Universidad de las Ciencias Informáticas

Facultad 3

Título: Componente para la ejecución de flujos de trabajo en el marco de trabajo Sauxe.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor(es): Alexis Castro Díaz

Jean Pablo Matamoros Pifarret

Tutor(es): MSc. Raykenler Yzquierdo Herrera

Co-tutor: Ing. Yoriangel Rivero González

Junio 2012

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Informatización y Gestión de Entidades (CEIGE) 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 _____.

_____	_____
Alexis Castro Díaz	Jean Pablo Matamoros Pifarret
Autor	Autor
_____	_____
MsC. Raykenler Izquierdo	Ing. Yoriangel Rivero González

DATOS DE CONTACTO

Tutor: Raykenler Yzquierdo Herrera*:* Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el 2007. Se desempeñó durante el 2007 como analista principal del proyecto CCV y durante el 2008 como gerente general del proyecto. Durante el 2008 se incorpora al proyecto ERP Cuba, desempeñando el rol de jefe Soporte hasta el 2009. Se desempeñó como arquitecto de la línea Planificación presupuestada y empresarial durante el 2010. En septiembre del 2010 defiende su tesis de maestría. Actualmente trabaja en su doctorado. Ha participado en múltiples talleres, fórums y eventos en los cuales ha obtenido resultados relevantes. Ha impartido clases a pregrado generalmente de asignaturas de la disciplina de Programación y en posgrado ha impartido cursos de Soporte y Minería de proceso.

Co-tutor: Yoriangel Rivero González. Ingeniero en Ciencias Informáticas, graduado en el año 2010.

AGRADECIMIENTOS

A mis padres y mi hermano, que siempre estuvieron pendientes de cada movimiento dándome apoyo cuando lo necesité. A Karli el pelú, Collado, Juan Carlos, a Albertón, a todos mis amigos, a lo que se mantuvieron aquí, a los que por alguna razón no están. A Máximo, Jose, "Maneco", "Cuju", Legna, Pedrito, por estar ahí, para diversión, trabajo y todo lo que se nos ocurrió inventar. A la gente del proyecto, sobre todo a Javier por aguantar mis preguntas todos los días. A mis tutores, por discutir, preguntar y exigir. En fin, a todo el que de una forma u otra, dio su apoyo, su cuenta, su tiempo, para que 'esto', saliera adelante. Y sobre todo, miles de gracias a Saily, mi novia, por aguantar mis malcriadeces, darme todo su amor y confianza y preocuparse más que yo por mi tesis aun estando bien lejos.

Alexis

A mis padres por su apoyo incondicional, a los tutores y todos los que me ayudaron de alguna forma, especialmente los profesionales de el proyecto por su ayuda y preocupación. A mis compañeros de siempre Yarenis, Collado, Gloria, Yeremi , Luis, Aime, Pedro Luis, William, Mato que a pesar de estar en grupos separados nos mantuvimos muy unidos. A los compañeros de los dos proyectos en los que estuve. A mis compañeros del grupo siete que compartimos los dos últimos cursos.

Yan

DEDICATORIA

De Alexis:

A mi familia, que siempre estuvo ahí cuando la necesité.

A Saily, que espero siempre este aquí, cuando la necesite.

De Jean Pablo:

A mis abuelos paternos Mami y Papi, ya fallecidos.

A mis padres por su cariño y atención.

A toda mi familia, mis hermanos, primos, tías.

A Celia por su atención y apoyo todo el tiempo.

RESUMEN

En Cuba, se está desarrollando un Sistema de gestión empresarial sobre el marco de trabajo Sauxe, también producto nacional; ambos bajo el modelo de desarrollo de software propuesto por el Centro de Informatización de Gestión de Entidades (CEIGE), perteneciente a la Universidad de las Ciencias Informáticas. El marco de trabajo en cuestión contiene un componente para la gestión de flujos de trabajo, los cuales deben ser ejecutados de manera tal que satisfagan las necesidades actuales de Sauxe. El objetivo de este trabajo es lograr que dicha aplicación ejecute flujos de trabajo, de acuerdo con las restricciones existentes. Para esto será utilizado el componente de ejecución del conjunto de librerías ezComponents, el cual deberá ser adaptado para que ejecute las tareas de usuarios y de servicios, de manera que pueda interactuar con Sauxe, dando como resultado final un componente capaz de ejecutar flujos de trabajo adaptados para el mismo.

Después de concluida la implementación de este componente, se realizaron pruebas de rendimiento y caja blanca, las cuales arrojaron resultados satisfactorios, cumpliendo con los estándares de calidad definidos para estos parámetros, establecidos por el modelo mencionado anteriormente.

PALABRAS CLAVE

Flujos de trabajo, ejecución, ezComponents, Sauxe

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Actualmente, las empresas están optando por informatizar la gestión de la información que circula a través de ellas. La misma se logra con la implantación de sistemas informáticos, los cuales tienen la misión de gestionar de la forma más ágil esta información. A medida que pasa el tiempo, agregan más funcionalidades, significando aumentos en los ingresos de las empresas que los utilicen.

El término PAIS[1] surge para hacer referencia a los sistemas utilizados para establecer formas para mejorar los procesos empresariales. Por definición, un PAIS es un sistema que gestiona y ejecuta procesos que involucran tanto recursos como fuentes de información, estando conscientes de los procesos de negocio tanto de manera implícita como explícita. La conciencia implícita del proceso de negocio implica que el mismo sea conocido y manejado de forma interna, mientras que cuando se conoce de forma explícita el sistema es capaz de manipular procesos cambiantes. Los PAIS brindan una infraestructura que apoya un cambio en la programación del sistema, además permite que las tareas, tanto manuales como automatizadas, sean manejadas de forma que la empresa pueda sacar el mayor provecho de ellas. Ejemplo de los PAIS, son los sistemas de gestión de flujos de trabajo, WfMS[2] por sus siglas en inglés, y algunas plataformas de integración empresarial, conocidos por EAI [1, 2].

Entre los sistemas de definición implícita se encuentran los de planificación de recursos empresariales, ERP[3] por sus siglas en inglés, que se han vuelto muy populares, debido a las grandes ventajas que brinda. Estos sistemas son adaptables a las necesidades de una determinada empresa ya que son hechos expresamente para responder a un determinado modelo de procesos de negocio. Debido a que uno de sus principales objetivos es la coordinación de los negocios de la empresa, su implantación influye positivamente en la capacidad de la empresa para hacer frente a la competitividad del mercado, también le permite obtener un control sobre la información que maneja y la integración de dicha información con áreas claves para un buen funcionamiento. Los ERP reportan varias ventajas como por ejemplo, la reducción del costo de gerencia, un incremento en los ingresos, la centralización del control de la información, la posibilidad de medir constantemente los resultados obtenidos, una mejora en la administración financiera, además de proporcionar un solo sistema para gestionar muchos procesos

[1] Siglas de Process Aware Information Systems.

[2] Siglas de Workflow Management Systems.

[3] Siglas de Enterprise Resource Planning.

comerciales. Hay muchas otras ventajas, estas varían en dependencia del sistema ERP en cuestión y de la empresa en que se encuentra implantado, ya que la existencia de un mismo sistema ERP en diferentes empresas no implica necesariamente que estos tengan las mismas funcionalidades. También, tienen como desventajas principales el tiempo que demora su implementación e implantación y que al ser creados específicamente para un modelo de procesos, un cambio en este modelo tiene como consecuencia una modificación en el sistema a diferentes escalas[3].

Por su parte, los sistemas de gestión de flujos de trabajo, pertenecen a los sistemas con una declaración explícita, dándole a la empresa la oportunidad de adaptarse rápidamente a cambios en sus procesos de negocio. Se le conoce como WfMS a la automatización de procesos de negocio, este tiene como objetivo el apoyo a una determinada ruta de actividades realizadas en una empresa, para que estas se ejecuten de manera eficiente, en el momento requerido y por la persona específica que tiene la responsabilidad de ejecutarlas. Los WfMS funcionan sobre un conjunto definido de procesos utilizando sus especificaciones, representándolo como una plantilla de ejecución de casos concretos de un flujo de trabajo.

El ciclo de vida de un flujo de trabajo comienza con el reconocimiento de sus necesidades, después pasa a su modelación y evaluación, de aquí que cada flujo de trabajo pueda tener un ciclo de vida diferente, dependiendo de la modelación y evaluación del mismo. Llegado a este punto, ya se han definido los eventos que darán inicio y fin a la ejecución del flujo, es válido aclarar que los flujos de trabajo pueden ser rediseñados tantas veces como sea necesario para la formalización de métodos de análisis y la definición de técnicas de simulación. Luego de informar y capacitar al personal implicado, el flujo de trabajo se implementa en la organización, creando una instancia específica del mismo, posteriormente se puede proceder al reconocimiento de la necesidad de modificar la instancia, en caso de ser aceptada la misma, el flujo de trabajo vuelve a ser modelado y evaluado en correspondencia con las necesidades detectadas, creando una nueva instancia y archivando la anterior[1, 4].

Asimismo, cada instancia tiene su ciclo de vida particular conocido en la bibliografía especializada como modelo de ejecución, que pasa por varias fases, comenzando con el inicio del flujo de trabajo por un evento definido, pasa por una etapa de ejecución, dicha ejecución puede bajo ciertas condiciones desviarse del esquema propuesto, y finaliza cuando se llega a un estado de culminación determinado que puede representar una terminación exitosa o excepcional. Las instancias de un mismo flujo de trabajo pueden estar activas en un momento determinado sin que esto traiga consecuencias negativas para el flujo[1, 4].

Con la correcta implementación e implantación de un sistema de flujo de trabajo, las empresas pueden obtener varios beneficios, entre ellos la mejora y estandarización de procesos, la reducción de tiempo dedicado a diversas labores y por tanto de dinero, la reducción del trabajo manual, el manejo de información concisa, incluyendo la seguridad y privacidad de la misma. Debido a todas las funcionalidades que poseen estos sistemas, su implementación tiene un grado de complejidad relativamente alto en comparación con otros sistemas PAIS[5, 6].

Una lista de los beneficios específicos que provee un WfMS puede ser bastante extensa, de estas, su capacidad para brindar flexibilidad a las empresas a la hora de realizar cambios en cualquiera de sus procesos de negocio, dio pie al surgimiento de sistemas pensados para integrar en un solo sistema las funcionalidades de un ERP y las de un sistema de gestión de flujos de trabajo, eliminando de ese modo las restricciones del ERP para soportar un cambio en los procesos de negocio.

Al estar las empresas a nivel mundial inmersas en este proceso de informatización y optimización, Cuba debe desarrollar un sistema de este tipo para insertarse de manera gradual en el mercado. Dicho sistema no debe ser importado ya que debe ajustarse a las características del modelo económico cubano. El sistema en cuestión, el ERP Cedrux, se está desarrollando actualmente en la Universidad de las Ciencias Informáticas, sobre el marco de trabajo Sauxe. Con el objetivo de darle flexibilidad a este ERP, se decidió incorporarle a Sauxe un componente para la gestión de flujos de trabajo, que interactuará con Cedrux. Esta aplicación consta de varios componentes, un modelador encargado de permitir al usuario construir visualmente un flujo de trabajo, hecho esto, el sistema transformará esa definición a una homóloga en lenguaje PHP; después un validador hará validaciones a dicho flujo de trabajo, para comprobar que este responde a las especificaciones de BPMN[4], posteriormente el sistema llevará esta definición del flujo de trabajo al estándar XPDL[5], añadiéndole ciertas peculiaridades para que se ajuste al marco de trabajo; por último, este XPDL será transformado en un lenguaje de ejecución que servirá de entrada al motor de flujos de trabajo encargado de ejecutar las definiciones, el cual estará basado en la librería ezComponents[7], y por tanto no ejecuta los flujos de trabajo de acuerdo a las necesidades de Sauxe.

Esto plantea el siguiente **problema a resolver**: ¿Cómo contribuir en la ejecución de los flujos de trabajo en las soluciones desarrolladas con Sauxe a partir de las especificaciones de los procesos en un lenguaje de ejecución de flujos de trabajo?

[4] Siglas de Business Process Management Notation.

[5] Siglas de XML Process Definition Language.

Para la solución de este problema se definió como **objetivo general**: Desarrollar un componente que contribuya en la ejecución de los flujos de trabajo en las soluciones desarrolladas con Sauxe a partir de las especificaciones de los procesos en un lenguaje de ejecución de flujos de trabajo.

Este objetivo general se desglosa en los siguientes **objetivos específicos**:

- Realizar el marco teórico acerca de las herramientas de ejecución de flujos de trabajo
- Realizar un diseño de la solución
- Realizar la implementación de la solución
- Validar la solución

Se plantea como **objeto de estudio**: Los marcos de trabajo que manejan flujos de trabajo.

Y el **campo de acción** se define específicamente sobre: La ejecución de los flujos de trabajo.

Se plantea como **idea a defender**: El desarrollo de un componente de ejecución de flujos de trabajo para el marco de trabajo Sauxe permitirá la ejecución de flujos de trabajos definidos y adaptados para el mismo.

Para el desarrollo de la solución se tienen las siguientes tareas:

- Análisis de las herramientas de ejecución de flujos de trabajo
- Diseño del componente para la ejecución de flujos de trabajo en el marco de trabajo Sauxe
- Integración la gestión de las trazas producidas por ezComponents considerando las especificaciones del marco de trabajo Sauxe
- Integración del marco de trabajo Sauxe con ezComponents para la persistencia de datos
- Integración del marco de trabajo Sauxe con ezComponents para la ejecución de servicios
- Integración del marco de trabajo Sauxe con ezComponents para el manejo de tareas semiautomáticas

Para el desarrollo del presente trabajo se utilizaron los siguientes métodos de la investigación científica:

Analítico sintético: Mediante el uso de este método se realizó un análisis de los principales marcos de trabajo que implementaran motores de flujo de trabajo, facilitando su estudio y logrando definir una estrategia para llegar al resultado final con más facilidad. Este método permite la extracción de los elementos más importantes de cada documento analizado.

Histórico lógico: El uso de este método permitió conocer y comprender qué beneficios trae consigo el uso de los motores de flujo de trabajo en marcos de trabajo o plataformas en el mundo, qué funcionalidades incorporan los motores a sistemas y cómo se integran, conociendo así su evolución y desarrollo hasta la actualidad.

El presente trabajo estará estructurado en los 3 capítulos enunciados a continuación:

Capítulo 1 Fundamentación Teórica

En este capítulo se abordan una serie de conceptos necesarios para un mejor entendimiento del problema y la solución propuesta. También se realiza un estudio del estado del arte orientado a herramientas utilizadas para la gestión de flujos de trabajo, con el objetivo de determinar cuál de ellas permite su uso en Sauxe. Asimismo se hace el análisis correspondiente al entorno de desarrollo existente, describiendo sus características, así como las tecnologías y herramientas disponibles.

Capítulo 2 Análisis y diseño

Este capítulo está dirigido a definir el análisis y diseño de la solución propuesta. Se licitan y describen los requisitos funcionales y no funcionales del sistema. Son generados y explicados los artefactos correspondientes, siguiendo el modelo de desarrollo de software orientado a componentes propuesto por CEIGE.

Capitulo 3 Implementación y prueba

En el tercer y último capítulo se describen las acciones tomadas para la implementación de la solución. También se realizan pruebas al sistema para validar la calidad del diseño y la implementacion de la solución.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Este capítulo está encaminado a realizar un estudio de algunos sistemas gestores de flujos de trabajo y plataformas o marcos de trabajo conocidos, que tengan incorporados estas herramientas. Este análisis será de carácter crítico, ya que tiene como objetivo determinar cuál motor es más factible para adaptarlo a Sauxe, incluidos aquellos que están integrados a plataformas o marcos de trabajo.

1.2 Conceptos

Se estarán abordando un conjunto de conceptos, siendo estos considerados importantes para la elaboración de la investigación en cuestión, para dar un mejor entendimiento de la problemática a resolver.

¿Qué es un proceso de negocio?

Se conoce como proceso de negocio al conjunto de actividades que se llevan a cabo en un ambiente empresarial u organizacional con el fin de llegar al cumplimiento de un objetivo de negocio. Debido al contexto en el cual se desarrollan, en estos procesos intervienen roles funcionales que establecen relaciones, ya sea entre ellos mismos o con recursos o actividades específicas. Un proceso, ya sea de negocio o no, debe tener requisitos de entrada, estos requisitos serán introducidos en las funciones que los necesiten para su ejecución y estas proporcionarán una serie de datos de salida o resultantes; es decir, un proceso realmente definido debe transformar datos de entrada en resultados deseados utilizando acciones definidas[8, 9] .

Pueden ser de carácter intra-organizativo, ejecutados dentro de una sola organización o empresa, o inter-organizativo, aplicable para varias unidades, como por ejemplo las relaciones cliente-proveedor.

¿Qué es un workflow o flujo de trabajo?

En un workflow o flujo de trabajo, se estructuran las actividades que deben ser realizadas en un proceso de negocio, la realización de ellas y el orden en que deben realizarse, así como la sincronización entre ellas, el flujo de la información que les da soporte y el seguimiento de su cumplimento. En otras palabras, describe los posibles caminos por los cuales pueden transitar los procesos de una empresa, definiendo las reglas para tomar alguno de ellos y las acciones necesarias para transitarlos[8, 9].

¿Qué es BPM?

