Universidad de las Ciencias Informáticas

Facultad 3

Título: Componente para transformar ficheros XPDL al lenguaje de ejecución de flujos de trabajos para Sauxe.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autores: Legna Cortina Almanza.

Reinier Rivas Silva

Tutores: Ing. René Bauta Camejo

Ing. Yoriangel Rivero González

La Habana, 2013

Podrán cortar todas las flores, pero nunca terminarán con la primavera. Seamos realistas y hagamos lo imposible

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

Declaramos ser autores del presente trabajo y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

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

Legna Cortina Almanza

_____

Firma del Autor

Reinier Rivas Silva

_____

Firma del Autor

Ing. Yoriangel Rivero González

_____

Firma del Tutor

Ing.Rene Bauta Camejo

_____

Firma del Tutor

AGRADECIMIENTOS

Reinier Rivas:

Ante todo agradecerle su colaboración en todo momento a mis tutores ya que han demostrado además de ser unos profesionales capaces, unos seres humanos estupendos, agradecer a mi compañera de tesis Legna por su comprensión, ayuda y permanecer a mi lado hasta el final, al tribunal ya que con su mano dura nos fue forjando en el camino para hacer de nosotros unos profesionales, a mi familia que me apoyo y me apoya en todo momento. A mis amigos que me ha acompañado en esta larga trayectoria de mi vida esos nombres que voy a recordar por siempre Carlos Vingut, Julio Jesús, y por último y no menos importante a mi esposa por la comprensión, amor y respeto demostrados.

Legna Cortina:

A mis padres que son todo lo que tengo en mi vida, a mis amigos que han recorrido conmigo estos 5 años y que son lo más importante que me llevo de esta etapa de mi vida, a mis tutores y en especial a Yoriangel que además de tutor ha sido un amigo en las buenas y las malas y esta tesis no se habría concretado de no ser por él….es por eso que si hoy soy ingeniera se lo debo a él…. A mi novio que ha soportado todos mis cambios de humor que han sido muchos en estos 8 meses y a todos los que de una forma u otra han colaborado para que hoy pueda llamarme ingeniera.

DEDICATORIA

Reinier Rivas:

Quiero dedicarle el presente trabajo de diploma especialmente a mis padres y hermano los cuales han sabido guiarme, aconsejarme hasta el final de esta carrera sin ellos no fuera posible, a mi esposa eh hijo por todo su amor y apoyo, y a todas aquellas personas que en el transcurso de la carrera creyeron en mí y me extendieron su mano.

Legna Cortina:

Esta tesis se la dedico a mis padres y abuelos, mis amigos y a todos los que creyeron en mí….

RESUMEN

Los procesos de negocio constituyen un esfuerzo para mejorar las operaciones y funciones de las empresas. En Cuba, se desarrolló un Sistema de gestión empresarial sobre el marco de trabajo Sauxe; 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 (UCI). Sauxe contiene un componente para la gestión de flujos de trabajo basado en la librería ezComponents de PHP (PHP Hypertext Preprocessor), la cual posee un lenguaje de definición nativo, no estandarizado, descrito en sintaxis XML(Extensible Markup Language) y con características muy propias. Lo que dificulta en gran escala la interpretación directa del flujo de procesos de negocio definido en XPDL además de implementar siete patrones de flujo de trabajo muy básicos, limitando en gran medida la expresividad del mismo. La presente investigación constituye en una propuesta de solución para el desarrollo de un componente que permita la transformación de XPDL a un lenguaje de ejecución de procesos basado en el estándar BPEL, aumentando la expresividad con la implementación de nueve patrones incluyendo los que ya contiene. El desarrollo está respaldado por el estudio de sistemas de gestión de procesos relacionados con la transformación de un lenguaje de definición a uno de ejecución. Se generan los artefactos atendiendo la estructura del modelo de desarrollo a seguir y se realiza la implementación atendiendo al diseño propuesto, así como la validación del mismo.

Palabras Claves:

BPEL, flujo de trabajo, lenguaje de definición, lenguaje de ejecución, Proceso de negocio, patrones de flujo de trabajo, XPDL.

ÍNDICE

INDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El mundo de hoy, cambia vertiginosamente y exige que las empresas cuenten con la agilidad, flexibilidad y capacidad de adaptarse a nuevos entornos de una manera rápida, por lo que uno de sus objetivos fundamentales es mantener una posición en el mercado que les permita obtener un óptimo desarrollo financiero. Debido a esto han centrado su atención en los Procesos de Negocio (Bussines Process, BP por sus siglas en inglés). En aras de lograr estos objetivos puede ser de ayuda realizar una adecuada gestión de procesos, que permita mejorar la ejecución y resultados de los mismos.

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. Estos sistemas 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. (1) (2) Ejemplo de PAIS, son los sistemas de gestión de flujos de trabajo, WfMS[2] por sus siglas en inglés, y algunas plataformas de integración empresarial. Se le conoce como WfMS a un sistema que define, crea y gestiona la ejecución de un flujo de trabajo a través de un software, y que permite interpretar el proceso de definición y además interactuar con los participantes del flujo de trabajo.

Los sistemas de Workflow ayudan a gestionar los procesos de negocio, asegurando que las actividades serán ejecutadas:

- Lo más rápido posible
- Por las personas adecuadas
- En el orden justo

Otra de las grandes ventajas de implantar una herramienta Workflow es la estandarización de los procesos y un mayor control sobre los mismos, lo que permite reducir los costes de operación, elevar la eficiencia de los empleados y a la vez ofrecer un mejor servicio al cliente (3) (4)

[1] Siglas de Process Aware Information System

[2] Siglas de Workflow Management Systems.

En el proceso realizado por los WfMS se pueden distinguir distintas perspectivas:

- **La perspectiva de flujo de control**: capta los aspectos relacionados con el control de la información, de las dependencias entre diversas tareas (por ejemplo, el paralelismo, la elección, sincronización)
- **La perspectiva de los Datos**: se refiere a la transmisión de información, alcance de variables entre otras
- **La perspectiva de Recursos:** se refiere a recursos para la asignación de tareas

Centrando la atención en la perspectiva de flujo de control; esta es definida a partir de la implementación de diferentes patrones de flujo de control, que tienen como propósito describir la capacidad de modelar la complejidad de los procesos de negocio, es decir, la expresividad con que van a contar los lenguajes que utilizan los WfMS para la definición y ejecución de los mismos. (5)

Cuba no se encuentra indiferente a los cambios que ocurren con el mundo empresarial a nivel internacional, y muestra de ello es el trabajo realizado por el Centro de Informatización de Gestión de Entidades (CEIGE) perteneciente a la Universidad de las Ciencias Informáticas donde se desarrolla un componente para la gestión de flujos de trabajo. El mismo es confeccionado sobre el marco de trabajo Sauxe y a su vez está subdividido en otros componentes, entre los cuales se encuentra uno para la ejecución de procesos de negocio, basado en ezcWorkflow de ezComponents. De manera general, este componente permite definir y ejecutar procesos de negocio. Para ello cuenta con una API de definición de procesos y una para la ejecución de los mismos. La principal dificultad del componente de workflow de ezComponents en general y de la API Workflow Definition en particular es que los modeladores de los procesos deben tener un alto conocimiento de programación en PHP.

Como parte del desarrollo del componente de workflow que se realiza para Sauxe, se ha logrado una integración de un componente de definición de procesos de negocio de manera visual usando la notación BPMN. Una definición de procesos de negocio concreta con BPMN es almacenada haciendo uso del lenguaje XPDL (XML Process Definition Language o Lenguaje de Definición de Procesos XML), la cual es luego traducida a una definición de procesos para ejecución en el lenguaje nativo de ezcWorkflow. Este lenguaje de definición, que es también basado en sintaxis XML, representa una serialización de un proceso ejecutable por la API Workflow Execution, por lo tanto limita las definiciones del proceso en XML a la estructura que el motor de ejecución define y ejecuta. Los

procesos en este componente de ejecución representan un grafo, donde los nodos representan actividades y las transiciones son representadas como una lista de nodos definidos como atributo de cada tipo de nodo.

Una vez realizado un estudio del componente ezcWorkflow, se pudo observar que existe un número limitado de nodos, consistente fundamentalmente de nodos split, multi-choice, multi-merge, simple-choice y simple-merge; los que representan los patrones para el control de flujos. Además presenta un solo nodo de tipo Action, el cual en su definición contiene como atributo un objeto de tipo ServiceObject declarado como una interfaz. En tiempo de diseño de los procesos, se deben definir clases (escritas en php) que implementen esa interfaz y definir comportamientos según lo que se desee que se haga una vez que el motor de ejecución alcance el nodo Action.

BPMN, por su parte, define un conjunto de elementos a los que el lenguaje de definición de procesos de ezcWorkflow no da soporte en ninguna manera. A modo de ejemplo, se puede mencionar que el lenguaje de ejecución de ezcWorkflow define un evento de inicio por defecto, al igual que un evento de fin. Estos no representan una acción concreta, ni constituyen, según definen las especificaciones de BPMN (BPMN Specifications, Versión 1.0 - Mayo 2004), algo que ocurre durante el curso de un proceso y que puede tener una causa y un resultado. Otro ejemplo de las limitaciones que presenta dicho componente, es que solo definen una acción (nodeAction), y que no representan ningún tipo de comportamiento según define esta misma especificación. Para ilustrar se puede mencionar los tipos de tareas Enviar un mensaje, Recibir un mensaje, Consumir un servicio, Ejecutar código, Invocar aplicaciones (en términos BPMN: receive, send, service, script, invoke. No maneja el uso de los disparadores de eventos que según BPMN son el mecanismo de activación o desactivación de un evento, los que están relacionados a los eventos de inicio y fin, y ser de tipo tiempo, mensaje, regla, cancelación (en términos BPMN : timer, message, rule, cancel, terminate) A consecuencia de ello, se ha decidido extender dicho componente para soportar todos los tipos de actividades que define BPMN.

Además de los anteriormente mencionados, existe el problema de que el lenguaje para ejecución solo provee soporte a patrones básicos de control de flujo, limitando esto la capacidad del mismo de modelar la complejidad de un proceso de negocio, en resumen su expresividad. Adicionalmente, no

existe una estructura del lenguaje, una semántica clara y precisa de cómo se debe definir un proceso de negocio para ezcWorkflow.
Una vez descrita la problemática queda como **problema a resolver:** ¿Cómo garantizar una definición de proceso de negocio con mayor expresividad en el componente de gestión de procesos de negocio de Sauxe?

Se define como **objeto de estudio** los Sistemas de Gestión de Procesos y como **campo de acción** las herramientas de intercambios de lenguajes definición a lenguajes de ejecución de flujo de trabajo.

Teniendo en cuenta el problema, se define como **objetivo general**: Desarrollar un componente que permita obtener la definición de un proceso de negocio ejecutable para el componente de flujos de trabajos del marco de trabajo Sauxe a partir de su definición usando XPDL.

Para darle cumplimiento al objetivo general se plantean como **objetivos específicos**:

1. Realizar el marco teórico acerca de las herramientas de gestión de flujos de trabajo.
2. Realizar un estudio sobre los lenguajes de definición y los lenguajes de ejecución de flujos de trabajo existentes para determinar posibles áreas de reutilización.
3. Realizar un estudio de los diferentes patrones de flujo de trabajo existentes para determinar los utilizados por la solución propuesta.
4. Realizar el análisis y diseñó de la solución.
5. Definir los algoritmos transformación.
6. Implementar algoritmos de transformación
7. Validar la solución propuesta.

La investigación parte de la siguiente **idea a defender**, Si se desarrolla un componente que permita obtener la definición de un proceso de negocio ejecutable para el componente de flujos de trabajos del marco de trabajo Sauxe a partir de su definición usando XPDL se garantizará una definición de proceso con mayor expresividad.

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 las principales herramientas de gestión de procesos de negocio en el mundo, facilitando su estudio y logrando definir una estrategia para llegar al resultado final con facilidad.

Histórico lógico: El uso de este método permitió conocer y comprender qué beneficios trae consigo el uso de los lenguajes de ejecución de BP en el mundo, qué funcionalidades incorporan y cómo se integran, conociendo así su evolución y desarrollo hasta la actualidad.

El trabajo se estructura en tres capítulos:

CAPÍTULO 1. **Fundamentación Teórica:** Se describen los aspectos y conceptos asociados al dominio del problema a resolver, siendo estos esenciales para entender el entorno del mismo. Se presenta el estado del arte de las técnicas implicadas en el objeto de estudio y el conjunto de tecnologías involucradas en el desarrollo de la propuesta.

CAPÍTULO 2. **Propuesta de Solución**: Se exponen los procesos de negocios y requisitos a cumplir por el componente además de los artefactos generados durante el diseño de la misma.

CAPÍTULO 3. **Implementación y Prueba**: Se exponen los artefactos generados durante la implementación de la solución así como las métricas y pruebas utilizadas para la validación de la misma

CAPÍTULO 1

1.1. Introducción

En el capítulo se tiene como objetivo realizar un análisis y estudio de varios sistemas gestores de flujos de trabajo, específicamente en los lenguajes de definición y ejecución que implementan, así como las herramientas y procedimientos utilizados para la transformación de un modelo de procesos de negocios a un lenguaje de ejecución y la forma más factible para adaptarlo al marco de trabajo Sauxe y a su herramienta de flujos de trabajo.

