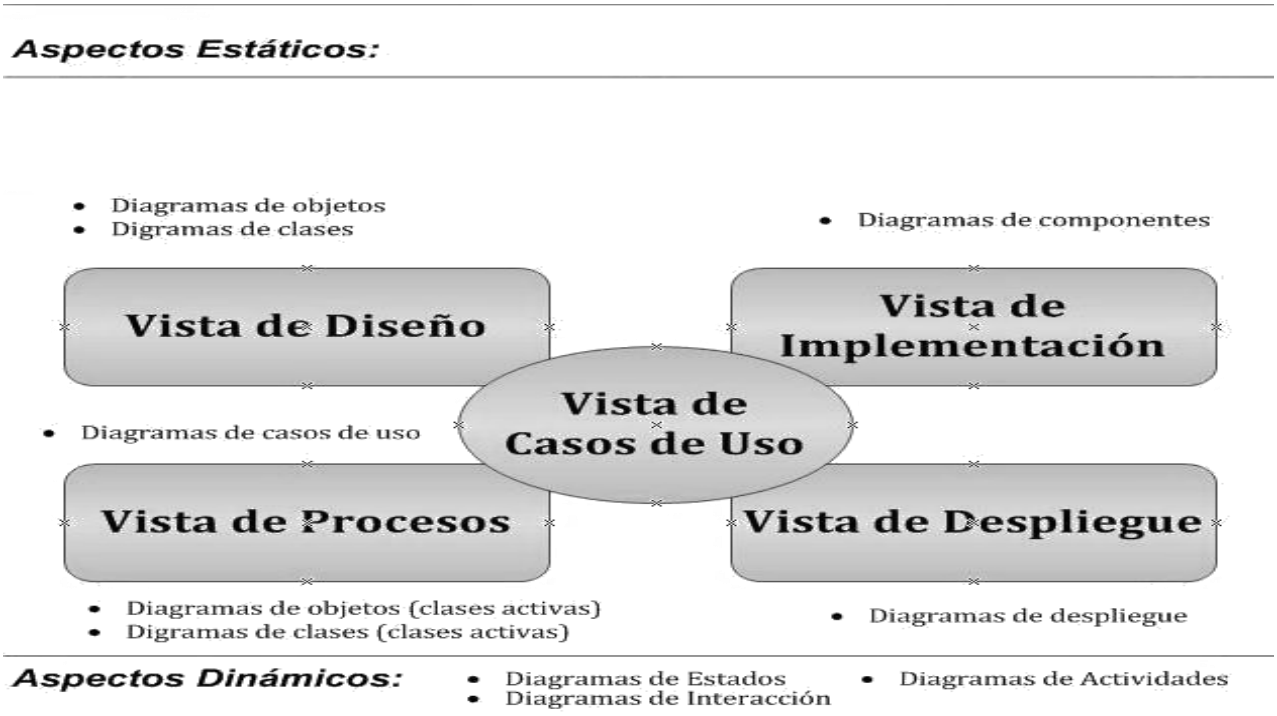


Figura 3: 4+1 vistas de RUP. (Kruchten, 2003)

- Vistas propuestas por Microsoft (Mena, 2009):
  - ✓ Vista conceptual: Define los requerimientos funcionales y la visión que los usuarios del negocio tienen de la aplicación. Muestra los subsistemas y módulos en los que se divide la aplicación:
    Casos de Uso, Diagramas de Actividad, Procesos de Negocio, etc. Ver Figura 4.

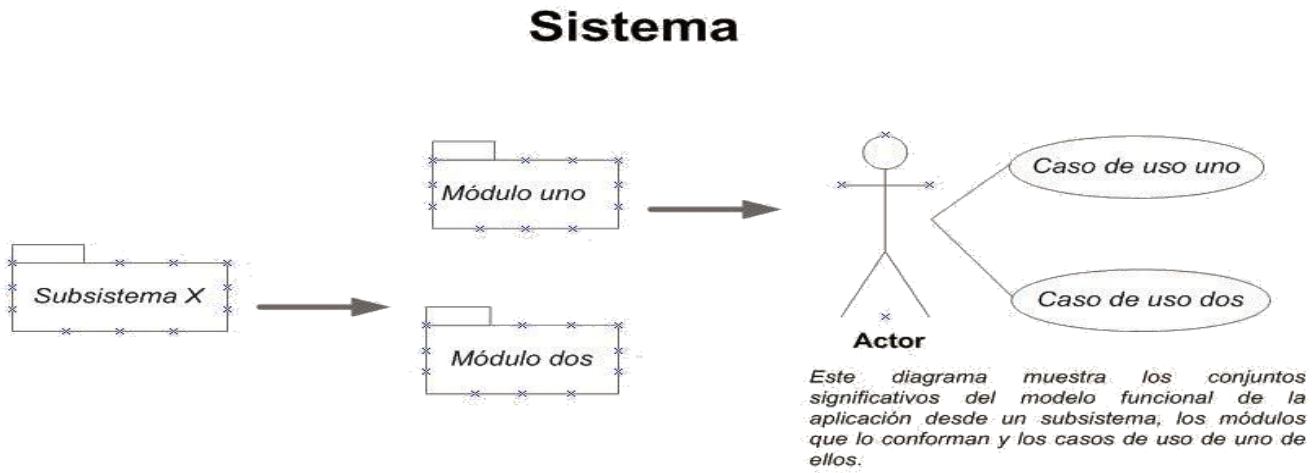


Figura 4: Vista Conceptual. (Mena, 2009)

  - ✓ Vista Lógica: Muestra los principales componentes de diseño y sus relaciones de manera independiente de los detalles técnicos:
    Realización de Casos de Uso, subsistemas, paquetes y clases de los casos de uso más significativos. Ver Figura 5.

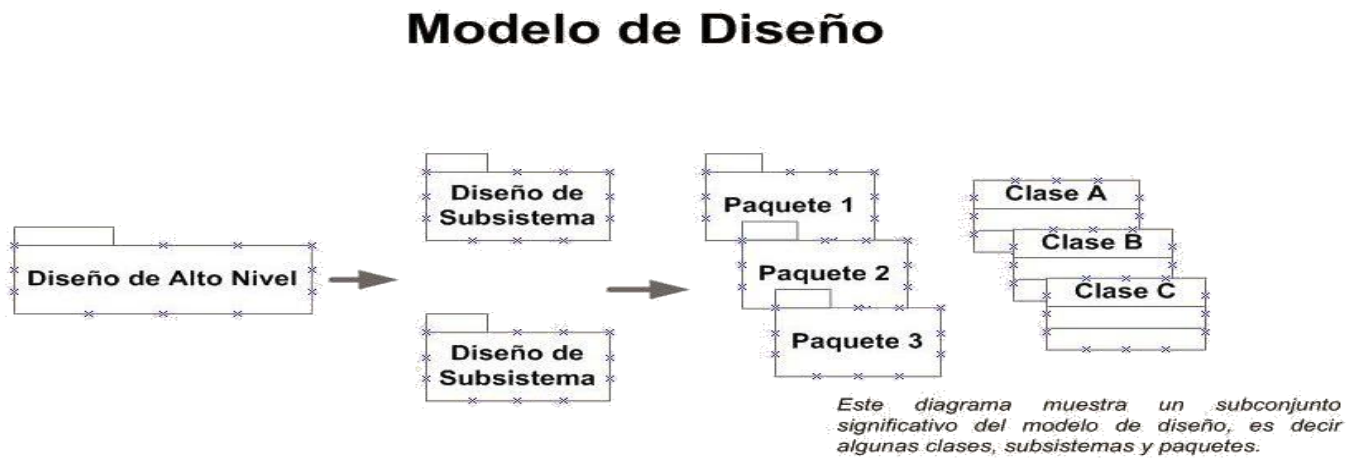


Figura 5: Modelo de Diseño. (Mena, 2009)

Dentro de la vista lógica uno de los patrones de diseño más empleado es el de Capas, donde se dividen los elementos de diseño en paquetes de interfaz de usuario, lógica de negocio y acceso a datos y servicios. Ver Figura 6.

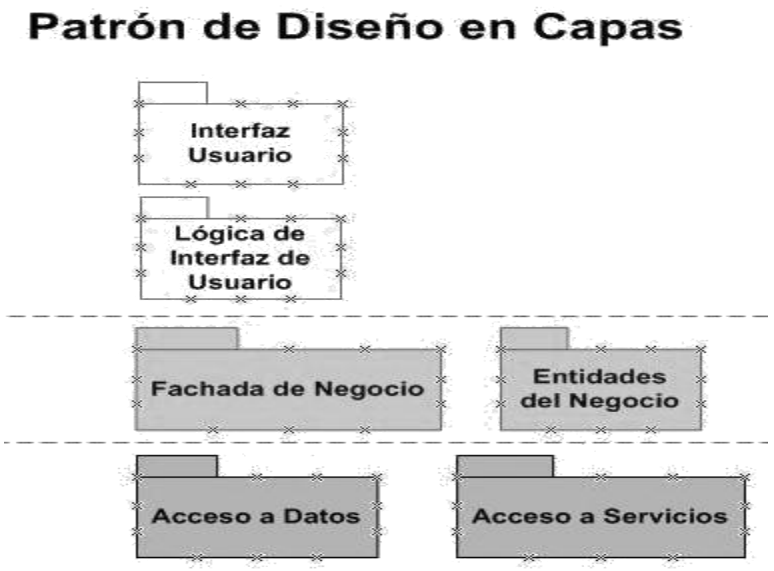


Figura 6: Diagrama de Paquetes. (Mena, 2009)

- ✓ Vista Física: Muestra la distribución de los distintos equipos que conforman la solución, incluyendo los servicios y procesos de base. Ver Figura 7.

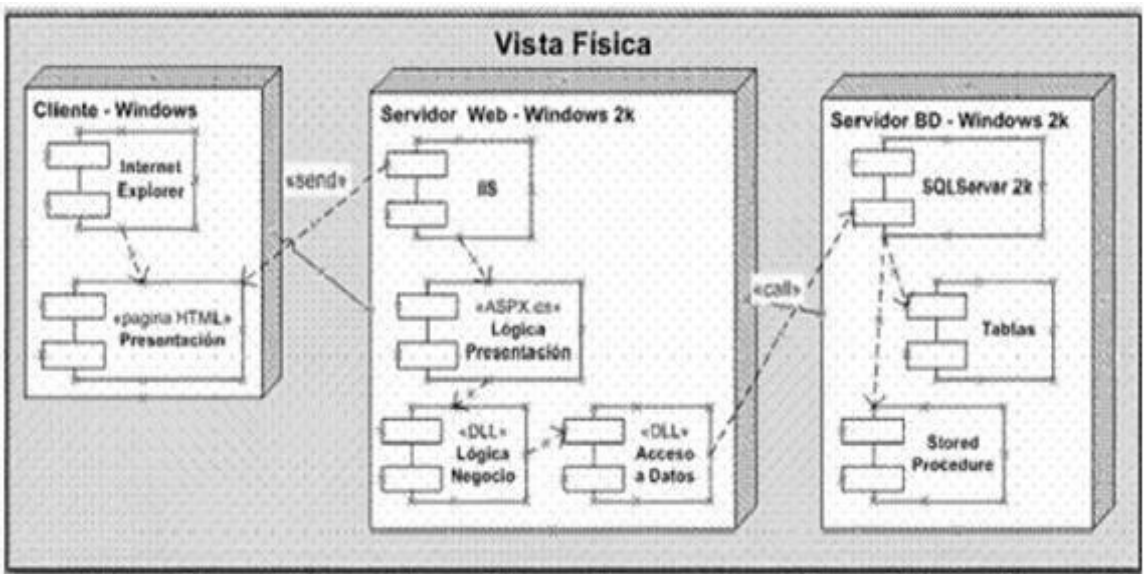


Figura 7: Vista Física (Diagrama de despliegue). (Mena, 2009)

- ✓ Vista de Implementación: Describe cómo se implementan los componentes físicos mostrados en la vista de distribución agrupándolos en subsistemas organizados en capas y jerarquías. Ver Figura 8.

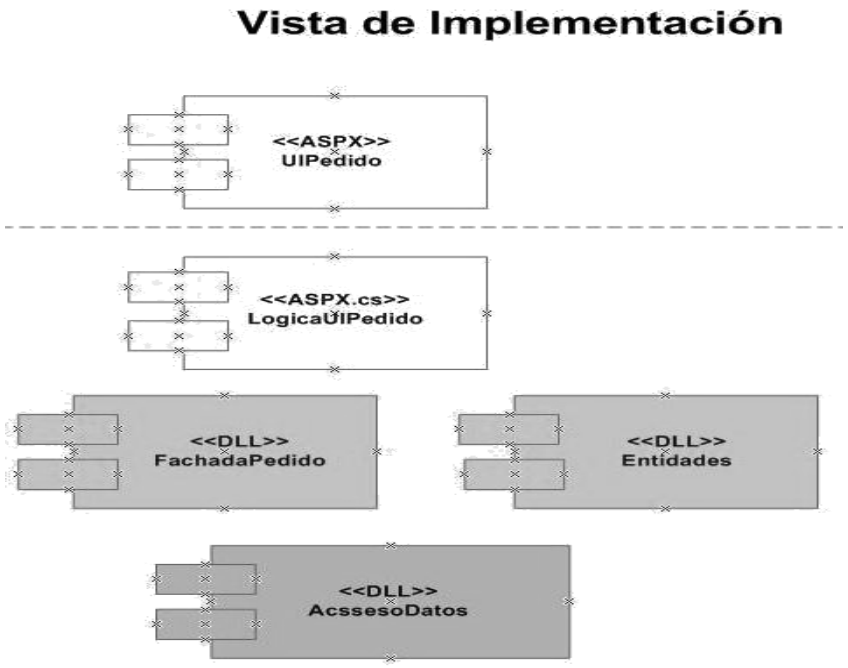


Figura 8: Vista de Implantación (Diagrama de componentes). (Mena, 2009)

Luego de haber analizado las anteriores definiciones de los diferentes autores acerca de las vistas, muchas de ellas no se ajustaban a las necesidades del ERP Cuba, enfocadas en el objetivo de agilizar el proceso de desarrollo de software y lograr que el soporte y mantenimiento del sistema fuera basado en los artefactos netamente necesarios. Debido a todo lo antes mencionado se definieron las siguientes vistas por el equipo de trabajo de arquitectura, estas son: la vista de tecnología, la vista de integración, la vista de infraestructura, la vista de presentación, la vista de seguridad, la vista de sistema y la vista de datos. Seguidamente se explican las vistas concernientes al presente trabajo de diploma: vista de sistema y vista de datos.

Vista de sistema

La vista de sistema es la forma de representación y descripción de la arquitectura de sistema que es una de las más complejas dentro de la arquitectura de software. Es la encargada de definir las principales interacciones, los conectores, y las configuraciones que serán asumidas por los componentes computacionales en función de los elementos del negocio que los mismos abstraen. La vista de sistema

constituye una proyección simétrica de alto nivel de los procesos de negocio o arquitectura de negocio que se trabaja, expresada en elementos, conectores, restricciones y configuraciones.

Los principios de empaquetamiento del diseño arquitectónico de un sistema de software tienen alto impacto en el diseño de la solución, debido a que esta actividad está vinculada con los niveles de colaboración de los elementos computacionales, los niveles abstracción y encapsulamiento y los principios de reutilización en función del problema que se modela. La correcta selección de los mismos implica mejor organización del equipo de desarrollo y facilita los procesos de Gestión de Configuración de Software. (Mena, 2009).

Vista de datos

La vista de datos describe y representa la arquitectura de datos la cual es la responsable de diseñar las soluciones de datos en cada uno de los subsistemas, diseña las soluciones de integración en los mismos, construye la vista de datos general y la vista de datos particular para cada subsistema. Dicha vista tiene como objetivos fundamentales definir las políticas de integración y los objetivos de trabajo en función del Centro de Datos, definir la metodología de trabajo de la arquitectura de datos, definir metodología de implementación de réplica, definir, construir y actualizar permanentemente el expediente de la vista de datos, definir método de actualización y control de versiones de la base de datos y definir indicadores y métricas para el manejo de los datos. (Mena, 2009).

1.3. Tipos de Arquitecturas existentes

Son varios los tipos de arquitecturas que existen en la actualidad, seguidamente se muestran algunos de estos tipos, tomados de un estudio realizado por (Carrascoso y otros, 2008).