El término BPM[6] se ha ido formando paulatinamente a partir de experiencias de colectivos dedicados a la gestión empresarial. En teoría, BPM asegura la subsistencia de las entidades, permitiéndoles gestionar y optimizar sus procesos en aras de resultados satisfactorios. Abarca un conjunto bastante amplio de elementos, dígase personas, aplicaciones, sistemas, funciones, negocios, clientes, proveedores e incluso socios. En BPM coexisten tecnologías y teorías de gestión de procesos, permitiendo la realización de nuevos enfoques para los diseños de procesos de negocio. Básicamente, el uso de la metodología BPM implica una convergencia entre las tecnologías de la información (TI) y las actividades de negocio, además de la coordinación de acciones y comportamientos, ya sea de personas o de sistemas, al interactuar con procesos de negocio [10].

¿Qué es un BPMS?

Un BPMS[7] es un sistema dedicado enteramente a la gestión de procesos de negocio, que permite a las empresas la modelar, implementar y gestionar sus procesos de negocio, incluyendo el uso de aplicaciones empresariales. Está definido bajo la tecnología de BPM e integra la infraestructura de apoyo, las capacidades técnicas y los módulos funcionales necesarios para la correcta aplicación de dicha tecnología. Estos sistemas deben tener la facultad de extraer procesos de las diferentes aplicaciones con las cuales interactúan y almacenarlos en un repositorio, con el objetivo de que dichas aplicaciones puedan posteriormente acceder al repositorio para consultarlos y trabajar sobre los procesos.

Un BPMS debe, entre otras funcionalidades:

- Permitir la modelación de procesos
- Brindar entornos de desarrollo para aplicaciones de colaboración
- Generar, actualizar y publicar documentación de los procesos
- Automatizar los procesos[10-12]

Un sistema de gestión de flujos de trabajo es un sistema informático que automatiza un proceso de negocio, gestionándolo como una secuencia de actividades cuya realización requiere del empleo de recursos humanos y tecnológicos, los cuales están asociados a las actividades en las diversas etapas de su desarrollo. Hay dos tipos de sistemas de gestión de flujo de trabajo: los que están basados en actividades y aquellos que son basados en entidades. Los primeros tienen su enfoque en las actividades

[6] Del inglés Business Process Management.

[7] Del inglés Business Process Management System.

que van a completar todo el flujo de trabajo, los segundos saltan su atención por las entidades, tales como documentos, que son procesados por un flujo de trabajo[7, 8].

¿Qué es un lenguaje de ejecución?

Se conoce como lenguaje de ejecución al lenguaje que utilizan los sistemas y motores de flujo de trabajo para realizar la ejecución de los mismos, uno de los más utilizados por los desarrolladores es BPEL4WS[8], poseedor de una gran expresividad en cuanto a implementación de patrones de flujos de trabajo, esto no significa que sea el único, también son utilizados BPELJ[9] y JPDL[10][12]. Normalmente los lenguajes de ejecución orientados a servicios web, utilizan un componente conocido como WSDL[11] para la comunicación entre componentes.

1.3 Gestores de flujos de trabajo

1.3.1 JBoss jBPM

JBoss es un gestor de procesos, tiene incorporado una herramienta de flujo de trabajo conocida como jBPM, implementada totalmente en Java; funciona independiente de algún servidor de aplicaciones y es totalmente libre, siguiendo la ideología de la empresa sobre la potencialidad del código abierto para impulsar los nuevos estándares de BPM.

JBPM es aplicado esencialmente en tres escenarios, uno para empresas que tengan sus sistemas implementados sobre la plataforma J2EE, que pueden incorporar JBoss jBPM simplemente con la inclusión de su librería; otro para aplicaciones de ERP, las cuales pueden implementar jBPM y utilizar sus potencialidades para adquirir mayor flexibilidad y capacidad de personalización; el tercer escenario es la utilización de la herramienta como un componente independiente de la arquitectura de una empresa, proporcionando estándares de administración de negocios bastante confiables.

La herramienta provee un motor de procesos que da seguimiento a los estados y variables de los procesos activos, incluyendo:

[8] Del inglés Business Process Execution Language for Web Services.

[9] Del inglés BPEL for Java technology.

[10] Del inglés Java Process Definition Language.

[11] Del inglés Web Services Description Language.

- Controlador de servicios como infraestructura para el envío de tareas a los procesos, usuarios o aplicaciones requeridas
- Servicios de interacción: conjunto de servicios tanto estándares como personalizados que interactúan con aplicaciones existentes para funciones y datos que serán usados en determinados procesos
- Administrador de estado: encargado de procesos potenciales, incluida la persistencia de registros y datos, además de manipular bases de datos con los resultados de las acciones realizadas

Como complemento, el motor posee un monitor de proceso: Un módulo diseñado para proporcionar visibilidad del estado del proceso, también permite la visualización del estado de las aplicaciones con las cuales está interactuando dicho proceso.

El motor de jBPM está basado en JPDL, siendo éste el lenguaje por defecto de la herramienta, que admite otros estándares como BPELJ, BPML, el XPDL definido por la WfMC[12]y BPEL como lenguaje estándar de la comunidad de flujo de trabajo.

Independientemente de ser bastante flexible, jBPM solo puede funcionar o bien sobre una máquina virtual de Java, dentro de algún servidor de aplicaciones J2EE o formando parte de buses o canales empresariales conocidos como Enterprise Service Bus (ESB), éste último posibilita la combinación de un motor de flujos de trabajo con un sistema de mensajería[13, 14].

1.3.2 Bonita Open Solution

Bonita Open Solution es un paquete ofimático para la gestión de procesos de negocio (BPM) y realización de flujos de trabajo, que establece un estándar propio para BPM, creado en el 2001. Es código abierto y se encuentra bajo la licencia GPL v2. Bonita está compuesto de tres partes principales combinadas en una misma solución, dando como resultado un **estudio** para el diseño de procesos, un **motor de BPM** y una **interfaz de usuario**, definidas a continuación:

[12] Workflow Management Coalition.

- Bonita Studio (**estudio**) que permite al usuario modificar gráficamente los procesos de negocio siguiendo el estándar BPMN, así como interactuar con procesos diseñados con otros estándares y tecnologías como XPDL o jBPM
- Bonita User Experience (**interfaz de usuario**) es un portal web que permite a cada usuario final gestionar en una interfaz similar a la del correo web (webmail-like) todas las tareas y procesos en las cuales está involucrado. También permite al propietario de un proceso administrarlo y obtener informes sobre otros procesos
- Bonita BPM Engine (**motor BPM)** es una JAVA API que permite al usuario interactuar programáticamente con el proceso o los procesos. Está disponible bajo licencia LGPL.

El motor de ejecución de procesos de negocio de Bonita, puede ampliarse para integrar nuevos servicios o las normas de BPM a medida que surgen[14, 15].

1.3.3 OpenWFE

OpenWFE es una solución libre para la gestión de flujos, las versiones más difundidas se encuentran implementadas en Java, no obstante existen otras versiones conocidas como OpenWFEru, desarrolladas en Ruby. Entre sus componentes se encuentran:

- Un entorno Web llamado DorFlo, para la definición gráfica de los modelos de procesos de negocio, la misma será persistida en formato XML haciendo uso de un lenguaje de definición nativo de la solución, conocido con el mismo nombre. Este lenguaje carece de documentación bibliográfica suficiente para hacer una definición más específica del mismo
- Una interfaz gráfica del motor conocida como WebClient o appserver, concebida para brindar al cliente un listado de las tareas que el motor distribuye
- Una interfaz web para gestionar recursos llamada UMAN[14]

1.4 Marcos de trabajo

Un marco de trabajo o framework como también se le conoce, es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Normalmente debe incluir soporte de programas, librerías y otros sistemas, para ayudar al desarrollo o integración de diferentes componentes de una aplicación. Define una filosofía de trabajo, proporcionando funcionalidades que facilitan las acciones del programador, ahorrando trabajo y tiempo; también al evitar el código duplicado permite producir aplicaciones más fáciles de mantener.

Un framework agrega funcionalidad extendida a un lenguaje de programación, esta automatiza muchos de los patrones de programación para orientarlos a un determinado propósito. Además, proporciona una estructura al código haciendo que los desarrolladores implementen de una manera más entendible. Hace la programación más fácil, convirtiendo complejas funciones en sencillas instrucciones[16].

1.4.1 SAFE

La plataforma SAFE (Sage Application Framework for Entreprise) soporta el ERP Sage ERP X3, cuenta con un motor de flujo de trabajo de segunda generación que permite poner en marcha circuitos de información, dentro y fuera de la organización, acciones de seguimiento, alertas y flujos de validación a partir de cualquier suceso registrado en el ERP, siguiendo los criterios establecidos para la gestión de eventos excepcionales o críticos. Este se encuentra integrado al área de Compras, cuando en el sistema se introduce un nuevo pedido de compra, Sage ERP X3 puede informar inmediatamente a un comprador particular, en función de la naturaleza del pedido y avisar, también automáticamente, a un responsable de la empresa, si dicha compra supera un importe determinado. Gracias al motor de flujo de trabajo, los procesos de firma electrónica y la trazabilidad de los datos son flexibles, asegurando la simplicidad y el control de todas las actividades; su uso permite a los usuarios desencadenar eventos(iniciar un proceso, enviar un e-mail, entre otros), basados en acciones de usuarios tales como impresión, creación de nuevos registros, edición, eliminación de datos o un evento basado en el resultado de una consulta a la base de datos, también activar el envío de correos (incluidos documentos adjuntos), permitiendo que los destinatarios inicien acciones haciendo clic en los botones de opción embebidos y/o marcando sobre la entrada en el ERP que desencadenó el mensaje. Debido a que es un software propietario no existe documentación referente a las especificidades del motor más allá de lo antes expuesto[17, 18].

1.4.2 SAP NetWeaver

La plataforma SAP NetWeaver es el framework sobre el cual está implantado el ERP-SAP. Esta plataforma cuenta con un motor llamado Business Workflow, desarrollado en ABAP[13]. Este lenguaje fue definido por SAP específicamente para implementar sus sistemas. El motor permite la definición de flujos de trabajo de procesos de negocio, proporcionando un ambiente gráfico, así como la ejecución del flujo de actividades en un orden específico, por varios usuarios. Las tareas asignadas a un usuario en particular,

[13] Siglas de Advanced Business Application Programming.

pueden ser visualizadas en una especie de bandeja de entrada a la cual el usuario tiene acceso. Por otro lado, el motor mantiene un control sobre todo el proceso, distribuyendo de forma adecuada los elementos del trabajo. Es válido aclarar que debido a que el motor Business Workflow está implementado en ABAP, lenguaje propiedad de SAP, la bibliografía y datos más específicos sobre el motor, no son de dominio público[19].

1.4.3 Enhydra Shark

Enhydra es una plataforma de código abierto desarrollada en Java, cuenta con un editor visual para flujos de trabajo y con un motor de ejecución. El editor, conocido como Enhydra JaWE[14], da al usuario la posibilidad de crear, gestionar y revisar una definición de un proceso de negocio de forma visual, también tiene la funcionalidad de exportar dicha definición a XPDL, así como importar una definición válida, ya sea en XPDL o una representación gráfica de la misma.

Enhydra Shark es el motor utilizado por esta plataforma para la ejecución de flujos de trabajo basándose en las especificaciones definidas por la WfMC y el OMG[15], cuenta con un administrador llamado Shark Admin, siendo este uno de los elementos del paquete de herramientas Shark, cuenta con dos modalidades diferentes de administración, una es su uso directamente como una biblioteca, la otra es mediante la comunicación con otros elementos de Shark implementados usando la interfaz CORBA de dicho paquete. El Admin es utilizado para manipular XPDL externos residentes en un repositorio, ya sea para cargarlos en un momento determinado como para eliminar alguno existente que no sea necesario. Una vez que un XPDL es cargado, puede ser actualizado e instanciado, a la vez que se monitoriza su ejecución, haciendo la asignación entre las definiciones de los participantes y los usuarios reales.

El motor posee un árbol con las instancias de procesos disponibles, permitiendo la selección de aquella que se desea ejecutar. Cuando la instancia es seleccionada, son visualizadas sus principales características, así como el estado de la misma en cuanto a las actividades que estén ejecutándose en ese momento. Enhydra Shark también admite, entre otras, realizar sobre las instancias las siguientes actividades:

- Dar inicio, suspensión, cancelación y continuación al proceso instanciado
- Ver un historial de las acciones realizadas por el proceso

[14] Siglas de Java Workflow Editor.

[15] Object Management Group.

- Ver la descripción del proceso
- Editar las variables del proceso, dando la posibilidad de gestionar el flujo en caso de ser necesario
- Mostrar un listado de las actividades del proceso, visualizando el estado de una actividad escogida por el usuario, estas actividades también pueden ser suspendidas, reanudas, interrumpidas o iniciadas manualmente
- Eliminar procesos terminados o seleccionados
- Administrar las cuentas existentes en un servidor
- Asignar paquetes de procesos y participantes de los mismos a usuarios reales

En esencia, Enhydra Shark es considerado como un motor de flujo de trabajo extensible que incluye una implementación estándar de los flujos de trabajo, basada en las especificaciones de la WfMC, haciendo uso de XPDL como extensión nativa para la persistencia de las definiciones de flujos de trabajo[20].

1.4.4 Microsoft Dynamics

Microsoft Dynamics es una plataforma de código abierto desarrollada por la empresa Microsoft, sobre la cual está implementado el ERP Microsoft Dynamics NAV. Esta plataforma, al ser de código abierto, ha soportado muchísimos cambios en su funcionamiento, por parte de varias empresas asociadas. Estos cambios han sido conocidos como complementos y se han agrupado en piezas funcionales, algunos son específicamente para la implementación de sistemas ERP, otros pueden ser utilizados para la definición y ejecución de flujos de trabajo. En un principio la plataforma estuvo implementada en C++, posteriormente se migró a .NET con el lanzamiento de la versión del 2009 del sistema ERP, que contenía una interfaz de usuario completamente nueva, mucho más amigable que la anterior.

La empresa ve las ventajas del uso de flujo de trabajo en la facilidad que provee el mismo para guiar el proceso definido hasta su final, por ejemplo, las ventas realizadas mediante ese sistema son gestionadas por un motor de flujo de trabajo, lo mismo pasa con la creación y aprobación de documentos. El sistema está basado en eventos y provee de una interfaz conocida como “Role Center”, que permite al usuario identificar las actividades de un determinado flujo de trabajo que tiene pendientes en un momento específico[21].

La bibliografía disponible no aporta datos diferentes a los antes expuestos.

1.4.5 Windows Workflow Foundation

La compañía Microsoft, también provee otra aplicación para la gestión de flujos de trabajo, se trata de Workflow Foundation. Esta herramienta, básicamente, es un entorno para el desarrollo de aplicaciones utilizando flujos de trabajo sobre Visual Studio. Soporta diversos escenarios como por ejemplo:

- La habilitación de flujos de trabajo dentro de aplicaciones empresariales
- Flujos de trabajo centrados en documentos
- Flujos de trabajo compuestos por aplicaciones orientadas a servicios
- Flujos de trabajo para la administración de sistemas

Entre sus características cuenta con la asimilación de dos tipos de flujos de trabajo, de acuerdo con una filosofía de control del mismo, los llamados secuenciales, utilizados para dar control principalmente al proceso definido y por ende para la automatización de procesos; y los de máquina de estado, empleados cuando el control del flujo se debe mayormente a la interacción de los usuarios con el sistema y por tanto definir una secuencia de actividades resulta prácticamente imposible. Permite la persistencia de flujos de trabajo y sus ejecuciones, así como la resolución de problemas haciendo uso de un modelo de dominio específico, logrando la utilización de una terminología común para cada dominio. También posee un motor de reglas y un modelador gráfico, todo esto integrado al Visual Studio, proporcionando una especie de ecosistema para el desarrollo con flujos de trabajo.

Según la empresa, esta solución para la gestión de flujos de trabajo es extensible, debido a una serie de puntos de extensión para la modificación del comportamiento de los flujos de trabajo, en caso de que este no se adapte a las necesidades de los clientes[22, 23].

1.4.6 EzComponents

El framework de desarrollo de aplicaciones cliente-servidor ezComponents es una biblioteca de alta calidad de componentes independientes, que ayudan en el desarrollo de aplicaciones centradas en la web, desarrollado bajo la licencia de software libre BSD[16], que permite el uso del código fuente. Todos sus componentes están desarrollados en PHP[17], donde cada componente es completamente independiente de los demás, lo que posibilita elegir exactamente qué partes de la biblioteca se desea utilizar en una aplicación. EzComponents no dicta la forma de organizar una aplicación, sino que proporciona la

[16] Siglas de Berkeley Software Distribution.

[17] Siglas de Hypertext Pre-processor.

flexibilidad para decidir cómo y cuándo usar cada componente. Entre los componentes que integran el framework se encuentran Archivo, Autentificación, Configuración, Base de Datos, Ejecución, Correo, Flujo de trabajo entre otros, algunos descritos brevemente a continuación:

- Archivo: el componente le permite crear, modificar y extraer archivos comprimidos en varios formatos, soporta actualmente Tar.
- Flujo de trabajo: su propósito es proporcionar la funcionalidad básica de un sistema de flujo de trabajo basado en la actividad, incluyendo la definición y ejecución de las especificaciones del mismo
- Correo: permite construir y / o analizar mensajes de correo conforme a la norma electrónica. Tiene soporte para archivos adjuntos, mensajes de varias partes y el correo HTML[18]. También interactúa con SMTP[19] para enviar correo o IMAP[20], POP3[21] o mbox[22] para recuperar el correo electrónico
- Base de Datos: una capa de base de datos ligera en la parte superior de PDO[23] que permite utilizar una base de datos sin necesidad de hacerse cargo de las diferencias entre los dialectos SQL[24].[7]

1.5 Modelo de desarrollo

Define un conjunto de actividades, acciones, tareas, fundamentos y productos de trabajo que se requieren para desarrollar software de alta calidad. Debe llenar cada actividad del marco de trabajo con un acumulado de acciones de ingeniería de software, y definir cada acción en cuanto a un grupo de tareas que identifique el trabajo que deben completarse para alcanzar las metas de desarrollo.