1.2. Conceptos

Flujo de trabajo (Workflow o WF):.Se define como un sistema de secuencia de tareas de un proceso de negocio. Organiza y controla tareas, recursos y reglas necesarias para completar el proceso de negocio. (6)

Patrones de flujos de trabajo: soluciones que permiten hacer cumplir las dependencias de flujo de control entre las tareas que componen un flujo de trabajo. (7)

Expresividad o poder expresivo: Capacidad de modelar la complejidad procesos de negocio. Su regla de medida son los patrones de flujo de trabajo. (8) (9)

Proceso de Negocio (Business Process o BP): Conjunto de tareas relacionadas lógicamente, que utilizan recursos para transformar en un resultado de negocio definido los elementos de entrada que reciban, ya que con ellos describen su comportamiento y flujos de trabajo (3) (10)

BPM (Business Process Management): se llama Business Process Management a la metodología empresarial cuyo objetivo es mejorar la eficiencia a través de la gestión sistemática de los procesos de negocio, que se deben modelar, automatizar, integrar, monitorear y optimizar de forma continua. (10)

XPDL: es un lenguaje de definición de proceso de negocio establecido por la WfMC[3] que usa una sintaxis basada en XML. Tiene como objetivo almacenar e intercambiar diagramas de procesos y permitir que un motor de flujos de trabajo pueda interpretar los procesos para su posterior ejecución. (11)

BPEL4WS: es un lenguaje de ejecución con sus variables y operaciones para los sistemas o gestores de flujo de trabajo basado en servicios web, utilizado por la gran mayoría de desarrolladores

[3] Del inglés Workflow Management Coalition

de sistemas de gestión de procesos de negocios. El objetivo de BPEL es ofrecer una forma de orquestar servicios web, la secuencia de interacciones subyacente y el flujo de datos punto a punto. (12)

1.3. Sistemas de gestión de flujo de trabajo

El desarrollo del presente epígrafe está encaminado a la descripción de distintos sistemas de gestión de flujo de trabajo que emplean los lenguajes XPDL y BPEL para la definición y ejecución de los procesos de negocio.

1.3.1. ProcessMaker

ProcessMaker es una solución de software de flujos de trabajo, de código abierto simple y rentable. Ayuda a las organizaciones de todos los tamaños para diseñar, automatizar e implementar procesos de negocio. Entre sus características se encuentran las mencionadas a continuación:

- La caja de herramientas ProcessMaker permite a los usuarios de negocio crear formas y mapas de flujos de trabajo completamente funcionales
- El software está completamente basado en web, lo que facilita la coordinación del flujo de trabajo entre los usuarios, departamentos y organizaciones
- Utiliza WSLD[4] para la descripción de los servicios web y una Arquitectura Orientada a Servicios (SOA por sus siglas en inglés) de gran alcance, que posibilita interconexión con sistemas que incluyen la gestión de documentos, ERP, CRM[5] y aplicaciones de inteligencia empresarial
- Utiliza BPMN en su versión 2.0 para el modelado de procesos y soporta XPDL para la definición de procesos así como BPEL como lenguaje de ejecución de procesos (13) (14)

1.3.2. TIBCO

TIBCO en un software de modelado de negocio basado en los estándares que permite a los expertos en negocios modelar, implementar y manejar procesos de negocios. Business Studio es el entorno de

[4] Del inglés WSDL Web Services Description Languaje

[5] Del inglés Customer Relationship Management. Es un modelo de gestión de toda la organización, basada en la orientación al cliente.

desarrollo basado en modelos para TIBCO iProcess Suite para BPM y TIBCO ActiveMatrix suite para SOA. Entre sus características se encuentran:

- TIBCO Business Studio proporciona modelos BPM de gran calidad es un ambiente desarrollo basado en Eclipse para la construcción de aplicaciones compuestas a través de los software **TIBCO's BPM** y **SOA**
- Proporciona un entorno más colaborativo que permite a los usuarios de negocio modelar, simular y gestionar los procesos de negocio de forma correcta.
- Es compatible con los estándares y soporta BPMN y XPDL, puede implementar fácilmente los modelos que los usuarios han creado
- Debido a la creciente complejidad de la orquestación en un formato basado en estándares para las empresas que deseen llevar su arquitectura orientada a servicios (SOA) para el siguiente nivel es implementado en la versión 5.4 WS-BPEL 2.0. (15) (16)

1.3.3. JbossjBPM

JbossjBPM no utiliza XPDL para la definición de procesos de negocios, pero tiene soporte para este estándar, en sustitución a este usan JPDL[6], lenguaje propio desarrollado por sus creadores para esta solución, su diseñador está basado en Eclipse, no usa la nomenclatura BPMN pero no deja de ser herramienta muy completa y poderosa. Viene con una consola sobre JBOSS[7], completamente modificable y siempre sobre Hibernate, eso permite correr el flujo de trabajo sobre cualquier base de datos. En su versión 3.0 adiciona una extensión jBPM BPEL para fusionarse con el lenguaje de ejecución BPEL. (17) (12) (18)

1.3.4. ADONIS

ADONIS es un software de Gestión de Procesos de Negocios desarrollado por la compañía BOC Group. Se encuentra soportado sobre el marco de trabajo ADONIS Process Portal o APP, este complementa a ADONIS mediante un sencillo acceso web online permitiendo con esto que cada rol tenga una interfaz definida y las funciones que requieran para el desempeño de sus tareas. APP accede directamente a la base de datos de ADONIS, posibilita la inserción de datos online y únicamente necesita un navegador Web, no hay necesidad de instalación de software en el cliente.

[6] Del inglés jBPM Process Definition Language.

[7] Es un servidor de aplicaciones de código abierto implementado en Java puro

Este tiene soporte en la modelación con diferentes estándares como BPMN, UML, EPK, LOVEM y para la implementación de procesos soporta XDPL, BPEL, XMI. ADONIS incluye un potente editor gráfico de modelación, de manejo intuitivo y fácil de aprender, que permite visualizar modelos (procesos de negocio, procedimientos, mapas de procesos, organigramas y mapas de sistemas). La reutilización de estructuras garantiza una modelación eficiente, mientras que las distintas vistas de los modelos resaltan las características del modelo en cada caso. El componente Import/Export ofrece una interfaz abierta mediante la cual es posible importar y exportar todos los modelos contenidos en ADONIS gracias a los formatos ADL[8] y XML. Esto garantiza la migración de datos entre instalaciones ADONIS distribuidas (independencia de la base de datos) y con otras aplicaciones. (19) (20)

Una vez estudiadas las herramientas son descartadas ya que aunque todas soportan XPDL para la definición de sus procesos de negocio, orientan la ejecución de los mismos a WS-BPEL utilizando WSLD[9] para la descripción de los servicios web .Esto incumple con la arquitectura orientada a servicios que implementa Sauxe ya que este implementa un IoC[10] para el consumo de servicios y la integración de componentes.

1.4. Lenguajes de definición de flujos de trabajo

La realización del presente epígrafe se encuentra orientada a la descripción de diferentes lenguajes de definición de procesos de negocio.

1.4.1. XPDL

XPDL (XML Process Definition Language) es un formato de archivo basado en XML[11] cuya función es intercambiar modelos de Procesos de Negocio entre distintas herramientas. Para llevar a cabo lo que se propone con XPDL la WfMC define un metamodelo para el mismo que cubre:

- Las entidades de más alto nivel en el dominio de la definición de procesos
- Atributos de procesos
- Agrupaciones de diferentes procesos en modelos relacionados
- Definiciones de datos comunes que puedan ser usados en variedad de modelos

[8] Del inglés ADONIS Definition Language.

[9] Del inglés WSDL Web Services Description Languaje

[10] Del inglés Inversion of Controller

[11] Del inglés Extensible Markup Language

Para todos estos aspectos se tienen dos metamodelos principales:

El metamodelo Package, se encarga de las agrupaciones de procesos, del intercambio de mensajes entre estos y de las diferentes características que poseen los mismos. (21)

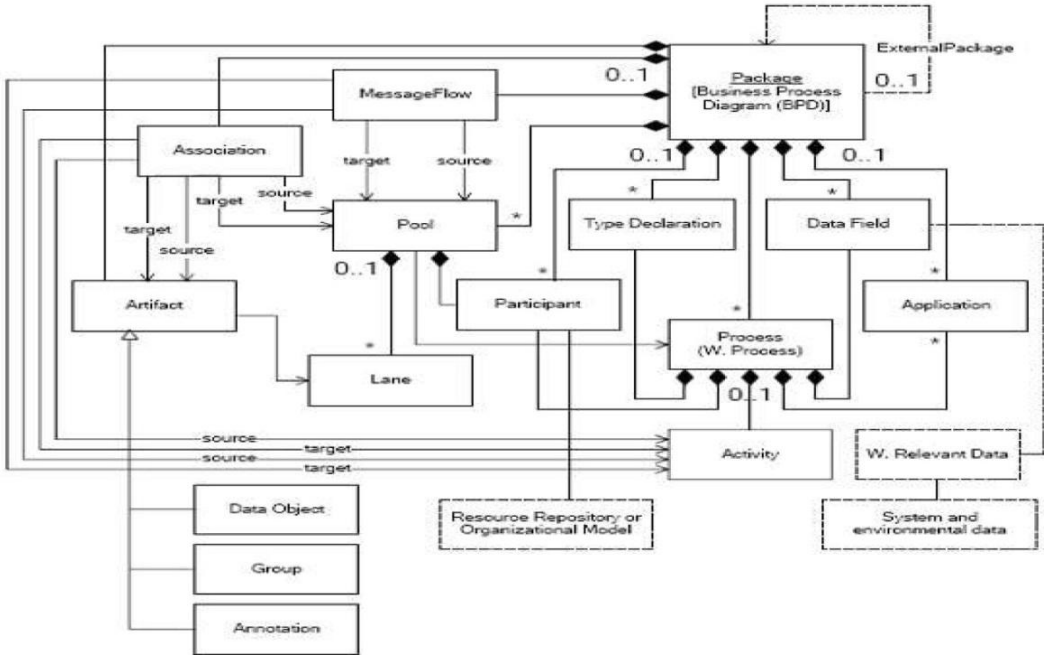

Figura 1: Metamodelo Package

El metamodelo Process que describe las principales entidades que componen un proceso así como los atributos de estas.

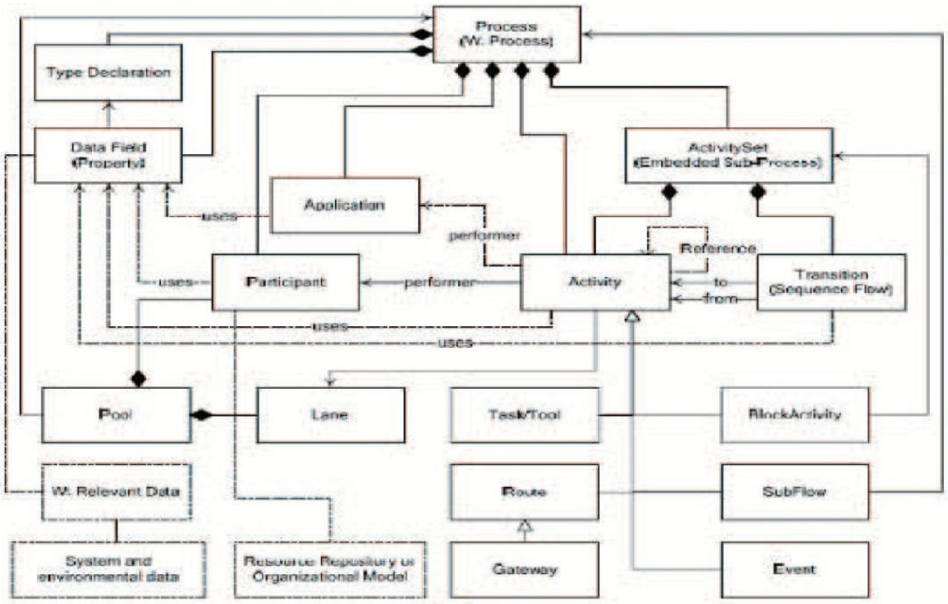

Figura 2: Metamodelo Process

Para una mejor comprensión se muestra en la figura 3 una integración de los metamodelos anteriormente ilustrados. (22)

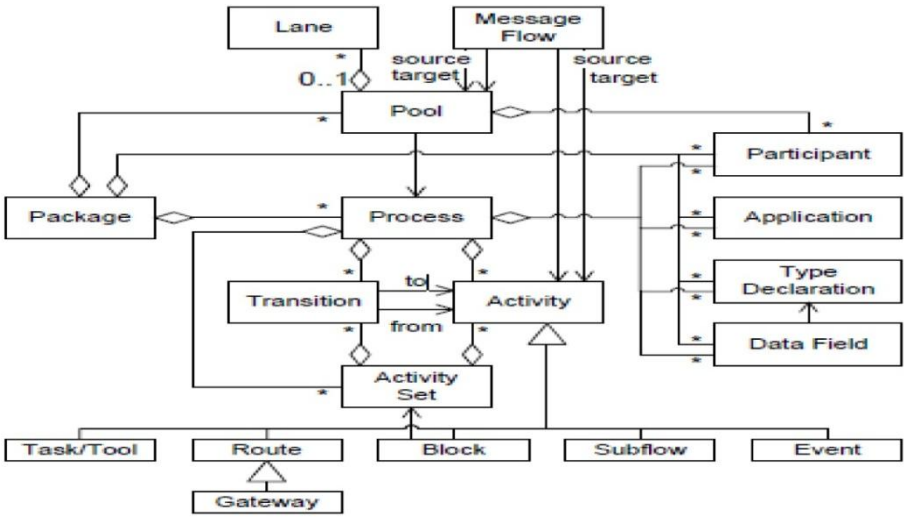

Figura 3: Integración de los metamodelos Package y Process

1.4.2. YAWL

YAWL (del inglés Yet Another Workfow Languaje) es un lenguaje para la definición de flujos de trabajo. En el mismo toda definición de flujo de trabajo está compuesta de tareas y condiciones:

- Una tarea es una representación abstracta de un trabajo a realizar (ej. invocación de una operación de un servicio Web)
- Una condición es un requisito que se debe cumplir para poder continuar con el flujo de trabajo (ej. que una tarea previa se haya completado)

En la Figura4 se presentan algunos de los símbolos utilizados en YAWL para modelar flujos de trabajo. (23)

Nombre y Símbolo	Descripción
Condición de Entrada	Marca el inicio de un flujo de trabajo.
Condición de Salida	Marca el fin de un flujo de trabajo.
Tarea Atómica	Representa una tarea.
Tarea AND-Split	Especifica que una vez completada la tarea todas las tareas subsecuentes podrán ejecutarse.
Tarea XOR-Split	Especifica que una vez completada la tarea sólo una tarea subsecuente podrá ejecutarse.
Tarea OR-Split	Especifica que una vez completada la tarea una o más tareas subsecuente podrán ejecutarse.
Tarea AND-Join	Especifica que la tarea se ejecutará sólo cuando todas sus tareas precedentes se hayan completado.
Tarea XOR-Join	Especifica que la tarea se ejecutará una vez que una de las tareas precedentes se complete.
Tarea OR-Join	Especifica que la tarea se ejecutará sólo cuando todas las tareas previamente activadas se completen.