Arquitectura orientada a objetos:

A este tipo de arquitectura se le conoce de varias maneras, entre ellas: arquitecturas basadas en objetos, abstracción de datos y organización orientada a objetos. Los componentes de esta son los objetos, o más bien instancias de los tipos de datos abstractos. En la caracterización clásica de David Garlan y Mary Shaw, los objetos representan una clase de componentes que ellos llaman managers, debido a que son responsables de preservar la integridad de su propia representación. Un rasgo importante de este aspecto es que la representación interna de un objeto no es accesible desde otros objetos. (Garlan, 1994). Según Billy Reinoso (Reynoso, 2004), si hubiera que resumir las características de las arquitecturas orientadas a

objetos, se podría decir que: Los componentes se basan en principios orientados a objetos como encapsulamiento, herencia y polimorfismo. Son así mismo las unidades de modelado, diseño e implementación. Los objetos y sus interacciones son el centro de las incumbencias en el diseño de la arquitectura y en la estructura de la aplicación.

Las interfaces están separadas de las implementaciones. En general la distribución de objetos es transparente, y en el estado de arte de la tecnología apenas importa si los objetos son locales o remotos. El mejor ejemplo de orientación a objetos para sistemas distribuidos es Common Object Request Broker Architecture (CORBA), en la cual las interfaces se definen mediante Interface Description Language (IDL); un Object Request Broker (intermediarios en peticiones a objetos) media las interacciones entre objetos clientes y objetos servidores en ambientes distribuidos.

En cuanto a las restricciones, puede admitirse o no que una interfaz pueda ser implementada por múltiples clases. Hay muchas variantes; algunos sistemas, por ejemplo, admiten que los objetos sean tareas concurrentes; otros permiten que los objetos posean múltiples interfaces.

Ventajas:

- Se puede modificar la implementación de un objeto sin afectar a sus clientes.
- Un objeto es una entidad reutilizable en el entorno de desarrollo.
- Encapsulamiento.

Desventajas:

- Para poder interactuar con otro objeto a través de una invocación de procedimiento, se debe conocer su identidad.
- Presenta problemas de efectos colaterales en cascada: si A usa B y C también lo usa, el efecto de C sobre B puede afectar a A.
- Granularidad muy fina para sistemas grandes.

Arquitectura en Capas:

Garlan y Shaw la definen como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior (Reynoso, 2004).

Los conectores se definen mediante los protocolos que determinan las formas de la interacción. Las restricciones topológicas pueden incluir una limitación, más o menos rigurosa, que exige a cada capa operar sólo con capas adyacentes, y a los elementos de una capa entenderse sólo con otros elementos

de la misma. Es de suponer que si esta exigencia se relaja, el estilo deja de ser puro y pierde algo de su capacidad heurística (MSDN, 2004); también se pierde, naturalmente, la posibilidad de reemplazar totalmente una capa sin afectar a las restantes, disminuye la flexibilidad del conjunto y se complica su mantenimiento. En muchas ocasiones se sacrifica la pureza de la arquitectura en capas precisamente para mejorarla, colocando, por ejemplo: reglas de negocios en los procedimientos almacenados de las bases de datos o articulando instrucciones de consulta en la capa de la interfaz de usuario.

Ventajas:

- Modularidad.
- Soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
- Proporciona amplia reutilización.
- Soporta fácilmente la evolución del sistema; los cambios sólo afectan a las capas vecinas. Se pueden cambiar las implementaciones respetando las interfaces con las capas adyacentes.

Desventajas:

- Es difícil razonar a priori sobre la separación en capas requeridas.
- Formatos, protocolos y transportes de la comunicación entre capas suelen ser específicos y propietarios.
- No todos los sistemas pueden estructurarse en capas.

Arquitectura orientada a servicios (SOA):

La arquitectura orientada a servicios (SOA: Service Oriented Architecture) está formada por servicios de aplicación débilmente acoplados y altamente ínter operables. Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma y del lenguaje de programación (por ejemplo WSDL: Web Services Description Language). La definición de la interfaz encapsula las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo (como Java o .NET).

Con esta arquitectura, se pretende que los componentes de software desarrollados sean muy reusables, ya que la interfaz se define siguiendo un estándar; así, un servicio desarrollado en CSharp podría ser usado por una aplicación Java.

Una de las características más relevante de SOA, es que está basada en contratos, donde el proveedor establece las reglas de comunicación, el transporte, y los datos de entrada y salida que serán intercambiados por ambas partes. Otras características de SOA son:

- Es una arquitectura conceptual.
- Organiza funciones de negocio como servicios ínter operables.
- Permite reutilización de servicios para satisfacer necesidades de negocio.
- Es basada en estándares. Independencia de fabricantes.
- Es una arquitectura en la cual se exponen los procesos de negocio del sistema a construir como servicios independientes de alta cohesión y bajo acoplamiento que encapsulan dichos procesos y pueden ser invocados a través de interfaces bien definidas.

Ventajas:

- Mejora la toma de decisiones.
- Panorámica unificada. Más información con mejor calidad.
- Potencia la relación entre clientes y proveedores.
- Mayor capacidad de respuesta a los clientes.
- Aplicaciones más productivas y flexibles.
- Aplicaciones más seguras y manejables.

Desventajas:

- Los tiempos de llamado no son despreciables, gracias a la comunicación de la red, tamaño de los mensajes, entre otros. Esto necesariamente implica la utilización de mensajería confiable.
- La respuesta del servicio es afectada directamente por aspectos externos como problemas en la red, configuración, entre otros.
- Debe manejar comunicaciones no confiables, mensajes impredecibles, reintentos, mensajes fuera de secuencia, etcétera.

Arquitectura basada en componentes:

Actualmente en el desarrollo de software hay una gran necesidad de hacer uso de la reutilización de partes o módulos de software existentes, que podrían ser utilizados para la generación de nuevas extensiones de las aplicaciones o las aplicaciones completas.

Al ser una arquitectura basada en componentes permite la reutilización de los mismos. Para ello los componentes deben satisfacer como mínimo el siguiente conjunto de características:

- **Identificable:** un componente debe tener una identificación clara y consistente que facilite su catalogación y búsqueda en repositorios de componentes.
- **Accesible sólo a través de su interfaz:** el componente debe exponer al público únicamente el conjunto de operaciones que lo caracteriza (interfaz) y ocultar sus detalles de implementación. Esta característica permite que un componente sea reemplazado por otro que implemente la misma interfaz.
- **Servicios son invariantes:** las operaciones que ofrece un componente, a través de su interfaz, no deben variar. La implementación de estos servicios puede ser modificada, pero no deben afectar la interfaz
- **Documentado:** un componente debe tener una documentación adecuada que facilite su búsqueda en repositorios de componentes, evaluación, adaptación a nuevos entornos, integración con otros componentes y acceso a información de soporte.

Ventajas:

- Reutilización del software. Lleva a alcanzar un mayor nivel de reutilización de software.
- Simplifica las pruebas. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
- Simplifica el mantenimiento del sistema. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
- Mayor calidad. Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

Desventajas:

- Si no existen los componentes, hay que desarrollarlos y se puede perder mucho tiempo, por otro lado, estos componentes pueden tener conflictos si de estos sale una nueva versión, por lo que es posible que haya que re-implementarlos.

Es importante mencionar que cada una de las arquitecturas antes descritas es usada para dar solución acorde a las necesidades y servicios que brinde un producto.

1.4. Arquitectura de un sistema ERP

Un ERP es una solución informática integral, está formado por un conjunto finito de módulos que pueden adquirirse total o parcialmente, existen en general tres grandes grupos: el primero correspondiente al área financiera, un segundo grupo al área logística y finalmente un grupo del área recursos humanos. Adicionalmente a los anteriores grupos existen soluciones específicas para sectores industriales particulares. Son varias las empresas de la industria del software que diseñan, desarrollan y comercializan estas soluciones, y aún existiendo diferencias en el producto final presentan ciertas características comunes, entre las que se encuentra la arquitectura, a continuación se muestra un estudio acerca de la arquitectura utilizada en algunos de los ERP existentes en el mundo.

OpenXpertya: Es el ERP de software libre en español y latinoamericano con la mayor red de soporte profesional, el mayor número de usuarios del sistema, e implantaciones finales reales y está situada en el grupo de cabeza de los de mayor madurez. Está desarrollado en tres capas e íntegramente en Java, con lo que funciona sobre cualquier sistema operativo y plataforma (Windows, Solaris, FreeBSD, Linux, UNIX, AIX, MacOS, etc.) sin dependencias de ningún tipo. (OpenXpertya, 2005).

OpenERP: Es un sistema ERP y CRM[1], utilizado para la gestión integrada de los recursos de una empresa. La arquitectura del sistema es cliente – servidor, lo que permite que todos los usuarios trabajen sobre el mismo repositorio de datos. Esto tiene la ventaja de que toda la información está disponible y sincronizada en todo momento, además de que descarga la mayor parte del trabajo de procesamiento de datos de las máquinas cliente (donde trabajan efectivamente los usuarios). (Openerp, 2001).

Openbravo ERP: Se desarrolla utilizando estándares abiertos. La arquitectura empleada por el sistema es MVC (Modelo Vista Controlador). Incorpora muchas ingeniosas características que le hacen destacarse entre la multitud y le convierten en un perfecto software para empresas. (Openbravo, 2007).

De manera general después del estudio realizado acerca de los diferentes tipos de arquitecturas y las soluciones arquitectónicas empleadas en los sistemas ERP antes mencionados, se puede concluir que el ERP Cuba emplea una solución diferente a las utilizadas por los sistemas estudiados, pues esta es basada en componentes unida al patrón arquitectónico Modelo-Vista-Controlador.

[1] CRM: Software CRM por sus siglas en inglés "Customer Relationship Management"(Gerencia de relaciones con los clientes). Mediante el software CRM lo que se logra es conocer más a fondo las necesidades y preferencias de los clientes, y de esta forma poder ofrecerles un producto con mayor valor agregado dependiendo de cada usuario.

1.4.1. Arquitectura del ERP Cuba

La solución arquitectónica del ERP Cuba está enfatizada en la modificabilidad y la escalabilidad, para lograr ambas cualidades la arquitectura seleccionada fue la basada en componentes debido a las características y ventajas que esta posee y que fueron explicadas en el epígrafe 1.3, además se usa como patrón arquitectónico el Modelo-Vista-Controlador, el cual es utilizado dentro de cada uno de los componentes que conforman los subsistemas del proyecto.

Modelo-Vista-Controlador (MVC): Se emplea cuando es necesario modularizar la interfaz de usuario, las reglas de negocio y el control de eventos.

El MVC divide una aplicación interactiva en tres componentes en donde, de manera general, el modelo contiene la capa de datos y la lógica de negocio:

Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador). Mantiene el conocimiento del sistema. No depende de ninguna vista y de ningún controlador.

Vista: Maneja la visualización de la información.

Controlador: Analiza los mensajes de eventos que recibe el sistema y modifica u obtiene datos del modelo en respuesta a las peticiones del usuario. Evita poner código indebido en la capa impropia, por ejemplo instrucciones de base de datos (modelo) en interfaz de usuario (vista).

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual. En aplicaciones Web la separación entre la vista (el browser) y el controlador (los componentes del lado del servidor que manejan los requerimientos de HTTP) está taxativamente muy bien definida. A continuación se muestra una figura que representa dicho patrón dentro del proyecto ERP Cuba.

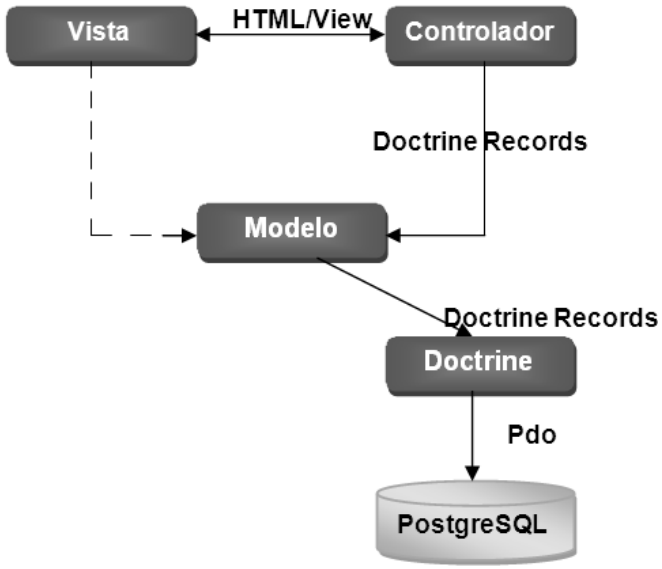


Figura 9: Modelo Vista Controlador del proyecto ERP Cuba

El Controlador representa el gestor de eventos del sistema y se encuentra dentro del paquete controllers, este atiende los pedidos que llegan de la vista accediendo al paquete models, este último paquete a su vez está dividido en los paquetes bussines y domain, el primero encierra las clases donde se realiza la lógica del negocio y el segundo comprende las clases del dominio, o sea, las que leen y/o escriben datos en las tablas de la base de datos.

Dicho patrón fue re-implementado, por el equipo central de arquitectura, de forma tal que las clases controladoras de los diferentes componentes tienen la posibilidad de obtener los datos mediante instanciaciones a objetos de clases del modelo del negocio, o tomándolos directamente del modelo del dominio.

Después de haber analizado la arquitectura del proyecto es necesario aclarar que todo el flujo de trabajo del proceso de construcción de dicha arquitectura debe mantenerse organizado, estructurado y ubicado en un sitio de acceso para el equipo de desarrollo, facilitando el control de versiones y que sirva de guía para el proceso de desarrollo del software en general, para esto el proyecto cuenta con un expediente tecnológico. (CEIGE, 2009).

Expediente tecnológico del proyecto ERP Cuba

El expediente tecnológico de la arquitectura de software del proyecto ERP Cuba está conformado por todas las vistas definidas en el proyecto:

- Arquitectura de Datos.
- Arquitectura de Información.
- Arquitectura de Infraestructura.
- Arquitectura de Presentación.
- Arquitectura de Seguridad.
- Arquitectura de Sistema.
- Arquitectura Tecnológica.

En cada uno de los espacios definidos en el expediente se almacenan todos los artefactos generados en cada una de las áreas, se guarda además toda la documentación con la que se debe contar para el mantenimiento, control y soporte de la solución. En el presente trabajo serán generados y explicados los artefactos que se recogen dentro de la vista de sistema y la vista de datos.

1.4.2. Artefactos generados en el ERP Cuba

De todos los artefactos contenidos en los espacios de los expedientes de la arquitectura de sistema y la arquitectura de datos, en la siguiente tabla se muestra la relación Artefacto-Descripción-Roles, de los que conciernen al presente trabajo. En los capítulos 2 y 3 se abordará el tema de forma más amplia.

Tabla 1: Relación artefacto-descripción-rol. (Leyet, 2010)

Artefacto	Descripción	Roles
Documento Línea Base del Subsistema	Delimita las responsabilidades arquitectónicas de un subsistema. Permite establecer la comunicación y dependencias entre los componentes y módulos que lo integran, así como plasmar la criticidad y complejidad de los componentes con vistas a su reutilización y prioridad en el desarrollo.	Arquitecto de Componentes

Complejidad y Criticidad	Contiene todos los componentes definidos en los subsistemas y por cada componente los requisitos funcionales que implementan, la complejidad de dichos requisitos y una serie de datos desde el punto de vista de integración.	Arquitecto de sistema
Especificación de Componentes	Se listan todos los componentes definidos en los subsistemas y de cada uno de ellos se especifica el nombre, una descripción además de sus características fundamentales y los servicios que brinda.	Arquitecto de sistema
Matriz de Integración	Contiene todos los componentes definidos en los subsistemas de forma matricial y en las intercepciones se especifican los servicios que consume dicho componente.	Arquitecto de sistema
Modelo de datos	Contiene todas las tablas, atributos y relaciones.	Arquitecto de Datos
Diccionario de datos	Contiene una descripción detallada de las tablas, atributos y relaciones del subsistema.	Arquitecto de Datos
Matriz de trazabilidad de datos	Establece las relaciones existentes entre las tablas de los subsistemas.	Arquitecto de Datos
Pruebas de Conceptos	Describe los diferentes escenarios de prueba, junto con los resultados arrojados del subsistema en que se realizan.	Arquitecto de Datos

Para la construcción de algunos artefactos mencionados en la tabla anterior se decidió utilizar como herramienta Visual Paradigm para UML 6.3.

Visual Paradigm 6.3

Visual Paradigm para UML es una herramienta muy empleada en el mundo del software. Esta herramienta tiene capacidad para la ingeniería directa e inversa en varios lenguajes de programación, de igual forma existe disponibilidad de múltiples versiones para cada necesidad. Permite la integración de aplicaciones

empresariales a las bases de datos. Posibilita el manejo de grandes estructuras de manera eficiente, solo requiere una configuración de escritorio común. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegues. (VP, 2006).