[18] Siglas de HyperText Markup Language.

[19] Siglas de Simple Mail Transfer Protocol.

[20] Siglas de Internet Message Access Protocol.

[21] Siglas de Post Office Protocol.

[22] Acrónimo de mailbox.

[23] Siglas de PHP Data Objects.

[24] Siglas de Structured Query Language.

1.5.1 Modelo de desarrollo de software propuesto por CEIGE

El modelo de desarrollo propuesto fue elaborado por el equipo de producción del CEIGE[25], teniendo en cuenta los principales riesgos con los que se cuentan en el proyecto.

Este modelo está orientado a la reutilización de componentes parametrizables, donde se simplifican las actividades y se eleva el nivel de especialización de los involucrados. Proporciona una guía para regir el proceso de desarrollo de software tecnológico, centrado en la arquitectura.

Dicho modelo está basado en principios, prácticas propuestas y algunos elementos de las metodologías SCRUM, RUP y XP, fue elaborado teniendo en cuenta las características especiales que presenta la Universidad de Ciencias Informáticas (UCI).

En general, propone una solución, que se centra en el desarrollo de componentes como base tecnológica, con una mayor calidad y en menor tiempo, para su posterior uso en la construcción de productos concretos; además propone dividir el trabajo y el equipo de desarrollo para lograr mayor especialización. Su buena aplicación proporcionará en gran medida la independencia tecnológica de los sistemas finales, pudiendo así ahorrar grandes sumas de dinero al país. [24]

1.5.2 Características

Centrado en la arquitectura

La arquitectura determina la línea base, los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades del desarrollo y resuelve las necesidades tecnológicas y de soporte para el desarrollo.[24]

Orientado a componentes

Las iteraciones son orientadas por el nivel de importancia arquitectónica de los componentes, los mismos son abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan, el componente es la unidad de medición y ordenamiento de las iteraciones.[24]

Iterativo e incremental

[25] Siglas de Centro de Informatización para la Gestión de Entidades.

Las iteraciones son planificadas y coordinadas con el equipo de arquitectura, los clientes y la alta gerencia. Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo de esta manera la evolución incremental del producto.[24]

Ágil y adaptable al cambio

El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados.[24]

1.6 Librerías y Marcos de Trabajo definidos por CEIGE

Desde el punto de vista informático una librería, es un conjunto de funciones destinadas a formar parte de programas independientes, contienen códigos que pueden ser reconocidos y utilizados por dichos programas en cualquier momento, aportando funcionalidades útiles para su funcionamiento.

1.6.1 Sauxe

Sauxe es un marco de trabajo que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor agilidad en el proceso de desarrollo.Está basado en Zend framework, utiliza en la capa de acceso a datos el Lenguaje de Consulta de Datos (DQL) que implementa Doctrine. Utiliza ExtJS en la capa de presentación, por la gran gama de componentes que se pueden reutilizar y para mostrarle al usuario una interfaz más amigable.[25]

1.6.2 Zend Framework

Es un framework para el desarrollo de aplicaciones y servicios con PHP, que brinda soluciones para construir sitios web modernos, robustos y seguros. Es código abierto y trabaja con PHP5. Entre sus principales características figuran las siguientes:

- Trabaja con MVC (Modelo Vista Controlador)
- Cuenta con módulos para manejar archivos PDF y servicios web (Amazon, Flickr, Yahoo) entre otros
- El marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple acceder a la suya, sin tener que escribir ninguna consulta SQL

- Una solución para el acceso a base de datos que balancea el ORM[26] con eficiencia y simplicidad
- Completa documentación y pruebas de alta calidad
- Robustas clases para autenticación y filtrado de entrada[26]

1.6.3 ZendExt Framework

Es elaborado a partir de Zend Framework cumpliendo con todas sus características. Posee un controlador vertical para el control de las acciones realizadas por las vistas, un motor de reglas para las validaciones en el servidor, además se le incluyó el IoC[27] para la comunicación entre los módulos o componentes, la integración con el ORM Doctrine Framework para trabajo en la capa de abstracción a base de datos, el ExtJS Framework para el desarrollo de las vistas y un controlador de trazas para controlar las acciones del sistema (acción, excepciones, rendimiento, integración y excepción de integración).[26]

1.6.4 Doctrine

Doctrine es un potente y completo sistema ORM para PHP con un DBAL[28] incorporado que permite exportar una base de datos a sus clases correspondientes y viceversa, o sea, a partir de las clases creadas y siguiendo las especificaciones de ORM, generar las tablas de la base de datos. Se encuentra en la parte superior de la Capa de Abstracción de Base de Datos. Una de sus principales características es la opción de escribir las consultas de base de datos en un objeto con una propiedad orientada al dialecto SQL conocida como DQL[29], inspirada en Hibernate HQL. Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación innecesaria de código.[27]

[26] Siglas de Object Relational Mapping.

[27] Siglas de Inversion of Control

[28] Siglas en inglés de capa de abstracción de bases de datos.

[29] Siglas de Doctrine Query Language.

1.7 Herramientas y Tecnologías de desarrollo definidas por CEIGE

1.7.1 Apache

Apache es el servidor web por excelencia, su facilidad de configuración, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Se ejecuta en gran cantidad de sistemas operativos, lo que lo hace prácticamente universal. Es una tecnología gratuita, de código abierto y altamente configurable, de diseño modular por lo que resulta muy sencillo ampliar sus capacidades. Actualmente existen muchos módulos para Apache que son adaptables al mismo, y están disponibles para su instalación cuando sean necesarios. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor y es posible configurarlo para que ejecute un determinado script cuando esto suceda.[28]

1.7.2 Visual Paradigm

Aunque es un software propietario; la UCI posee una licencia para el trabajo con dicha herramienta, ya que es una de las más potentes que existen en la actualidad, es multiplataforma y orientado a BPM; siendo BPMN el lenguaje de modelado que se empleará.

Visual Paradigm proporciona soporte al modelado visual con UML[30] 6.0, ofreciendo distintas perspectivas del sistema independiente de la plataforma y dotada de una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un software, así como garantizar la calidad del producto final.

Posee entre sus principales características la capacidad de crear un amplio conjunto de artefactos, utilizados con frecuencia durante la confección de software. Posibilita generar código a partir de los diagramas, para plataformas como .Net, Java y PHP.[29]

1.7.3 PostgreSQL

PostgreSQL es un servidor de base de datos relacional, libre. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados y almacenamiento de objetos de gran tamaño. Se destaca en ejecutar consultas complejas, consultas sobre vistas, sub-consultas y joins de gran tamaño.

Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta de software libre PostgreSQL tiene entre otras ventajas las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y algo muy

[30] Siglas de Unified Modeling Language.

importante es que dicha herramienta es multiplataforma. Fue diseñado para ambientes de alto volumen.[27]

1.7.4 PGAdmin III

Es una aplicación gráfica usada para la gestión de PostgreSQL, siendo la más completa y popular con licencia Open Source[31]. PGAdmin está escrito en C++ y utiliza la librería gráfica multiplataforma wxWidgets[32], permitiendo que se pueda usar en sistemas operativos como: GNU/Linux[33], MacOS y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3, ejecutándose en cualquier plataforma. PGAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente su administración.

1.7.5 Mozilla Firefox

Mozilla Firefox es un navegador bastante ágil, que está en renovación constante. Tiene la capacidad de ser modificado totalmente a gusto del usuario y según las necesidades del mismo. Esto se consigue gracias a la multitud de "extensiones" existentes, que permiten añadirle nuevas funciones de todo tipo.[30]

1.7.6 Subversión

Es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS[34], producto a las numerosas deficiencias de este último. Es un software libre, bajo una licencia de tipo Apache/BSD y se le conoce también como SVN, por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de este, es que los archivos versionados no tienen cada uno un número de revisión independiente. Las principales ventajas que provee el uso de esta herramienta son:

[31] Open Source: Código abierto, es el término con el que se conoce al software distribuido y desarrollado libremente.

[32] Bibliotecas multiplataforma y libres, para el desarrollo de interfaces gráficas programadas en lenguaje C++. Están publicadas bajo una licencia LGPL, similar a la GPL con la excepción de que el código binario producido por el usuario a partir de ellas, puede ser propietario, permitiendo desarrollar aplicaciones empresariales sin coste.

[33] GNU/Linux: Es el término empleado para referirse al sistema operativo Unix-like que utiliza como base las herramientas de sistema de GNU y el núcleo Linux.

[34] Siglas de Concurrent Versions System.

- La creación de ramas y etiquetas es una operación muy eficiente. Tiene costo de complejidad constante (O (1))
- Maneja eficientemente archivos binarios
- Los programas asociados a Subversión que se ejecutan por línea de comandos, pueden ejecutarse tanto en plataformas Unix, Linux, Solaris o Microsoft Windows[31]

1.8 Lenguajes de desarrollo y de modelado

Lenguaje de modelado

Se conoce como lenguaje de modelado de objetos a un conjunto estandarizado de símbolos y de formas de disponerlos para modelar un diseño de software o parte de este.

1.8.1 Lenguaje Unificado de Modelación (UML)

Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Es probablemente, una de las innovaciones conceptuales en el mundo tecnológico del desarrollo de software que más expectativas y entusiasmo han generado en muchos años. Es un estándar en la industria del software, creado por Grady Booch, James Rumbaugh e Ivar Jacobson.

UML también intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos se crea una documentación que cualquier desarrollador con conocimientos de UML será capaz de entender. Su utilización es independiente del lenguaje de programación y de las características de los proyectos ya que ha sido diseñado para modelar cualquier tipo de proyecto.[32]

Lenguaje del lado del servidor

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

1.8.2 PHP

PHP es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Es conocido como una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz, aplicaciones web dirigidas a bases de datos. Es un potente lenguaje de secuencia de comandos

diseñado específicamente para permitir a los programadores crear aplicaciones en la web con distintas prestaciones de forma rápida. No requiere definición de tipos de variables, no es un lenguaje de marcas.
Su interpretación y ejecución se realiza en el servidor en el cual se encuentra almacenada la página y el cliente sólo recibe el resultado de la ejecución. Permite la conexión a numerosas bases de datos de forma nativa tales como Postgres, MySQL, Oracle, ODBC, Microsoft SQL Server, entre otras, lo cual posibilita la creación de aplicaciones web robustas.
PHP tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos como por ejemplo UNIX, Linux, Windows y Mac OS X y puede interactuar con distintos los servidores web.

Ventajas:

- Es un lenguaje multiplataforma
- Capacidad de expandir su potencial utilizando gran cantidad de módulos
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda
- Es libre, lo que representa que una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente
- Permite las técnicas de Programación Orientada a Objetos[35]
- Posee una biblioteca nativa de funciones sumamente amplia e incluida

Desventajas:

- No posee una abstracción de base de datos estándar, sino bibliotecas especializadas
- Por sus características promueve la creación de código desordenado y complejo de mantener
- Todo el trabajo lo realiza el servidor, por tanto puede ser más ineficiente a medida que aumenten las solicitudes
- La orientación a objetos es aún muy deficiente en aplicaciones grandes[33]

Lenguaje del lado del cliente

[35] La terminología Programación Orientada a Objetos (POO u OOP según siglas en inglés) representa un paradigma de programación donde se definen los programas en términos de "clases de objetos", tales objetos son entidades que combinan estado (datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto).

Un lenguaje del lado del cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. El código, tanto del hipertexto como de los scripts, es accesible por cualquier usuario y ello puede afectar a la seguridad.

1.8.3 JavaScript

JavaScript es un lenguaje de scripting basado en objetos, que se utiliza principalmente para crear páginas web dinámicas y permite el desarrollo de interfaces de usuario mejoradas. Una página web dinámica es aquella que permite la interacción entre el contenido de la misma y el usuario. JavaScript permite incorporar a dichas páginas efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.[34]

1.9 Conclusiones parciales

Una vez analizado el ambiente de desarrollo, quedan establecidos los parámetros a tener en cuenta para la selección del motor para la ejecución de flujos de trabajo sobre el marco de trabajo Sauxe, dicho motor debe:

- Estar disponible bajo licencia libre, que permita su adquisición y modificación de forma gratuita
- Estar implementado en un lenguaje compatible con los utilizados en el marco de trabajo
- Usar un lenguaje de ejecución distinto de BPEL o a algún otro que requiera de servicios web
- Permitir su integración al marco de trabajo siguiendo el diseño arquitectónico de Sauxe
-

- **Tabla 1: Resultado del estudio en cuanto a los parámetros definidos**

Herramienta	Libre	Propietario	Lenguaje	Otros
JBoss jBPM	X	-	Java	Necesita de una Maquina Virtual
Bonita	X	-	Java	Necesita de una Maquina Virtual
OpenWFE	X	-	Java o Ruby	-
SAFE	-	X	?	-

Sap Netweaver	-	X	ABAP	Implementado en ABAP
Enhydra Shark	X	-	Java	Depende de otras librerías
Microsoft Dynamics	X(relativo)	-	.Net	-
ezComponents	X	-	PHP	-

De los sistemas y plataformas estudiados, el motor de flujos de trabajo del marco de trabajo ezComponents es el que mejor se ajusta a las especificaciones del ambiente de desarrollo de acuerdo con los parámetros descritos; dicho motor se encuentra bajo una licencia libre, permitiendo acceder y modificar su código, está implementado totalmente en PHP, no utiliza servicios web durante la ejecución de los flujos de trabajo y tiene funcionamiento independiente de cualquiera de los demás componentes de la librería ezComponents.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

2.1 Introducción

A continuación se describe la propuesta de solución. Se realiza la descripción de los procesos que la componen mediante un modelo de dominio abarcando los conceptos fundamentales y sus relaciones. Asimismo se trataran los requisitos capturados, el diagrama que contiene la estructura de las clases y la relación que existe entre estas, así como los patrones de diseño identificados.

2.2 Modelo conceptual

Con la elaboración de un modelo conceptual se obtiene un mejor entendimiento del dominio del problema, ya que expresa de manera visual aquellos objetos o conceptos reales que son significativos para el problema, no representa componentes de software, sino clases conceptuales y las relaciones entre ellas. Para la realización de este modelo conceptual fue necesario identificar conceptos y asociaciones que se consideren importantes y así obtener un desglose del dominio del problema.[35]

En el diagrama se muestra gráficamente como el motor de ejecución del ezComponents recibe información referente a un flujo de trabajo de un modelo de datos para ejecutarlo e interactuar con el marco de trabajo Sauxe, haciendo uso principalmente de las actividades automáticas y semiautomáticas para la ejecución de tareas y de una lista de trabajo para mantener un control sobre las actividades pendientes para cada usuario.

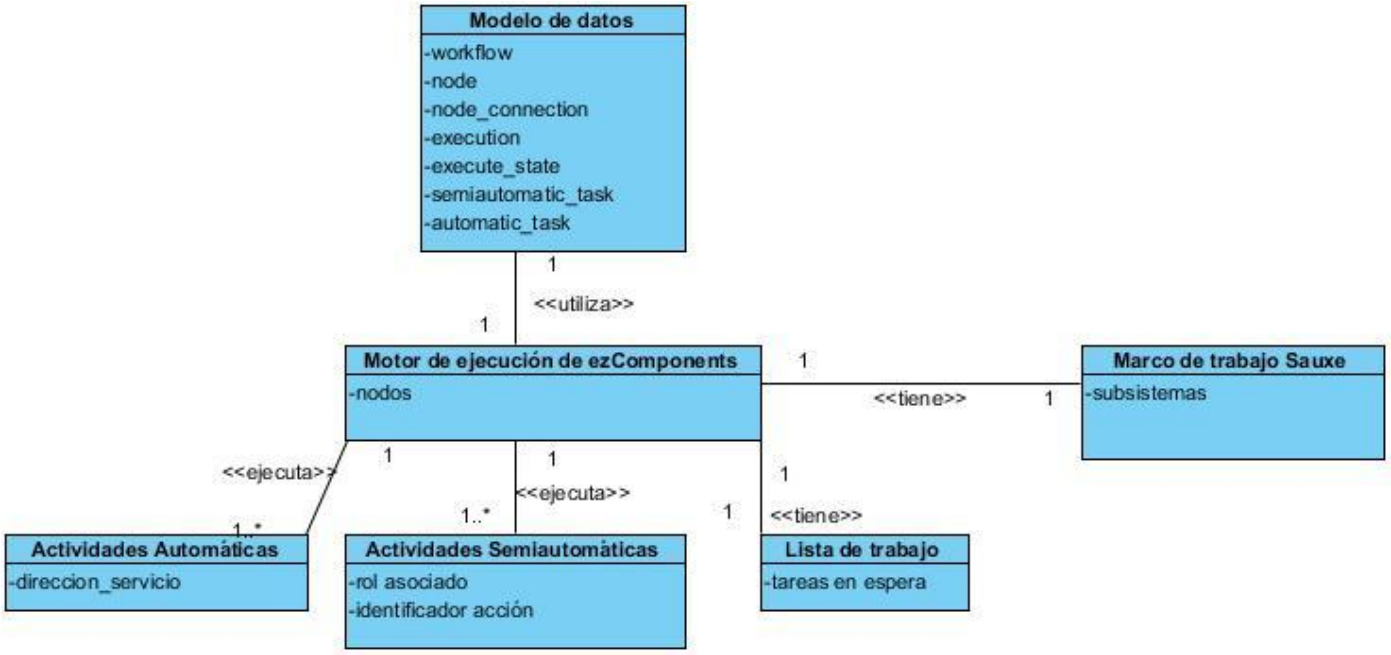

Figura 1: Modelo Conceptual

2.3 Requisitos de software

Los requisitos cumplen un papel esencial en el proceso de producción de software, ya que se enfocan en un área fundamental: la definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento del sistema; de esta manera, se pretende minimizar los problemas relacionados al desarrollo de sistemas.