Figura 4: Elementos del lenguaje gráfico de YAWL

1.4.3. JPDL

JPDL (Java Process Definition Language) es un lenguaje específico definido por el equipo de desarrollo de JBoss para los procesos definidos en Java., permite la descripción de procesos de negocio, mediante la definición de tareas y actividades humanas a través de un lenguaje orientado a grafos. Se trata de un lenguaje maduro y estable, es utilizado por el motor jBPM (aunque a partir del 2007 también incorpora una extensión de BPEL). (17)

1.5. Lenguajes de ejecución de flujos de trabajo

El desarrollo del presente epígrafe se encuentra orientado a la descripción de diferentes lenguajes de ejecución de procesos de negocio.

1.5.1. BPEL4WS

BPEL4WS (del inglés Business Process Execution Lenguaje for Web Services) es un lenguaje de ejecución diseñado para proporcionar una definición de orquestación de servicios web. La orquestación de servicios web debe ser dinámica, flexible y adaptable para descubrir necesidades de cambios de negocio. Una separación clara entre la lógica de proceso y los servicios web usados fomentan la flexibilidad.

BPEL proporciona una gramática basada en XML para la descripción de la lógica, para controlar y coordinar los servicios Web que participan en un flujo de proceso. Esta gramática puede ser interpretada y ejecutada por un motor BPEL, que está controlado por una de las partes de negocio participantes. El motor coordina todas las actividades en el proceso y controla las actividades de corrección del sistema cuando se producen excepciones. BPEL se basa en XML y se extiende y las especificaciones de servicios web (10) (22)

Los principales conceptos de BPEL son actividades básicas, actividades estructuradas, variables, enlaces asociados y controladores. A continuación se ilustran los conceptos principales de BPEL y sus interrelaciones mediante un diagrama de clases UML. (22)

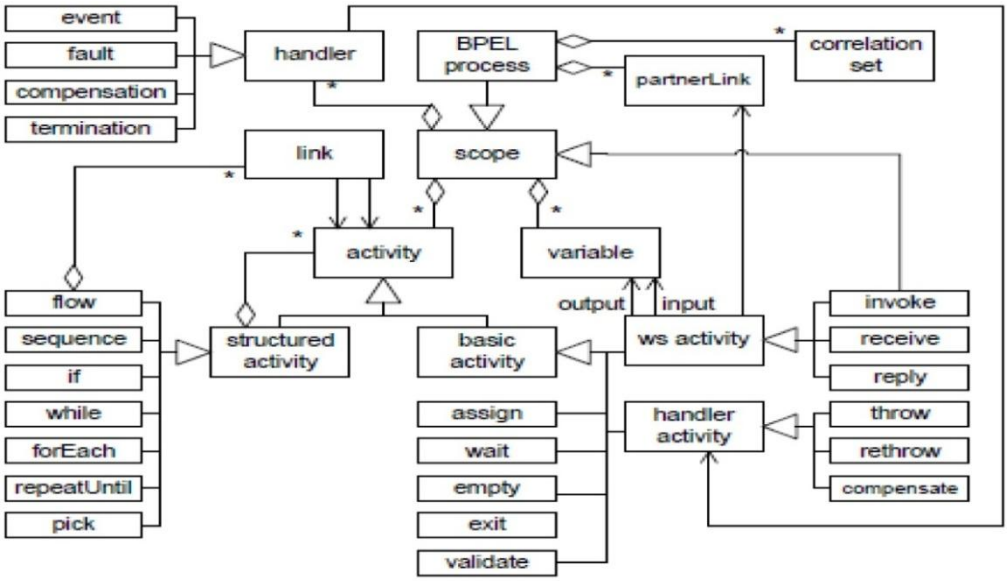

Figura 5: Meta-modelo de BPEL

En cuanto a la capacidad de representación de los Patrones de flujo de trabajo, BPEL representa menos patrones que los modelos orientados a la definición. Entre las limitaciones que presenta en

cuanto a la expresividad cabe destacar que no puede representar patrones de MultiFusión, es decir, no permite la fusión de ramas heterogéneas, patrones de Ciclos Arbitrarios, por lo que no permite la definición de ciclos de cualquier tipo, ni la definición de hitos es decir, que impide representar transiciones dependientes de hitos alcanzados.

La principal ventaja que presenta es que es un lenguaje estándar que se ha convertido en el lenguaje de referencia para la interacción de Servicios web. Tanto es así que la mayoría de los motores del mercado tienen un módulo que les permite cargar módulos BPEL. (12)

1.6. Patrones de flujos de trabajo

Los patrones de flujo de trabajo han tenido un impacto significativo en el campo de la tecnología de flujo de trabajo. Son utilizados para una variedad de propósitos, incluyendo la selección y evaluación de soluciones de flujo de trabajo, la mejora del diseño de las ofertas comerciales y de código abierto, además de garantizar el poder expresivo de cualquier lenguaje de definición de procesos. La iniciativa Workflow Patterns realizada por el profesor Wil van der Aalst, en su investigación ha identificado 43 patrones de control de los cuales en la solución presentada son implementados 3 más de los ya posee ezComponents sumando 10 patrones de flujo de trabajo , con el fin de dar solución a las limitaciones que esta presenta en cuanto a la cancelación dentro del flujo de trabajo y además resolver la situación problemática antes planteada, los mismos se muestran a continuación: (11) (24)

Tabla 1: Patrones de flujo de trabajo

Patrón	Descripción
Sequence	El patrón de secuencia sirve como bloque de construcción fundamental para los procesos. Se utiliza para construir una serie de tareas consecutivas que se ejecutan a su vez, una tras otra. Dos tareas de formar parte de una secuencia si hay un borde de flujo de control de una de ellas a la siguiente, que no tiene guardias o condiciones asociadas con ella.
Parallel Split	Un punto en el proceso donde un simple flujo de control se divide en dos o más flujos de control que pueden ser ejecutados en paralelo (actividades pueden ser ejecutadas simultáneamente o en cualquier orden).

Synchronization	Un punto en el proceso donde múltiples actividades convergen en un único flujo de control, sincronizando múltiples flujos. Cada flujo de control de entrada a un sincronizador es ejecutado una sola vez, y todos necesitan ser completados para que el sincronizador continúe el flujo de control.
Exclusive Choise	Un punto en el proceso donde basado en la evaluación de datos del proceso (variables de instancia del proceso), uno de varios flujos alternativos es seleccionado.
Simple Merge	Un punto en el proceso donde dos o más flujos alternativos se unen sin sincronización(Supuesto: los flujos alternativos no se ejecutan en paralelo)
Multi-Choice	El modelo multi-elección consiste en la divergencia de una rama en dos o más ramas de tal manera que cuando la rama entrante está activado, el hilo de control se pasa inmediatamente a una o más de las ramas salientes basado en un mecanismo que selecciona una o más ramas salientes. Este patrón es esencialmente un análogo del patrón elección exclusiva, en el que múltiples ramas salientes se puede habilitar.
Synchronizing Merge	Una forma de sincronización en la cual la ejecución procede si y sólo si uno de los flujos o caminos de entrada ha finalizado. • Si se ejecutó más de un camino, estos son sincronizados • Si se ejecutó un sólo camino, se continúa la ejecución sin sincronización
Implicit Termination	Un proceso debería ser finalizado cuando no existan más actividades habilitadas para su ejecución
Cancel Task	Una actividad habilitada es deshabilitada, es decir, el flujo

	esperando por la ejecución de la actividad es removido.
Cancel Case	Una instancia de proceso es cancelada y removida completamente (aún si existen partes del proceso que son instanciadas varias veces).

1.7. Mapeo de XPDL a BPEL

Con el objetivo de garantizar la comprensión y traducción de los puntos comunes que presentan los lenguajes de definición XPDL y BPEL, fue realizado un mapeo de los mismos, el cual arrojó los resultados que se muestran en la siguiente tabla. (22) (25)

Tabla 2: Mapeo de XPDL a BPEL

XPDL	Significado	BPEL	Significado
Process -id -name -priority -duration -effectiveDate -expiryDate -author	Un proceso de negocio que contiene uno o más de los objetos siguientes	**Process** objects name targetNamespace	Un proceso de negocio que contiene uno o más de los objetos siguientes
Transition -name -id -toActivity -fromActivity -condition -type	Representa el control en el flujo de trabajo y son los arcos entre las actividades.	**Link** -name	Sincronización de servicios en el proceso
TransitionRestrictions -Type(Exclusive, Inclusive, Parallel) -ExclusiveType -IncomingCondition		**sequence**	Actividad que las representa

TransitionRestrictions Split -Transition Refs (lista de transiciones salientes de una actividad) -Type **-**ExclusiveType **-**OutgoingCondition	La restricción de transición de división se utiliza para especificar el orden en el que las transiciones salientes son evaluadas.		
TransitionRestrictions (OR: Exclusive, OR: Inclusive)	Restrinciones de las transiciones	**switch, if, else if**	Actividades que las representa
TransitionRestrictions (AND: Parallel)		**flow**	Actividad que las representa
Activity -name -id -duration -priority		**Activity** Name id	
Block Activity	Una actividad de bloque es un conjunto de actividades que tienen su propio ámbito de ejecución.	**Sequence ,switch ,while**	Actividades estructurales de BPEL
Route Activity	Reenvía a controlar las actividades dirigidas por sus transiciones salientes; actividades de ruta se utiliza típicamente para administrar el flujo de control.	**Pick, flow, empty, If**	
TypeDeclaration	Declaración de los tipos de datos a usar declarados en una referencia externa.	**XML.SchemaType**	Define que se usara el esquema XML para el intercambio de

			mensajes y demás acciones a ejecutar en el proceso.
Task		**Invoke, Recive, Reply**	
Assigment	Cambia el valor a una propiedad	**Assing**	Se puede utilizar para copiar datos de una variable a otra, así como para construir e insertar nuevos datos utilizando expresiones
DataField -dataType -name -id -type -isArray -length -initialValue	Variables del proceso	**Variable** -name -element -type	Propiedades de un proceso
ActivitySet -id	Espacio donde se realizan las actividades	**Scope** -isolated(yes/no) exitOnStandarFault(yes/no) -estándar Attributes	Ámbito donde se realizan las actividades
MessageFlow	Flujo de mensajes entre las actividades	**CorrelationSet** -name -properties	Propiedades compartidas por un conjunto de mensajes
Participant -id -name -type		**PartnerLink** -name -PartnerRole -myRole -Type	Relaciones de intercambio de mensajes

PartnerLinkType -name -id -role -RoleName -PortType	Define la información general del PartnerLink y permite a varios procesos utilizar el mismo PartnerLinkType	**partnerLinkType** -roles	Caracteriza la relación de intercambio de mensajes entre los servicios.
PartnerLink -name -id -MyRole -MyRoleRoleName -PartnerRole -EndPoint -PartnerRole RoleName -ServiceName -PortName	Se define el papel que el proceso va a jugar y el papel de la pareja va a usar. Myrole elemento define el papel que el proceso se está reproduciendo y el elemento PartnerRole define el papel de la pareja se está reproduciendo.	**partnerLink** -name -PartnerRole -myRole -Type	Relaciones de intercambio de mensajes
Loop **-**LoopType(Standard o MultiInstance) **Standard** **-** LoopCondition - LoopCounter - LoopMaximum - TestTime **MultiInstance** -LoopCounter -MI_Condition	ciclos	**repeatUntil** **While** **forEach**	Ciclos y actividades para patrones de WF

-MI_Ordering -MI_FlowCondition			
Event **StartEvent** **intermediateEvent** **EndEvent**	Tipos de eventos que presenta	• **catch** • **catchAll,** • **compensate,** • **compensation Handler** • **eventHandlers** • **exit** • **faultHandlers** • **onEvent** • **onAlarm** • **terminationHandler** • **throw** • **wait**	Controladores de eventos y todo lo relacionado a el manejo de eventos (controladores de eventos y llamadas)
Package -id -name -createDate -author	Contiene la información asociada con la definición de procesos	**Process**	

1.8. Modelo de desarrollo de software propuesto por CEIGE

La Universidad de las Ciencias Informáticas (UCI), cuenta con el Centro de Informatización de la Gestión de Entidades (CEIGE), que desarrolla productos y servicios asociados para la gestión integral de entidades. A partir de las características del centro se definió el Modelo de desarrollo de software para el CEIGE.

La estructura organizativa del modelo de desarrollo permite que cada departamento lleve a cabo un modelo de gestión que permite la orientación a dominios del negocio, la especialización y reutilización de los recursos humanos en varios proyectos y el aumento de la productividad. Todo esto con el propósito de mantener entre dichos departamentos un mejor balance de los compromisos y recursos tanto humanos como materiales.

De ahí que se define el modelo de ciclo de vida de los proyectos de CEIGE, que describe las fases por las que transitarán los proyectos de desarrollo de software. El mismo tiene en cuenta las actividades de cada una de las fases y áreas de procesos que plantea el nivel dos de CMMI. (26)

Figura 6: Etapas del modelo de desarrollo

1.9. Tecnologías y herramientas para el desarrollo (Vita ARQ-Entorno de desarrollo tecnológico)

1.9.1. Herramientas CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de software Asistida por Computadora) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero.

La tecnología CASE supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información y se plantean los siguientes objetivos:

- Permitir la aplicación práctica de metodologías estructuradas, las cuales al ser realizadas con una herramienta se consigue agilizar el trabajo
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.