1.5. Conclusiones

En el presente capítulo se ofrece una visión acerca de los diferentes conceptos relacionados con la arquitectura de software, vistas de la arquitectura, soluciones arquitectónicas empleadas por otros sistemas, etc. Según el estudio realizado se concluye que la definición de arquitectura de software que adopta el presente trabajo es la que brinda el IEEE, por otro lado, con respecto a las vistas arquitectónicas se identificaron nuevas vistas según las necesidades del proyecto ERP Cuba, además permitió identificar de igual forma la solución arquitectónica empleada en dicho proyecto, por último se menciona la herramienta a utilizar en el desarrollo del trabajo y se brindan aspectos que justifican su uso. De manera general todo lo desarrollado en el capítulo enmarca la importancia de cada uno de los temas que en él aparecen creando las condiciones para conocer la solución propuesta, después de haber sido introducidos los conceptos fundamentales para su comprensión.

CAPÍTULO 2: VISTA DE LA ARQUITECTURA DE SISTEMA.

2.1. Introducción

En el presente capítulo se realiza un análisis de los artefactos del negocio generados en los subsistemas Contabilidad y Costos y Procesos. Este análisis no solo facilita la comprensión del negocio, también es utilizado como punto de partida para proponer una vista arquitectónica de sistema para los subsistemas Contabilidad y Costos y Procesos.

2.2. Análisis de los artefactos generados en el negocio

Los artefactos del negocio constituyen la entrada para definir la vista de arquitectura de sistema y generar los artefactos correspondientes a esta, de ahí la importancia de realizar su análisis. El presente trabajo recoge el estudio de dos de los artefactos del negocio: el mapa de procesos y el listado de los requisitos para cada uno de los subsistemas en cuestión.

Para realizar el análisis de los artefactos antes mencionados es necesario tener en cuenta las descripciones de los procesos en ambos subsistemas. En el subsistema Contabilidad los procesos que se definieron fueron:

Emitir comprobante operaciones: este proceso es el encargado de llevar un registro de todas las operaciones contables que se realizan en una entidad cubana.

Emitir estados financieros: este proceso es el responsable de representar y estructurar la situación financiera y el rendimiento financiero de la entidad.

Realizar apertura de ejercicio contable: este proceso es el encargado de iniciar el ejercicio o año contable para las entidades, poniendo en vigencia todas las cuentas, nomencladores y demás recursos.

Realizar apertura inicial: este proceso es el responsable de hacer la apertura del primer ejercicio contable de la entidad.

Realizar apertura por cambio de sistema: este proceso es el encargado de comenzar el ejercicio con un nuevo sistema contable ya sea por cambio de un sistema anterior o porque la entidad decide adoptar por primera vez un sistema contable.

Realizar apertura: este proceso es el responsable de iniciar los períodos contables para las entidades poniendo en vigencia todas las cuentas, nomencladores y demás recursos.

Realizar cierre cuando es mes 12: este proceso permite conocer el estado de la entidad en el período 12.
Realizar cierre cuando es mes 13: este proceso permite conocer el estado de la entidad en el período 13 o en el ejercicio contable.
Realizar cierre cuando no es mes: este proceso permite conocer el estado de la entidad en un período determinado dentro del ejercicio contable.
Realizar cierre: este proceso permite conocer el estado de la entidad en un período de tiempo definido.
Por otro lado en el subsistema Costos y Procesos los procesos definidos fueron:
Ajuste al costo: este proceso es el encargado de ajustar los costos de producción al final del período.
Definir secuencia de traspasos de gastos: este proceso es el responsables de registrar todos los datos necesarios para definir una secuencia de traspaso.
Ejecutar secuencia de traspasos de gastos: este proceso es el encargado de ejecutar la(s) secuencia(s) de traspasos que haya(n) sido configurada(s).
Cierre de ejercicio: este proceso es el responsable de, una vez que cierran todos los períodos, hacer una serie de validaciones que permiten cerrar el ejercicio, dejando todas las subcuentas en cero y cargando el saldo a la cuenta de proceso.
Cierre de período: este proceso es el encargado de, una vez que termina el período, realizar el cierre del mismo a través de una serie de validaciones.
Inicializar cuentas de procesos al cierre de ejercicio: este proceso es el responsable de inicializar las cuentas de procesos para cerrar las cuentas del cierre de ejercicio.
Definir los centros de costos: este proceso es el encargado de definir los centros de costos, realizar el registro correctamente de los centros de costos y mostrar en un documento los centros de costos.
Definir los elementos de gastos: este proceso es el responsable de definir los elementos de gastos, realizar el registro correctamente de los elementos de gastos y mostrar en un documento los elementos de gastos.
En ambos subsistemas fueron analizadas además la descripción de los flujos básicos de los procesos, la descripción de sus flujos paralelos y la descripción de sus extensiones, a continuación se muestran los mapas de procesos correspondientes a los subsistemas Contabilidad y Costos y Procesos respectivamente. (Ver figura 10 y figura 11).

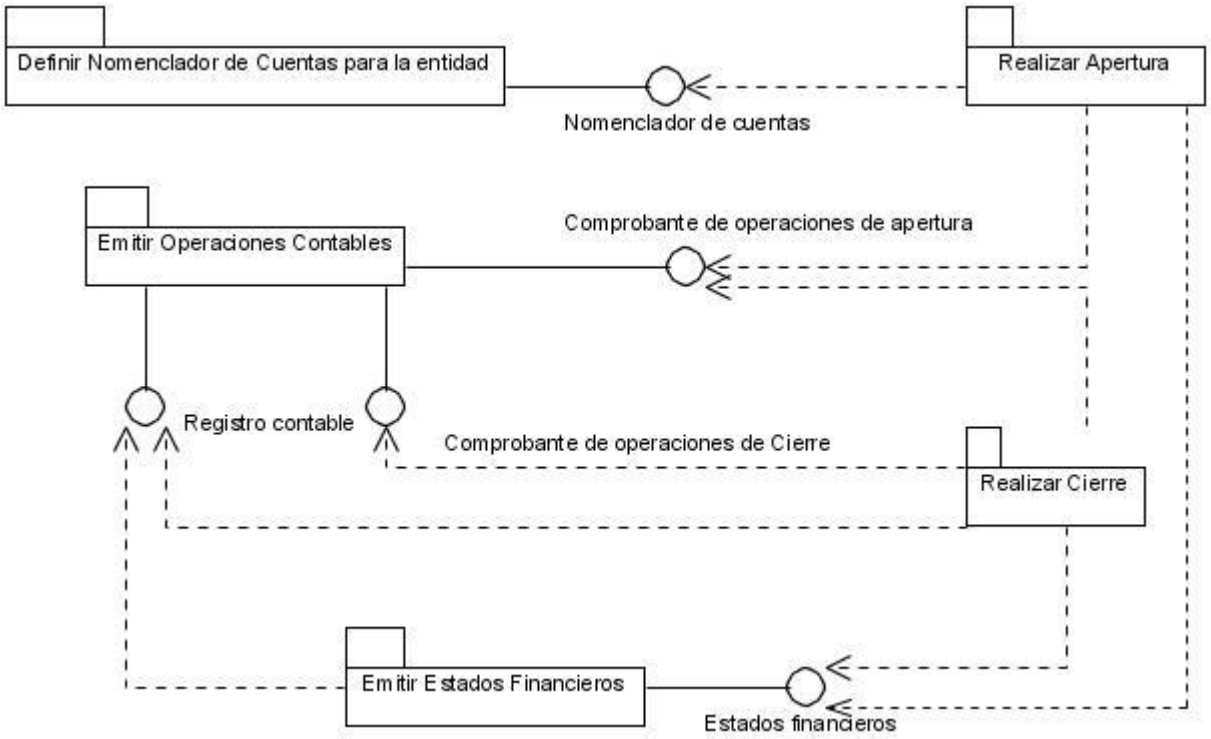


Figura 10: Mapa de procesos Subsistema Contabilidad

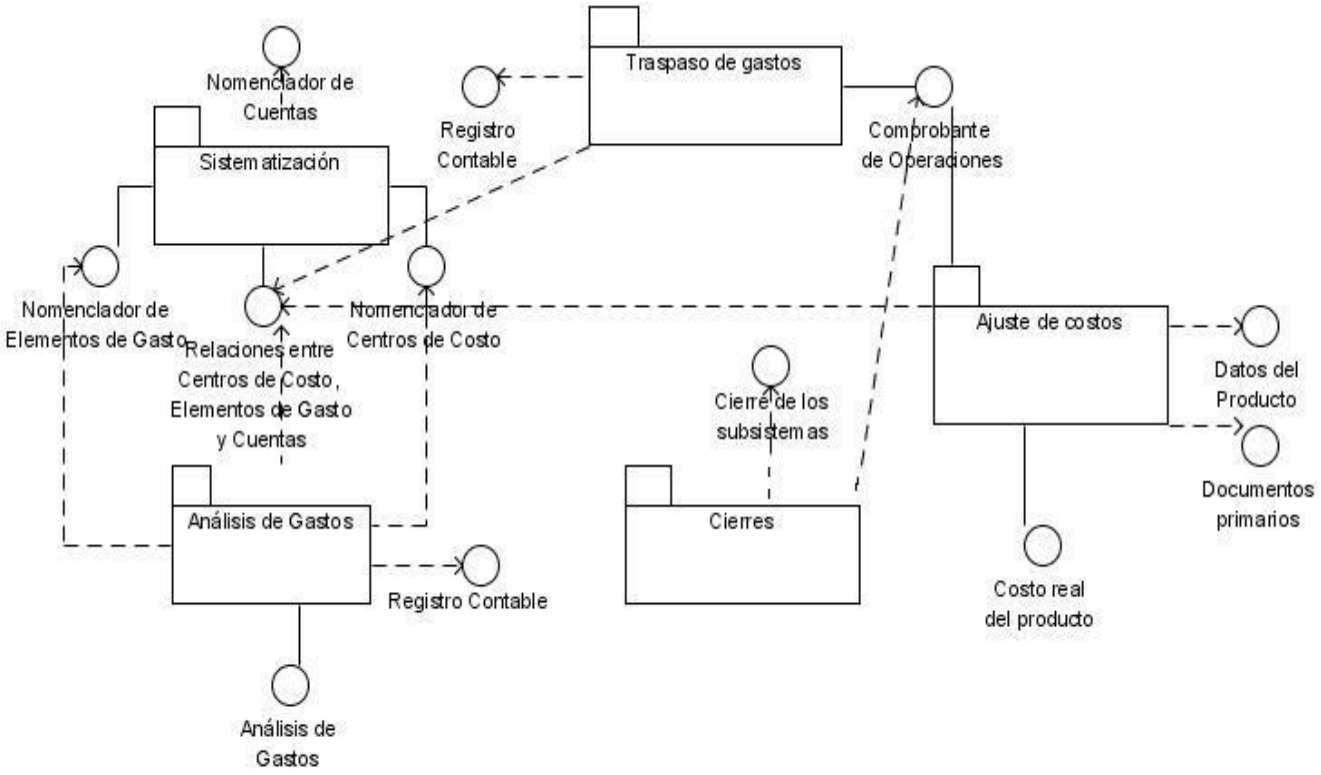


Figura 11: Mapa de procesos Subsistema Costos y Procesos

De manera general el estudio de los mapas de procesos permite conocer con exactitud lo que ocurre realmente en el negocio.

Análisis de los requisitos funcionales

Después del análisis de los procesos se realizó un estudio de los requisitos, y la forma en que se agrupan estos según sus funcionalidades, a continuación se muestra un ejemplo de las agrupaciones de requisitos y las funcionalidades a realizar en cada uno: Gestionar comprobante de operaciones, Gestionar registros anexos al pase, Gestionar pase y Gestionar comprobante tipo de operaciones. Todos estos se encuentran vinculados al comprobante de operaciones en el subsistema Contabilidad.

Tabla 2: Requisitos funcionales asociados al comprobante de operaciones del subsistema Contabilidad

	Requisitos Funcionales
Comprobante de operaciones	Adicionar comprobante de operaciones.
	Modificar comprobante de operaciones.
	Eliminar comprobante de operaciones.
	Listar comprobante de operaciones.
	Consultar comprobante de operaciones.
	Buscar comprobante de operaciones.
	Revertir comprobante de operaciones.
	Duplicar comprobante de operaciones.
	Cambiar el número del comprobante de operaciones.
	Terminar comprobante de operaciones.
	Asentar comprobante de operaciones.
	Pasar día contable.
	Importar comprobante de operaciones.
	Exportar comprobante de operaciones.
	Imprimir comprobante de operaciones.
	Adicionar registro anexo al pase.
	Modificar registro anexo al pase.
	Eliminar registro anexo al pase.
	Listar registro anexo al pase.
	Adicionar pase.
	Modificar pase.
	Eliminar pase.
	Duplicar pase
	Listar pase.
	Generar comprobante resumen de operaciones.
	Cambiar saldo de cuenta.
	Adicionar comprobante tipo de operaciones.

	Modificar comprobante tipo de operaciones.
	Eliminar comprobante tipo de operaciones.
	Listar comprobante tipo de operaciones.
	Realizar búsqueda avanzada de comprobante tipo de operaciones.
	Revertir comprobante tipo de operaciones.
	Duplicar comprobante tipo de operaciones.
	Cambiar el número del comprobante tipo de operaciones.
	Terminar comprobante tipo de operaciones.
	Asentar comprobante tipo de operaciones.
	Importar comprobante tipo de operaciones.
	Exportar comprobante tipo de operaciones.
	Imprimir comprobante tipo de operaciones.

Después de conocer las actividades a automatizar por medio de los requisitos, la descripción de los procesos y los mapas de procesos, se puede afirmar que estos en conjunto, constituyen la primera aproximación a la estructura de empaquetamiento, la cual posteriormente se convertirá en componentes.

2.3. Descripción de los subsistemas en cuestión

El proyecto ERP Cuba está dividido en varios subsistemas, con el objetivo de facilitar todo tipo de soluciones y cada uno estos subsistemas está estructurado en componentes, seguidamente se presenta una breve descripción de los subsistemas Contabilidad y Costos y Procesos especificando los componentes que los conforman.

Subsistema Contabilidad

La contabilidad tiene el propósito de proporcionar información sobre el desarrollo de la actividad económica que realiza en determinada entidad y es la encargada de desarrollar y comunicar la información de forma tal que sirva de ayuda en la planificación y control de actividades.

La contabilidad permite controlar de forma más abarcadora las principales funciones de la entidad como son las finanzas, la administración, la producción, las ventas y la distribución, etc.

El subsistema Contabilidad es el encargado de llevar el control de todos los movimientos u operaciones contables que se realizan en cualquier entidad, por ejemplo: la apertura de una cuenta y la creación y recepción de comprobantes. Se creó para organizar y gestionar todo lo relacionado con la contabilidad acorde a las normativas de nuestro país con el objetivo de mejorar el control y el seguimiento de las operaciones contables en las empresas cubanas.

El subsistema está dividido en 6 componentes:

- comprobante_operaciones
- nomenclador_cuentas
- configuraciones
- recuperaciones
- consistencias
- cierre

Ver figura 12.

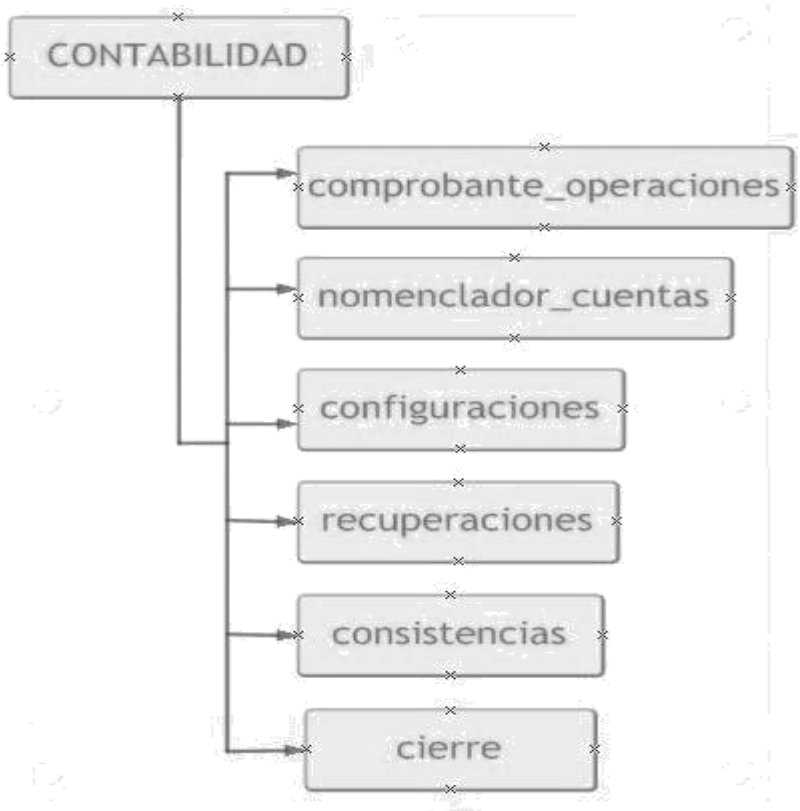


Figura 12: Gráfico estructural de la Arquitectura de Sistema del subsistema Contabilidad

Subsistema de Costos y Procesos

El costo constituye un elemento normativo y evaluador de la gestión de una entidad, de aquí su importancia como herramienta de dirección, por lo que se requiere por parte del personal dirigente el dominio de los aspectos que caracterizan su contenido.