Los requisitos de software se clasifican en funcionales y no funcionales. Los requisitos funcionales describen qué es lo que el sistema debe hacer para dar soporte a las funciones y objetivos del usuario. Los requisitos no funcionales imponen restricciones de cómo los requisitos funcionales deben ser implementados.[36]

2.3.1 Técnicas empleadas en la captura de requisitos funcionales

En el proceso de desarrollo de software cuando se realiza la actividad captura de los requisitos el equipo de desarrollo extrae de diversas fuentes de información que estén disponibles, las necesidades que debe cubrir el sistema que se va a desarrollar. Por la complejidad que puede tener este proceso, se han creado técnicas que permitan realizarlo de una forma eficiente y precisa, la mejor forma de poner estas técnicas

en práctica consiste en una interacción entre los clientes y el equipo de desarrollo para la comprensión del problema, la proposición de soluciones, negociación de diferentes perspectivas y especificación de un conjunto básico de requisitos de la solución. De las técnicas existentes se utilizó:

- ✓ **Tormenta de ideas**: nombre con el que se le conoce al conjunto de reuniones realizadas con el objetivo de generar ideas, concibiendo la mayor cantidad de requisitos que puede tener el sistema. En estas reuniones participan integrantes de varias disciplinas, siempre tratando de que estos tengan un nivel aceptado de experiencia, ninguna idea es descabellada ni debe ser criticada o evaluada, evitando en todo momento la creación de un ambiente hostil[37].

2.3.2 Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, se mantienen invariables sin importar con qué propiedades o cualidades se relacionen[38]. Los requisitos funcionales identificados se enuncian a continuación:

- ✓ RF1 Integrar el componente de flujos de trabajo del ezComponents con el marco de trabajo Sauxe
- ✓ RF2 Ejecutar tareas automáticas
- ✓ RF3 Ejecutar tareas semiautomáticas
- ✓ RF4 Conocer el estado de las instancias de los flujos de trabajo

Especificación de requisitos

Descripción del RF1:

Tabla 2: Integrar el componente de flujos de trabajo del ezComponents con el marco de trabajo Sauxe

Precondiciones	La versión de la librería Zend debe ser superior a 1.8.
Flujo de eventos	
Flujo básico Integrar ezComponents con el Marco de Trabajo Sauxe	
1.	Consultar el index.php del subsistema Portal
2.	Modificar la función autoload de la librería Zend añadiéndole la función autoload del ezComponents
3.	Invocar la función autoload
Pos-condiciones	
1.	El Marco de Trabajo Sauxe reconoce las clases de tipo ezComponents
2.	
Flujos alternatives	
Flujo alternativo	

1	N/A
Pos-condiciones	
1	N/A
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual: 0127_Modelo conceptual Componente de ejecución

Conceptos	N/A	Utilizados internamente:
Requisitos especiales		
Asuntos pendientes	N/A	

Descripción del RF2:

Tabla 3: Ejecutar tareas automáticas

Precondiciones	La versión de la librería Zend debe ser superior a 1.8. El nodo anterior se haya ejecutado. El flujo de trabajo debe estar activo.
Flujo de eventos	
Flujo básico Ejecutar tareas automáticas	
1	Capturar la dirección del servicio que le corresponde a la actividad
2	Llamar al servicio que le corresponde a la actividad
3	Cambiar estado de la actividad
Pos-condiciones	
1	Se ejecutó la tarea automática
Flujos alternatives	
N/A	
Pos-condiciones	
1	N/A
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual: 0127_Modelo conceptual Componente de ejecución

Conceptos	N/A Tareas automáticas	Utilizados internamente: Estado Dirección del servicio
Requisitos especiales	Librería Zend, PostgreSQL, Doctrine	
Asuntos pendientes	N/A	

Descripción del RF3:

Tabla 4: Ejecutar tareas semiautomáticas

Precondiciones	La versión de la librería Zend debe ser superior a 1.8. El nodo anterior se haya ejecutado. El flujo de trabajo debe estar activo.
Flujo de eventos	
Flujo básico Ejecutar tareas semiautomáticas	
1	El usuario hace la petición de la acción que quiere realizar
2	Verificar si el usuario tiene permisos para realizar la acción solicitada
3	Verificar si la acción solicitada está en espera
4	Ejecutar la acción
5	Cambiar el estado de la tarea semiautomática
6	Actualizar la lista de trabajo
Pos-condiciones	
1	El Marco de Trabajo Sauxe reconoce las clases de tipo ezComponents
2	
Flujos alternativos	
Flujo alternativo 2. a El usuario no tiene permisos	
1	Se deniega la petición
Flujo alternativo 3. a La acción no está en espera	
1	Se deniega la petición
Flujo alternativo 4. a La acción no se ejecuta correctamente	
1	Lanza excepción de error
2	Regresa al flujo básico, paso 1.
Pos-condiciones	
1	Se actualizó la lista de trabajo
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual:

	0127_Modelo conceptual Componente de ejecución	
Conceptos	N/A Tareas semiautomáticas	Utilizados internamente: Estado IdRol IdWorkflow IdNodo IdAction
Requisitos especiales	Librería Zend, PostgreSQL, Doctrine	
Asuntos pendientes	N/A	

Descripción del RF4:

Tabla 5: Conocer el estado de las instancias de los flujos de trabajo

Precondiciones	La versión de la librería Zend debe ser superior a 1.8. El flujo de trabajo debe estar activo.	
Flujo de eventos		
Flujo básico Conocer estado de las tareas semiautomáticas		
1	Consultar la lista de trabajo	
2	Buscar la actividad que se desea en la lista de trabajo	
3	Consultar el estado de la actividad	
Pos-condiciones		
1	Se conoce el estado de la actividad	
Flujos alternativos		
N/A		
Pos-condiciones		
1	N/A	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual: 0127_Modelo conceptual Componente de ejecución	
Conceptos	N/A Tareas semiautomáticas	Utilizados internamente: Estado
Requisitos especiales	Librería Zend, PostgreSQL, Doctrine	
Asuntos pendientes	N/A	

2.3.3 Requisitos no funcionales

La solución propuesta fue iniciada en el proyecto paquete de herramientas, el cual está desarrollado sobre el marco de trabajo Sauxe. De esta manera los requisitos no funcionales a los que se debe acoger el componente a desarrollar son los establecidos al inicio del proceso de desarrollo. A continuación son descritos algunos de estos requisitos.

Software

Para el cliente:

- ✓ Navegador Mozilla Firefox 3.0 o superior
- ✓ Sistema operativo Windows 98 o superior o Linux

Para el servidor

- ✓ Sistema operativo Linux en cualquiera de sus distribuciones
- ✓ Un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible. Este debe estar configurado con la extensión “pgsql” incluida
- ✓ Un servidor de base de datos PostgreSQL 8.3

Rendimiento

Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 5 segundos para las actualizaciones y 20 para las recuperaciones.

Seguridad

Autenticación y Autorización (Contraseña de acceso). Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos. La atención al sistema incluyendo el mantenimiento de las bases de datos, así como la salva de la información, se realizará de forma centralizada por el administrador.

Hardware

Para el servidor:

- ✓ Requerimientos mínimos: Procesador Pentium IV a 2GHz de velocidad de procesamiento y 1Gb de memoria RAM
- ✓ Al menos 40Gb de espacio libre en disco duro
- ✓ Tarjeta de red

Para el cliente:

- ✓ Requerimientos mínimos: Procesador Pentium III a 1GHz con 256Mb de memoria RAM
- ✓ Tarjeta de red.[39]

2.3.4 Validación de los requisitos funcionales

Los requisitos una vez definidos necesitan ser validados. Dicha validación tiene como objetivo demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos, para ello se evalúa su calidad en la fase de validación. La misma examina las especificaciones para asegurar que todos los requisitos del sistema que han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos. Una especificación es considerada con calidad por el estándar IEEE 83 cuando es correcta, no ambigua, completa, consistente, ordenada por importancia y estabilidad, verificable, modificable y trazable [38].

Existen algunas técnicas que pueden aplicarse para ello, por ejemplo: Reviews o Walk-throughs, Auditorías, Matrices de Trazabilidad, Revisión técnica formal y la utilización de Prototipos. Para validar los requisitos se utilizó una técnica:

- ✓ **Revisión técnica formal:** es una actividad que garantiza la calidad, su objetivo primario es encontrar errores durante el proceso, incluye recorridos, inspecciones y revisiones cíclicas. Cada Revisión Técnica Formal (RTF) se lleva a cabo mediante una reunión y solo tiene éxito si esta es bien planificada, controlada y atendida. Una vez terminadas las especificaciones de requisitos se efectuó una reunión para revisarlos con el cliente[40]

Se aplicó la técnica de RTF**,** donde una vez realizado el levantamiento de los requisitos funcionales, se realizó la validación de los mismos.

2.4 Diagrama de clases

Según la clasificación UML, un diagrama de clases es una estructura estática que representa los requerimientos a través de clases del sistema y las interrelaciones entre ellas. Estos diagramas representan una abstracción del dominio, constituyendo un elemento básico del modelado, a la vez que muestra de manera general, exactamente qué debe hacer el sistema.

En el diagrama se muestra la forma en que el componente obtiene datos de los flujos de trabajo de la base de datos mediante clases de acceso a datos y pasa a la ejecución del flujo de trabajo accediendo a clases del ezComponents implementadas para ese fin, de forma tal que es posible realizar tareas automáticas y semiautomáticas, interactuando con el marco de trabajo y guardando los resultados de las ejecuciones y los estados de las tareas en una base de datos.

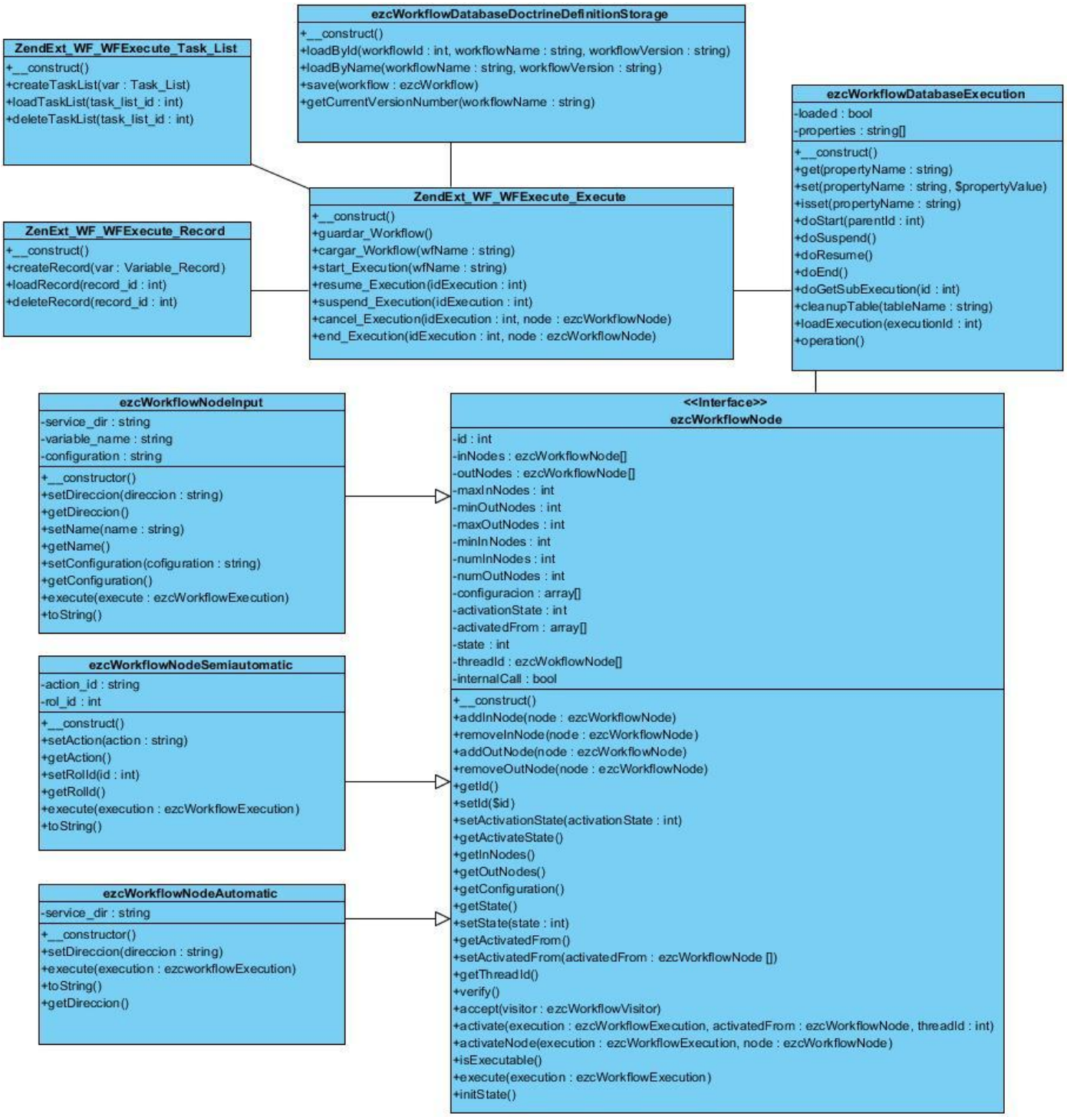

Figura 2: Diagrama de clases

2.5 Modelo de Datos

Un modelo de datos es un lenguaje encaminado a describir las estructuras de datos, las restricciones de integridad y las operaciones de manipulación de los datos; orientado a resolver el problema y que describa los elementos de la realidad que intervienen en el mismo.[41]

El modelo propuesto consta de 10 tablas, para su elaboración se tuvo en cuenta la reducción a la mínima expresión de los campos nulos.

Teniendo en cuenta el requisito funcional “Ejecutar tareas semiautomáticas” se crea la tabla semiautomatic_task, la misma contiene los campos node_id, action_id, rol_id y hereda de la tabla node.

Teniendo en cuenta el requisito funcional “Ejecutar tareas automáticas” se crea la tabla automatic_task, la misma contiene los campos node_id, service_dir y hereda de la tabla node.

Teniendo en cuenta el requisito funcional “Conocer el estado de las instancias de los flujos de trabajo” se crea la tabla task_list, la misma contiene los campos, action_name y rol_id y se encuentra relacionada con la tabla node y workflow.

Entre las principales tablas del modelo se encuentra workflow que es la encargada de guardar la información referente flujo de trabajo, conteniendo la versión y el nombre de este y está relacionada con las tablas node, execution, task_list y variable_record. La tabla node contiene el identificador, la clase del nodo, y la configuración, datos que son manejados por ezComponents, por su parte la tabla execution contiene los datos de la ejecución y a su vez está relacionada con la tabla execution_state que almacena la información relacionada con el estado de la ejecución.

A continuación se muestra el diseño lógico inicial de la base de datos.

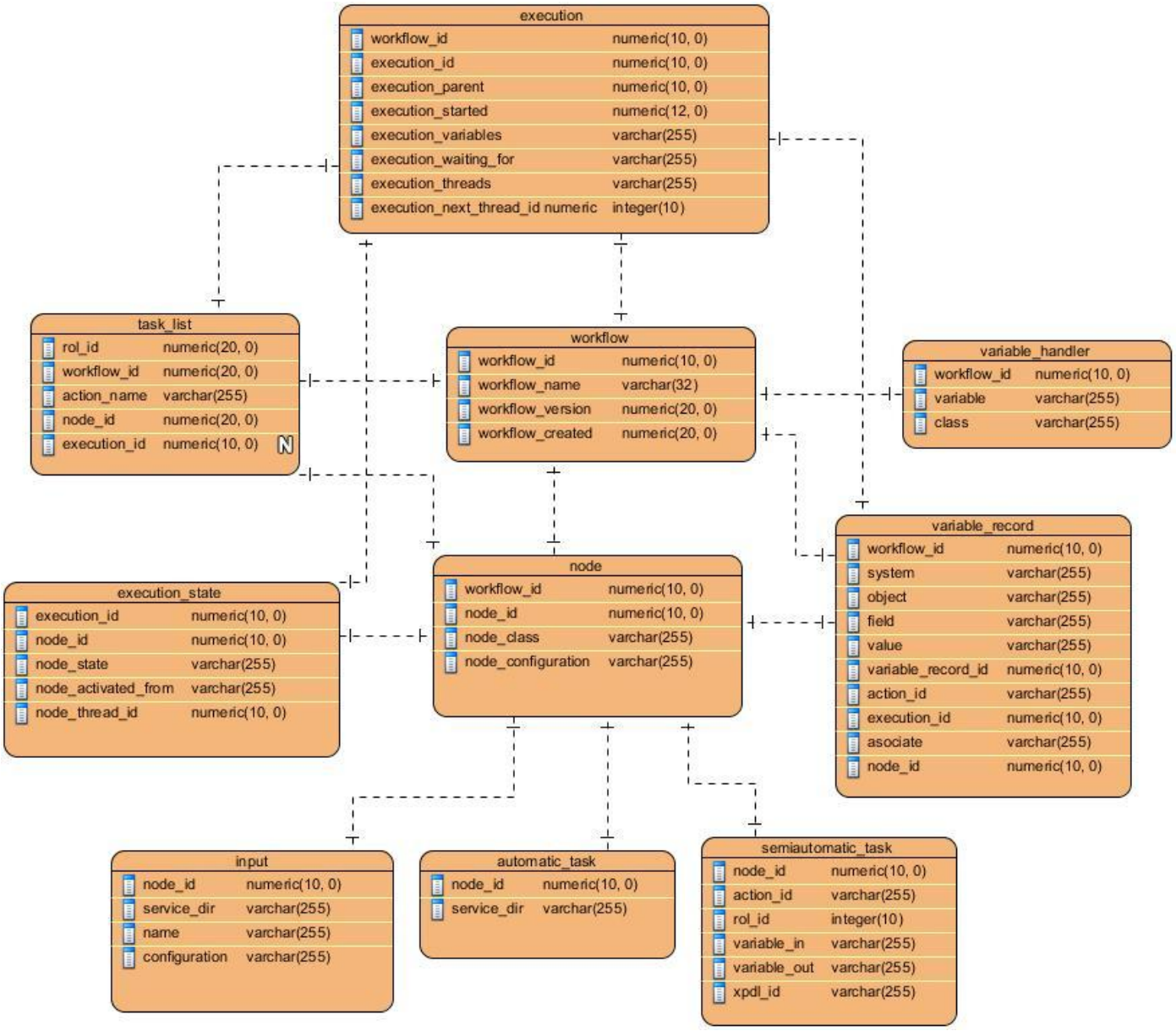

Figura 3: Diagrama del Modelo de datos

2.6 Patrones de Diseño empleados

Los patrones de diseño son una solución estándar para un problema común de diseño dentro de un contexto dado y constituyen una manera más práctica de describir ciertos aspectos de la organización de un programa.