- Simplificar el mantenimiento de los programas.
- Mejorar y estandarizar la documentación.
- Aumentar la portabilidad de las aplicaciones.
- Facilitar la reutilización de componentes software.
- Permitir un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos. (27)

1.9.1.1. Visual Paradigm 8.0

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una construcción más rápida de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (27)

1.9.2. Herramientas para el desarrollo colaborativo

1.9.2.1. Subversión

Subversión es un sistema de control de versiones diseñado específicamente para reemplazar al popular CVS del inglés (Concurrent Versions System). Es software libre bajo una licencia de tipo Apache_Bsd y se le conoce también como svn por ser el nombre de la herramienta utilizada en la línea de comando.

Una característica importante de Subversion es que, a diferencia de CVS, los demás archivos con versionamiento no tienen cada uno un número de revisión independiente, en cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en un instante determinado del repositorio que se está trabajando. Presenta además las siguientes características:

- Mantiene versiones no sólo de archivos, sino también de directorios
- Mantiene versiones de los metadatos asociados a los directorios.
- Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
- Es un registro oficial de eventos durante un rango de tiempo en particular.
- Es un sitio centralizado donde se almacena y mantiene información, habitualmente bases de datos o archivos informáticos.

- Atomicidad de las actualizaciones, una lista de cambios constituye una única transacción o actualización del repositorio, esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
- Soporte tanto de ficheros de texto como de binarios.
- Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos. (28)

1.9.2.2. RapidSVN

RapidSVN es un cliente gráfico de Subversión multiplataforma. Que se distribuye bajo la Licencia Pública General de GNU. Entre sus características se encuentran:

- **Simple** - proporciona una interfaz fácil de usar para las características de Subversion
- **Eficiente** - simple para los principiantes pero lo suficientemente flexible como para aumentar la productividad para los usuarios de Subversion con experiencia
- **Portable** - se ejecuta en cualquier plataforma en la que Subversion y wxWidgets puede ejecutar: Linux, Windows, Mac OS / X, Solaris.
- **Rápido** - completamente escrito en C + +
- **Multilingüe** - que ha sido traducido a muchos idiomas ya: alemán, francés, italiano, portugués, ruso, ucraniano, chino simplificado, japonés
- **Soporte** completo para Unicode (28) (29)

1.9.2.3. Morzilla Firefox 12

Es un Navegador web libre descendiente de Mozilla Application Suite, desarrollado por la Corporación Mozilla, la Fundación Mozilla y un gran número de voluntarios externos. Es un navegador multiplataforma y está disponible en varias versiones de Microsoft Windows, Mac OS X, GNU/Linux y algunos sistemas basados en Unix. Su código fuente es software libre, publicado bajo una triple licencia GPL/LGPL/MPL. (30)

1.9.2.4. Apache 2.0

Apache es un servidor web HTTP de código abierto multiplataforma que implementa el protocolo HTTP/1.12. Es altamente configurable y su capacidad modular permite 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. (31)

1.9.3. Herramientas para el desarrollo

1.9.3.1. PostgreSQL

PostgreSQL es un sistema gestor de base de datos basado en software libre y liberado bajo la licencia BSD, esto significa la reutilización de su código fuente, modificarlo a voluntad y redistribuirlo libremente. Está considerado como el sistema de gestión de bases de datos de código abierto más potente del mercado, ofrece nuevos conceptos como son: clases, herencia, tipos y funciones. En cierta forma aporta potencia y flexibilidad adicional como son las restricciones, los disparadores, las reglas y la integridad transaccional, posee una amplia variedad de tipos nativos, o sea, presenta soporte para números de precisión arbitraria, texto de largo ilimitado, figuras geométricas con una variedad de funciones asociadas, direcciones IP, arreglos entre otros. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (32)

1.9.3.2. Netbeans 7.1

Netbeans es un entorno de desarrollo integrado, libre hecho principalmente para el lenguaje de programación java. Existe además un número importante de módulos para su extensión. Netbeans IDE1 es un producto libre y gratuito sin restricciones de uso. La plataforma Netbeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de Netbeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma Netbeans pueden ser extendidas fácilmente por otros desarrolladores de software. (33)

1.9.4. Librerías y marcos de trabajo

1.9.4.1. Sauxe 2.0

Es un marco de trabajo desarrollado con tecnologías libres que contiene un conjunto de componentes reutilizables. Presenta una arquitectura en capas e implementa el patrón arquitectónico Modelo Vista Controlador en su capa superior.

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. (34)

1.9.4.2. ExtJS 3.3.1

ExtJS es una librería JavaScript que permite construir aplicaciones complejas en Internet. Esta librería incluye:

- Componentes UI del alto performance y personalizables.
- Modelo de componentes extensibles.
- Un API fácil de usar.
- Licencias Open source y comerciales.

Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts similar al que provee Java Swing, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, y otros).
Además la ventana flotante que provee ExtJS es excelente por la forma en la que funciona. Al moverla o redimensionarla solo se dibujan los bordes haciendo que el movimiento sea fluido lo cual le da una ventaja tremenda frente a otros. (34) (35)

1.9.4.3. Zend Framework 1.4

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.Utiliza el patrón Modelo-Vista-Controlador, incluye objetos de las diferentes bases de datos, por lo que no es necesario escribir ninguna consulta SQL para acceder a la suya. Una solución para el acceso a base de datos que balancea el ORM (Mapeo Objeto-Relacional) con eficiencia y simplicidad. (36)

1.9.4.4. Doctrine ORM

Doctrine es un mapeador de objetos relacional (ORM) escrito en PHP que proporciona una capa de persistencia para objetos PHP. Es una capa de abstracción que se sitúa justo encima de un SGBD. Una característica de Doctrine es el bajo nivel de configuración que necesita para empezar un proyecto. Puede generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas. No es necesario generar o mantener complejos esquemas XML de base de datos como en otros frameworks.
Otra característica importante de Doctrine es la posibilidad de escribir consultas de base de datos utilizando un dialecto de SQL denominado **DQL** (Doctrine Query Language) que está inspirado en Hibernate (Java).

Otras características notables de Doctrine son:

- Soporte para datos jerárquicos;
- Soporte para hooks (métodos que pueden validar o modificar las escrituras y lecturas de la base de datos) y eventos para manejar la lógica de negocio relacionada
- Herencia
- Un framework de caché que utiliza diversos motores como memcached, SQLite o APC
- Diversos comportamientos del modelo (conjuntos anidados, internacionalización, log, índice de búsqueda)
- Una función "compilar" que combina varios archivos PHP del framework en uno solo para evitar el descenso de rendimiento que provoca incluir varios archivos PHP. (34) (37)

1.9.5. Lenguajes de modelado y desarrollo

1.9.5.1. Lenguajes de Modelado

UML

El Lenguaje Unificado de Modelado (UML, del inglés Unified Modeling Lenguage) es un lenguaje de modelado de sistemas de software y está respaldado por el OMG. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Ofrece un estándar para describir una visión del sistema (meta-modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquema de bases de datos y componentes reutilizables. (5)

1.9.5.2. Lenguaje de desarrollo

PHP 5.3

Lenguaje de programación diseñado originalmente para la creación de Página web dinámicas. Posee entre sus características:

- Es un lenguaje interpretado, solo se necesita un navegador web para ejecutarlo.
- Es un lenguaje del lado del servidor, por lo que los script se ejecutan remotamente y el resultado aparece en la máquina cliente (local).
- Tiene soporte para muchos tipos de bases de datos, entre las principales están MySQl, PostgreSQL, SQLite, entre otras.
- Es embebido en código HTML.
- Soporte de orientación a objetos. (38)

Entre sus Ventajas se encuentran:

- 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.
- Posee una biblioteca nativa de funciones sumamente amplia e incluida.

JavaScript 3.0

El JavaScript es un lenguaje de programación interpretado, lo que significa que no necesita ser compilado. Proviene del Java y se utiliza principalmente para la creación de páginas web. (41)

Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuarios y páginas web dinámicas, en bases de datos locales al navegador. Aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS).

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM). (39)

1.10. Conclusiones

Al terminar este capítulo se arribó a las siguientes conclusiones:

- Una vez realizado el marco teórico acerca de las herramientas de gestión de flujos de trabajo se concluye que aunque todas las herramientas analizadas en su ciclo de gestión transforman del lenguaje de definición XPDL a el lenguaje de ejecución BPEL, el BPEL generado utiliza un WSLD para la descripción de servicios web incumpliendo esto con la arquitectura orientada a servicios que implementa Sauxe ya que este utiliza un IoC para la integración de componentes y el consumo de servicios.

- Se realizó una estudio de los principales estándares en lenguajes de definición y ejecución de procesos de negocio , identificando posibles áreas de reutilización evidenciadas en el mapeo realizado de los lenguajes XPDL y BPEL
- Se realizó un estudio de los diferentes patrones de flujo de trabajo existentes identificando 10 patrones a utilizar por la solución propuesta
- Se establecen los requerimientos y herramientas necesarias a tener en cuenta en el desarrollo de la solución propuesta

2. CAPÍTULO 2: Análisis y Diseño

2.1. Introducción

El siguiente capítulo parte de los conocimientos adquiridos en la fundamentación teórica. Se ilustra el modelo conceptual con el objetivo de esclarecer los conceptos fundamentales con los que se trabajarán en el desarrollo de la aplicación y seguido los requisitos funcionales y no funcionales de la

solución para lograr un mejor entendimiento de la misma. Posteriormente se realiza una descripción del diseño elaborado que servirá de entrada para la posterior implementación.

2.2. Requisitos de Software

Un requisito es una “condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado”. También se aplica a las condiciones que debe cumplir o poseer un sistema o uno de sus componentes para satisfacer un contrato, una norma o una especificación. (40)

2.2.1. Identificación de requisitos

En el desarrollo de software los requisitos tienen un papel importante ya que son la condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de software para satisfacer un contrato, estándar, u otro documento impuesto formalmente. Estos se pueden dividir en requisitos funcionales y no funcionales. Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Los no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Para la captura de requisitos se utilizaron las técnicas descritas a continuación:

- **Tormenta de ideas** que no es más que reuniones en grupos cuyo objetivo es que cada participante muestre su idea libremente. Para esto se realizaron talleres con los integrantes del equipo de desarrollo tanto como los tutores con el objetivo de debatir del tema y así definir las principales funcionalidades.
- **Entrevista con el cliente:** Las entrevistas fueron realizadas con el fin de conocer sus exigencias con mayor claridad.

2.2.2. Especificación de requisitos

2.2.2.1. Requisitos Funcionales

Los requisitos funcionales identificados son enuncian a continuación:

- El sistema debe:
    - ✓ RF1: Cargar archivo XPDL

- ✓ RF2: Reconocer el patrones de flujo de trabajo
- ✓ RF3: Transformar patrones de flujo de trabajo del lenguaje de definición XPDL al lenguaje de ejecución BPEL
- ✓ RF4: Generar archivo XML
- ✓ RF5: Guardar archivo XML

RF1: Cargar el archivo XPDL

Tabla 3: RF1 Cargar el archivo XPDL

Precondiciones	Debe existir un archivo XPDL
Flujo de eventos	
Flujo básico Cargar archivo XPDL	
1	El usuario accede a la opción cargar XPDL
2	El sistema devuelve un de objeto de tipo XPDL
3	El sistema muestra un mensaje de¨ Archivo Cargado ¨
Pos-condiciones	
1	Se guarda el objeto en una variable
2	
Flujos alternativos	
Flujo alternativo 1.a No se cargó correctamente el archivo	
1	El sistema muestra un mensaje de error.
	Flujo alternativo 2.a El usuario presiona la opción Cancelar
1	El sistema cancela la operación.
Pos-condiciones	
1	El sistema no carga el archivo
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual

Conceptos	**XPDL**	Visibles en la interfaz: Botón cargar Label contenedor del archivo Utilizados internamente: Etiquetas
Requisitos especiales	Lenguajes de definición de procesos de negocio	
Asuntos pendientes	Posibles mejoras al requisito.	

RF2: Reconocer los patrones de flujo de trabajo

Tabla 4: RF2 Reconocer los patrones de flujo de trabajo

Precondiciones	Se debe haber cargado el archivo XPDL
Flujo de eventos	
Flujo básico Reconocer patrones de flujo de trabajo	
4	Se realiza una llamada a la función getActivities()
5	El sistema devuelve una arreglo de objetos Activities
6	Se realiza una llamada a la función getTransitions()
7	El sistema devuelve una arreglo de objetos Transitions
8	Se realiza una llamada a la función getTransitionsRef()
9	El sistema devuelve una arreglo de objetos TransitionsRef
10	Se realiza una llamada a la función readWorkflowPatterns () de la clase TranslatorBPEL
Pos-condiciones	
3	Transformar los patrones de XPDL a BPEL
4	
Flujos alternativos	
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>> <Condición que dio lugar a la extensión>	

2	Seleccionar módulo.	
3		
Pos-condiciones		
2	N/A	
Validaciones		
2	Se validan los datos según lo establecido en el Modelo conceptual	
Conceptos	**Patrones de Workflow**	Utilizados internamente: Etiquetas
Requisitos especiales	Patrones de flujo de trabajo	
Asuntos pendientes	Posibles mejoras al requisito.	

RF3: Transformar patrones de flujos de trabajo del lenguaje de definición XPDL al lenguaje de ejecución BPEL

Tabla 5: RF3 Transformar patrones de flujos de trabajo del lenguaje de definición XPDL al lenguaje de ejecución BPEL

Precondiciones	Debe existir un archivo XPDL cargado en el sistema Se deben haber reconocido los patrones de Workflow
Flujo de eventos	
Flujo básico Transformar patrones de flujos de trabajo del lenguaje de definición XPDL al lenguaje de ejecución BPEL	
11	Se realiza una llamada a la función readWorkflowPatterns () de la clase TranslatorBPEL.