El subsistema Costos y Procesos fue creado con la misión fundamental de gestionar los costos y procesos conforme a las normativas establecidas en Cuba con el propósito de optimizar el control y el seguimiento de los gastos en las empresas cubanas. La tarea esencial del subsistema es llevar el control de todos los movimientos u operaciones de costos que se realicen en cualquier entidad por ejemplo: las operaciones para ajustar los costos de los productos en producción terminada.

El subsistema está dividido en 4 componentes:

- ajustes_costos
- cierres_traspasos
- nom_conf
- cierre

Ver figura 13.

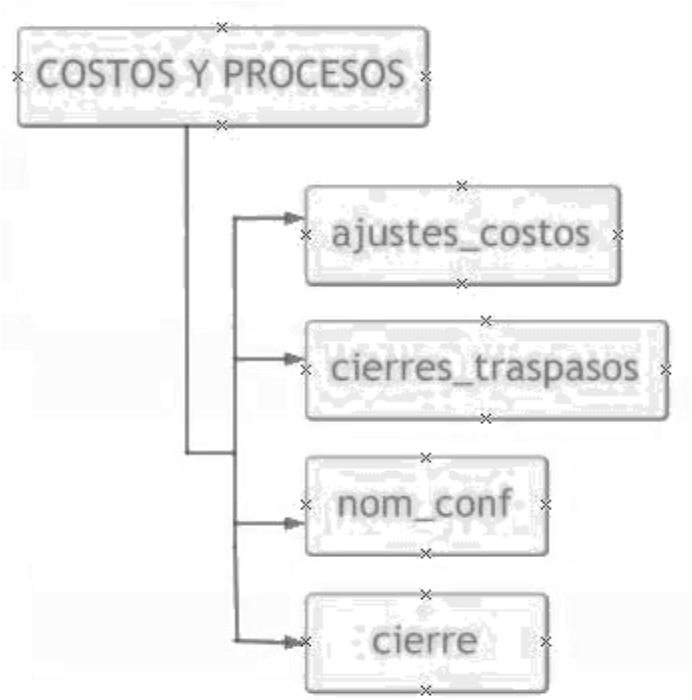


Figura 13: Gráfico estructural de la Arquitectura de Sistema del subsistema Costos y Procesos

En ambos subsistemas los componentes agrupan un conjunto de funcionalidades y requerimientos específicos del cliente, interactúan y comparten datos en dependencia de la funcionalidad de cada uno, por otro lado, para cada componente se definen los paquetes, presentación, controlador y modelo además de un paquete validaciones que almacena las reglas del negocio y un paquete servicios.

Cada subsistema cuenta con un expediente, el cual almacena todos los artefactos que deben ser generados en la vista de sistema. (Ver figura 14)

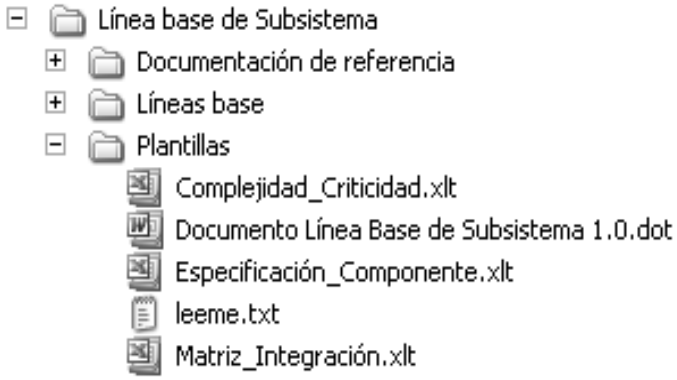


Figura 14: Estructura de los espacios de los subsistemas

Como aparece en la Figura 14 el expediente recoge los artefactos generados en cada subsistema:

- **Documento Línea Base**
- **Complejidad y Criticidad**
- **Especificación de Componentes**
- **Matriz de Integración**

En el capítulo dichos artefactos se generan en el orden lógico a la hora de desarrollar el producto.

2.4. Especificación de componentes

Se listan todos los componentes definidos en el subsistema, y de cada uno de ellos se especificará el nombre del componente, una descripción basada en su objetivo fundamental y las principales características del mismo, así como los servicios que provee con sus detalles.

Objetivo: Tener registrada la formalización de cada componente y la relación de los servicios que brinda.

De forma más detallada el artefacto contiene para cada componente una descripción breve de la responsabilidad arquitectónica de estos, una lista de cada uno de los servicios que brinda con su descripción y atributos de entrada y salida de los servicios. Ver figura 15.

Componentes	Descripción	Servicios que brinda	Código	Descripción	Entrada	Salida
cierres_traspasos	Este componente tiene como función principal realizar el traspaso de los gastos directos entre centros de costos y elementos del gasto, mediante la creación de secuencias de traspaso y su ejecución, generando comprobantes con el resplado de las operaciones realizadas.	Obtenerultimaejecucior	CostosNomConf38	Devuelve el id de la última secuencia de traspasos ejecutada		enterospos
		Obtenercentrosorigen	CostosNomConf39	Devuelve un arreglo con los centros de origen		array
		Obtenercentrosdestino	CostosNomConf40	Devuelve un arreglo con los centros de destino		array
		saldosCuenta	CostosNomConf41	Devuelve el saldo de una cuenta y un centro	idEstructuraComun(enterospos) idcuenta(enterospos) idcentro(enterospos)	ireal
		estadoComprobante	CostosNomConf42	Devuelve true o false en dependencia de si el comprobante emitido fue asentado	iddocumento(enterospos)	bool

Figura 15: Especificación del componente ajustes_costos del subsistema Costos y Procesos

2.4.1. Responsabilidad arquitectónica de los componentes

Subsistema Contabilidad:

- **Nomenclador de cuentas:** Este componente es el responsable de definir el clasificador de cuentas del país, gestiona el mismo, así como las aperturas y las clasificaciones de las cuentas.
- **Configuración:** Este componente es el encargado de configurar los nomencladores de grupos y subgrupos, crear el nomenclador de contenidos económicos y asignarlo a un grupo contable, además crea la estructura económica y le asocia a cada una un criterio económico (centro rector padre).

- **Comprobante de operaciones:** Existen tres tipos de comprobantes, los cuales pueden ser: comprobante tipo, comprobante cuentas íntegras y comprobante cuentas desglosadas. El componente es el encargado de registrar los documentos básicos de la contabilidad por cualquiera de los tres tipos antes mencionados, donde se registran todos los asientos pertenecientes a una acción contable.
- **Recuperaciones:** Este componente como su nombre lo indica es el encargado de las básicas recuperaciones de contabilidad (mayor, submayor y balance de comprobación). En cada uno de los casos antes mencionados permite obtener el saldo de una cuenta, en caso del mayor devuelve el saldo perteneciente a una cuenta de primer nivel (cuenta padre), el submayor por su parte devuelve el saldo de cualquier cuenta a partir del nivel dos (cuenta hija) y el balance devuelve el saldo de todas las cuentas afectadas hasta el momento.
- **Consistencias:** Este componente es el encargado de reconstruir los saldos de cuentas y centros de costo.
- **Cierre:** Este componente es el responsable del cierre de un período contable y un ejercicio económico. Posee dentro de sus precondiciones que todos los comprobantes hayan sido asentados.

Subsistema Costos y Procesos:

- **Nomenclador y configuración:** Este componente es el encargado de realizar toda la configuración que permite definir las cuentas de gastos, cuentas de procesos, los centros de costos/grupos presupuestados, los elementos del gasto/objetos del gasto y las asociaciones que se establecen entre ellos, así como las asociaciones que se definen entre las áreas de responsabilidad y los centros de costos.
- **Ajustes al costo:** Este componente tiene como objetivo fundamental realizar el ajuste de los productos en producción terminada, teniendo en cuenta si dicho ajuste se realiza por inventario, en este componente se definen configuraciones para dichos ajustes y sus ejecuciones y se generan comprobantes para almacenar las afectaciones ya sea por producto o por cuentas.
- **Cierres traspasos:** Este componente tiene como función principal realizar el traspaso de los gastos directos entre centros de costos y elementos del gasto mediante la creación de secuencias de traspasos y su ejecución, generando comprobantes con el respaldo de las operaciones realizadas.

- **Cierre:** Este componente es el responsable del cierre de un período y un ejercicio contable.

Para mayor información, consultar artefacto en el expediente de sistema de ambos subsistemas.

2.5. Matriz de Integración

Contiene todos los componentes definidos en el subsistema, de forma matricial, y en las intercepciones se especifican los servicios que consume el componente.

Objetivo: Tener registradas las relaciones entre componentes mediante servicios y saber mediante cuáles servicios específicos.

Dicho artefacto específicamente contiene la relación de los componentes de ambos subsistemas establecida por los servicios ya sean consumidos por cada componente dentro del propio subsistema o por otro componente fuera del mismo, es decir, tanto interno como externo. De forma vertical aparecen los componentes que consumen servicios y de forma horizontal los componentes que brindan servicios. Ver figura 16 y 17 respectivamente.

La matriz sirve posteriormente para confeccionar el mapa de componentes el cual es la forma de visualizar la relación establecida entre los componentes que consumen y brindan servicios. Dicho mapa se presenta en el epígrafe 2.8.1.

Costos y Procesos	Brindan			
Componentes	ajustes_costos	cierre	nom_conf	cierres_traspasos
cierre	Obtenerultimaejecucion			Obtenerultimaejecucion

Figura 16: Matriz de Integración Interna del componente cierre del subsistema Costos y Procesos

		Brindan			
	Componentes	**Contabilidad**	**Costos Procesos**	**Seguridad**	**Parámetros**
Consumen	**recuperaciones**	ObtenerEstructuraE	obtenerCuentasGastod escripcionCentro descripcionElemento	getUsersProfile	BuscarMonedaConta BuscarMoneda ObtenerPeriodos ObtenerPeriodo ObtenerEjercicio ObtenerEjercicios BuscarTasa

Figura 17: Matriz de Integración Externa del componente recuperaciones del subsistema Contabilidad.

Para una mejor comprensión, ver artefactos en el expediente de sistema de ambos subsistemas.

2.6. Complejidad y Criticidad de los componentes

Contiene todos los componentes definidos en el subsistema, y por cada componente los requisitos funcionales que implementa, la complejidad de esos requisitos, y una serie de datos desde el punto de vista de integración. A partir de la dependencia entre componentes y la prioridad de requisitos se determinan la complejidad de los componentes y la criticidad de los mismos.

Objetivo: Evaluar por cada componente su complejidad así como qué tan importante es desde el punto de vista de la integración, ambos criterios son variables de peso a la hora de definir una prioridad de implementación, así como a la hora de la aceptación o no de un cambio determinado.

De forma más específica este artefacto almacena para cada componente datos como: cantidad de requisitos, nombre de los requisitos, complejidad de cada uno de estos. Es importante destacar que la complejidad para cada requisito es determinada por los analistas de los subsistemas, es un valor numérico definido del 1-10, además se recoge la cantidad de servicios brindados por cada componente, cantidad de componentes dependientes, tipo de componente: Núcleo, Configuración, Integración, Tecnología y Prueba, en ambos subsistemas solo existen componentes de dos tipos Núcleo y Configuración, definidos de la siguiente forma:

Núcleo: Representan aquellos componentes con la responsabilidad de abstraer las principales funcionalidades del negocio de la organización, comúnmente agrupan la mayor parte de los conceptos del modelo conceptual, y en términos arquitectónicos agrupan el deseo funcional del sistema que solicita el cliente.

Configuración: Representan aquellos componentes con la responsabilidad de abstraer las configuraciones y parametrizaciones estáticas o dinámicas del sistema, así como las características de los estándares a utilizar en la conversión de formato y validaciones de procesos del negocio.

Por último se calcula la complejidad y criticidad para cada componente mediante las siguientes fórmulas definidas por el equipo de trabajo de arquitectura:

Complejidad del componente: cantidad de requisitos más (complejidad de los requisitos por cuatro) más (la cantidad de servicios por dos).

Criticidad de integración del componente: cantidad de dependencias más la sumatoria de la complejidad de los componentes dependientes.

Después de un análisis se llegó a las conclusiones siguientes: en el subsistema Costos y Procesos el componente que agrupa mayor cantidad de requisitos funcionales es nom_conf (35). Siendo los requisitos de mayor complejidad: definir cuenta de proceso (10), adicionar cuenta de gastos (10), eliminar cuenta de gastos (10), listar cuenta de gastos (10) y buscar cuenta de gastos (10) y los de menor complejidad: listar asociación entre área de responsabilidad y centro de costos o grupo presupuestario (3), consultar asociación entre área de responsabilidad y centro de costos o grupos presupuestarios (3), imprimir asociaciones de un área de responsabilidad (3), consultar centro de costos o grupo presupuestario (3), buscar centro de costos o grupo presupuestario (3), imprimir centro de costos o grupo presupuestario (3). El componente de mayor complejidad es nom_conf (144) y el de menor es cierre (14). En el caso de la criticidad el de mayor criticidad de integración es cierres_traspasos (160) y de igual forma el de menor criticidad es el componente cierre (0).

Del mismo modo en el subsistema Contabilidad se obtuvieron las siguientes conclusiones: el componente que concentra el mayor número de requisitos funcionales es comprobante_operaciones (39). Los requisitos que poseen mayor complejidad son: adicionar estructura económica, adicionar grupo o subgrupo contable, adicionar contenido económico, adicionar cuenta contable, adicionar apertura del árbol de cuenta, adicionar comprobante de operaciones, adicionar pase, adicionar comprobante tipo de operaciones, cerrar período contable cuando no es mes doce, cerrar período contable cuando es mes doce y cerrar ejercicio económico, todos con un valor igual a diez, los de menor complejidad son: exportar el nomenclador de cuentas y pasar día contable, ambos con un valor igual a dos. Los componentes de menor y mayor complejidad son: consistencias (38) y comprobante_operaciones (127). Por último el

mayor valor de criticidad de integración lo tiene el componente comprobante_operaciones (326) y los componentes de menor valor son recuperaciones, consistencias y cierre, todos con un valor igual a cero. Seguidamente se muestra una imagen con todos los datos recogidos en dicho artefacto, pertenecientes al componente cierre del subsistema Contabilidad.

Ver figura 18.

Componente	Cant. de requisitos	No.	Requistos	Código	Complejidad del Requisito	Servicios que brinda	Componentes dependientes	Tipo de Componente	Complejidad	Criticidad de Integración
cierre	5	104	Cerrar Período Contable cuando no es mes12	Cont_Cierre_Cerrar Período Contable cuando no es mes12	10	2	0	Núcleo	189	0
		105	Cerrar Período Contable cuando es mes12	Cont_Cierre_Cerrar Período Contable cuando es mes12	10					
		106	Cerrar ejercicio económico	Cont_Cierre_Cerrar ejercicio económico	10					
		107	Reapertura de período contable	Cont_Cierre_Reapertura de período contable	6					
		108	Realizar cierre de cuentas	Cont_Cierre_Realizar cierre de cuentas nominales	9					

Figura 18: Complejidad y Criticidad del componente cierre del subsistema Contabilidad

Para conocer los datos del resto de los componentes, consultar artefacto en el expediente de sistema de ambos subsistemas.

2.7. Línea Base

Este documento recoge todos los artefactos antes mencionados, además del modelo de integración de componentes y otras especificaciones.

Objetivo: Delimitar las responsabilidades arquitectónicas del subsistema, establecer la comunicación y dependencias entre los componentes y módulos que lo integran, así como plasmar la criticidad y complejidad de los componentes con vistas a su reutilización y prioridad en el desarrollo.