Estos patrones se emplearon por su capacidad de brindarle al diseño flexibilidad y extensibilidad. A continuación se explica brevemente en qué consisten y cómo se utilizaron:

Patrones GRASP

Son los patrones generales de software para asignar responsabilidades, GRASP[36], describen en su totalidad los principios fundamentales sobre la asignación de responsabilidades a objetos, todo en forma de patrones.

- **Experto:** en la definición de las clases se evidencia que a partir de la información que se maneja las funcionalidades que se deben realizar, un ejemplo la clase *Execute* la cual es responsable de la gestión de la ejecución de los flujos de trabajo, y a su vez delegando funcionalidades a los encargados de la información requerida para llevar a cabo esta acción[35]
- **Creador:** este patrón se evidencia en las clases encargadas de crear objetos, por ejemplo la clase *ezcWorkflowDatabaseExecution* es responsable de crear instancias de la clase *ExecutionState[35]*
- **Bajo acoplamiento:** el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. El uso de este patrón se evidencia en el caso de la relación que se establece entre la clase *ezcWorkflowDatabaseExecute* y la clase *ExecutionState*. En este caso, la clase *ezcWorkflowDatabaseExecute* no depende del *ExecuteState*, porque recibe como parámetro para las operaciones definidas un flujo de trabajo, y los cambios que ocurran en la clase *ExecuteState* no acarrean cambios en *ezcWorkflowDatabaseExecute*, y viceversa[35]
- **Alta cohesión:** este patrón fue utilizado para agrupar las clases por requerimientos, de modo que las pertenecientes a los requisitos funcionales de tipo gestionar ejecución se encargan solamente de estas acciones, no realiza otra operación, de modo que cada clase implementa las operaciones que se encuentran en su misma área funcional asegurando así que no realicen un trabajo enorme[35]
- **Polimorfismo:** cuando varía el tipo de clase o comportamientos relacionados, se asigna la responsabilidad del comportamiento mediante operaciones polimórficas, a los tipos en que varia el

[36] Por sus siglas en inglés General Responsability Asignment Software Paterns.

comportamiento. Se evidencia este patrón en el caso de las clases *Automáticas* y *Semiautomáticas* ya que varía su comportamiento[35]

2.7 Descripción de la arquitectura

La arquitectura de software es una disciplina que tiene por objetivo definir la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre estos y el entorno y los principios que orientan su diseño y evolución. El objetivo primario de la arquitectura de software es el de aportar elementos que ayuden en la toma de decisiones significativas y al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre todos los que tienen parte en el proceso de desarrollo del software. Estas decisiones significativas se toman teniendo en cuenta:

- La organización del sistema de software
- La selección de los elementos estructurales y sus interfaces a través de los cuales se constituye el sistema
- Su comportamiento, según resultados de las colaboraciones entre esos elementos
- El estilo arquitectónico que guía esta organización: los elementos estáticos y dinámicos; sus interfaces, su colaboración y su composición

2.7.1 Arquitectura en capas

La arquitectura basada en capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada.[42]

El estilo de arquitectura basado en capas se identifica por las siguientes características:

- Describe la descomposición de servicios de forma que la mayoría de la interacción ocurre solamente entre capas vecinas
- Las capas de una aplicación pueden residir en la misma máquina física (misma capa) o puede estar distribuido sobre diferentes computadores (n-capas)
- Los componentes de cada capa se comunican con otros componentes en otras capas a través de interfaces muy bien definidas
- Este modelo ha sido descrito como una “pirámide invertida de re-uso” donde cada capa agrega responsabilidad y abstracción a la capa directamente sobre ella

Los principios comunes que se aplican cuando se diseña para usar este estilo de arquitectura incluyen:

- Abstracción: la arquitectura basada en capas abstrae la vista del modelo como un todo mientras que provee suficiente detalle para entender las relaciones entre capas
- Encapsulamiento: ll diseño no hace asunciones acerca de tipos de datos, métodos, propiedades o implementación
- Funcionalidad claramente definida: el diseño claramente define la separación entre la funcionalidad de cada capa. Capas superiores como la capa de presentación envía comandos a las capas inferiores como la capa de negocios y la capa de datos y los datos fluyen hacia y desde las capas en cualquier sentido
- Alta cohesión: cada capa contiene funcionalidad directamente relaciona da con la tarea de dicha capa
- Reutilizable: las capas inferiores no tienen ninguna dependencia con las capas superiores, permitiéndoles ser reutilizables en otros escenarios

La descomposición más habitual es dividir la aplicación en tres capas:

- Capa de presentación: se ocupa de toda la interacción entre el usuario y el software, pudiendo tratarse de un sistema de menús muy simple o una interfaz gráfica de usuario relativamente compleja
- Capa del dominio: también conocida como la lógica de negocio, que es todo lo que necesita conocer la aplicación para poder trabajar con el dominio en cuestión. Implica realizar cálculos basados en datos de entrada o almacenados, validación de cualquier dato proveniente de la capa de presentación y la ejecución de algoritmos específicos en función de los comandos de la presentación
- Capa de acceso a datos: se encarga de manejar el mecanismo de almacenamiento

De esta manera se desarrolla usando Zend Framework para el controlador y el Doctrine para el modelo.[42]

De las tres capas descritas anteriormente en el desarrollo del componente solo se hace uso de las capas de dominio y de acceso a datos, debido a que el mismo no requiere una interfaz visual para la ejecución de los flujos de trabajo.

2.8 Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean estos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes[43]. A continuación se describe el diagrama de componentes del componente ***WFExecute***, el cual depende del componente ***Ioc*** para la realización de las tareas automáticas y los nodos de decisión, a su vez el mismo depende de la clase ***Controller*** contenida en ***ZendExt*** para la gestión de las tareas semiautomáticas. El componente ***Portal*** es el encargado de cargar la librería ***ezComponent*** llevándose a cabo todo el proceso en el marco de trabajo ***Sauxe***, gestionando la seguridad el subsistema de seguridad ***Acaxia***.

El componente ***WFExecute*** depende también del ***EZSEL*** encargado de traducir proceso modelado que se encuentra en un fichero XPDL al lenguaje de ejecución que conoce ***ezComponent*** y persistirlo en la base de datos para su ejecución.

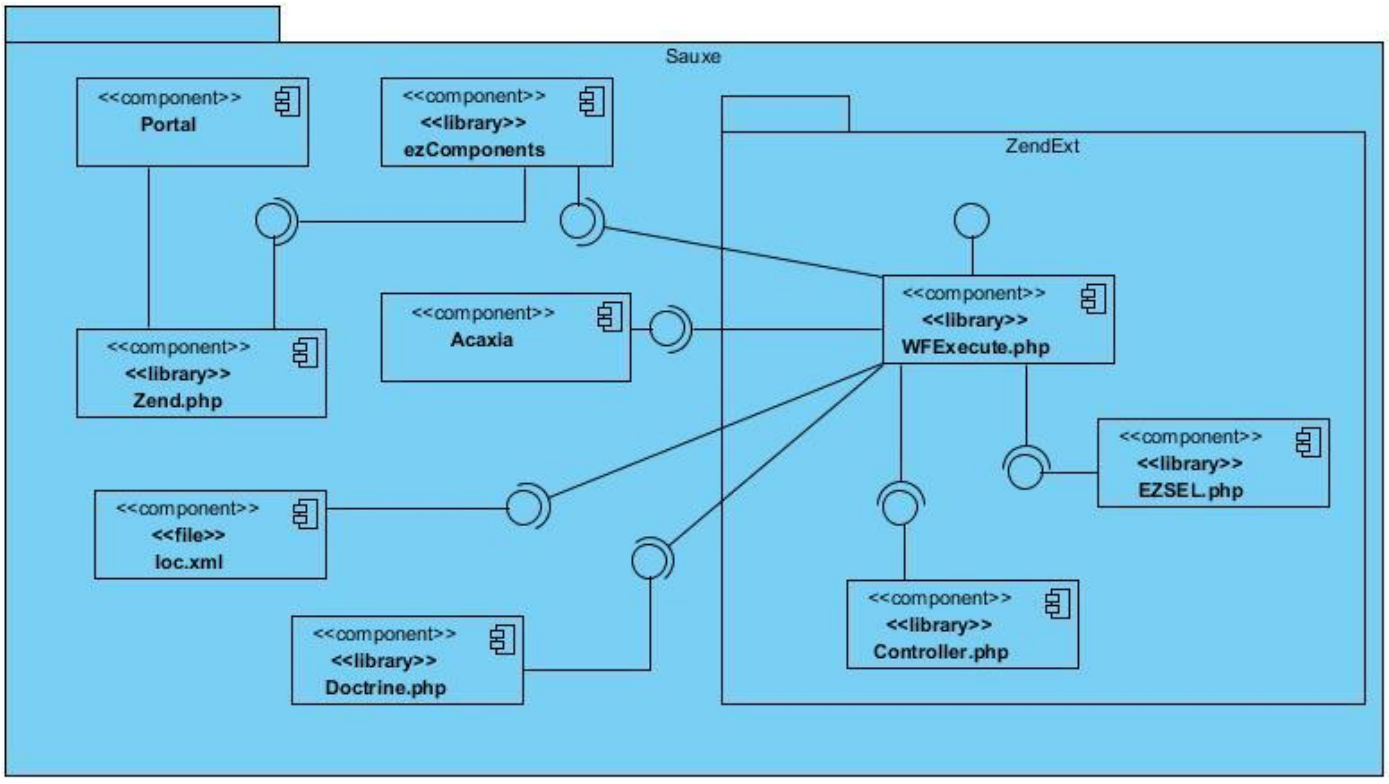

Figura 4: Diagrama de Componentes

2.9 Conclusiones Parciales

Al concluir este capítulo, quedaron definidos el análisis y el diseño de la solución propuesta, cumpliendo con la arquitectura que posee el CEIGE. El diseño de las clases y las funcionalidades de las mismas, responden al modelo de desarrollo propuesto y deberán implementar los patrones necesarios para esto. El problema fue completamente analizado y la solución diseñada ya se encuentra en un estado que posibilita pasar a su implementación, de manera que se adecúe a las características de la arquitectura y el modelo de desarrollo previsto.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

En el presente capítulo se muestran los artefactos generados en la fase de implementación, guiados por el modelo de desarrollo orientado a componentes propuesto por CEIGE. También se especifica cuáles son los estándares de codificación utilizados, el conjunto de validaciones y las pruebas que evalúan la calidad del componente desarrollado.

3.2 Estándares de codificación

Se definen como estándares de codificación los estilos de programación en un proyecto determinado, permitiendo que todos los participantes del mismo puedan entenderlos en menos tiempo y que el código sea mantenido. Debido al numeroso personal involucrado en la implementación del sistema CEDRUX, su complejidad y el alto nivel de integración, el grupo arquitectónico del proyecto definió normas de codificación con el fin de obtener un estándar en la implementación por el equipo de desarrollo, que permitiera asegurar la calidad del software y de esta forma obtener un código más legible y reutilizable. A continuación se describen estos estándares empleados en la implementación del componente para la validación de procesos de negocios modelados con BPMN en el Marco de trabajo Sauxe.

3.2.1 PascalCasing

El estándar PascalCasing establece que los identificadores, nombres de clases, variables, métodos o funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula. La nomenclatura de las clases del componente para la ejecución de los flujos de trabajos modelados con BPMN se realizó sobre la base de este estándar, usando palabras compuestas sugerentes acordes al propósito de la misma[44].

Nomenclatura de clases según su tipo

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

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

- **Clases de los modelos**

Business: clases modelo del negocio

Las clases modelo tendrán por identificador el nombre de la tabla en la que trabajan seguido por la palabra Model y heredarán de la super clase del framework Zend_Ext llamada ZendExt_Model. Ejemplo: **AutomaticTaskModel**.

3.2.2 CamelCasing

El estándar CamelCasing es parecido al PascalCasing con la particularidad de que la letra inicial del identificador no comienza con mayúscula. Esta notación se utilizó para el nombre de funciones y atributos[44].

- **Nomenclatura de las funciones**

El identificativo a emplear para las funciones o métodos se escribe con la primera palabra en minúscula utilizando la notación CamelCasing y nombres que deduzcan su propósito. Ejemplo: **cargarWorkflow**. Los denominadores de las acciones de las clases controladoras tienen la peculiaridad de ir seguidos por la un palabra “Execution” identificando la acción. Ejemplo: **startExecution**.

- **Nomenclatura de los atributos**

Se definió que el nombre a emplear para las variables se escribe en minúscula y en caso de ser un nombre compuesto se escriben ambas palabras en minúscula separadas por un guión bajo. Ejemplo**:** **node_id y workflow_id**.

3.3 Diagrama de Despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos). Estarán formados por instancias de los componentes software que representan manifestaciones del código en tiempo de ejecución.

Figura 5: Diagrama de Despliegue

3.4 Métricas de software

En [40], Pressman plantea: “El proceso del software y las métricas del producto son una medida cuantitativa que permite a la gente del software tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software”.

Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

- **Responsabilidad**: se le asigna a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta
- **Complejidad de implementación**: grado de dificultad en la implementación de un diseño de clases determinado
- **Reutilización**: nivel de reutilización que tiene una clase o estructura de clase, dentro de un diseño de software determinado
- **Acoplamiento**: valor de dependencia de una clase o estructura de clase con otras. Este atributo está muy ligado al de Reutilización
- **Complejidad del mantenimiento**: categoría de esfuerzo para realizar un arreglo, mejora o rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto
- **Cantidad de pruebas**: número de esfuerzos para realizar las pruebas de calidad (Unidad) del producto (componente, clase, conjunto de clases, etcétera) diseñado

Debido a que este software se realizó bajo la POO y las clases constituyen la unidad básica y fundamental de un sistema orientado a objetos, es indiscutible que la validación del mismo se centró en la aplicación de métricas dirigidas a sus clases de forma individual, sus jerarquías y colaboraciones. A continuación se encuentran desarrolladas dichas métricas:

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

Tabla 6: Tamaño operacional de clase (TOC)

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

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

Tabla 7: Rango de valores para los criterios de evaluación de la métrica Tamaño Operacional de Clase (TOC)

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

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

Tabla 8: Atributos de calidad evaluados por la métrica RC

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase

Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase

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

Tabla 9: Criterios de evaluación para la métrica RC

Atributo	Categoría	Criterio
Acoplamiento	Ninguna	0
	Baja	1
	Media	2
	Alta	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	>2*Promedio

3.4.1 Resultados obtenidos al aplicar la métrica de Tamaño Operacional de Clase (TOC)

Al aplicar la métrica TOC se determinó que la solución está compuesta de 8 clases y 75 procedimientos, reflejados en la siguiente tabla junto a los atributos de calidad de Responsabilidad, Complejidad de Implementación, Reutilización. Se obtuvo, un promedio de procedimientos por clase de 9.4, y los siguientes datos posibilitaron categorizar los atributos por cada clase:

Tabla 10: Instrumento de evaluación de la métrica TOC

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
ZendExt_WF_WFExecute_Execute	15	Media	Media	Media
ezcWorkflowDatabaseExecution	11	Media	Media	Media
ezcWorkflowDatabaseDoctrine DefinitionStorage	4	Baja	Baja	Alta
ZendExt_WF_WFExecute_Record	8	Baja	Baja	Alta
ZendExt_WF_WFExecute_Task_List	5	Baja	Baja	Alta
ezcWorkflowNodeSemiautomatic	20	Alta	Alta	Baja
ezcWorkflowNodeAutomatic	4	Baja	Baja	Alta
ezcWorkflowNodeInput	8	Baja	Baja	Alta

De acuerdo a los resultados obtenidos anteriormente se construyeron las siguientes gráficas para un mejor entendimiento de los mismos:

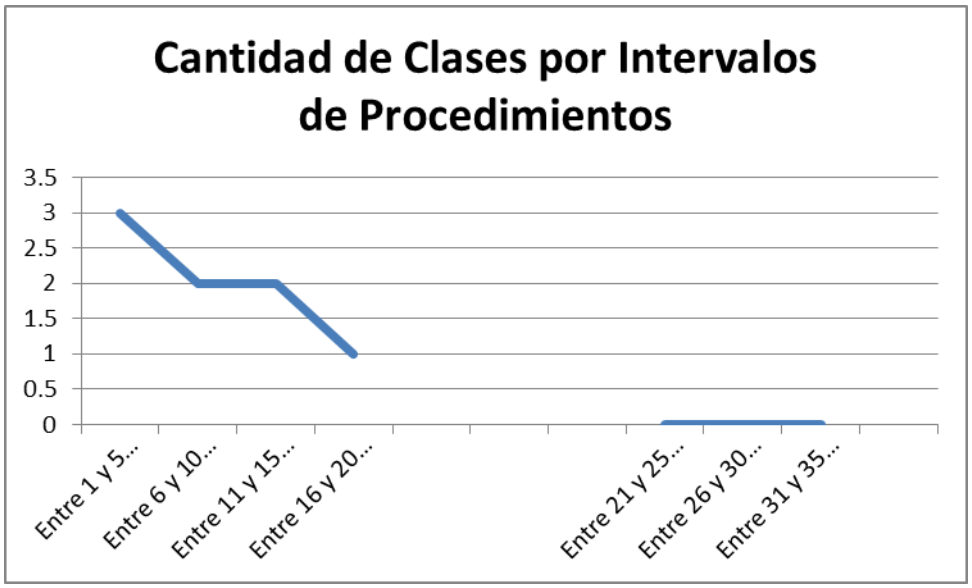

Figura 6: Cantidad de clases por intervalos de procedimientos

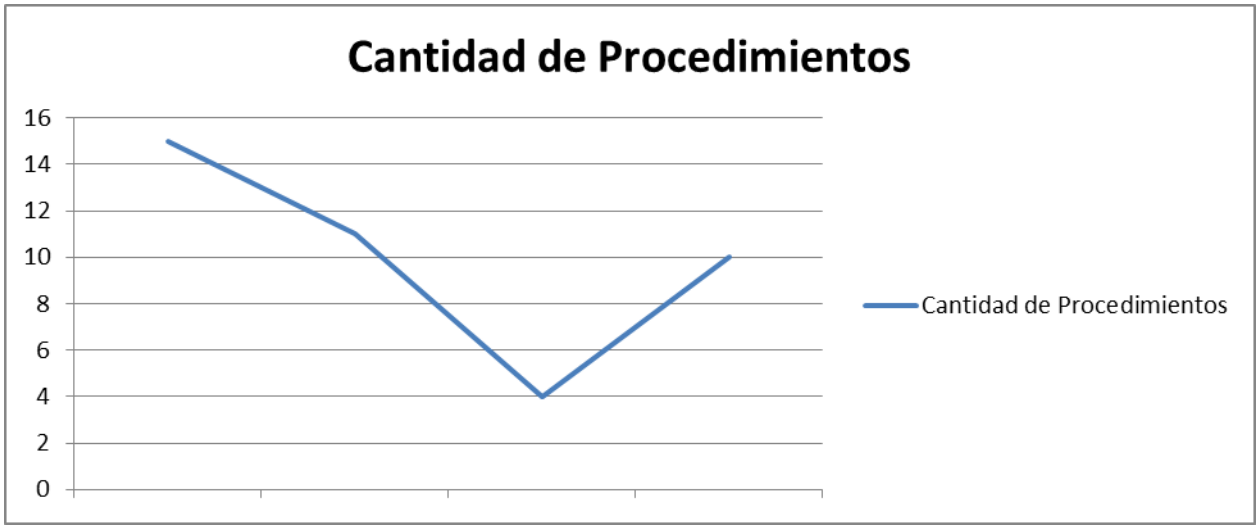

Figura 7: Cantidad de procedimientos

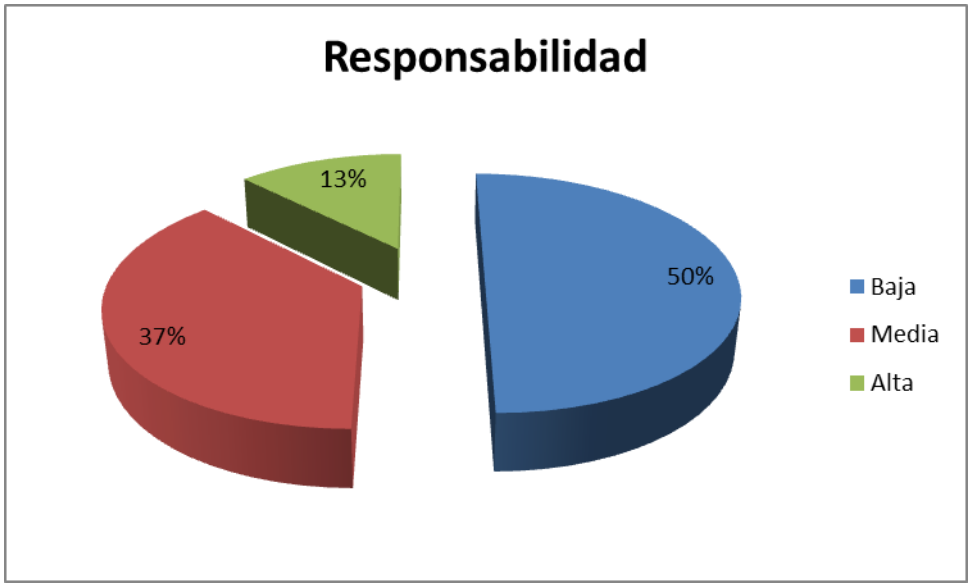

Figura 8: Resultados obtenidos de la evaluación de la métrica TOC para el atributo Responsabilidad

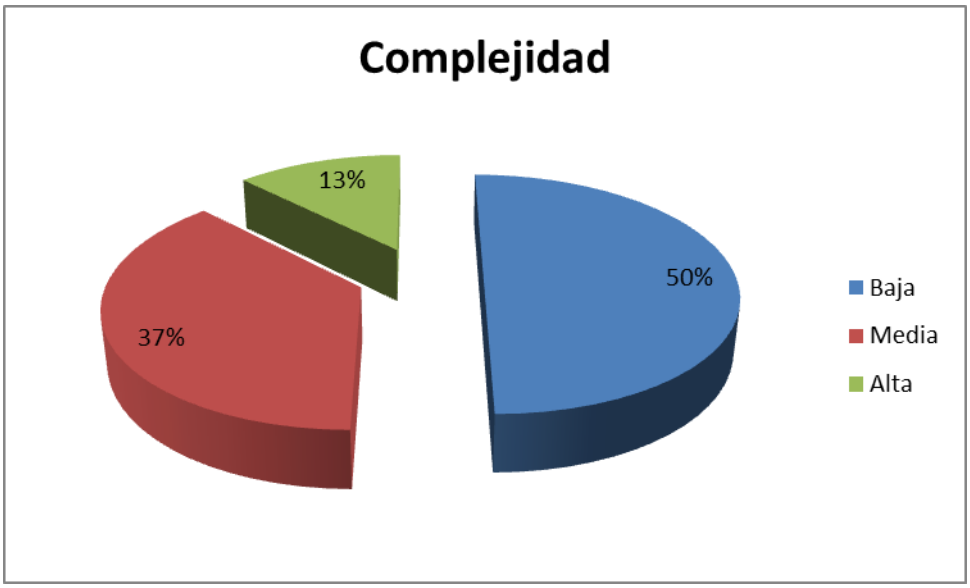

Figura 9: Resultados obtenidos de la evaluación de la métrica TOC para el atributo Complejidad

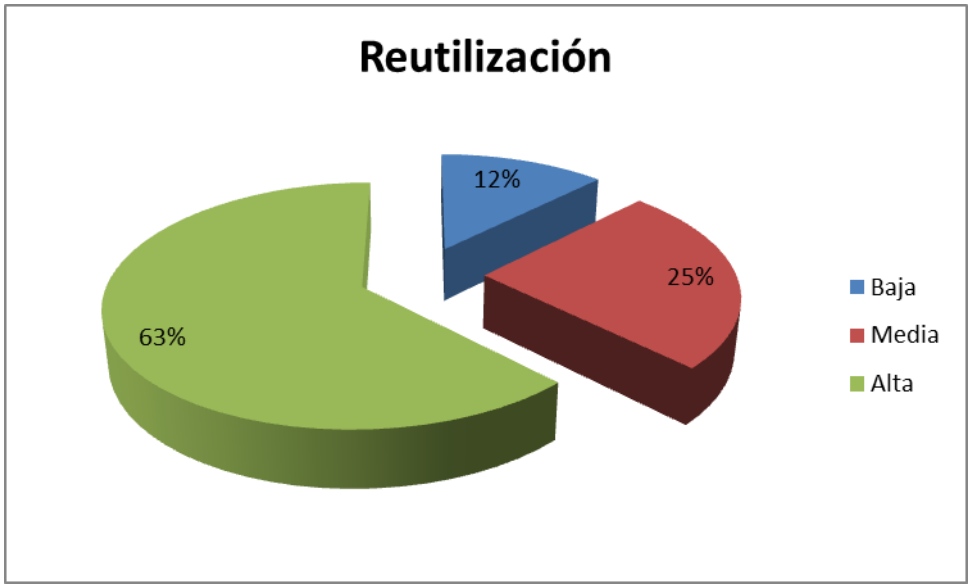

Figura 10: Resultados obtenidos de la evaluación de la métrica TOC para el atributo Reutilización

3.4.2 Resultados obtenidos al aplicar la métrica de Relaciones entre clases (RC)

Las 8 clases existentes contienen 23 dependencias entre ellas, a continuación en la siguiente tabla estarán reflejadas junto a los atributos de calidad de Acoplamiento, Complejidad de Mantenimiento,

Reutilización y Cantidad de Pruebas. Se obtuvo un promedio de relaciones de dependencia por clase de 2.9.

Tabla 11: Instrumento de evaluación de la métrica de Relaciones entre clases (RC)

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mantenimiento	Reutilización	Cantidad de Pruebas
ZendExt_WF_WFExecute_Execute	7	Alta	Alta	Baja	Alta
ezcWorkflowDatabaseExecution	2	Media	Baja	Alta	Baja
ezcWorkflowDatabaseDoctrine DefinitionStorage	6	Alta	Alta	Baja	Alta
ZendExt_WF_WFExecute_Record	1	Baja	Baja	Alta	Baja
ZendExt_WF_WFExecute_Task_List	1	Baja	Baja	Alta	Baja
ezcWorkflowNodeSemiautomatic	4	Alta	Media	Media	Media
ezcWorkflowNodeAutomatic	1	Baja	Baja	Alta	Baja
ezcWorkflowNodeInput	1	Baja	Baja	Alta	Baja

De acuerdo a los resultados obtenidos anteriormente se construyeron las siguientes gráficas para un mejor entendimiento de los mismos:

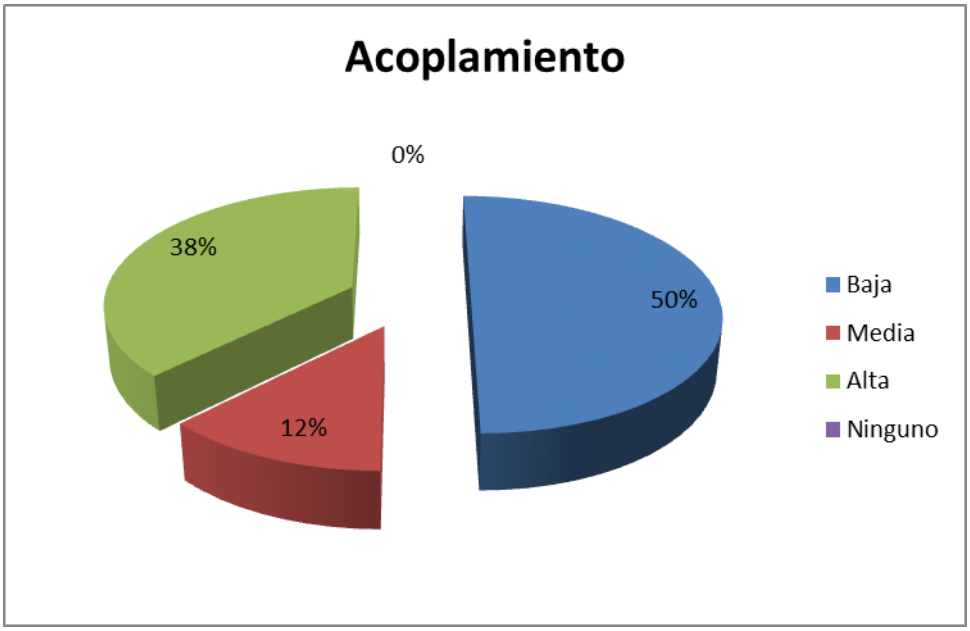

Figura 11: Resultados de la evaluación de la métrica RC para el atributo Acoplamiento

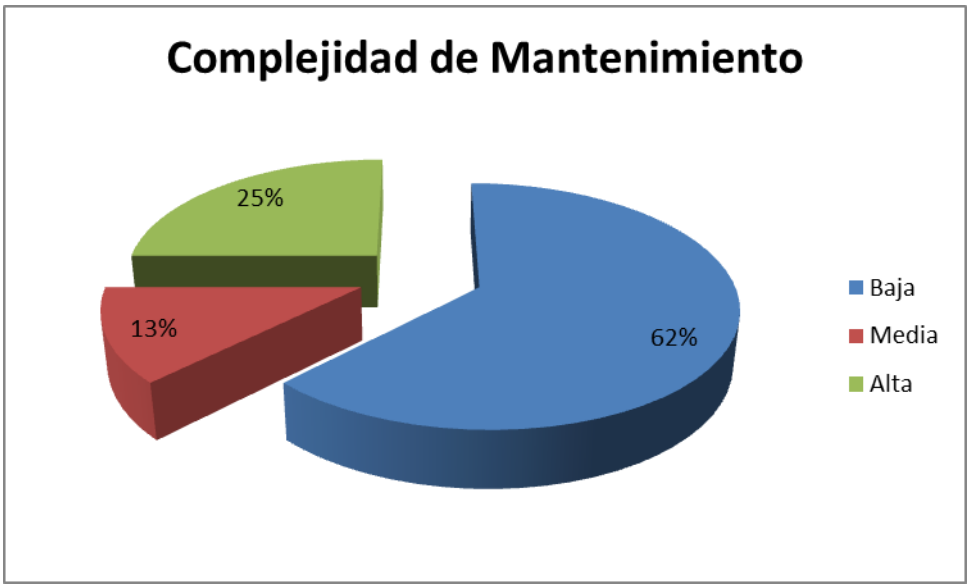

Figura 12: Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento

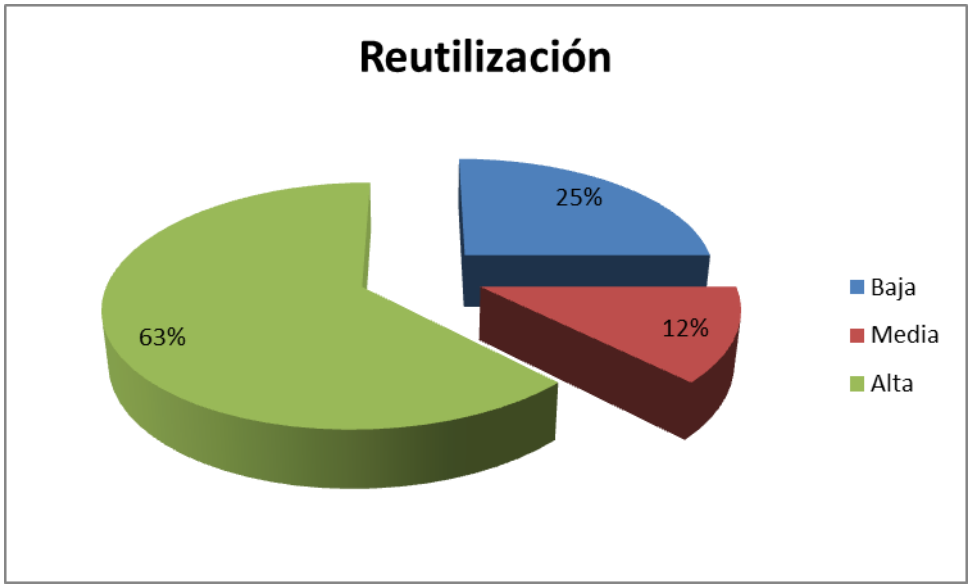

Figura 13: Resultados de la evaluación de la métrica RC para el atributo Complejidad de Reutilización

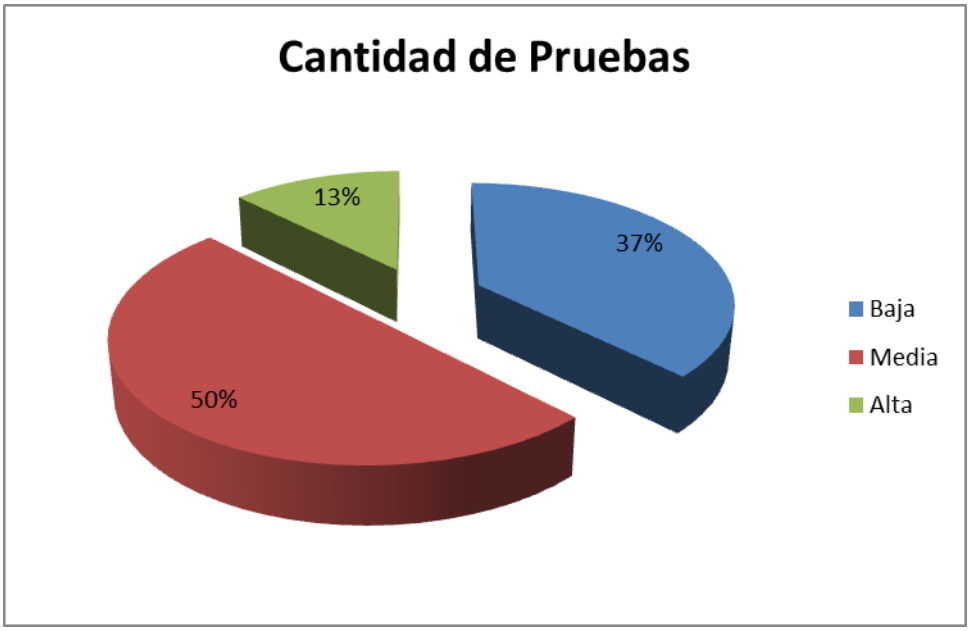

Figura 14: Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas

3.4.3 Matriz de cubrimiento o matriz de inferencia de indicadores de calidad

La matriz de cubrimiento o matriz inferencia de indicadores de calidad es una representación estructurada de los atributos de calidad y métricas utilizadas en los epígrafes anteriores para evaluar la calidad del diseño de los componentes que integran la solución propuesta. La misma permite conocer si el resultado obtenido de la relación atributo/métricas para cada componente es positivo o negativo. Llevando estos resultados a una escala numérica donde, si los resultados son positivos tendrá un valor de 1, si son negativos de 0 y si no existe relación alguna se tomará como nula (-). Una vez completado los datos de dicha relación se realiza un cálculo donde se promedia la sumatoria de los valores obtenidos de un atributo por cada métrica evaluada, y la división de dicha sumatoria por la cantidad de métricas evaluadas (solo se promedian las que arrojan un resultado, las nulas no). Este valor es el que va a tener el atributo dentro de una tabla que medirá si los atributos fueron buenos, regulares o malos.