12	El sistema devuelve un BPELModel
Pos-condiciones	
5	Quedan transformados los objetos de tipo patrones a objetos de tipo BBEL
Flujos alternativos	
Flujo alternativo1.a El sistema genera un error a la hora de transformar	
4	El sistema lanza una excepción de error
Pos-condiciones	
3	No son transformados los patrones a BPEL
Validaciones	
3	Se validan los datos según lo establecido en el Modelo conceptual
Conceptos	**BPEL** Utilizados internamente: Etiquetas
Requisitos especiales	Patrones de Workflow
Asuntos pendientes	Posibles mejoras al requisito.

RF4: Generar archivo XML

Tabla 6: RF4 Generar archivo XML

Precondiciones	Se debe haber transformado los patrones de flujo de trabajo de XPDL a BPEL.

Flujo de eventos	
Flujo básico Generar archivo XML	
13	Se realiza una llamada a la función readWorkflowPatterns () de la clase TranslatorBPEL.
14	El sistema devuelve un objeto de tipo BPELModel
15	Se realiza una llamada a la función ToXML() de la clase BaseXML
16	El sistema devuelve un objeto DomDocument
17	Se realiza una llamada a la función SaveToFile () de la clase TranslatorBPEL.
Pos-condiciones	
6	Se generó el XML
7	
Flujos alternativos	
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>> <Condición que dio lugar a la extensión>	
5	N/A
6	N/A
Pos-condiciones	
4	N/A
Validaciones	
4	Se validan los datos según lo establecido en el Modelo conceptual
Conceptos	**XML** Utilizados internamente: **Etiquetas**
Requisitos especiales	Patrones de flujo de trabajo
Asuntos pendientes	Posibles mejoras al requisito.

RF5: Guardar archivo XML

Tabla 7: RF5 Guardar archivo XML

Precondiciones	Se debe de haber generado un archivo XML
Flujo de eventos	
Flujo básico Guardar archivo XML	
18	El usuario selecciona la opción guardar
19	El sistema brinda la opción de donde guardar
20	El usuario accede a la opción aceptar
21	El sistema guarda el XML
22	
Pos-condiciones	
8	Se ha guardado el XML
9	
Flujos alternativos	
Flujo alternativo 1.a El usuario presiona el botón cancelar	
7	El sistema cancela la operación de guardar
Pos-condiciones	
5	El archivo no es guardado.
Validaciones	
5	Se validan los datos según lo establecido en el Modelo conceptual
Conceptos XML	Utilizados internamente: **Etiquetas**

Requisitos especiales	Patrones de flujo de trabajo
Asuntos pendientes	Posibles mejoras al requisito.

2.2.2.2. Requisitos No Funcionales

Ya que la solución que se desea implementar esta propuesta para el marco de trabajo SAUXE, los requisitos no funcionales que debe cumplir deben acogerse a los establecidos al inicio del proceso de desarrollo. A continuación se describen algunos de estos requisitos

Tabla 8: Requisitos no funcionales

	Cliente	**Servidor**
Software	• Navegador Mozilla Firefox 3.0 o superior. • Sistema operativo Windows 98 o superior o Linux.	• Sistema operativo Linux en cualquiera de sus distribuciones. • Un servidor Apache 2.0 o superior con módulo PHP 5.3 disponible y la extensión “pgsql” incluida. • Un servidor de base de datos PostgreSQL 8.3.

Hardware	• Requerimientos mínimos: Procesador Pentium III a 1GHz con 256Mb de memoria RAM. • Tarjeta de red el mismo del anterior	• Requerimientos mínimos: Procesador Pentium IV a 2GHz de velocidad de procesamiento y 1Gb de memoria RAM. • Tarjeta de red.

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.

2.2.3. Aplicación de técnicas de validación de requisitos

La validación de requisitos 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 830 cuando es correcta, no ambigua, completa, consistente, ordenada por importancia y estabilidad, verificable, modificable y trazable.

Muchas son las técnicas para la validación de los requisitos, entre ellas se puede mencionar: auditorías, matrices de trazabilidad, generación de casos de pruebas, análisis de consistencia

automático (CASE, BD requerimientos), revisión técnica formal y el prototipado. Para validar los requisitos propuestos se utilizaron las técnicas:

- **Revisión técnica formal:** proceso manual que involucra a ambas partes, tanto cliente como equipo de desarrollo, en la revisión formal el equipo de desarrollo conduce al cliente a través de los requerimientos, explicándole las implicaciones de cada uno. Los conflictos, contradicciones, errores y omisiones deben señalarse durante la revisión y registrarse formalmente. Se revisan aspectos como consistencia, integridad, verificabilidad, comprensibilidad, rastreabilidad y adaptabilidad
- **Creación de manuales de usuario:** Este método consiste en verificar si la especificación de requisitos contiene el suficiente detalle como para preparar el manual de usuario del sistema. De no ser así, podría ocurrir que la especificación fuera incompleta
- **Generación de casos de prueba:** Este método tiene como objetivo comprobar la verificabilidad de los requisitos. Consiste en la definición de casos de prueba que permitan verificar el cumplimiento de los requisitos funcionales

2.3. Modelo conceptual

Para obtener un mejor dominio del sistema se realizó el siguiente modelo conceptual en el que se describen aquellos objetos o conceptos reales que son significativos para el problema. En el diagrama se muestra gráficamente como los procesos de negocio definidos en el XPDL, y en los que se representan una serie de patrones de flujo de trabajo, son definidos en BPEL para su posterior ejecución por la librería ezComponents. Estos patrones definen la expresividad que van a poseer BPEL y XPDL es decir que entre más patrones implementen los lenguajes mayor será su expresividad.

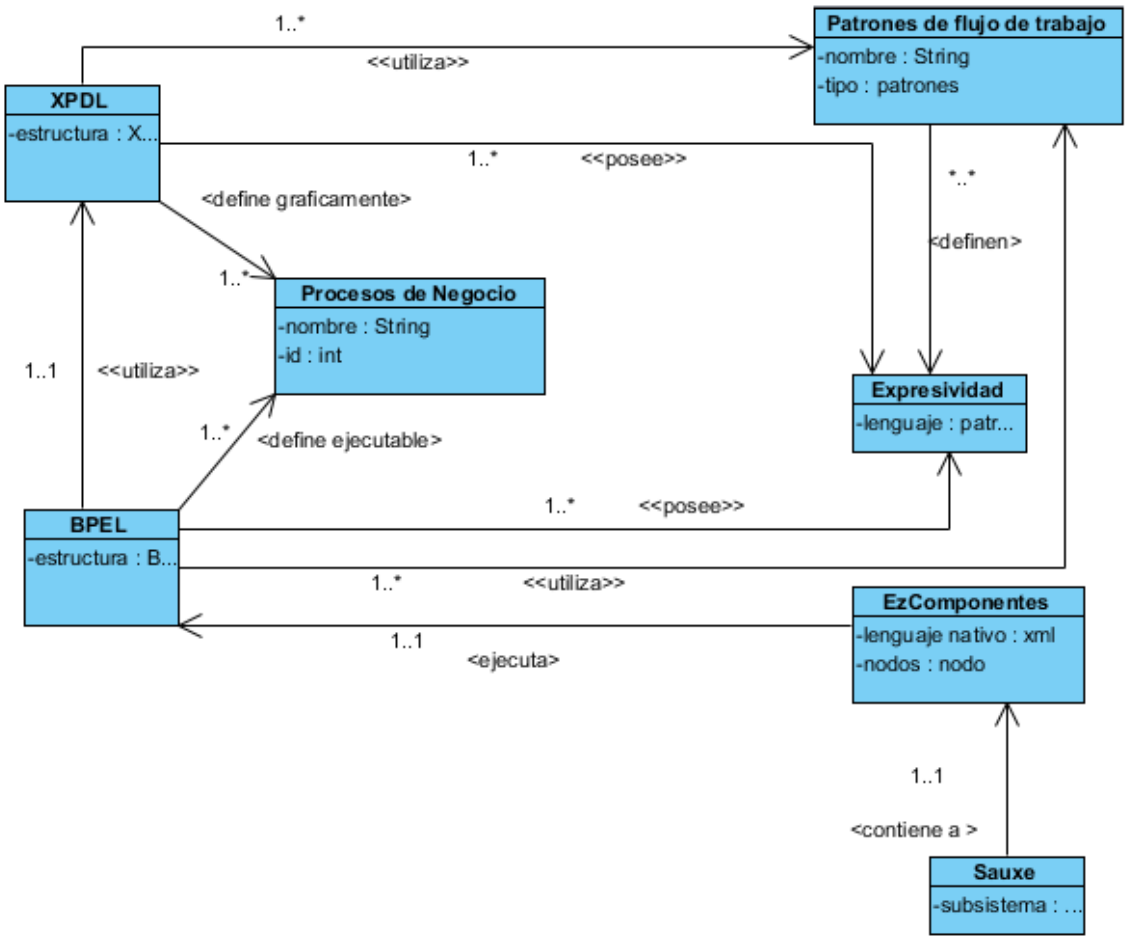

Figura 7: Modelo Conceptual

2.4. Patrón arquitectónico

E n la definición de la arquitectura fue utilizado el patrón Modelo-Vista-Controlador (MVC), el cual divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

- **Modelo** (Model): encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada
- **Vista** (View): muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador

- **Controlador** (Controller): reciben las entradas, usualmente como Eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, entre otras. Los Eventos son traducidos a solicitudes de servicio para el modelo o la vista

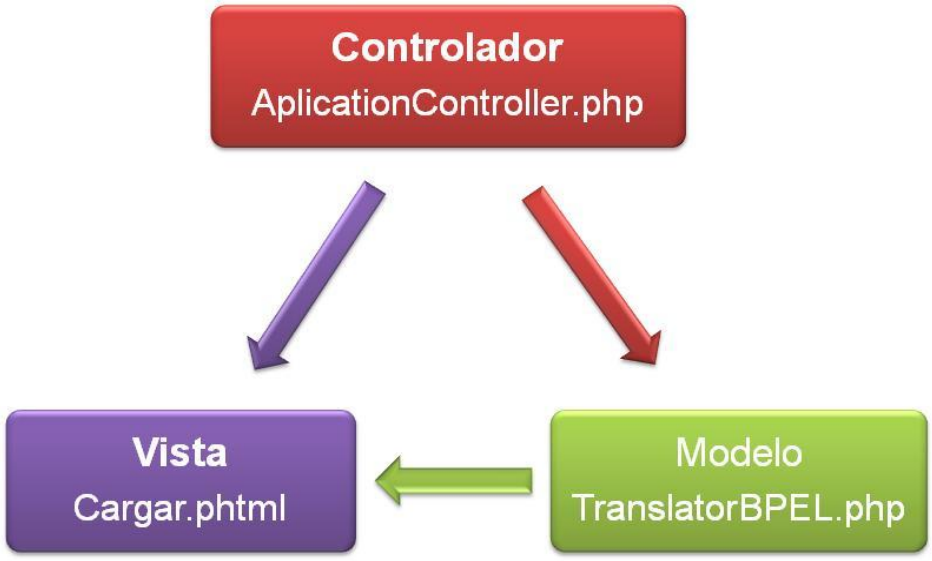

Figura 8: Patrón MVC

La arquitectura MVC fue diseñada para que los cambios realizados en una aplicación afecten lo menos posible a la programación que ya se encuentra implementada. Esto es posible gracias a que su arquitectura desacopla en diferentes capas los datos, la lógica del negocio y la lógica de presentación. Todo esto hace posible la actualización y desarrollo de cada uno de los componentes de forma independiente.

2.5. Diseño

2.5.1. Diseño de clases

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases, contiene información como asociaciones, atributos, métodos y dependencias. A continuación se muestra el diagrama de clases del diseño basado en estereotipos web que se genera durante el proceso de desarrollo.

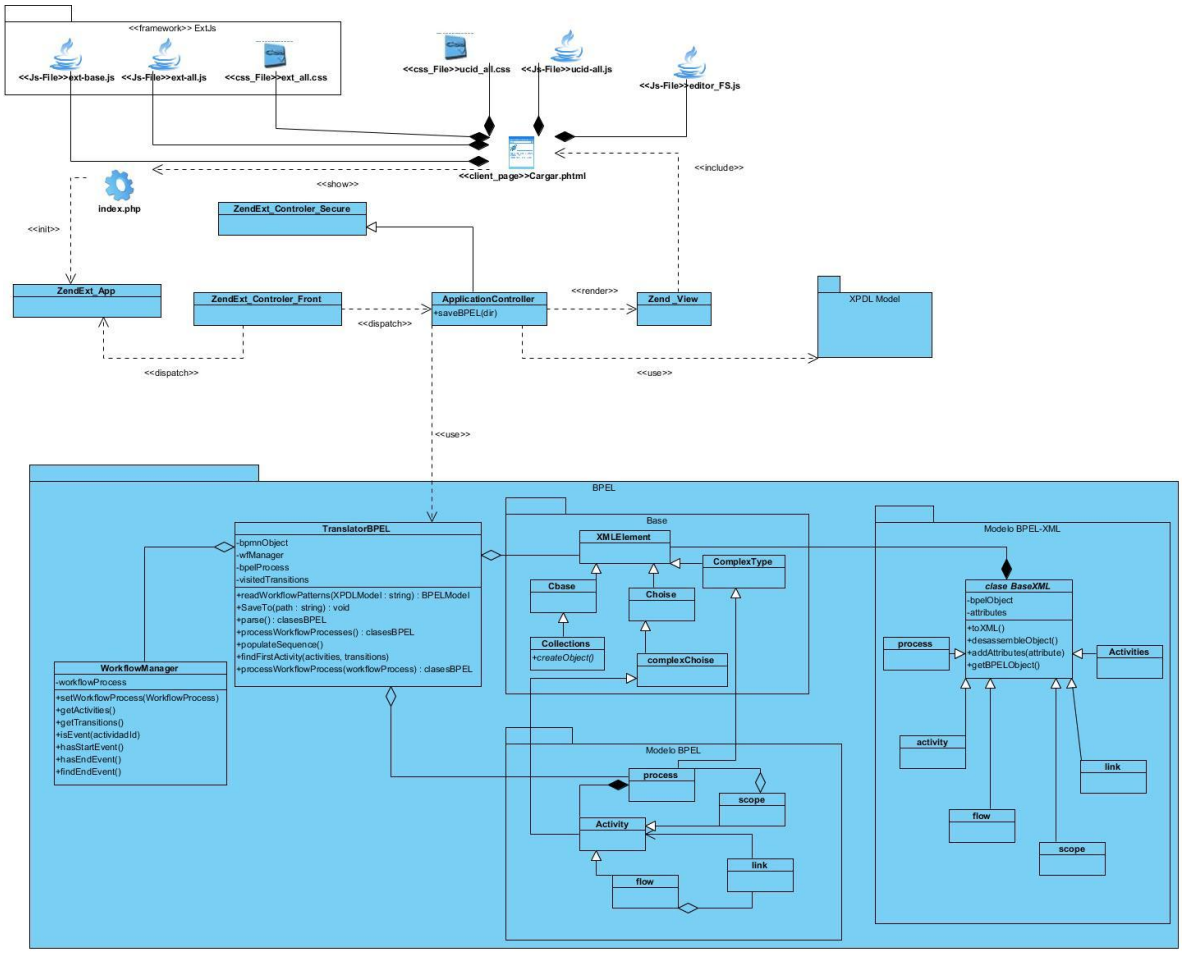

Figura 9: Diagrama de clases del diseño

En el diagrama se muestra la forma en que el componente obtiene los datos del flujo de trabajo del XPDL que recibe como entrada, pasa al reconocimiento de patrones y transformación de los mismos al lenguaje BPEL haciendo uso de la clase TranslatorBPEL y las clases que conforman al paquete ModeloBPEL implementadas para ese fin, una vez transformados los patrones TranslatorBPEL con la funcionalidad ToXML() hace uso del paquete ModeloBPEL-XML para convertir el código generado hasta el momento a XML mediante la clase Base XML de la cual heredan todas las clases que contiene dicho paquete y de esta forma generar un archivo XML con la definición ejecutable basada en BPEL del flujo de trabajo mencionado anteriormente.