La Línea Base de ambos subsistemas además de lo antes mencionado contiene la priorización desde el punto de vista arquitectónico de los componentes, el mapa de componentes, la descripción de los patrones y estilos arquitectónicos utilizados y por último el diseño de las clases, donde se lista y describen cada una de las clases controladoras y del modelo de negocio que contiene cada componente.

Priorización de los componentes

La priorización arquitectónica de los componentes consiste en listar los componentes con su priorización respectiva. El valor de priorización equivale al orden de desarrollo que deben tener los componentes (a menor valor mayor prioridad). Para establecer esta prioridad se toma como partida la criticidad calculada para cada componente en el artefacto Complejidad y Criticidad. En las tablas se muestra la priorización de los componentes de los subsistemas de Contabilidad y Costos y Procesos respectivamente.

Tabla 3: Priorización de los componentes subsistema Contabilidad

Componentes	Criticidad	Prioridad
comprobante_operaciones	326	1
configuraciones	279	2
nomenclador_cuentas	262	3
recuperaciones	0	4
consistencia	0	5
cierre	0	6

Tabla 4: Priorización de los componentes subsistema Costos y Procesos

Componentes	Criticidad	Prioridad
cierres_traspasos	160	1
nom_conf	88	2
ajustes_costos	15	3
cierre	0	4

2.7.1. Mapa de Componentes de los subsistemas Contabilidad y Costos y Procesos

El mapa de componentes no es más que una forma de representar de forma visual las dependencias existentes entre los componentes de un subsistema.

El subsistema Contabilidad está dividido en seis componentes los cuales concentran un grupo de funcionalidades específicas para los clientes. Los componentes que conforman dicho subsistema son: nomenclador_cuentas, configuraciones, comprobante_operaciones, recuperaciones, consistencias y cierre.

Dichos componentes están integrados de la siguiente forma. Los componentes cierre, consistencias, recuperaciones, nomenclador_cuentas y configuraciones consumen servicios del componente comprobante_operaciones a través de la interfaz IComprobante, los componentes recuperaciones, nomenclador_cuentas y comprobante_operaciones por medio de la interfaz IConfiguraciones consumen servicios del componente configuraciones, por último, los componentes recuperaciones, configuraciones y comprobante_operaciones consumen servicios del componente nomenclador_cuentas a través de la interfaz ICuenta. Ver figura 19.

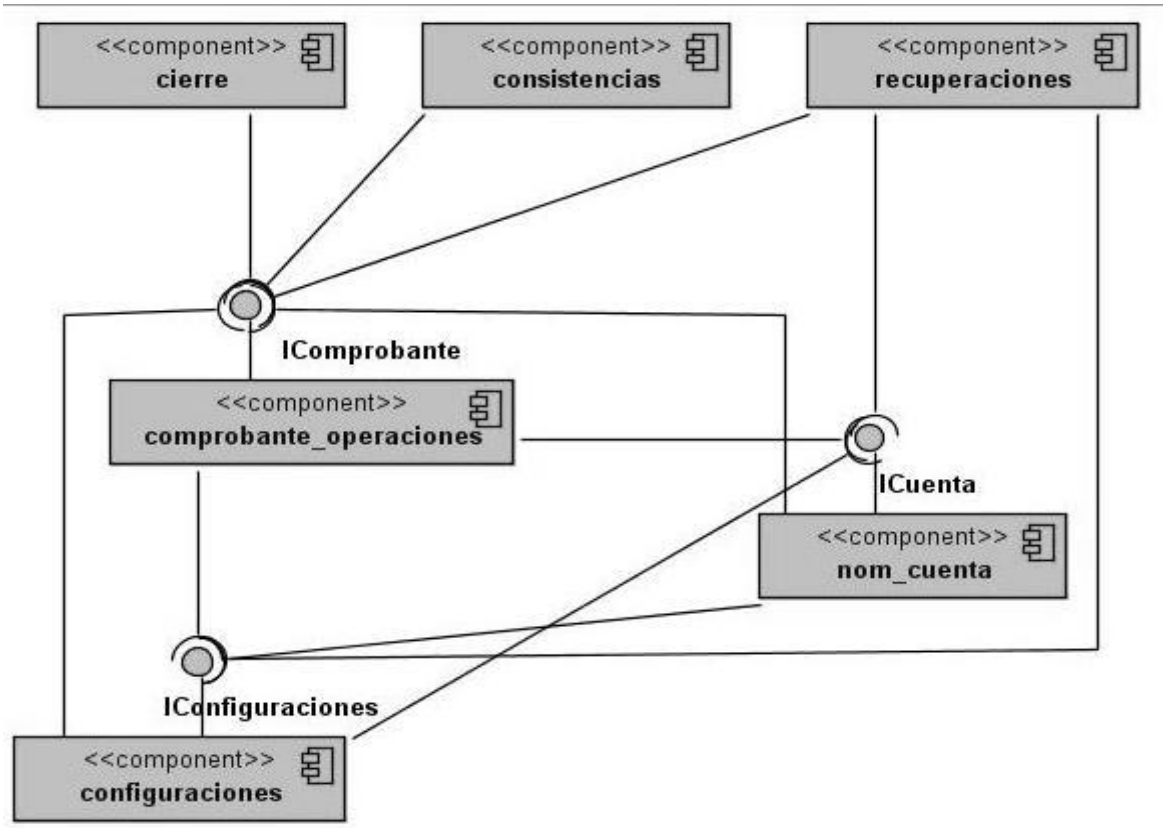


Figura 19: Mapa de componentes del subsistema Contabilidad

Por su parte el subsistema Costos y Procesos de igual forma está dividido en varios componentes los cuales son: ajustes_costos, cierres_traspasos, nom_conf y cierre.

La integración entre dichos componentes está definida de la siguiente forma, los componentes: ajustes_costos y cierres_traspasos consumen servicios del componente nom_conf a través de la interfaz IConfiguraciones, el componente nom_conf consume del componente cierres_traspasos mediante la interfaz ITraspazos y por último el componente cierre consume servicios de los componentes ajustes_costos y cierres_traspasos a través de las interfaces IAjustes y ITraspasos respectivamente. Ver figura 20.

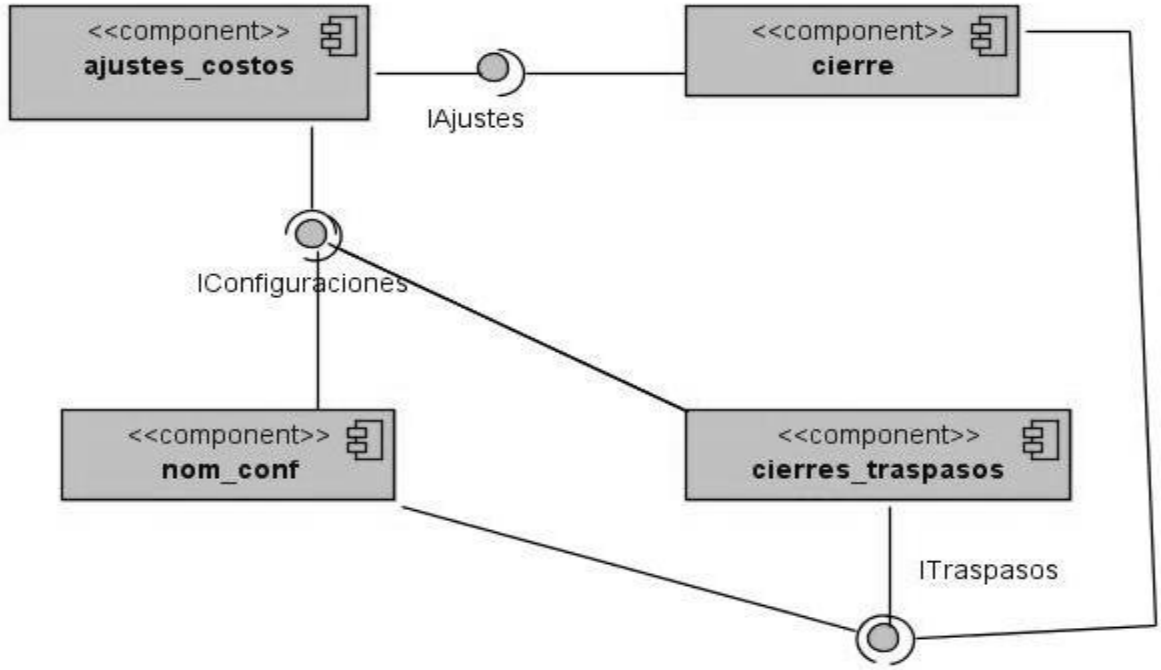


Figura 20: Mapa de componentes del subsistema Costos y Procesos

Estilos y Patrones arquitectónicos aplicados

El empleo de ciertos estilos y patrones arquitectónicos mejora o disminuye las posibilidades de satisfacer los atributos de calidad de un sistema, por tanto, la decisión del uso de alguno de los existentes depende de los requisitos de calidad a alcanzar por el sistema.

En los subsistemas Contabilidad y Costos y Procesos para sus diseños arquitectónicos se emplea el estilo basado en componentes el cual permite definir componentes que dan respuesta a los requisitos, por otro

lado se utilizan principios del estilo orientado a servicios para lograr el intercambio de información entre dichos componentes. Dentro de cada componente se aplica el patrón Modelo-Vista-Controlador (MVC), con el uso del Zend Framework para la implementación de lógica del negocio.

2.7.2. Diseño de clases

Para cumplir con los estilos y patrones definidos para los subsistemas se hizo necesario realizar un diseño de clases que respondiera a los requerimientos que plantea cada patrón.

En ambos subsistemas fueron analizadas las funcionalidades de las clases de diseño, en este caso sólo se analizaron las clases controladoras y las clases del modelo de negocio, a continuación aparecen una serie de tablas que muestran las clases controladoras y del modelo de negocio pertenecientes al componente nom_conf del subsistema Costos y Procesos y al componente comprobante_operaciones correspondiente al subsistema Contabilidad.

Tabla 5: Clases Controladoras componente nom_conf subsistema Costos y Procesos

Clases	Descripción
OpasociacionccgpController	Clase donde se gestionan las asociaciones entre las áreas de responsabilidad y los centros de costos/grupos presupuestados.
OpasociacionController	Clase donde se gestionan las asociaciones entre las cuentas, los centros y los elementos de gastos ya sea para la actividad patrimonial como para la presupuestada.
OpcuentasgastosController	Clase donde se gestionan las cuentas de gastos y de ellas, las de saldo de inicio de año, así como el tipo de contabilización y análisis.
OpnomencladorController	Clase donde se gestionan los nomencladores de centros de costos/grupos presupuestados y elementos del gasto/objetos de gasto.
OprecuperadorController	Clase donde se gestionan los datos de entrada para mostrar los reportes que se generan en el subsistema.

Tabla 6: Clases del Modelo de Negocio componente nom_conf subsistema Costos y Procesos

Clases	Descripción
OpcuentasgastosModel	Clase que permite la inserción, modificación y eliminación de las cuentas de gastos.
OpasociacionModel	Clase que permite la inserción, modificación y eliminación de las asociaciones entre las cuentas, los centros y los elementos del gasto.
OpasociacionareaModel	Clase que permite la inserción, modificación y eliminación de las asociaciones entre las áreas de responsabilidad y los centros de costos.

Tabla 7: Clases Controladoras componente comprobante_operaciones subsistema Contabilidad

Clases	Descripción
MostrarcomprobanteoperacionesController	Clase que permite controlar los datos a mostrar del comprobante de operaciones.
ComprobantetController	Clase que controla los casos de uso relacionados con el comprobante de operaciones de cuentas tipo.
ComprobanteiController	Clase que controla los casos de uso relacionados con el comprobante de operaciones de cuentas íntegras.
ComprobantedController	Clase que controla los casos de uso relacionados con el comprobante de operaciones de cuentas desglosadas.
ComprobanteController	Clase que controla los casos de uso relacionados con el comprobante de operaciones.
CambiarsaldocuentaController	Clase que permite realizar el cambio de saldo entre 2 cuentas.

VerificarsaldonaturalezaController	Clase controladora encargada de verificar la naturaleza de las cuentas con saldo inverso a su naturaleza.
RegistrocomprobantesController	Clase que controla la gestión de los cambios de los comprobantes y actualiza el historial.
PrinterController	Clase controladora encargada de la impresión de los reportes.

Tabla 8: Clases del Modelo de Negocio componente comprobante_operaciones subsistema Contabilidad

Clases	Descripción
ResCentrocuentaElementoPeriodoModel	Clase donde se inserta, actualiza y elimina un centro, una cuenta, un elemento y un período.
ResBloqueoComprobanteModel	Clase donde se boquea o se desbloquea un comprobante.
DatRegistroanexoModel	Clase donde se insertan, actualizan y registran los registros anexos de un comprobante.
DatHistorialCmpModel	Clase que gestiona los casos de uso relacionados con el historial de los comprobantes.
VerificarSaldoNaturalezaModel	Clase que verifica si el saldo de la cuenta es inverso a su naturaleza.
DatPaseModel	Clase donde se insertan, actualizan y registran los pases.
DatComprobanteModel	Clase que gestiona los casos de uso relacionados con el comprobante (insertar, modificar, eliminar y guardar un comprobante).

DatAsientoModel	Clase donde se insertan y se modifican los asientos.
CierrePeriodoModel	Clase donde se cierra un período contable.
CierreEjercicioModel	Clase donde se cierra un ejercicio contable.
CambiarSaldoCtaModel	Clase donde se cambia el saldo de una cuenta.

Para realizar una consulta más detallada, ver artefacto en el expediente de sistema de ambos subsistemas.

2.8. Conclusiones

En el presente capítulo se realizó la vista de la arquitectura de sistema correspondiente a los subsistemas Contabilidad y Costos y Procesos, establecida por los siguientes artefactos: complejidad y criticidad, especificación de componentes, matriz de integración y el documento Línea Base. Es importante mencionar que mediante el análisis de los artefactos del negocio fue identificada la primera aproximación a la estructura de empaquetamiento, por otro lado, al realizar los artefactos correspondientes a dicha vista, es posible optimizar los servicios brindados, para evitar sus repeticiones innecesarias, además al contar con los servicios conjuntamente con sus parámetros de entrada y salida específicos se hace más fácil identificar las relaciones que se establecen entre componentes de forma interna y externa. De forma general todo lo obtenido en dicho capítulo permite conocer y delimitar las responsabilidades arquitectónicas de ambos subsistemas.

CAPÍTULO 3: VISTA DE LA ARQUITECTURA DE DATOS.

3.1. Introducción

La vista de datos no es más que la representación global de la arquitectura de datos, encargada de tareas como la definición, construcción y actualización constante del expediente que recoge todos los artefactos generados en el flujo de trabajo de datos. El presente capítulo se centra en la arquitectura de datos de los subsistemas Contabilidad y Costos y Procesos y recoge los principales conceptos dentro del dominio del problema propuesto. Se definen cuatro artefactos para cada subsistema los cuales son: el modelo de datos, el diccionario de datos, la matriz de trazabilidad y las pruebas de conceptos. Cada uno de los artefactos antes mencionados cuenta con una explicación detallada que facilita su comprensión.

3.2. Análisis de los artefactos que dan entrada a la Vista de Datos de los subsistemas

Para poder realizar un correcto diseño de una Base de Datos es necesario identificar primeramente cuáles son los conceptos fundamentales del negocio, cuáles de estos conceptos son necesarios almacenar en la base de datos, y específicamente cuáles de sus atributos son los necesarios para responder a los requisitos. Para lograr esto se hizo necesario realizar un modelo de conceptos o entidades, ya que dichas entidades posteriormente serán representadas físicamente a través de tablas con sus atributos en el modelo de datos físico de la base de datos. Ejemplo de esto en el subsistema de Contabilidad lo constituyen las definiciones de grupo, cuenta y naturaleza, ver figura 21, de la misma forma sucede con el subsistema de Costos y Procesos, ver anexos 1 y 2 respectivamente.