Tabla 12: Rango de valores definidos para la evaluación

Atributos de Calidad	Intervalos de evaluación		
	Malo	Regular	Bueno
Responsabilidad	>=1.3	>0.7 y <1.3	<=0.7
Complejidad de Implementación	>=1.3	>0.7 y <1.3	<=0.7

Reutilización	<=0.7	>0.7 y <1.3	>=1.3
Acoplamiento	>=1.3	>0.7 y <1.3	<=0.7
Complejidad de Mantenimiento	>=1.3	>0.7 y <1.3	<=0.7
Cantidad de pruebas	>=1.3	>0.7 y <1.3	<=0.7

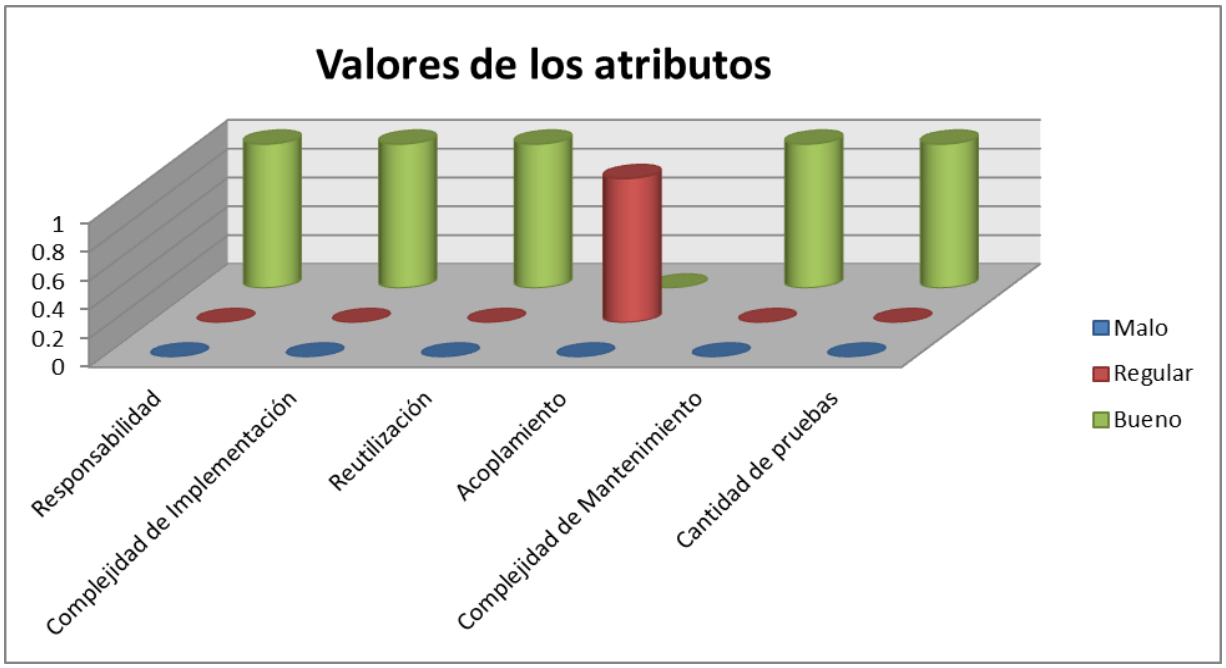

Figura 15: Resultados del rango de valores aplicados a cada atributo

Tabla 13: Resultados de la evaluación de la relación Atributo/Métrica

Atributo/Métrica	TOC	RC	Promedio
Responsabilidad	1	-	1
Complejidad de Implementación	1	-	1
Reutilización	1	1	1
Acoplamiento	-	1	1
Complejidad de Mantenimiento	-	1	1
Cantidad de pruebas	-	1	1

Los valores obtenidos en los atributos de calidad de la tabla 13 fueron positivos, el 38% de Alto del atributo Acoplamiento es debido a que las clases implementada hacen uso de la librería ezComponents que tiene un alto nivel de recursividad, lo cual provoca un aumento de la RC, y los restantes atributos con

evaluación Bueno, por lo que se demuestra que el diseño propuesto para el componente Motor de Ejecución se encuentra dentro de los niveles requeridos.

3.5 Pruebas de software

Las pruebas de software forman parte de una etapa imprescindible durante el proceso de desarrollo del software, pues permiten detectar y corregir el máximo de errores posibles antes de ser liberado el producto final. Por lo que el éxito de las mismas incide directamente sobre la percepción de calidad del usuario final. Cuando se considera que un componente está terminado se realizan las pruebas sistemáticas. A continuación se describen las pruebas realizadas al componente de ejecución de procesos de negocios modelados con BPMN en el marco de trabajo Sauxe.

3.5.1 Pruebas de Caja Blanca

Las pruebas de la caja blanca se realizan sobre las funciones internas de un módulo en concreto, están dirigidas a las funciones internas. Entre las técnicas usadas se encuentran; la cobertura de caminos, pruebas sobre las expresiones lógico-aritméticas, pruebas de camino de datos y comprobación de bucles.

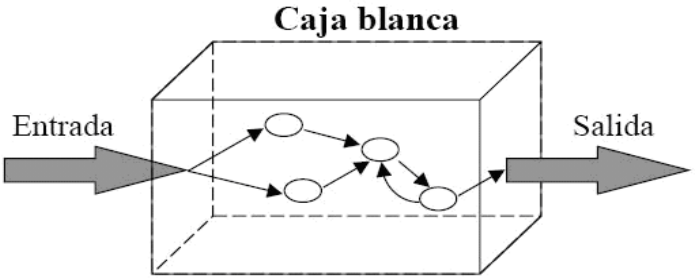

Figura 16: Representación de pruebas de Caja blanca

En el marco de las pruebas de caja blanca, la técnica que se utilizó fue la de prueba del camino básico, la cual se realizó mediante diseño casos de prueba que se centraron en obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Estas pruebas también garantizaron que en los casos de prueba obtenidos a través del camino básico se ejecute cada sentencia del programa al menos una vez. Para esto primero se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado.

Para aplicar la técnica del camino básico se debe introducir la notación para la representación del flujo de control, este puede representarse por un Grafo de Flujo en el cual:

- Cada nodo del grafo corresponde a una o más sentencias de código fuente
- Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo
- Se calcula la complejidad ciclomática del grafo

  Para construir el grafo se debe tener en cuenta la notación para las instrucciones

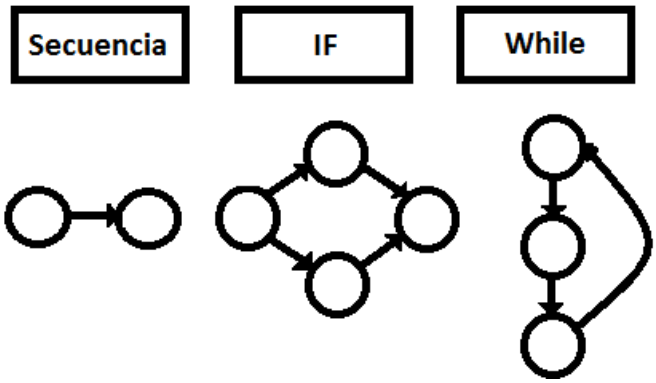

Figura 17: Notación de Grafos de flujo

Un grafo de flujo está formado por 3 componentes fundamentales que ayudan a su elaboración y comprensión, estos brindan información para confirmar que el trabajo se está haciendo adecuadamente.

Componentes del grafo de flujo:

Nodo: son los círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

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

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

Para realizar la técnica de prueba de caja blanca, específicamente la prueba del camino básico es necesario calcular antes la ***complejidad ciclomática*** del algoritmo o fragmento de código a analizar. A

continuación se enumeran las sentencias de código del procedimiento realizado sobre el componente para la ejecución de flujos de trabajo en el caso de prueba ***selectExecution***.

```
public function selectExecution($actividad)
{
    $result = -1; //1
    If ( count($actividad) == 1 ) //2
    {
        $result = 0; //3
    }
    Else
    {
        $node_id = $actividad[0]['node_id']; //4
        $semiautomaticTaskModel = new SemiautomaticTaskModel(); //4
        $semiautomaticTask     = $semiautomaticTaskModel->searchByIdObject($node_id); //4
        $variable_in    = $semiautomaticTask->getVariable_in(); //3
        $act            = variable_in['act']; //4
        $field          = variable_in['field']; //4
        $asociationModel = new AsociationModel(); //4
        $asociation       = $asociationModel->searchFieldAct($act, field); //4

        $post_field     = $_POST[$variable_in['post']]; //4
        $i = 0; //4
         $find = false; //4
        while( $i > count( $asociation ) && !$find ) //5
        {
            if($value['value'] == $post_field) //6
            {
                $result = $value['execution_id']; //7
                $find = true; //7
            }
            $i++; //8
        }
    }
    return $result; //9
}
```

Figura 18: Código fuente de la funcionalidad Seleccionar Ejecución

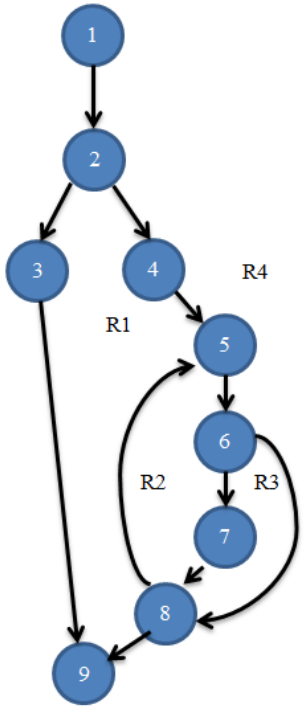

Figura 19: Notación de Grafos de flujo asociado a la funcionalidad

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

A continuación se realiza el cálculo de la complejidad ciclomática de la funcionalidad ***selectExecution*** mediante tres fórmulas descritas, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad sea correcto.

Fórmula 1. V (G) = (A – N) + 2

Donde “A” es la cantidad total de aristas y “N” la cantidad total de nodos.

V (G) = (11-9) + 2

V (G) =4

Fórmula 2. V (G) = P + 1

Donde “P” es la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) =3+1

V (G) =4

Fórmula 3. V (G) = R

Donde “R” es la cantidad total de regiones, para cada fórmula “V (G)” representa el valor del cálculo.

V (G) =4

Con la correcta realización del cálculo de complejidad se determinó el número de posibles caminos por donde el flujo debe circular y el número de casos pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. El cálculo arrojó como valor 4 por lo que seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo:

Camino #1: (1-2-3-9).

Camino #2: (1-2-4-5-6-7-8-9).

Camino #3: (1-2-4-5-6-8-9).

Camino #4: (1-2-4-5-6-8-5-6-7-8-9).

Para cada camino se realiza un caso de prueba, y es preciso cumplir con las siguientes exigencias:

- **Descripción:** se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo
- **Condición de ejecución:** se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento
- **Resultados Esperados:** se expone el resultado que se espera que devuelva el procedimiento
- **Evaluación de los resultados:** se exhibe la evaluación que dio el resultado final del procedimiento

Caso de prueba para el camino básico #1:

- **Descripción:** necesita como parámetros de entrada para ejecutarse los datos de una actividad en espera de ejecución
- **Condición de ejecución:** hay una sola actividad en espera de ejecución
- **Entrada:** una actividad
- **Resultados Esperados:** se retorna cero en el resultado

Caso de prueba para el camino básico #2:

- **Descripción:** necesita como parámetros de entrada para ejecutarse los datos de las actividades en espera de ejecución
- **Condición de ejecución:** el valor de la variable Post de la actividad en espera coincida el Post de la actividad en pera de la ejecución

- **Entrada:** un arreglo de actividades
- **Resultados Esperados:** se retorna el identificador de la ejecución

Caso de prueba para el camino básico #3:

- **Descripción: n**ecesita como parámetros de entrada para ejecutarse los datos de las actividades en espera de ejecución
- **Condición de ejecución:** el valor de la variable Post de la actividad en espera no coincida el Post de ninguna actividad en pera de la ejecución
- **Entrada:** Un arreglo de actividades
- **Resultados Esperados:** se retorna -1 en el resultado

Caso de prueba para el camino básico #4:

- **Descripción:** necesita como parámetros de entrada para ejecutarse los datos de las actividades
- **Condición de ejecución:** el valor de la variable Post de la actividad en espera coincida el Post de la actividad en pera de la ejecución luego de más de una iteración
- **Entrada:** Un arreglo de actividades
- **Resultados Esperados:** se retorna el identificador de la ejecución

Evaluación de los resultados: al realizar las pruebas de caja blanca al caso ***selectExecution*** se obtuvieron los resultados esperados, debido a que las tres fórmulas descritas arrojaron el mismo resultado y se cumplió la ejecución de cada camino al menos una vez con los parámetros de entrada y las salidas esperados.

3.5.2 Pruebas de Rendimiento

Según la IEEE: “Las pruebas de rendimientos es el grado en que un sistema o componente realiza sus funciones designadas dentro de las limitaciones dadas, tales como la velocidad, precisión o el uso de la memoria”.

Existen diferentes tipos de pruebas de rendimiento que ayudarán a mejorar las capacidades de una aplicación observando y evaluando las respuestas del sistema ante todas las posibles situaciones que se puedan dar. Cada una de estas pruebas tiene sus objetivos y características.

A la solución se le aplicaron las pruebas de rendimiento de carga y estrés, utilizando para ello la herramienta JMeter. Estas pruebas fueron realizadas por el equipo de calidad del centro. Debido a que el

motor de ejecución se encuentra relacionado con todas las acciones de cualquier componente asociadas a un flujo de trabajo definido en el modelador.

Las pruebas de estrés son utilizadas normalmente para someter a la aplicación al límite de su funcionamiento mediante la ejecución de un número de usuarios muy superior al esperado. Esta prueba tiene como finalidad el determinar la robustez de una aplicación cuando la carga es extrema. Para realizar la prueba de estrés, la aplicación se sometió a la ejecución de 125 usuarios.

Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. El resultado de esta prueba nos dará el tiempo de respuesta de todas las transacciones críticas.

Con la prueba de Rendimiento realizada al Editor de procesos de negocio se logró medir el tiempo de respuesta del sistema para el caso de 125 usuarios utilizando la aplicación. Para comprobar si el sistema relevó resultados positivos es necesario que el valor no sobrepase los 5 segundos, ya que este es el tiempo de repuesta límite definido en el requisito no funcional de Rendimiento para el Marco de Trabajo Sauxe.

Después de obtener los resultados generados en JMeter, el Editor de procesos de negocios relevó un tiempo de respuesta entre 1,8 seg. y 3,4 seg.; resultado positivo para el componente ya que estos valores no sobrepasan los 5 segundos de tiempo de respuesta del sistema.