2.5.2. Patrones de Diseño

Los patrones de diseño brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Los patrones de diseño no son fáciles de entender, pero una vez entendido su funcionamiento, los diseños serán mucho más flexibles, modulares y reutilizables.

Patrones GRASP

Son los patrones generales de software para asignar responsabilidades, describen en su totalidad los principios fundamentales sobre la asignación de responsabilidades a objetos. (41) A continuación se enuncian los que son utilizados en la propuesta de solución:

Experto: es el que delega responsabilidades a las clases expertas en determinada funcionalidad, este patrón se evidencia en la clase **TranslatorBPEL** la cual implementa los métodos que manejan las funcionalidades que debe realizar el sistema.

Controlador: Este patrón es el que se encarga de manejar las peticiones entre la vista y el modelo y se evidencia en la clase **AplicationController**.

Creador: este patrón se evidencia en las clases encargadas de crear objetos, por ejemplo en las clases **TranslatorBPEL**, especializada en la creación de los objetos de tipo flow, switch y demás clases del paquete Modelo BPEL.

Patrones GOF

Consisten en 23 patrones presentados por la “pandilla de los cuatro” en el libro “Design Patterns”. Gamma, Helm, Johnson, Vlissides. 1995. Son clasificados en tres grupos: de creación, estructurales y de comportamiento. A continuación se muestran los utilizados por la propuesta solución. (42)

De creación

Método Factoría: Define una interface para crear un objeto, pero deja a las subclases decidir qué clase instanciar, permite que una clase delegue en sus subclases la creación de objetos. Este patrón se evidencia en la clase **Collections** ya que las clases que heredan de ella implementan el método **createObject** pero todas no lo hacen de la misma manera.

2.6. Conclusiones

Una vez concluido el desarrollo del capítulo se concluye que:

- Se realizó el análisis y diseño del componente para para la transformación de ficheros XPDL al lenguaje de ejecución de flujos de trabajos para Sauxe ilustrando el proceso con los principales artefactos que propone el modelo de desarrollo propuesto por el centro CEIGE
- Se realizó ingeniería de requisitos a la solución propuesta para guiar la implementación, al cumplimiento de las funcionalidades a las cuales debe responder el sistema
- Se elaboró el diagrama de clases representando sus clases y relaciones, cumpliendo con las características de los patrones de diseños definidos para la solución.

3. CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1. Introducción

En este capítulo se muestra el modelo de implementación que pone en práctica el diseño de la solución realizado en el capítulo anterior y se especifica el conjunto de validaciones y pruebas que evalúan la calidad del sistema.

3.2. Implementación

3.2.1. Diagrama de componentes

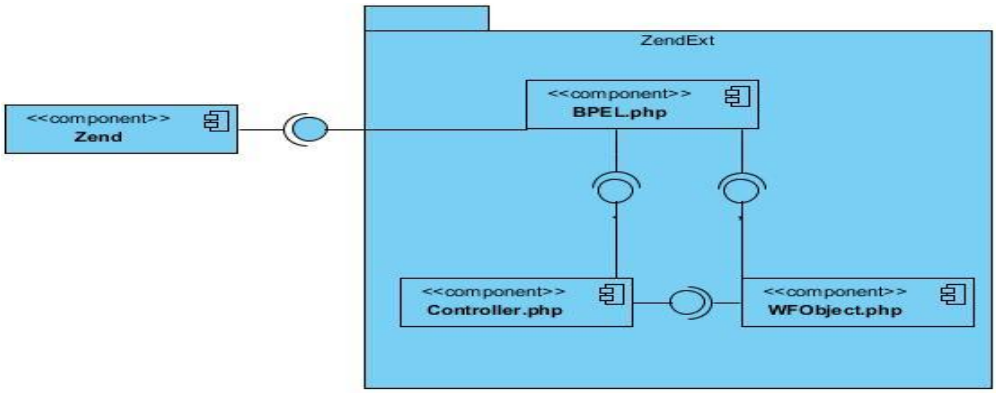

Figura 10 : Diagrama de componentes

A continuación se describe el diagrama de componentes del componente ***BPEL*** el cual depende del componente ***Zend*** ya que su implementación está basada en las funciones de esa librería, depende además del ***WFObject*** para obtener el Modelo XPDL a cargar y del ***Controller*** para realizar la petición de dicho modelo

3.2.2. 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. A continuación se describen los estándares empleados en la implementación del componente para transformar ficheros XPDL al lenguaje de ejecución de flujos de trabajos en el Marco de trabajo Sauxe.

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. (43)

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 seguido por la palabra Controller y heredar siempre de la super clase del framework ZendExt_Controller_Secure. Ejemplo: **AplicationController.**

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. (43)

- ✓ **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: **readPattern.**

3.3. Métricas para la validación del modelo de diseño propuesto

En Pressman (44) 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[12] 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 9: Tamaño operacional de clase (TOC)

Atributo de calidad	Modo en que lo afecta
Responsabilidad (Resp)	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase
Complejidad de implementación (CI)	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase
Reutilización (Reut)	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 10: Rango de valores para los criterios de evaluación de la métrica Tamaño Operacional de Clase (TOC)

Atributo	Categoría	Criterio

[12] Programación orientada a objeto

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 11: Atributos de calidad evaluados por la métrica RC

Atributo de calidad	Modo en que lo afecta
Acoplamiento (Ac)	Un aumento del RC implica un aumento del Acoplamiento de la clase
Complejidad de mantenimiento(CM)	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase
Reutilización (Reut)	Un aumento del RC implica una disminución en el grado de reutilización de la clase
Cantidad de pruebas (CP)	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 12: 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

3.3.1. 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 7 clases fundamentales y 72 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 10.28, y los siguientes datos posibilitaron categorizar los atributos por cada clase:

Tabla 13: Instrumento de evaluación de la métrica de Tamaño Operacional de Clase (TOC)

Clase	# de operaciones	Resp	CI	Reut
ZendExt_WF _BPEL_TranslatorBPEL	20	Alta	Alta	Baja
ZendExt_WF _BPEL_WorkflowManager	17	Media	Media	Media
ZendExt_WF_BPEL_ModeloBPELXML_BaseXML	4	Baja	Baja	Alta
ZendExt_WF _BPEL Base_Complex	10	Baja	Baja	Alta
ZendExt_WF_BPEL	7	Baja	Baja	Alta

_Base_Collections				
ZendExt_WF_BPEL_ModeloBPEL_ ComplexChoise	9	Baja	Baja	Alta
ZendExt_WF _BPEL Base_ Bfcace	5	Baja	Baja	Alta

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

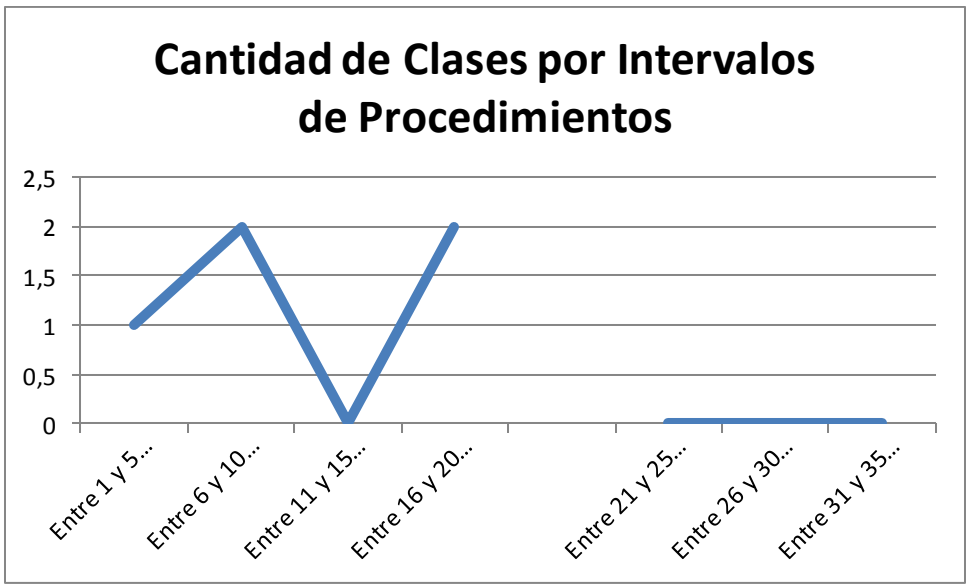

Figura 11: Cantidad de clases por intervalos de procedimientos

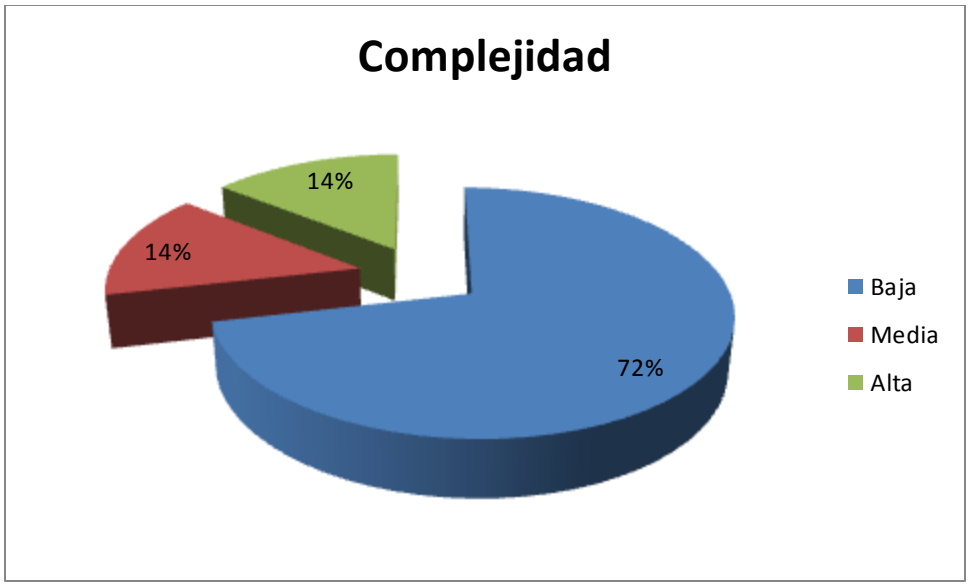

Figura 12: Resultados obtenidos de la evaluación de la métrica TOC para el atributo Responsabilidad

Figura 13: Resultados obtenidos de la evaluación de la métrica TOC para el atributo de Complejidad

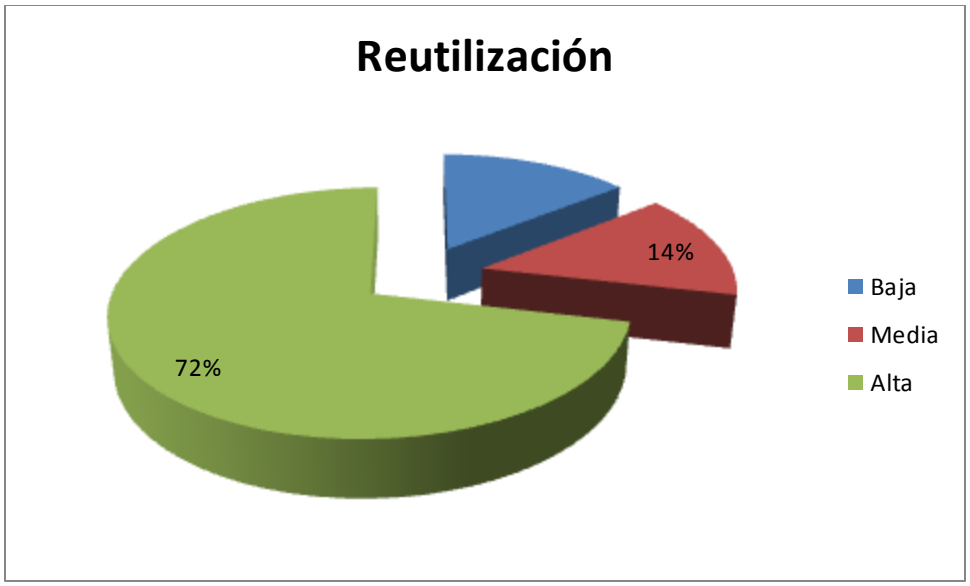

Figura 14: Resultados obtenidos de la evaluación de la métrica TOC para el atributo Reutilización