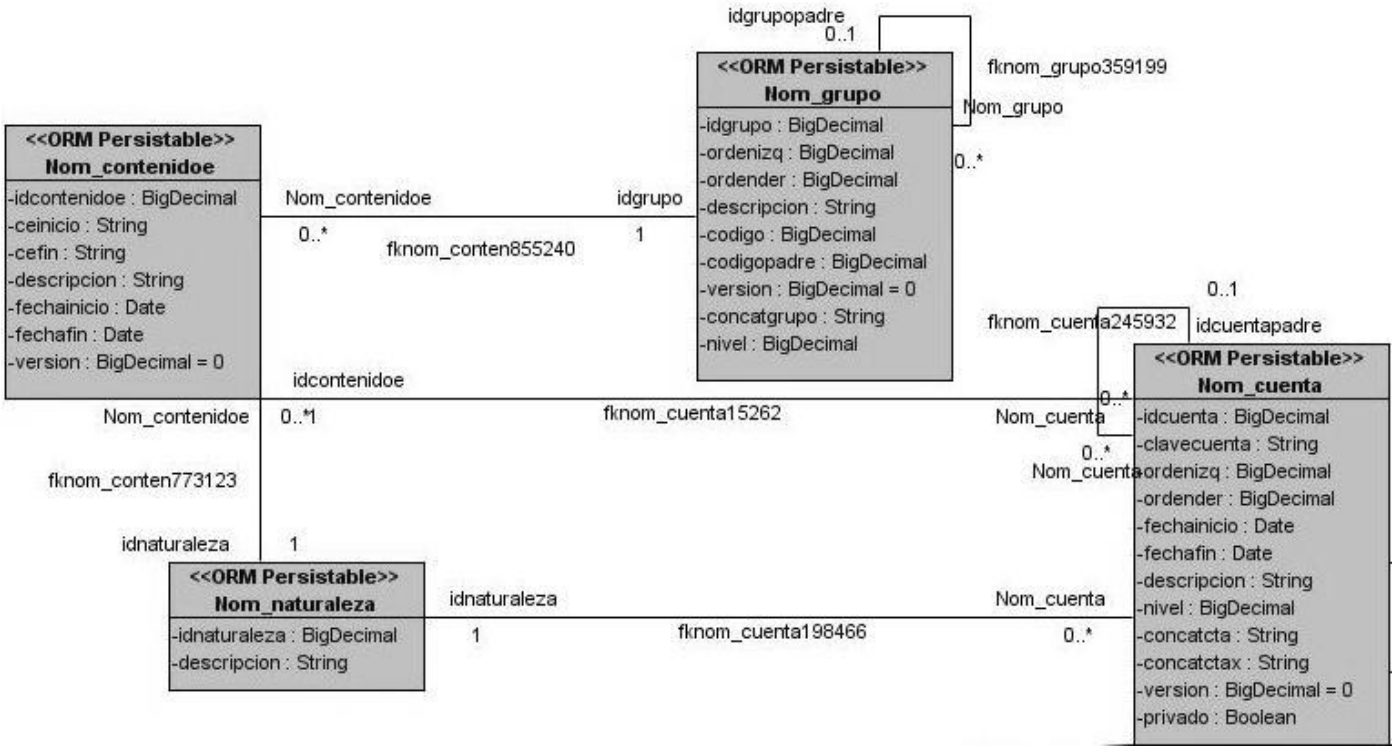


Figura 21: Relación de los conceptos contenido económico, grupo, cuenta y naturaleza, del subsistema Contabilidad

Integración entre la vista de sistema y la vista de datos de los subsistemas Contabilidad y Costos y Procesos

Tomando como referencia lo obtenido en la vista de sistema, para una mejor compresión de la integración entre la vista de sistema y datos de los subsistemas Contabilidad y Costos y Procesos es necesario explicar que después de contar con los requisitos funcionales agrupados para implementar los componentes y los conceptos asociados a los requisitos, se establecen dominios, los cuales corresponden a las tablas que aparecen en ambos modelos, es decir, cada tabla corresponde a un grupo de requisitos funcionales y a su vez al componente que los agrupa. Seguidamente se muestra la relación de las tablas correspondientes a cada componente en ambos subsistemas.

Tabla 9: Relación de tablas por cada componente (subsistema Contabilidad)

Componente	Tablas asociadas
	dat_historialcmp
	dat_comprobante
	dat_asiento

comprobante_operaciones	dat_pase
	dat_registroanexo
	dat_sumacontrolregistroanexo
	nom_tipocomprobante
	nom_estadocomprobante
	res_centrocuenta_elemento_periodo
	res_centrocuenta_elemento_periodo_terminado
	res_bloqueocomprobante
	res_resumenpase
	res_resumenpaseterminado
	res_reglapase
nomenclador_cuentas	conf_apertura
	dat_refapertura
	dat_confcuentaentidade
	dat_subsistemacta
	his_cuenta
	nom_cuenta
	nom_tipocuenta
configuraciones	conf_cuentaredondeo
	dat_estructurae
	dat_concepto
	dat_grupoconcepto
	dat_grupooperadores
	dat_desglose
	nom_grupo
	nom_contenidoe
	nom_naturaleza
	nom_criterioe
	nom_estadof

Tabla 10: Relación de tablas por cada componente (subsistema Costos y Procesos)

Componente	Tablas asociadas
ajustes_costos	conf_ajustescostos conf_ajustecalcajuste conf_ajusteproducto conf_calcularajuste conf_calcajustenoinv conf_calcajusteinv conf_docinventario conf_operacionesinv conf_destino dat_saldosantes dat_ajusteexistenciaantes dat_ajustedestinoantes dat_saldosdespues nom_baseditrib
nom_conf	conf_areascentrospres conf_areascentrospat conf_cuentasgastos conf_cuentascentrosegpres conf_cuentascentrospresupuestarios conf_cuentascentrospatrimonial conf_cuentascentrosegpat
cierres_traspasos	conf_baseejecucion conf_secuenciasdistribucion conf_criteriobusqueda conf_centrosorigen conf_centrosdestino conf_elementos conf_unidades conf_criteriodistribucion dat_ejecucionsecuencia

Expediente de datos de los subsistemas

Es importante aclarar que cada subsistema cuenta con un expediente de datos donde se almacenan los artefactos generados en la vista de datos. El mismo está estructurado como se muestra en la figura 22.

Figura 22: Estructura de los espacios para los subsistemas

Los artefactos almacenados en el expediente son:

- **Modelo de Datos**
- **Diccionario de Datos**
- **Matriz de Trazabilidad**
- **Pruebas de Conceptos**

3.2. Modelo de Datos

Subsistema Contabilidad

El modelo de datos del subsistema Contabilidad está conformado por un total de 32 tablas. Cuenta con: dos tablas que almacenan parámetros de configuración del sistema identificadas por el prefijo conf_ por ejemplo conf_apertura, catorce tablas que almacenan la mayor cantidad de características de una entidad (objeto) identificadas por el prefijo dat_ por ejemplo dat_pase, una tabla para almacenar datos por largos períodos de tiempo que sólo son utilizados para análisis esporádicos (Historiales) identificada por el prefijo his_ por ejemplo his_cuenta, nueve tablas nomencladores identificadas por el prefijo nom_ por ejemplo nom_grupo, y seis tablas resúmenes la cuales resumen la información almacenada en las tablas

originales, identificadas por el prefijo res_ por ejemplo res_resumen_pase. Dentro del modelo una de las tablas más significativa es el nomenclador cuenta. Esta tabla guarda las cuentas de acuerdo a una codificación definida. Contiene los siguientes campos: idcuenta el cual es la llave primaria de la tabla y es un número generado por una función definida; clavecuenta es la definición numérica de la cuenta y debe estar dentro del rango del Contenido Económico al cual estará asociada; fechainicio y fechafin las cuales son las fechas en que se iniciará y finalizará la vigencia de una cuenta respectivamente; descripción es el nombre que identifica la cuenta; idcontenidoe, idnaturaleza, idestructurae, idtipocuenta son llaves primarias de las siguientes tablas nom_contenidoe, nom_naturaleza, dat_estructurae, nom_tipocuenta respectivamente y al igual que el idcuenta su valor se crea a partir de un número generado por una función; el atributo privado es para saber cuando la cuenta es privada='1' solo se usa en esa entidad o pública='0' la pueden usar las entidades que se subordinen a la que creó la cuenta, además cuenta con un campo versión que se utiliza para llevar el control de la concurrencia y atributos que respaldan el comportamiento de esta como una estructura arbólica, ambos aspectos serán explicados en el epígrafe 3.4.1.

Por otro lado el modelo registra todas las funciones y triggers definidos en el subsistema, para un total de 41 funciones y 18 triggers, seguidamente se muestra un ejemplo de cada caso:

Función**: f_anexo_costoventauee**(codigo : varchar, moneda : varchar, periodo : varchar, fechadesde : date, fechahasta : date, centro : varchar, nivelelemento : numeric, codigoelto : varchar, ejercicio : varchar)

Esta función es la responsable de mostrar todos los elementos que están en el nivel escojido y además los que están en el próximo nivel a él, en un centro de costo patrimonial. Recibe como parámetros de entrada (filtros) los siguientes campos:

código: El código de la entidad Ej.: 01, 04-05,01-AA.

moneda: Código ISO de la moneda Ej.: CUC, CUP.

período: Nombre del período en caso que se quiera un período y no un rango de fechas Ej.: Enero, Febrero, Diciembre.

fechadesde y fechahasta: En caso que se quiera escoger un rango de fechas y no un período Ej.:01/12/2008, 31/12/2008.

centro: Código del centro de costo patrimonial Ej.:17, 01-02, 01-AA.

nivelelemento: Nivel del cual se quieran los elementos en caso que se quiera el nivel y no un elemento en específico Ej.: 1, 2.

codigoelto: Código del elemento en caso que se quiera el elemento y no un nivel Ej.: 30, 17, 01-02, 01-AA.

ejercicio: Nombre del ejercicio en el que se encuentra el período en caso que se quiera un período y no un rango de fechas Ej.: 2009, 2010.

Trigger: **t_resumenterminados**

Este trigger tiene como función crear un disparador después de modificar en la tabla dat_comprobante ejecutando la función ft_resumenpaseterminados().

Subsistema Costos y Procesos

El modelo de datos del subsistema Costos y Procesos (ver Anexo 4) está conformado por un total de 30 tablas. Cuenta con: veinticuatro tablas que almacenan parámetros de configuración del sistema identificadas por el prefijo conf_ por ejemplo conf_ajustecosto, cinco tablas que almacenan la mayor cantidad de características de una entidad (objeto) identificadas por el prefijo dat_ por ejemplo dat_ saldosantes y una tabla nomenclador identificada por el prefijo nom_ por ejemplo nom_ basedistrib y tablas resúmenes que se encuentran en el subsistema Contabilidad debido a que estos datos pertenecen a él y son usados por Costos. Una de las tablas de suma importancia con que cuenta el modelo es conf_cuentasgastos, esta guarda las cuentas que son de gastos. Dicha tabla se relaciona con el nomenclador de cuentas de Contabilidad. Se especifica en ella el tipo de cuenta de gasto (patrimonial o presupuestado) y además se especifica si es una cuenta con saldo inicio, esto implica que su padre en el primer nivel sea una cuenta de procesos, que es a fin de cuentas una cuenta patrimonial. Señalar que estas cuentas pueden ser analizadas por elementos de gastos (patrimoniales o presupuestados) o por centros de costos (patrimoniales o presupuestados). Contiene los siguientes campos: idnomcuenta el cual es la llave primaria y foránea de la tabla; espresupuestaria indica si la cuenta de gasto es presupuestada o patrimonial; essaldoinicio indica si la cuenta de gasto tiene saldo de inicio; esgastossubelementos indica si la cuenta será analizada posteriormente por elementos de gastos o por centros de costos, cuenta también con un campo versión que se utiliza para llevar el control de la concurrencia, aspecto que será explicado en el epígrafe 3.4.1.

De igual forma que en el modelo explicado anteriormente, el esquema de Costos y Procesos cuenta con un total de 11 funciones y 8 triggers, seguidamente se muestra un ejemplo en cada caso:

Función: **f_ctas_gastosue**(codigo : varchar, tipocont : varchar)

Esta función es la encargada de mostrar las cuentas de gastos y su tipo de contabilidad por entidad, recibe como parámetros de entrada (filtros), los siguientes campos:

código: El código de la entidad Ej.: 01, 04-05,01-AA.

tipocont: El tipo de contabilidad, en caso que se quiera especificar Ej.: Presupuestada, Patrimonial.

Trigger: **t_conf_cuentasgastos**

La función de este trigger es crear un disparador antes de modificar en la tabla conf_cuentasgastos ejecutando la función ft_chequear().

Para un mejor entendimiento, ver artefacto en el expediente de datos de ambos subsistemas.

3.2.1. Especificaciones de los modelos de datos de los subsistemas

Para la realización de ambos modelos se tuvieron en cuenta diferentes aspectos de suma importancia, a los cuales se les hará referencia a continuación:

Pautas de Optimización

En el subsistema Contabilidad fueron empleadas tablas resúmenes, encargadas de almacenar un conjunto de datos para realizar sobre estas las recuperaciones (reportes), evitando la sobrecarga a las tablas originales, por ejemplo res_resumen_paseterminado recoge los datos de los pases terminados de una cuenta por cada entidad. Por otro lado en el subsistema Costos y Procesos también se llevó a cabo esta práctica con la particularidad que no aparecen en el mismo modelo sino en el modelo de Contabilidad debido a que los datos necesarios con los que opera el subsistema de Costos pertenecen a Contabilidad, una de estas tablas es res_centrocuenta_elemento_periodo esta contiene un resumen de los centros, cuentas y elementos. Se empleó también en ambos modelos el indexado. Un índice es una forma común para un mejor rendimiento, pues permite al servidor de base de datos encontrar y recuperar una fila en específico de manera más rápida que si no existiera, en Contabilidad una de las tablas que cuenta con indexado es res_resumen_pase, esta cuenta con un índice compuesto el cual está conformado por los siguientes campos: idejercicio, idperiodo, idestructurae, idcuenta y fecha. En Costos ocurre de forma

similar la tabla res_centrocuenta_elemento_periodo también posee un índice compuesto conformado por los siguientes campos: idestructuracomun, idperiodo, idmoneda y fechaemision.

La desnormalización también es otra de las prácticas utilizadas para optimizar el rendimiento. Al desnormalizar se pueden combinar relaciones, introducir grupos repetidos y particionar las tablas, en el subsistema de Costos las siguientes tablas se encuentran desnormalizadas conf_cuentascentrospatrimonial y conf_cuentascentrospresupuestarios, es decir, estas dos tablas pudieran estar en una sola debido a que presentan prácticamente los mismos campos pero sería una búsqueda muy engorrosa a la hora de que sólo se necesite conocer los datos de un centro patrimonial, o viceversa.

Árboles

El empleo de estructuras arbólicas es una de las vías empleadas para la mejora del rendimiento. En ambos modelos existen tablas contenedoras de datos jerárquicos lo que hace que se comporten como un árbol, es decir, cada nodo tiene un sólo padre, puede tener varios o ningún hijo (exceptuando al nodo raíz, el cual no tiene padre, pues es padre de sí mismo). El modelado de una estructura arbólica en la base de datos, permite el conocimiento de todos los hijos pertenecientes a un determinado padre y viceversa, para esto se cuenta con cuatro atributos: Id, Id_padre, ordenizq, ordender, los cuales determinan:

- Id: Id correspondiente a la estructura.
- Id_padre: Id correspondiente a la estructura padre.
- Ordenizq: valor izquierdo.
- Ordender: valor derecho.

Para determinar los valores izquierdos y derechos se comienza moviéndose al lado extremo izquierdo del nodo exterior y se continúa hacia la derecha, realizando un recorrido en pre orden. Cuando se trabaja con el árbol, se hace de izquierda a derecha, sólo una capa cada vez, bajando a los hijos de cada nodo, antes de asignar el número de la derecha.

La actualización de los campos ordenizq y ordender se realiza a nivel de trigger, para eso se crearon funciones disparadoras para cada tabla que represente una estructura de este tipo, es decir, cada tabla tiene una función insertar, eliminar y modificar, además la función modificar necesita otra función reordenar la cual reordena el árbol después de ser modificado.

En el subsistema Contabilidad una de las tablas que se comporta de esta forma es **nom_cuenta**. Costos cuenta también con tablas que se comportan de la misma forma con la particularidad que no aparecen en

el esquema del propio subsistema sino que se encuentran en un esquema denominado Nomencladores creado por el equipo de arquitectura, el cual permite que dichos nomencladores sean dinámicos. Para una mejor comprensión se muestra a continuación una imagen del modelo según el Visual Paradigm de una tabla arbólica (Figura 23).