Label	# Muestr...	Media	Mediana	Linea d...	Mín	Máx	% Error	Rendimi...	Kb/se
/portal/	125	11191	12641	16407	469	18266	0,00%	3,3/sec	862
/portal/index.php/index/cargardominio	125	7525	8328	9438	578	16063	0,00%	3,4/sec	975
/portal/index.php/index/entraralsistema	125	7200	7703	9094	766	9953	0,00%	2,7/sec	777
/portal/index.php/portal/portal	125	7541	8016	8844	984	9797	0,00%	2,5/sec	683
/portal/index.php/portal/cargarperfil	125	7626	7985	8765	985	9500	0,00%	2,1/sec	600
/portal/index.php/portal/cargaretiquetas	250	7527	7562	8656	2937	10188	0,00%	3,4/sec	974
/portal/index.php/portal/cargardatostabpanel	125	7070	7157	8188	3781	9312	0,00%	1,8/sec	560
/portal/index.php/portal/cargardesktop	125	6956	6907	8125	5297	9453	0,00%	2,0/sec	562
/lib/ExtJS/temas/default/images/images/des...	125	56	47	125	0	281	100,00%	2,6/sec	88
/Herramientas/editorft/index.php/aplicacion/a...	125	5294	5437	7453	1125	8312	0,00%	2,3/sec	692
/Herramientas/editorft/views/js/aplicacion/co...	125	34	16	78	0	234	0,00%	3,0/sec	587
/Herramientas/editorft/views/js/aplicacion/ent...	125	9	0	31	0	47	0,00%	3,0/sec	1
/Herramientas/editorft/views/js/components/t...	125	11	15	31	0	63	0,00%	3,0/sec	1659
/Herramientas/editorft/views/js/components/...	125	11	15	31	0	47	0,00%	3,0/sec	3610
/Herramientas/editorft/views/js/components/...	125	12	15	31	0	78	0,00%	3,0/sec	2977
/Herramientas/editorft/views/js/components/t...	125	10	15	31	0	47	0,00%	3,0/sec	1322
/Herramientas/editorft/views/js/components/t...	125	9	0	31	0	62	0,00%	3,0/sec	615
/Herramientas/editorft/views/js/components/...	125	12	15	31	0	172	0,00%	3,0/sec	3423
TOTAL	2375	3980	2437	8688	0	18266	5,26%	27,5/sec	11046

Figura 20: Resultados obtenidos de las pruebas de rendimiento realizadas con la herramienta JMeter

3.6 Conclusiones parciales

En el capítulo se realizaron pruebas para validar el diseño propuesto, se utilizaron métricas de Relaciones entre clases y Tamaño operacional de clases, en la que se evaluaron los atributos de calidad Reutilización, Complejidad de Implementación, Responsabilidad entre otros, dando resultados satisfactorios basados en la tabla de rangos definidas por los autores del presente trabajo.

Se realizó la validación del diseño propuesto efectuando las pruebas de caja blanca usando la técnica de los caminos básicos, las cuales arrojaron resultados satisfactorios sobre el funcionamiento del código implementado. Para esto se calculó la complejidad ciclomática del mismo, asegurando que la ejecución de cada camino ocurra al menos una vez.

Al componente le realizaron las pruebas de carga y estrés de las cuales se obtuvieron resultados satisfactorios, puesto que el tiempo de respuesta del sistema no excedió de los 5 segundos definidos por los requisitos no funcionales del marco de trabajo para un tiempo de respuesta bueno.

CONCLUSIONES

Después de un estudio realizado fue posible fundamentar la posibilidad de la integración de un Sistema ERP con un WfMS, dando como resultado un sistema flexible capaz de responder a cambios en los procesos de negocio. Se hizo un análisis de diferentes herramientas utilizadas para la gestión de flujos de trabajo, determinando que el conjunto de librerías de ezComponents, después de una serie de adaptaciones, podría ejecutar flujos de trabajo sobre Sauxe. Esta herramienta fue escogida debido a cumplir con las restricciones impuestas por el ambiente de desarrollo, para el uso de una herramienta de este tipo.

Se llevó a cabo el análisis y diseño correspondiente a la solución propuesta, arrojando los artefactos requeridos por el modelo de desarrollo de software orientado a componentes propuesto por CEIGE. Los requisitos funcionales fueron licitados, descritos y posteriormente validados de manera satisfactoria. Estos requisitos incluyen la ejecución de tareas automáticas y semiautomáticas. Al estar la solución propuesta necesariamente integrada a Sauxe, sus requisitos no funcionales coinciden con los definidos para el marco de trabajo.

El sistema fue implementado siguiendo las especificaciones del diagrama de clases y el modelo de datos respectivamente, dando como resultado un componente capaz de ejecutar flujos de trabajo definidos y adaptados previamente para el marco de trabajo Sauxe. Esta ejecución se hizo orientada específicamente al manejo de tareas automáticas y semiautomáticas, para una primera versión del componente de ejecución de flujos de trabajo. Se validó el diseño de la aplicación utilizando las métricas TOC y RC, las cuales arrojaron resultados satisfactorios teniendo en cuenta una tabla de rangos definidos expresamente para la validación de la presente solución. El sistema pasó por una fase de pruebas, la cual incluyó pruebas de caja blanca, utilizando el método de camino básico para determinar la complejidad lógica del código; esto fue aplicado a todas las funcionalidades implementadas. También se realizaron pruebas de carga y estrés, demostrando la capacidad de respuesta de la aplicación ante una determinada carga de usuarios.

Con la implementación de un componente para la ejecución de flujos de trabajo, se completó la aplicación para la gestión de flujos de trabajo incorporada al marco de trabajo Sauxe; con el objetivo de darle flexibilidad tanto a Cedrux como a otros sistemas que puedan ser implementadas sobre él.

RECOMENDACIONES

1- Hacer uso de formularios dinámicos para la ejecución de las tareas semiautomáticas.
2- Basado en los formularios dinámicos, hacer una nueva definición de variables a consumir por la ejecución de los flujos de trabajo.
3- Extender la ejecución a otros tipos de tareas definidos por BPMN.
4- Incorporar otros componentes de ezComponents, agregándole más funcionalidades a la aplicación para la gestión de flujos de trabajo.

BIBLIOGRAFÍA

1. Alicia Martínez, N. G., Hugo Estrada (2010). "Monitoreo de procesos de negocios a través de un enfoque orientado a metas."
2. Aquino, A. L., Yasser (2009). "Implementación del módulo de Contabilidad General del Sistema Integral de Gestión Cedrux."
3. Arquitectura-IEEE (2000). "Arquitectura-IEEE 1471."
4. autores, C. d. (2009). ". Business Process Management Systems. ."
5. Autores, C. d. (2009). "Sage ERP X3."
6. autores, V. (2009). ". ERP Cuba. Proceso de Desarrollo y Gestión de Proyectos de Software."
7. Bergmann, S. (2009). "Design and Implementation of a Workflow Engine."
8. Buenosvinos, C. and C. Crespo (2008). "Motores de Workflow, Más allá de las Aplicaciones CRUD."
9. Bukovics, B. (2009). "Windows Workflow in .NET 3.5."
10. Canales, C. d. F. (2010). "Workflows flexibles para procesos de desarrollo de software." tesis de maestría.
11. Centro Técnico de Informática, S. (2006.). "El Framework de desarrollo del Consejo Superior de Investigaciones Científicas.".
12. Coalition, W. M. (1999). "Terminology and Glossary."
13. Cybok, D. (2006). "A Grid Workflow Infrastructure."
14. Dr. Markus Nüttgens and D.-K. V. Z. Dipl.-Inform. Thomas Feld (1998). "Business Process Modeling with EPC and UML Transformation or Integration?"
15. DUMAS, M., W. v. d. AALST, et al. (2005). PROCESS-AWARE INFORMATION SYSTEMS Bridging People and Software Through Process Technology.
16. E.W.T. Ngai, C. C. H. L., F.K.T. Wat (2008). "Examining the critical success factors in the adoption of enterprise resource planning."
17. Esser, S. (2009). "Secure Programming with the Zend-Framework." Dutch PHP Conference.

18. F. Gottschalk, W. M. P. v. d. A., M.H. Jansen-Vullers, and M. La Rosa (2007). "ConfigurableWorkflow Models."
19. Firefox., M. "Mozilla Firefox. http://www.mozilla- europe.org."
20. Frederick, S., Ramsay, Colin y Blades, Steve 'Cutter' (2008). "Learning Ext JS."
21. Garimella, K. L., Michael y Bruce Williams (2008). "Introducción a BPM para Dummies."
22. Goetz, c. i. M. (2009). "Modeling Workflow Patterns through a Control-flow perspective using BPMN and the BPM Modeler BizAgi."
23. González, P. S., M. A. Martínez, et al. (2003). "Análisis y modelado con redes de workflow del proceso de tratamiento de experiencias operativas."
24. González., O. O. (2011). "Interoperabilidad del proceso inventario en el sistema Cedrux."
25. Hee, W. M. P. v. d. A. a. K. M. v. (2002). "Workflow Management: Models, Methods, and Systems."
26. Hollingsworth, D. (1995). "Workflow Management Coalition The Workflow Reference Model."
27. IEEE (1993). "Standards Collection: Software Engineering."
28. Ing. Oiner Gómez Baryolo, I. Y. M. B., Ing. Darien García Tejo (2009). "ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE."
29. Koenig, j. (2004). "JBoss Jbpm white paper."
30. LARMAN., C. (1999.). "UML Y PATRONES INTRODUCCION AL ANALISIS Y DISEÑO ORIENTADO A OBJETOS." PRIMERA EDICION.
31. Lemus, J. M. (2009). "Modelado de datos e implementación de la base de datos."
32. Lerate, R. M. D., P. R. Quijano, et al. (2009). "Sistemas para el Control de Versiones."
33. Linux., Á. t. d. (Citado el: 15 de Diciembre de 2009.). " Ciberaula. [En línea]."
34. Martin Owen, J. R. (2003). "BPMN and Business Process Management."
35. Martinez, M. M. A. (2011). "Integración de los Sistemas ERP en las organizaciones."
36. MSc. Patricia Noy Viamontes, I. Y. P. F. (2009). "La actualidad de la Gestión de Procesos de Negocio: Business Process Management (BPM)."
37. Nielsen, M. (2009). "Microsoft Dynamics NAV."

38. Nora Escalona, M. J. a. K. (2002). "Ingeniería de Requisitos en Aplicaciones para la Web-Un estudio comparativo."
39. Parsons, D. (2012). "Data Types, Arithmetic, and Arrays."
40. Patricia Bazán, R. G., F.Javier Diaz (2010). "Tecnologías para implementar un marco integrador de SOA y BPM."
41. Pérez, M. S. (2009). "Propuesta de modelo de desarrollo de software tecnológico del Centro de Soluciones de Gestión."
42. Petia Wohed, N. R., Arthur H.M. ter Hofstede, Birger Andersson, Wil M.P. van der Aalst (2009). "Patterns-based evaluation of open source BPM systems: The cases of jBPM, OpenWFE, and Enhydra Shark."
43. PHP., G. d. d. d. (2011). "Manual de PHP."
44. Plesumus, C. (2002). "Introduction to Workflow." The Workflow Handbook.
45. Pressman, R. S. (2002). "La Ingeniería del Software. Un enfoque práctico."
46. Pyke, J. (2006). "BPM: Now and in the Future." The Workflow Handbook.
47. Ríos, C. L. (2011). "GESTIÓN DE RRHH CON SAP."
48. Rumbaugh, J., Jacobson, Ivar and Booch, Grady (2000). "El Lenguaje Unificado de Modelado. Manual de Referencia."
49. Rumbaugh, J., I. Jacobson, et al. (1999). "El lenguaje unificado de modelado."
50. Sage (2008). "Sage ERP X3: La solución de gestión avanzada para medianas y grandes empresas."
51. Sehmi, A. "Flujo de Trabajo Generacional." http://www.ArchitectureJournal.net
52. Simone Pellegrini, F. G., Antonia Ghiselli (2006). "A PRACTICAL APPROACH FOR A WORKFLOWMANAGEMENT SYSTEM."
53. Smith, H., and Peter Fingar (2003). "Business Process Management: The Third Wave."
54. Stephen A. White, I. C. (2009). "Introduction to BPMN."
55. Systems Popkin, S. (2008). "Modelado de Sistemas con UML."
56. Trichkov, K. and E. Trichkova (2006). "Modeling and Execution of Web Service in Internet using Enhydra workflow platform."

57. UCI (2008). "Introducción al proceso de desarrollo de software." Tema 1.
58. W.M.P van der Aalst, A. K. (2001). "A reference model for team-enabled workflow next term management systems."
59. Weske Mathias (2008). "Business Process Management: Concepts, Languages, Architectures."
60. White, S. A. (2010). "Process Modeling Notations and Workflow Patterns."
61. Word, D. W. a. J. (2004). "SAP NetWeaver For Dummies."

REFERENCIAS BIBLIOGRÁFICAS

1. DUMAS, M., W.v.d. AALST, and A.H.M.t. HOFSTEDE, *PROCESS-AWARE INFORMATION SYSTEMS*
Bridging People and Software
Through Process Technology. 2005.
2. Hee., W.M.P.v.d.A.a.K.M.v., *Workflow Management: Models, Methods, and Systems.* 2002.
3. Martinez, M.M.A., *Integración de los Sistemas ERP en las organizaciones.* 2011.
4. Hollingsworth, D., *Workflow Management Coalition*
The Workflow Reference Model. 1995.
5. Pyke, J., *BPM: Now and in the Future.* The Workflow Handbook, 2006.
6. Plesumus, C., *Introduction to Workflow.* The Workflow Handbook, 2002.
7. Bergmann, S., *Design and Implementation of aWorkflow Engine.* 2009.
8. González, P.S., M.A. Martínez, and D.P. González, *Análisis y modelado con redes de workflow del proceso de tratamiento de experiencias operativas.* 2003.
9. Buenosvinos, C. and C. Crespo, *Motores de Workflow, Más allá de las Aplicaciones CRUD.* 2008.
10. MSc. Patricia Noy Viamontes, I.Y.P.F., *La actualidad de la Gestión de Procesos de Negocio: Business Process Management (BPM).* 2009.
11. Garimella, K.L., Michael y Bruce Williams, *Introducción a BPM para Dummies.* 2008.
12. Smith, H., and Peter Fingar, *Business Process Management: The Third Wave.* 2003.
13. KOENIG, J., *JBOSS jBPM WHITE PAPER.* 2004.
14. Petia Wohed, N.R., Arthur H.M. ter Hofstede, Birger Andersson, Wil M.P. van der Aalst *Patterns-based evaluation of open source BPM systems: The cases of jBPM, OpenWFE, and Enhydra Shark.* 2009.
15. Alicia Martínez, N.G., Hugo Estrada, *Monitoreo de procesos de negocios a través de un enfoque orientado a metas.* 2010.
16. Centro Técnico de Informática, S., *El Framework de desarrollo del Consejo Superior de Investigaciones Científicas.* 2006.
17. Sage, *Sage ERP X3:La solución de gestión avanzada para medianas y grandes empresas.* 2008.
18. Autores, C.d., *Sage ERP X3.* 2009.
19. Word, D.W.a.J., *SAP NetWeaver For Dummies.* 2004.
20. Trichkov, K. and E. Trichkova, *Modeling and Execution of Web Service in Internet using Enhydra workflow platform.* 2006.
21. Nielsen, M., *Microsoft Dynamics NAV.* 2009.
22. Canales, C.d.F., *Workflows flexibles para procesos de desarrollo de software.* tesis de maestría, 2010.
23. Bukovics, B., *Windows Workflow in .NET 3.5.* 2009.
24. Pérez, M.S., *Propuesta de modelo de desarrollo de software tecnológico del Centro de Soluciones de Gestión.* . 2009.
25. Ing. Oiner Gómez Baryolo, I.Y.M.B., Ing. Darien García Tejo, *ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE* 2009.

26. Esser, S., *Secure Programming with the Zend-Framework* . . 2009. **Dutch PHP Conference.**
27. Aquino, A.L., Yasser, *Implementación del módulo de Contabilidad General del Sistema Integral de Gestión Cedrux.* . 2009.
28. Linux., Á.t.d. *Ciberaula. [En línea]*. Citado el: 15 de Diciembre de 2009.
29. Prada Nicot, H.y.S.G., Kenner, *Desarrollo de los componentes, Puesto de Trabajo y Pagos Adicionales del subsistema Capital Humano integrado al sistema integral de gestión CEDRUX.* . 2009.
30. Firefox., M. *Mozilla Firefox. [En línea]* *http://www.mozilla-europe.org/es/firefox/3.0/releasenotes/*.
31. Lerate, R.M.D., et al., *Sistemas para el Control de Versiones.* 2009.
32. Systems Popkin, S., *Modelado de Sistemas con UML.* 2008.
33. PHP., G.d.d.d., *Manual de PHP.* 2011.
34. Frederick, S., Ramsay, Colin y Blades, Steve 'Cutter', *Learning Ext JS.* . 2008.
35. LARMAN., C., *UML Y PATRONES INTRODUCCION AL ANALISIS Y DISEÑO ORIENTADO A OBJETOS.* . 1999. **PRIMERA EDICION**.
36. González., O.O., *Interoperabilidad del proceso inventario en el sistema Cedrux.* 2011.
37. Nora Escalona, M.J.a.K., *Ingeniería de Requisitos en Aplicaciones para la Web-Un estudio comparativo.* . 2002.
38. IEEE, *Standards Collection: Software Engineering.* . 1993.
39. González., O.O., *Interoperabilidad del proceso inventario en el sistema Cedrux.* 2011.
40. Pressman, R.S., *La Ingeniería del Software. Un enfoque práctico.* . 2002.
41. Lemus, J.M., *Modelado de datos e implementación de la base de datos.* 2009.
42. Arquitectura-IEEE, *Arquitectura-IEEE 1471.* 2000.
43. Rumbaugh, J., I. Jacobson, and G. Booch, *El lenguaje unificado de modelado.* 1999.
44. Parsons, D., *Data Types, Arithmetic, and Arrays.* 2012.

1