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

Las 7 clases fundamentales contienen 45 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 6.42

Tabla 14: Instrumento de evaluación de la métrica de Relaciones entre clases (RC)

Clase	# RC	Ac	CM	Reut	CP
ZendExt_WF _BPEL_TranslatorBPEL	4	Alta	Baja	Alta	Baja
ZendExt_WF _BPEL_WorkflowManager	1	Baja	Baja	Alta	Baja
ZendExt_WF _BPEL_ModeloBPELXML_BaseXML	27	Alta	Alta	Baja	Alta
ZendExt_WF _BPEL Base_Complex	2	Media	Baja	Alta	Baja
ZendExt_WF _BPEL _Base_Collections	8	Alta	Media	Media	Media

ZendExt_WF_BPEL_ModeloBPEL_ ComplexChoise	1	Baja	Baja	Alta	Baja
ZendExt_WF _BPEL _Base_ Bfcace	2	Media	Baja	Alta	Baja

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

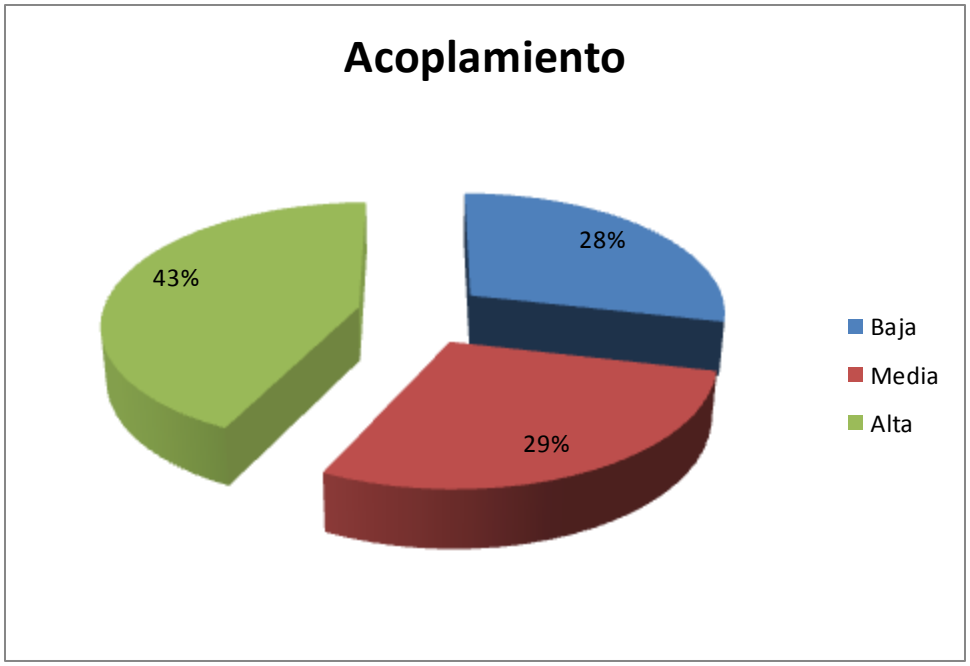

Figura 15: Resultados de la evaluación de la métrica RC para el atributo Acoplamiento

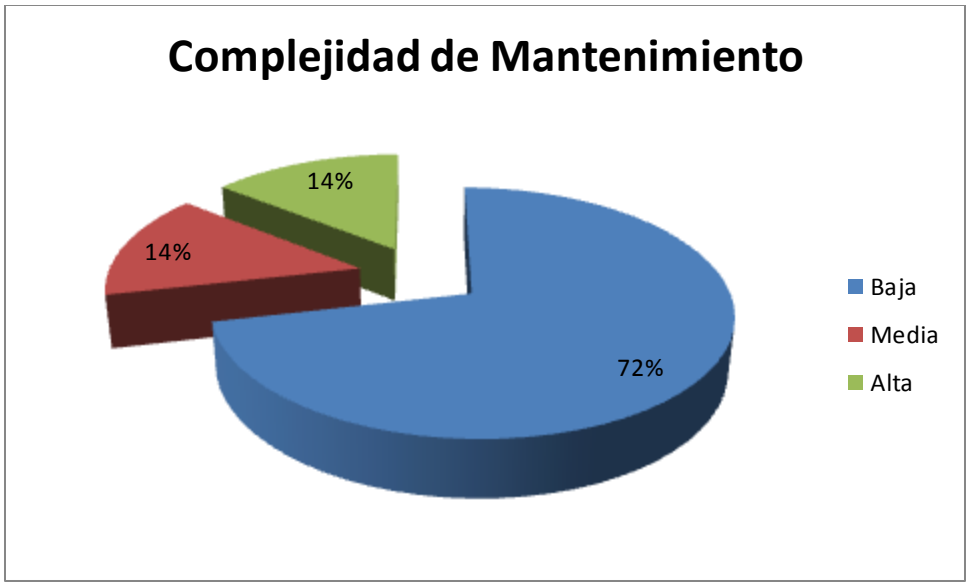

Figura 16: Resultados de la evaluación de la métrica RC para el atributo Complejidad de mantenimiento

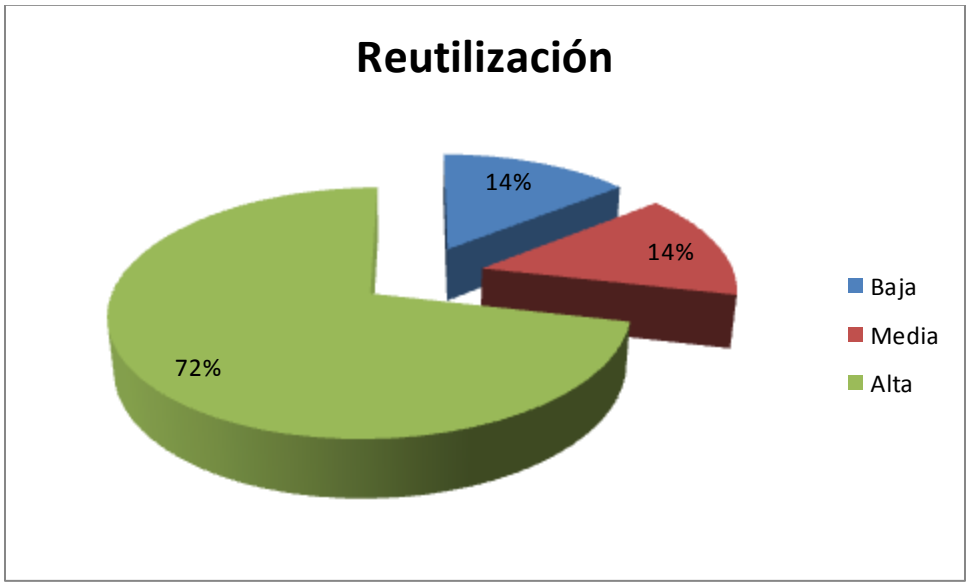

Figura 17: Resultados de la evaluación de la métrica RC para el atributo Reutilización

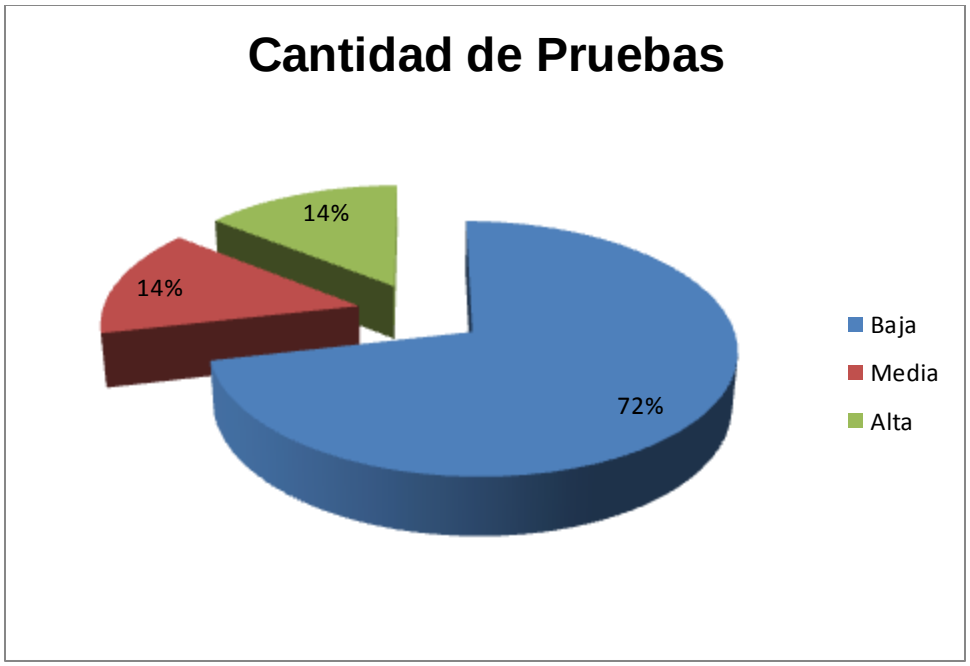

Figura 18: Resultados de la evaluación de la métrica RC para el atributo Cantidad de pruebas

3.3.3. 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.

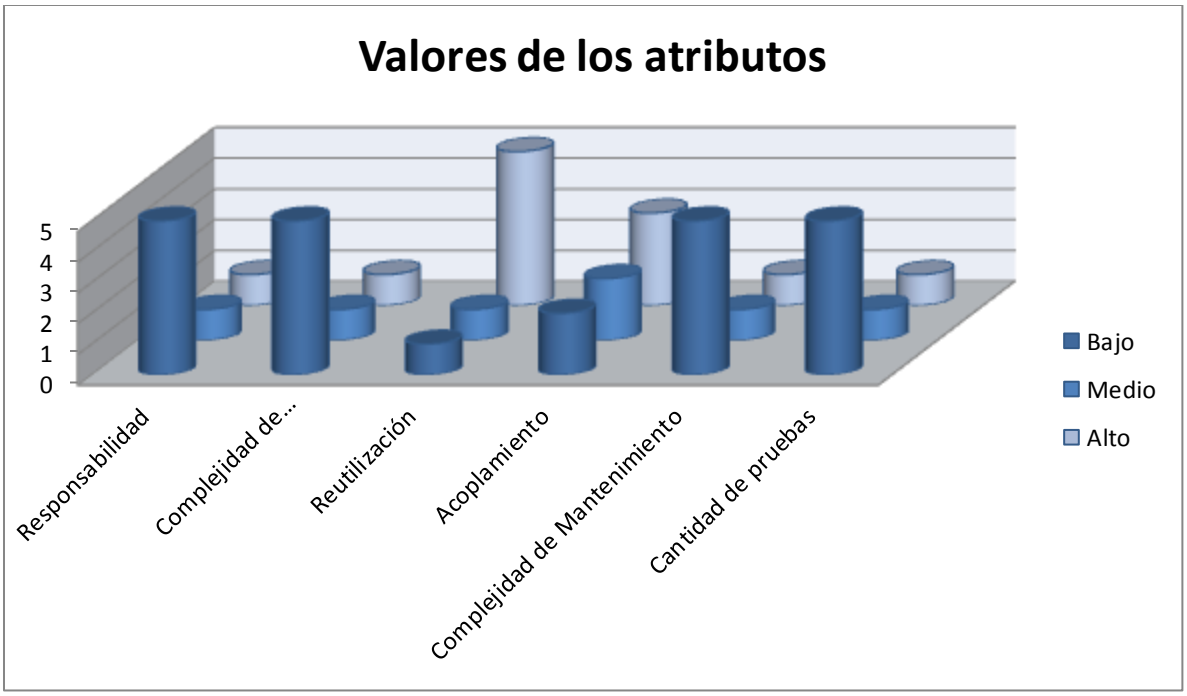

Figura 19: Resultados del rango de valores aplicados a cada atributo

Tabla 15: 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 fueron positivos, el 43% de Alto del atributo Acoplamiento es debido a que las clases implementadas en su mayoría hacen uso de la clase TranslatorBPEL la cual contiene un alto nivel de recursividad, provocando esto un aumento en las RC, y los restantes atributos con evaluación Bajo, solamente la reutilización está en su mayoría en

alto, por lo que se demuestra que el diseño propuesto para el componente para transformar ficheros XPDL al lenguaje de ejecución de flujos de trabajos para Sauxe se encuentra dentro de los niveles requeridos.

3.3.4. Validación del indicador expresividad

La expresividad de un lenguaje de definición o ejecución de procesos de negocio es medible a partir de la cantidad de patrones de flujo de trabajo que estos implementen, a raíz de esta aclaración la tabla anterior muestra como al implementar 3 patrones más de los que ya contiene ezComponents se garantiza una definición de proceso con mayor expresividad a la que se obtenía anteriormente.

Tabla 16: Resultados de la evaluación del indicador expresividad

	EXPRESIVIDAD	
	Antes (ezComponents)	**Después (BPEL)**
PATRONES	Sequence	Sequence
	Parallel Split	Parallel Split
	Simple Merge	Simple Merge
	Synchronization	Synchronization
	Exclusive Choise	Exclusive Choise
	Multi Choise	Multi Choise
	Synchronizing Merge	Synchronizing Merge
		Implicit Termination
		Cancel Task
		Cancel Case

3.4. 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 para transformar ficheros XPDL al lenguaje de ejecución de flujos de trabajos para Sauxe.

3.4.1. Pruebas de caja blanca

Las pruebas de la caja blanca se realizan sobre las funciones internas de un módulo en concreto. 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.