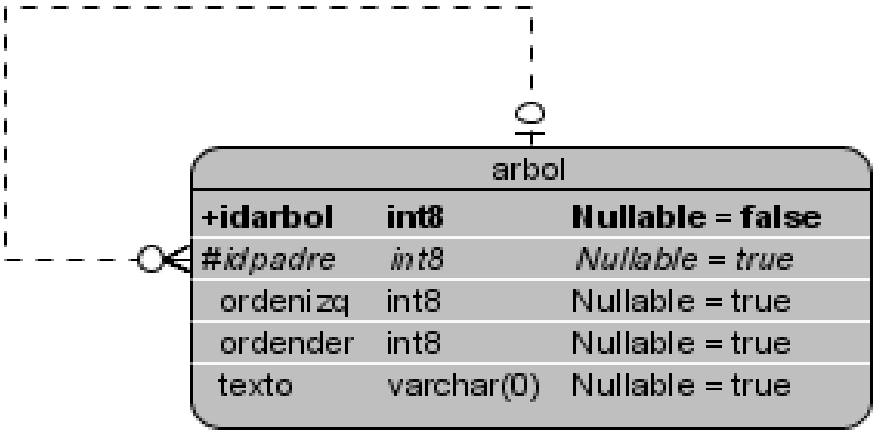


Figura 23: Modelo de un árbol con el Visual Paradigm

Concurrencia

La concurrencia no es más que cuando se accede a modificar algo por una o más personas al mismo tiempo, por tanto, el control de esta es otra de las soluciones presentes en el modelo. Dicho control evita que una misma tupla pueda ser accedida por uno o varios clientes al unísono con el propósito de realizar una acción determinada, en especial a la hora de hacer una modificación (Ver figura 24), esto a la vez evita la pérdida de datos o información, mantiene la integridad de los datos y no permite que un usuario sobrescriba la información de otro, siendo esta última la más actualizada.

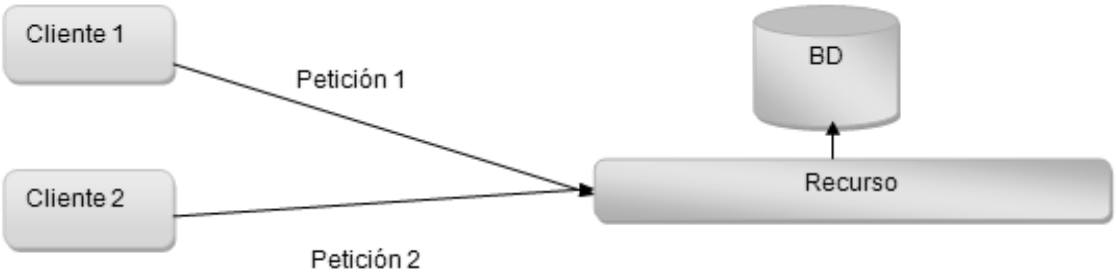


Figura 24: Ejemplo de concurrencia

Para evitar estas situaciones se le agrega a la tabla un campo versión con un límite X, el cual informa las veces que la tupla es modificada, algunas de las tablas que cuentan con esto son: dat_comprobante en

Contabilidad y conf_cuentasgastos en Costos. Dicha actualización es controlada mediante un trigger que verifica que la nueva versión que pone el cliente es igual a la antigua versión, si esto ocurre, entonces en este campo se incrementará en uno y permitirá modificar la tupla accedida.

3.3. Diccionario de Datos

El diccionario de datos en ambos subsistemas cuenta con una descripción detalla de cada una de tablas pertenecientes al modelo de datos. Cada tabla posee atributos los cuales cuentan también con una explicación propia que especifica si ese campo es llave primaria o llave foránea. Se encuentran además las relaciones entre tablas, especificando la cardinalidad de las mismas. Este artefacto incluye también la descripción de las funciones y los triggers que aparecen en cada subsistema, explicando de cada uno la función que cumple. Esta detallada descripción de cada uno de los elementos que conforman el modelo de datos proporciona un fácil entendimiento de la base de datos, tanto para el equipo de desarrollo como para un usuario, en caso de que desee reutilizar las soluciones.

Objetivo: El objetivo del diccionario de datos es proporcionar un registro organizado de todos los datos que pertenecen a los subsistemas Contabilidad y Costos y Procesos, evitar interpretaciones incorrectas y tergiversaciones de dichos datos.

Seguidamente se muestra la tabla nom_naturaleza perteneciente al subsistema Contabilidad.

nom_naturaleza

Tabla 11: Datos de la tabla nom_naturaleza perteneciente al subsistema Contabilidad

Nombre	Valor
Documentación	En la tabla nomenclador de naturaleza se encuentran los distintos tipos de naturalezas que pueden tener las cuentas y los contenidos económicos.
Esquema	mod_contabilidad
Llave Primaria	nom_naturaleza_pkey

Columnas

Nombre	Tipo de Dato	Limitaciones	Anulable	Documentación
idnaturaleza	numeric(19)	PK	No	Este campo es la llave primaria del nomenclador de naturaleza de las cuentas, el mismo es un número consecutivo generado a

				partir de una función definida por el equipo de arquitectura del ERP cubano.
descrpción	varchar(50)		No	En este campo se plasman como tal, las naturalezas que pueden tener las cuentas y los contenidos económicos, las mismas son: acreedora, deudora y mixta.

Relaciones

fknom_conten773123 : Relación	
A	nom_contenidoe
Documentación	Relación de 1 a muchos entre las tablas nom_naturaleza y nom_contenidoe. Llave primaria (nom_naturaleza): idnaturaleza. Llave foránea (nom_contenidoe):idnaturaleza. Permite conocer los tipos de naturalezas que puede tener un contenido económico.
Identifying	false
To Multiplicity	1..*
From Multiplicity	1
Sync To Association	0

fknom_cuenta198466 : Relación	
A	nom_cuenta
Documentación	Relación de 1 a muchos entre las tablas nom_naturaleza y nom_cuenta, Llave primaria (nom_naturaleza): idnaturaleza. Llave foránea (nom_cuenta):idnaturaleza. Permite conocer los tipos de naturalezas que puede tener una cuenta.
Identifying	false
To Multiplicity	1..*
From Multiplicity	1
Sync To Association	0

Para conocer la descripción del resto de las tablas, con sus campos y relaciones consultar el artefacto en el expediente de datos de ambos subsistemas.

3.4. Matriz de Trazabilidad

La matriz de trazabilidad como sugiere su nombre, es una traza para conocer la relación de los subsistemas Contabilidad y Costos y Procesos con el resto de los subsistemas existentes, así como el estado de integración que existe entre ellos. Se especifican además datos como: tabla de origen, tabla destino y la variable común. Para la confección de este artefacto fueron analizadas todas las tablas que conforman los esquemas de Contabilidad y Costos y Procesos.

Objetivo: La matriz de trazabilidad tiene entre sus objetivos fundamentales listar las relaciones que se establecen entre las tablas para agilizar el desarrollo y lograr identificar la integridad referencial que se establece entre las mismas, para de esta forma evitar pérdidas de información que puedan ocurrir en el futuro. Por otro lado por la agilidad que se le imprime al proceso de desarrollo de software, en ocasiones la actividad de integración es la última que se lleva a cabo entre los subsistemas con el fin de que cada uno finalice la programación, de ahí que no se establezcan las relaciones entre dichas tablas hasta culminar su tiempo de desarrollo y este artefacto posea vital importancia en el correcto funcionamiento de la base de datos.

Para mayor información ver el artefacto en el expediente de datos de ambos subsistemas.

3.5. Pruebas de Conceptos

El artefacto pruebas de conceptos almacena la descripción de los escenarios de pruebas en ambos subsistemas, visualiza las tablas, atributos y relaciones que conforman un escenario, además contiene los resultados específicos obtenidos al realizar cada prueba. Mediante estas pruebas se trata de lograr un mejor rendimiento y seguridad de los datos, dentro del proceso de desarrollo de software.

Objetivo: Las pruebas de conceptos tienen dentro de sus objetivos principales la descripción detallada de todas las pruebas que se realicen a nivel de diseño, así como validar el negocio implementado en la base de datos a través de funciones. Todo lo anteriormente expuesto permite obtener la validación de las decisiones que se toman para el diseño o el desarrollo de una base de datos de gestión.

En ambos subsistemas se realizaron las pruebas para los requisitos que corresponden a las funciones recogidas en la base de datos, los cuales de manera general son requisitos de reportes, por ejemplo el requisito de reporte **Imprimir comprobante de operaciones** (perteneciente al subsistema Contabilidad) se soluciona por medio de la función **f_imprimircomprobante (id numeric)**. Esta función es la encargada de devolver los datos necesarios que se muestran en el reporte del comprobante de operaciones.

Para realizar la prueba a dicha función se insertaron en las tablas a consultar (ver figura 25) un total de cinco mil tuplas de datos, para ejecutar luego la siguiente consulta:

EXPLAIN ANALYZE select * from mod_contabilidad.f_imprimircomprobante(4545422)

Obteniendo los resultados siguientes:

Result (cost=0.00..260.00 rows=1000 width=944) (actual time=23.984..23.984 rows=0 loops=1)

Total runtime: 24.047 ms.

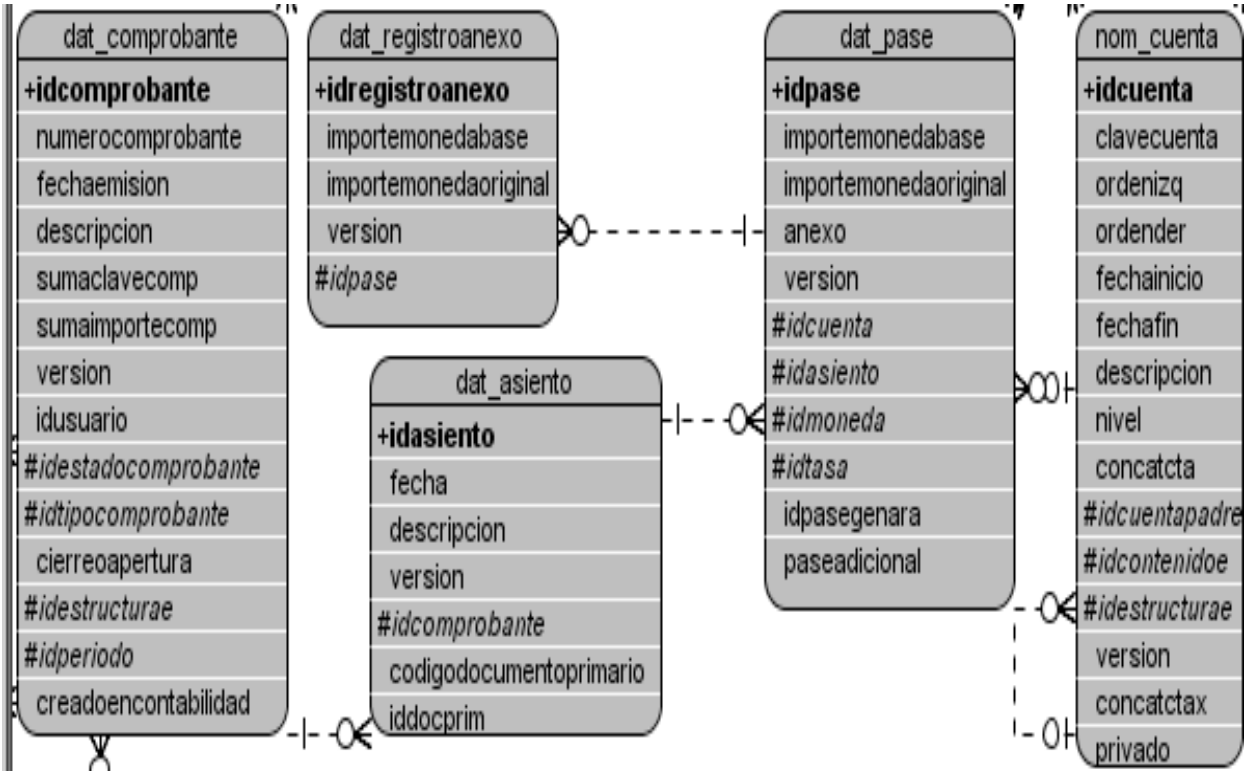


Figura 25: Tablas empleadas en la prueba para el requisito Imprimir comprobante de operaciones

Se puede afirmar que la base de datos para esta función cumple con un tiempo de respuesta bajo, pues devuelve dos filas en 63 ms. Para conocer el resto de las pruebas, consultar el artefacto en el expediente de datos de ambos subsistemas.

3.6. Conclusiones

En el presente capítulo se confeccionó la vista de la arquitectura de datos de los subsistemas Contabilidad y Costos y Procesos. Primeramente se posibilita establecer la correspondencia entre las tablas del modelo de datos y los componentes de cada subsistema, después del análisis realizado a los artefactos de entrada a dicha vista, por otro lado se contribuye a la mejora del rendimiento al identificar las pautas de optimización empleadas en ambos esquemas, se facilita la búsqueda para conocer las relaciones que se establecen entre subsistemas, mediante la traza obtenida en uno de los artefactos, por último, las pruebas realizadas permiten verificar que el diseño del modelo de datos cumpla con uno de los requisitos de calidad del cliente (rendimiento, específicamente el tiempo de respuesta exacto al realizar una consulta). De forma general la vista arquitectónica de datos permite controlar, describir y visualizar toda la información correspondiente al flujo de trabajo de la arquitectura de datos.

CONCLUSIONES GENERALES

Después de realizar un estudio de referencia sobre la arquitectura de software, tendiendo en cuenta conceptos relacionados con la disciplina tales como: vistas arquitectónicas, beneficios de la arquitectura de software, arquitectura de sistema, arquitectura de datos, diferentes tipos de arquitecturas existentes en la actualidad y la solución arquitectónica utilizada en el proyecto ERP Cuba, se ha llegado a las conclusiones siguientes:

- De todas las diferentes definiciones de vistas arquitectónicas y artefactos definidos se formalizaron los que por sus características se adaptaban al proyecto ERP Cuba.
- Las vistas arquitectónicas de sistema y datos de los subsistemas Contabilidad y Costos y Procesos son de suma importancia pues permiten la descripción general y el control de todo lo relacionado con la Arquitectura de Sistema y la Arquitectura de Datos de dichos subsistemas.
- Dada la importancia de contar con toda la información que se recoge en las vistas arquitectónicas de sistema y datos de los subsistemas Contabilidad y Costos y Procesos se hace necesario formalizar dichas vistas para el proyecto ERP Cuba.
- A partir de un análisis de los procesos de negocio de los subsistemas Contabilidad Costos y Procesos se generaron los artefactos de la vista de sistema y los artefactos de la vista de datos. Ambas soluciones contribuyen a la toma de decisiones por parte del equipo de trabajo, facilitan el entendimiento y la comprensión por parte de los involucrados y apoyan el soporte y desarrollo de nuevas versiones del sistema.

RECOMENDACIONES

Ningún sistema es estático en el tiempo debido a que la evolución del negocio y la tecnología así lo exigen es por ello que se recomienda:

- Actualizar las vistas arquitectónicas de sistema y datos de los subsistemas Contabilidad y Costos y Procesos para los cambios que surjan en versiones posteriores y para la comprensión de la solución y el apoyo al soporte y mantenimiento de nuevas versiones del sistema.
- Optimizar los servicios definidos en ambos subsistemas para evitar redundancias entre estos.
- Continuar efectuando pruebas al sistema para otros posibles escenarios que puedan existir.

BIBLIOGRAFÍA REFERENCIADA

Bass, Len, Clements, Paul y Kazman, Rick. 2003. *Software Architecture in Practice.* Second Edition. s.l. : Addison Wesley, 2003. 0-321-15495-9.

—. 1998. *Software Architecture in Practice.* 1998.

Boehm, Barry. 2005. Engineering Context (for Software Architecture). *Invited talk, First International Workshop on Architecture for Software Systems.* Seattle, Abril de 2005.

Carrascoso, Yoan A. y Chaviano, Enrique. 2008. *Propuesta de Arquitectura Orientada a Servicios para el Módulo de Inventario del ERP Cubano.* C. Habana : s.n., 2008.

CEIGE. 2009. *Línea Base del Proyecto ERP-Cuba.* C. Habana : s.n., 2009.

Clements, Paul. 1996. *A Survey of Architecture Description Languages.* Alemania : Proceedings of the International Workshop on Software Specification and Design, 1996.

Cristiá, Maximiliano. 2007. *Introducción a la Arquitectura de Software.* Rosario : s.n., 2007.

Garlan, David y Shaw, Mary. 1994. *An introduction to software architecture.* s.l. : CMU Software Engineering Institute Technical Report, 1994. CMU/SEI-94-TR-21, ESC-TR-94-21.

IEEE 1471. 2000. IEEE Standard 1471-2000, Recommended practice for Architectural. 2000.

Kontio, Mikko. 2005. ibm. *ibm.* [En línea] 2005. [Citado el: 15 de Febrero de 2010.] http://www.ibm.com/developerworks/webservices/library/wi-arch13.html.

Kruchten, Philippe. 2003. *Rational Unified Process.* s.l. : Addison-Wesley, 2003.

Leyet, Ing. Osmar. 2010. *Descripción de la Arquitectura de Software .* C. Habana : s.n., 2010.

Mena, Grette L. y Escalona, Arianna. 2009. *Expediente para la documentación técnica de la Arquitectura de Software de Sistemas de Gestión.* C. Habana : s.n., 2009.

2007. openbravo. *openbravo.* [En línea] 2007. [Citado el: 18 de Febrero de 2010.] http.//www.openbravo.com..

2001. openerp. *openerp.* [En línea] 2001. [Citado el: 18 de Febrero de 2010.] http://www.openerp.com/.

2005. openxpertya. *openxpertya.* [En línea] 2005. [Citado el: 19 de Febrero de 2010.] http://www.openxpertya.org/index.php?option=com_content&task=view&id=3&Itemid=4...

Rozanski, Nick y Woods, Eoin. 2005. *Software systems architecture: working with stakeholders using viewpoints and perspectives.* Boston : Addiso-Weley, 2005. 978-0-321-11229-3.

2006. visual-paradigm. *visual-paradigm.* [En línea] 2006. [Citado el: 26 de Febrero de 2010.] http://www.visual-paradigm.com/product/vpuml..

Wiggins, Andrew y Johnston, Richard. 2003. ibm. *ibm.* [En línea] 2003. [Citado el: 20 de Febrero de 2010.] http://www.ibm.com/developerworks/rational/library/754.html.

BIBLIOGRAFÍA CONSULTADA

Barry, Boehm. 1995. *Engineering Context (for Software Architecture).* Seattle : s.n., 1995.

Bass, Len, Clements, Paul y Kazman, Rick. 2003. *Software Architecture in Practice.* Second Edition. s.l. : Addison Wesley, 2003. 0-321-15495-9.

—. 1998. *Software Architecture in Practice .* s.l. : Adison Wesley, 1998.

2007. bsi-emea. *bsi-emea.* [En línea] 2007. [Citado el: 18 de Febrero de 2010.] http://www.bsi-emea.com/Integrated+Management/Overview/index.xalter..

Carrascoso, Yoan A. y Chaviano, Enrique. 2008. *Propuesta de Arquitectura Orientada a Servicios para el Módulo de Inventario del ERP Cubano.* C. Habana : s.n., 2008.

Casanova, Josep. 2004. desarrolloweb. *desarrolloweb.* [En línea] 2004. [Citado el: 16 de Febrero de 2010.] http://www.desarrolloweb.com/articulos/1622.php..

CEIGE. 2009. *Línea Base del Proyecto ERP-Cuba.* C. Habana : s.n., 2009.

Clemente, Daniel. 2005. danielclemente. *danielclemente.* [En línea] Junio de 2005. [Citado el: 19 de Febrero de 2010.] http://www.danielclemente.com/html/estandares.html...

Clements, Paul. 1996. *A Survey of Architecture Description Languages.* Alemania : Proceedings of the International Workshop on Software Specification and Design, 1996.

Cristiá, Maximiliano. 2007. *Introducción a la Arquitectura de Software.* Rosario : s.n., 2007.

David, Garlan. 2000. *Software Architecture: A Roadmap.* 2000.

epicor. *epicor.* [En línea] [Citado el: 17 de Febrero de 2010.] http://www.epicor.com/.

Garlan, David y Shaw, Mary. 1994. *An introduction to software architecture.* s.l. : CMU Software Engineering Institute Technical Report, 1994. CMU/SEI-94-TR-21, ESC-TR-94-21.

Georgas, John C., Dashofy, Eric M. y Taylor, Richard N. 2010. acm. *acm.* [En línea] 2010. [Citado el: 19 de Febrero de 2010.] http://www.acm.org/crossroads/espanol/xrds12-4/arqcentric.html#PerryWolf1992..

Gónzalez, Ing. Lariza. 2009. *Modelo de producción para el flujo de arquitectura.* C. Habana : s.n., 2009.

IEEE 1471. 2000. IEEE Standard 1471-2000, Recommended practice for Architectural. 2000.

Ing. Domínguez, Lisdanay y Ing. Pérez, Joisel. 2009. *Gestión y Control de los costos: solución informática para la gestión en las entidades en Cuba.* C. Habana : s.n., 2009.

Leyet, Ing. Osmar. 2010. *Descripción de la Arquitectura de Software .* C. Habana : s.n., 2010.

kontio, Mikko. 2005. ibm. *ibm.* [En línea] 2005. [Citado el: 15 de Febrero de 2010.] http://www.ibm.com/developerworks/webservices/library/wi-arch13.html.

Kruchten, Philippe. 2003. *Rational Unified Process .* s.l. : ADDISON-WESLEY, 2003.

Mena, Grette L. y Escalona, Arianna. 2009. *Expediente para la documentación técnica de la Arquitectura de Software de Sistemas de Gestión.* C. Habana : s.n., 2009.

2004. microsoft. *microsoft.* [En línea] 2004. [Citado el: 20 de Febrero de 2010.] http://msdn.microsoft.com/architecture/patterns/default.aspx?pull=/library/en-us/dnpatterns/html/ArcLayeredApplication.asp. Versión 1.0.1...

2007. openbravo. *openbravo.* [En línea] 2007. [Citado el: 18 de Febrero de 2010.] http.//www.openbravo.com.

2001. openerp. *openerp.* [En línea] 2001. [Citado el: 18 de Febrero de 2010.] http://www.openerp.com/.

2005. openxpertya. *openxpertya.* [En línea] 2005. [Citado el: 19 de Febrero de 2010.] http://www.openxpertya.org/index.php?option=com_content&task=view&id=3&Itemid=4..

Reynoso, Carlos Billy. 2004. *Introducción a la Arquitectura de Software.* Buenos Aires : s.n., 2004.

Reynoso, Carlos Billy y Kicillof, Nicolás. 2004. microsoft. *microsoft.* [En línea] 2004. [Citado el: 20 de Febrero de 2010.] http://www.microsoft.co.ke/spanish/msdn/arquitectura/roadmap_arq/style.aspx…

Rozanski, Nick y Woods, Eoin. 2005. *Software systems architecture: working with stakeholders using viewpoints and perspectives.* Boston : Addison-Wesley, 2005. 978-0-321-11229-3.

sei. *sei.* [En línea] [Citado el: 16 de Febrero de 2010.] http://www.sei.cmu.edu/architecture/..

Shaw, Mary y Garlan, David. 1996. *Software Architecture: Perspectives on an emerging discipline.* 1996.

2009. slideshare. *slideshare.* [En línea] 2009. [Citado el: 18 de Febrero de 2010.] http://www.slideshare.net/APCARDEN/erp-sistema-integral-de-gestion..

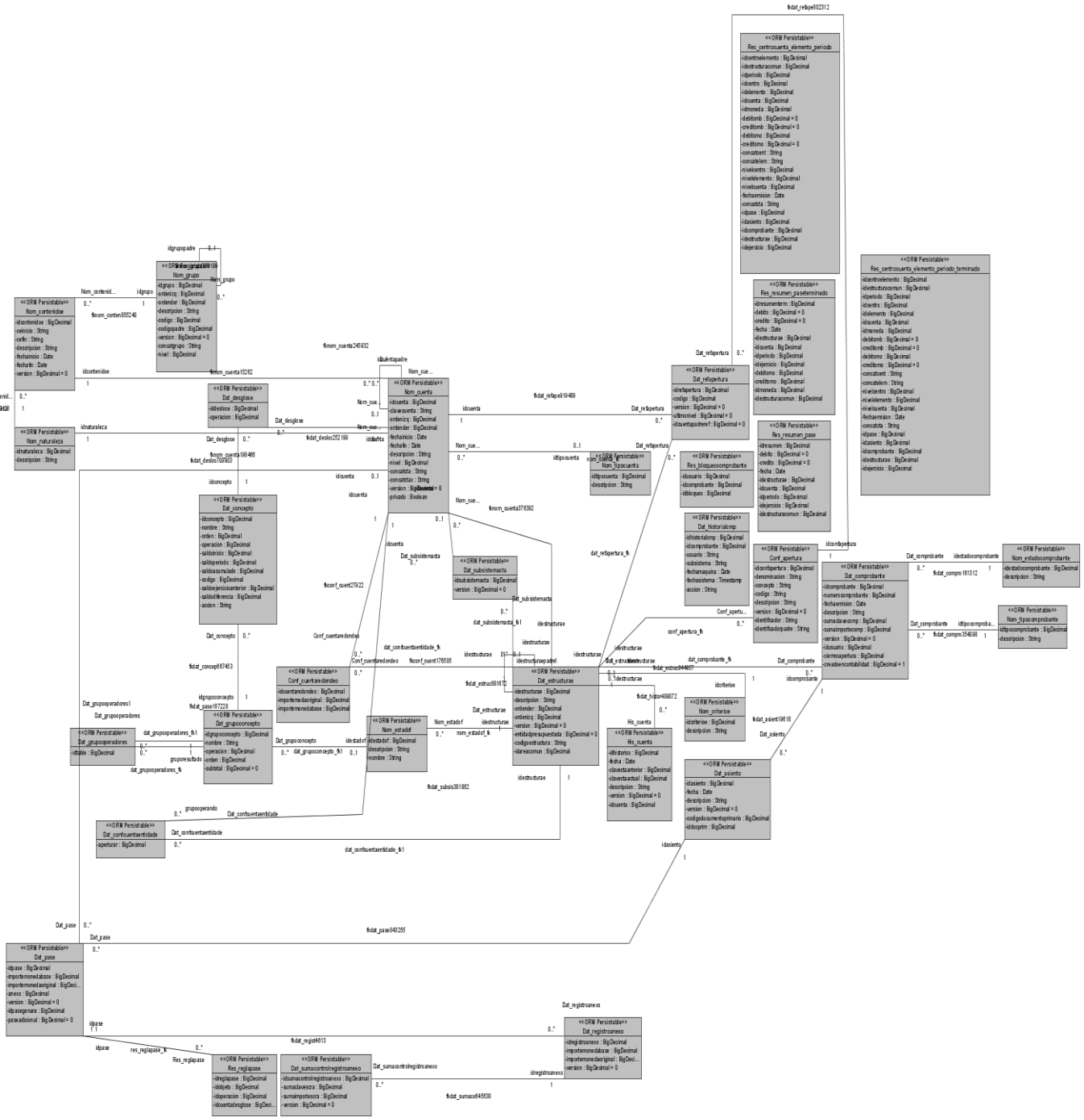
Sosa, Marisel y Cobo, Pedro H. 2009. betsime. *betsime.* [En línea] 2009. [Citado el: 18 de Febrero de 2010.] http://www.betsime.disaic.cu/secciones/eco_enemar_07.htm#2..

2006. visual-paradigm. *visual-paradigm.* [En línea] 2006. [Citado el: 26 de Febrero de 2010.] http://www.visual-paradigm.com/product/vpuml..

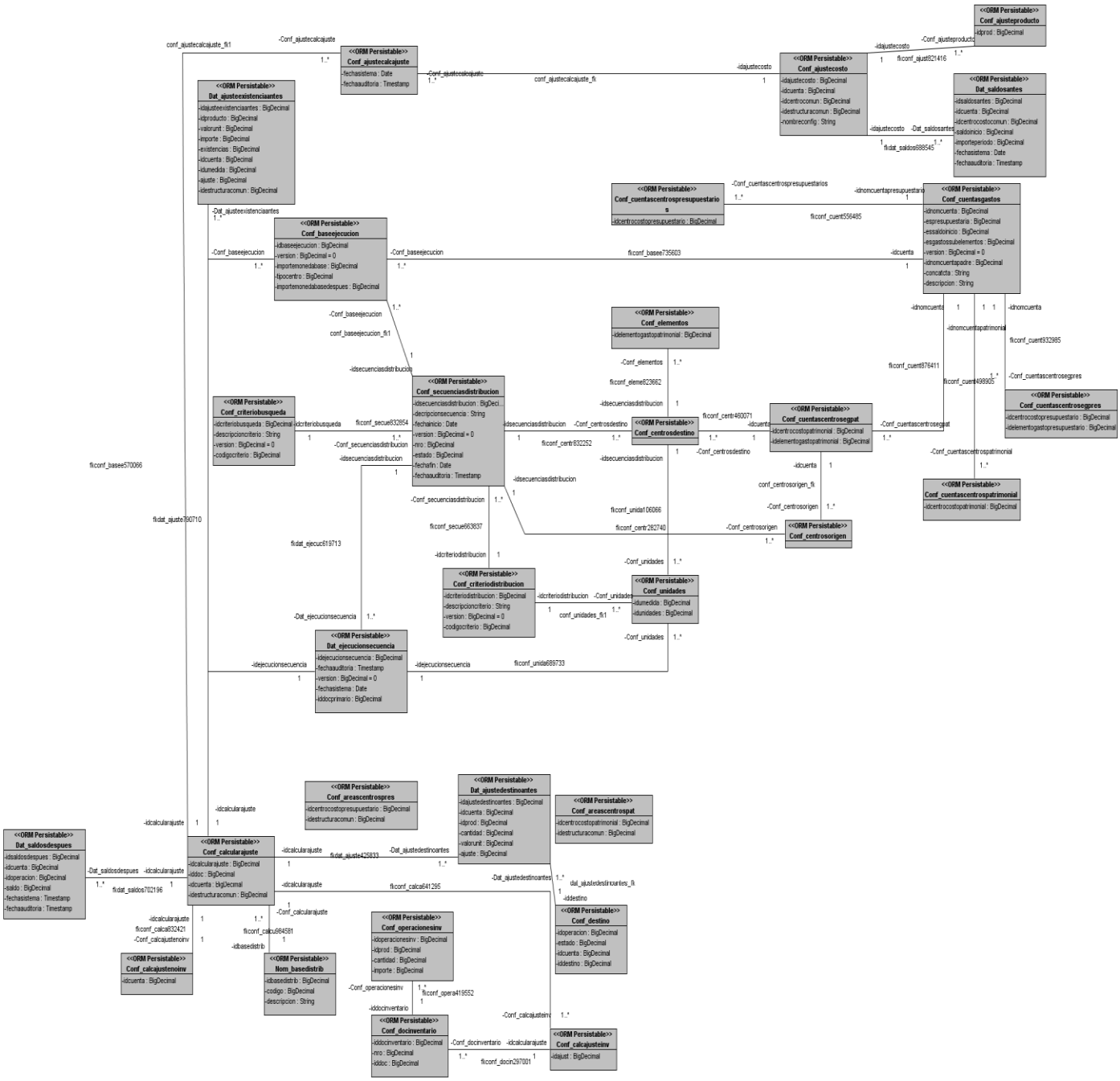
Wiggins, Andrew y Johnston, Richard. 2003. ibm. *ibm.* [En línea] 2003. [Citado el: 20 de Febrero de 2010.] http://www.ibm.com/developerworks/rational/library/754.html.

ANEXOS

Anexo 1- Modelo de Entidades del subsistema Contabilidad



Anexo 2- Modelo de Entidades Subsistema Costos y Procesos



GLOSARIO DE TÉRMINOS

ADLs: Por sus siglas en inglés, en español lenguajes de descripción arquitectónica.

CORBA: Por sus siglas en inglés, en español arquitectura común de intermediarios en peticiones a objetos

Concurrencia: Posibilidad de que varios usuarios modifiquen un mismo registro a la vez.

CSharp: Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET.

ERP: Por sus siglas en inglés, en español Sistema de Planificación de Recursos Empresariales.

Herramienta: Proporcionan el soporte automatizado para el proceso y los métodos.

HTTP: *(Protocolo de Transferencia de Hipertexto)*: Define la sintaxis y la semántica que utilizan los elementos software de la arquitectura Web para comunicarse; protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

IDL: Por sus siglas en inglés. Es un lenguaje de informática utilizado para describir la interfaz de componentes software.

Java: Lenguaje de programación orientada a objetos.

MVC: Patrón de diseño Modelo-Vista-Controlador.

.Net: Es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

SEI: Por sus siglas en inglés, es un instituto federal estadounidense de investigación y desarrollo,

SOA: Arquitecturas Orientadas a Servicios.

Software: conjunto de instrucciones que al ser ejecutadas proporcionan las características, funciones y el grado de desempeño deseados.

Trigger: Disparador en una Base de datos, es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

View Types: Puntos de vista.

WSDL: Por sus siglas en inglés, es un formato XML que se utiliza para describir servicios Web.