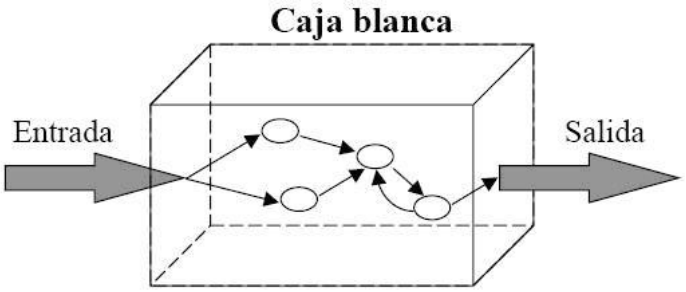

Figura 20: Representación de pruebas de Caja blanca

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

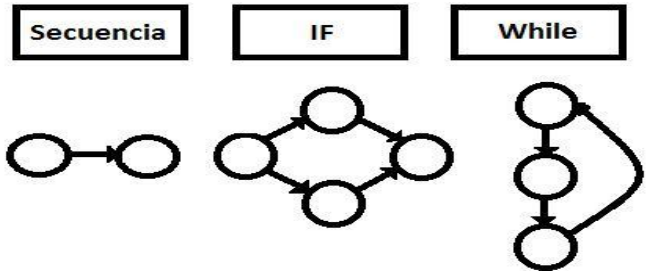

Figura 21: Notación de Grafos de flujo

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 transformar ficheros XPDL al lenguaje de ejecución de flujos de trabajos para Sauxe en el caso de prueba ***populateSequence.***

Llena una secuencia

```
private function populateSequence($xpdlActivity, $bpelSequence) {
        do {
            $bpel = $this->toBPEL($xpdlActivity, $bpelSequence); //1
            if ($bpel !== null) {//2
                $bpelSequence->getActivities()->add($bpel);//3
                $xpdlActivity = $this->nextActivity($xpdlActivity); //3
            } else {
                break; //4
            }
        } while ($xpdlActivity !== null);//5
    }
```

Figura 22: Figura de la función populateSequence

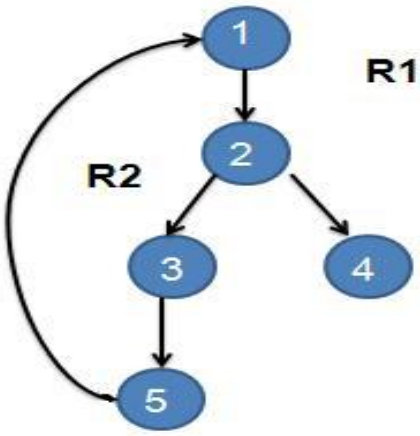

Figura 23: Notación de Grafos de flujo asociado a la funcionalidad *populateSequence*

Luego de haber construido el grafo se realiza el cálculo de la complejidad ciclomática mediante las tres fórmulas descritas a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

V (G) = (A – N) + 2

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

V (G) = (5- 5) + 2

V (G) = 5

1. V (G) = P + 1

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

V (G) = 1 + 1

V (G) = 2

2. V (G) = R

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

V (G) = 2

El cálculo efectuado anteriormente sobre las fórmulas ha dado el mismo valor, dando como resultado 2, esto indica que existen 2 posibles caminos por donde el flujo puede circular, y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo:

Camino básico #1: (1-2-3-4-5-1)

Camino básico #2: (1-2-4)

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 la actividad XPDL(xpdlActivity) que se va agregar a la secuencia y la secuencia BPEL(bpelSequence) la que va ser agregada dicha actividad
- **Condición de ejecución:** que la actividad XPDL mapeada a BPEL sea distinto de nulo

- **Entrada:** una actividad XPDL y un arreglo de actividades BPEL
- **Resultados Esperados:** adicionar la actividad a la secuencia

Caso de prueba para el camino básico #2:

- **Descripción:** necesita como parámetros de entrada para ejecutarse la actividad XPDL(xpdlActivity) que se va agregar a la secuencia y la secuencia BPEL(bpelSequence) la que va ser agregada dicha actividad
- **Condición de ejecución:** que la actividad XPDL mapeada a BPEL sea distinto de nulo
- **Entrada:** una actividad XPDL y un arreglo de actividades BPEL
- **Resultados Esperados:** No se adiciona la actividad a la secuencia

Evaluación de los resultados: al realizar las pruebas de caja blanca al caso ***populateSequence*** 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.

3.4.2. Pruebas de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, permiten al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. (44)

Categorías de errores a detectar por dichas pruebas:

- ✓ Funciones incorrecta o ausente, errores de interfaz
- ✓ Errores en estructuras de datos o en accesos a bases de datos externas
- ✓ Errores de rendimiento y errores de inicialización y de terminación

Estas pruebas se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa. Los casos de prueba pretenden demostrar que:

- ✓ Las funciones del software son operativas
- ✓ La entrada se acepta de forma correcta
- ✓ Se produce una salida correcta

Las pruebas funcionales en la mayoría de los casos son realizadas manualmente por el analista de pruebas, se apoyan en la especificación de requisitos del módulo.

La aplicación fue probada por el Departamento de Calidad del CEIGE), donde se comprobó el correcto funcionamiento de la misma a partir de los diseño de caso de prueba (Tablas de los Anexos).

3.5. Conclusiones parciales

- Fueron definidos e implementados los algoritmos de transformación de los lenguajes XPDL a BPEL
- Se validaron las disciplinas de requisitos, análisis y diseño mediante la aplicación de técnicas y métricas que certificaron la calidad y eficiencia necesaria para la implementación de las funcionalidades definidas
- Las pruebas de caja blanca y de caja negra realizadas a la aplicación, validaron el correcto funcionamiento de la misma
- El indicador de expresividad fue validado a través de la implementación de 3 patrones más de los que implementa ezComponents dando así solución al problema a resolver planteado al inicio de la investigación

CONCLUSIONES GENERALES

Con la finalización de la investigación se concluye que:

- Al realizar el marco teórico de la investigación se logró un estudio del estado del arte de las principales herramientas que son usadas para la gestión de flujo de trabajo, obteniendo como resultado la necesidad de la creación de una herramienta para transformar ficheros XPDL al lenguaje de ejecución de flujos de trabajos para Sauxe
- Al realizar el marco teórico de la investigación se analizaron los principales estándares en lenguajes de definición y ejecución de procesos de negocio, identificando BPEL como estándar a utilizar por la propuesta de solución.
- Se realizó un estudio de los diferentes patrones de flujo de trabajo existentes identificando 10 patrones a utilizar por la solución propuesta
- Se realizó el análisis y diseño de la propuesta de solución, sentando así las bases para su posterior implementación.
- Fueron definidos e implementados los algoritmos de transformación de los lenguajes XPDL a BPEL
- Se validaron las disciplinas de requisitos, análisis y diseño mediante la aplicación de técnicas y métricas que certificaron la calidad y eficiencia necesaria para la implementación de las funcionalidades definidas
- Se validó la solución propuesta a través de la realización de pruebas de caja blanca y caja negra las cuales arrojaron resultados satisfactorios
- El indicador de expresividad fue validado a través de la implementación de 3 patrones más de los que ya implementa ezComponents, dando así solución al problema a resolver antes planteado

RECOMENDACIONES

- Aumentar el número de patrones implementados para versiones posteriores
- Utilizar el presente trabajo como material de estudio en futuras investigaciones referentes al tema
- Se recomienda que el resultado obtenido en la investigación sea implantado en el marco de trabajo Sauxe para la gestión de procesos de negocio

BIBLIOGRAFÍA

1. **Nick Russell, Wil M.P. van der Aalst, Arthur H.M. ter Hofstede.** *Exception Handling Patterns in Process-Aware Information Systems.*

2. **Aalst, W.M.P. van der.** *Process-Aware Information Systems: Lessons to be Learned from Process Mining.* The Netherlands : s.n.

3. **Marlon Dumas, W.M.v.d.A., Arthur H,.** *Process Aware Information Systems: Bridging People and Software Through Process Technology.* 2005.

4. **Flattery, Mark.** *Workflow System.* 2005.

5. **Orúe, Sara Magaña.** *ESTUDIO COMPARATIVO DE LENGUAJES DE MODELADO DE PROCESOS DE NEGOCIO PARA SU INTEGRACIÓN EN PROCESOS DE DESARROLLO DE SOFTWARE DIRIGIDO POR MODELOS.* MADRID : s.n.

6. **Carlos Buenosvinos, Carlos Crespo.** *php | Motores de Workflow Más allá de las AplicacionesCRUD.* 2008.

7. **MARLON DUMAS, WIL van der AALST,ARTHUR H. M.** *PROCESS-AWARE INFORMATION SYSTEMS.* CANADA : s.n., 2005.

8. **W.M.P. VAN DER AALST, P. BARTHELMESS,C.A. ELLIS,J. WAINER.** *PROCLETS: A FRAMEWORK FOR LIGHTWEIGHT INTERACTING WORKFLOW PROCESSES. International Journal of Cooperative Information Systems.* s.l. : World Scienti.

9. **Kawtar Benghazi, José Luis Garrido Bullejos, Manuel Noguera García.** *Introducción al Modelado de Procesos de Negocio.* granada : s.n.

10. **Adolfo R. de Soto, Eva Cuervo Fernández.** *Nuevas Tendencias en Sistemas de Información:Procesos y Servicios.* 2006.

11. **Aalst, Wil M.P. van der.** *Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language.* The Netherlands : s.n.

12. **Llatas, Carlos Fernández.** *Representación, Interpretación y Aprendizaje de estandarización de Vías Clínicas Flujos de Trabajo basado en Actividades para la.* Valencia : s.n.

13. **Team, P.D.** *ProcessMaker manual.* 2006.

14. ProcessMaker. *Sitio oficial.* [Online] processMaker.com.

15. TIBCO. *sitio oficial.* [Online] tibco.com.

16. **Jesus Graterol, Francisco Hernández, Yamil Orozco.** Posgrado en ciencias de la computación. [Online] kuainasi.ciens.ucv.ve/adsi2010-2l.

17. Jboss. *Sitio oficial.* [Online] [Cited: 12 10, 2012.] http://www.jboss.org.

18. **KOENIG, JOHN.** *JBoss jBPM White Paper.* 2004.

19. **Harmon, Paul.** *BPM Software Tools Report on BOC's Adonis Version 4.0.* 2010.

20. Adonis: Comunity. [Online] http://www.adonis-community.com/.

21. **Pérez, Juan Diego.** *Notaciones y lenguajes de procesos. Una visión global.* Sevilla : s.n., 2008.

22. **Thomas Hornung, Agnes Koschmider,Jan Mendling.** *Integration of heterogeneous BPM Schemas:The Case of XPDL and BPEL.*

23. **García Bañuelos Gregorio, López López Luciano.** *Desarrollo de un Traductor de BPEL4WS a YAWL.* México : s.n.

24. Workflow Patterns. *sitio oficial.* [Online] [Cited: 1 30, 2013.] http://workflowpatterns.com.

25. **Havey, Mike.** *Essential Business Process Modeling.* EE.UU : s.n., 2005.

26. **Obregón, Ing. William González.** *MODELO DE DESARROLLO DE SOFTWARE.* Cuba : s.n., 2012.

27. **Alfaro, Félix Murillo.** *Herramientas Case.* 1999.

28. **Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato.** *Version Control with Subversion.* Chicago : s.n., 2004.

29. Tigris.org Open Source Software Engineering Tools. [Online] 2009. [Cited: 1 15, 2013.] http://rapidsvn.tigris.org/.

30. M Mozilla Firefox. [Online] [Cited: 1 15, 2013.] http://www.mozilla-europe.org/.

31. Apache. *sitio oficial.* [Online] apache.org.

32. PostgreSQL-es. *sitio oficial.* [Online] [Cited: 1 10, 2013.] www.postgresql.org.es.

33. NetBeans. *sitio oficial.* [Online] netbeans.org.

34. **Ing. Oiner Gómez Baryolo, Ing. Yoandry Morejón Borbón ,Ing. Darien García Tejo.** *ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE.* La Habana, Cuba : s.n.

35. **Jack Slocum, Brian Moeskau, Aaron Conran, Rich Waters.** *Sencha_Ext_JS.* 2007.

36. Zend Framework. *sitio oficial.* [Online] [Cited: 1 12, 2013.] framework.zend.com.

37. doctrine. *sitio oficial.* [Online] [Cited: 12 10, 2013.] http://www.doctrine-project.org.

38. PHP. *sitio oficial.* [Online] [Cited: 1 11, 2013.] www.php.net.

39. upcommons. [Online] [Cited: 1 13, 2013.] http://upcommons.upc.edu.

40. **Laguna, Miguel A.** *Requisitos .Ingeniería del Software I.*

41. **Mariñán, Martín Pérez.** *Patrones de Diseño(Design Patterns).*

42. **E. Gamma, R. Helm, T. Johnson, J. Vlissides.** *Design Patterns.* version española de 2003.

43. **Pearsons, David.** *Data Types, Arithmetic, and Arrays.* 2012.

44. **S.Pressman, Roger.** *La Ingeniería del Software. Un enfoque práctico.* 2002.

45. **Christopher Koch, D.S., E. Baatz.** *The ABCs of ERP.* 1999.

46. **Weske, M.** *Business Process Management. Concepts, Languages, Architectures. .* 2007.

ANEXOS

Tabla 17: Caso de prueba Cargar archivo XPDL

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Cargar archivo XPDL	Permite cargar un archivo XPDL.	EP 1.1: Seleccionar el botón de **cargar** el archivo y presionando el botón ***Aceptar***.	– Se presiona el botón ***Browse*** – Se muestra la interfaz de seleccionar el archivo. – Se selecciona el archivo a cargar Se presiona el botón ***Aceptar*.**
		EP 1.2 Cancelar Operación	– Se presiona el botón ***Cancelar*** Se cancela la operación.

Tabla 18: Caso de prueba Guardar archivo XML

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Guardar archivo XML	Permite guardar un archivo XPDL.	EP 1.1: seleccionar el botón de ***Guardar*** el archivo y presionando el botón ***Aceptar.***	– Se presiona el botón ***Guardar*** – Se muestra la interfaz de guardar el archivo. – Se selecciona donde guardar el archivo Se presiona el botón ***Aceptar*.**
		EP 1.2 Cancelar Operación	– Se presiona el botón ***Cancelar***

			Se cancela la operación.