Universidad de las Ciencias Informáticas
Facultad 3

Título: Componente para la transformación de XPDL a un lenguaje de ejecución de procesos para la solución de flujos de trabajo en el marco de trabajo Sauxe.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor: Máximo Enrique Marín López

Tutor: M.Sc. Raykenler Izquierdo Herrera

Co-tutor: Ing. Yoriangel Rivero González

"La Habana, 21 de junio de 2012"

DECLARACIÓN DE AUTORÍA

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

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

Máximo Enrique Marín López

_____

M.Sc. Raykenler Izquierdo Herrera

_____

Ing. Yoriangel Rivero González

_____

RESUMEN

Los Sistemas de Planificación de Recursos Empresariales tienen la característica de ser programados siguiendo procesos de negocios previamente definidos, lo que provoca, una vez cambiado estos, transformaciones en su estructura. Esta desventaja puede ser mitigada incorporándole Sistemas de Gestión de Procesos de Negocios, ya que permiten que los procesos puedan ser modelados, interpretados y ejecutados. En el Centro CEIGE se está desarrollando el ERP denominado Cedrux implementado sobre el marco de trabajo Sauxe. El centro, para dar una mayor flexibilidad a Cedrux, toma la iniciativa de adicionar a este marco de trabajo un gestor de flujo de trabajo para garantizar las entradas, funciones y salidas de tareas relacionadas lógicamente entre sí. Actualmente la solución, basada en librerías ezComponents de PHP, no puede interpretar directamente el flujo de información de los procesos de negocios definido en XPDL. La presente investigación constituye una propuesta de solución para el desarrollo de un componente de Sauxe que permita la transformación de XPDL a un lenguaje de ejecución de procesos para su solución de flujos de trabajo. El desarrollo está previamente respaldado por el estudio de sistemas de gestión de procesos y sus algoritmos 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. Con la implementación del sistema propuesto, se logra la traducción de procesos definidos permitiendo así su posterior ejecución.

PALABRAS CLAVE

XPDL, lenguaje de ejecución, lenguaje de transformación, flujo de trabajo, ezComponents.

TABLA DE CONTENIDOS

RESUMEN I

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 6

1.1 Introducción 6
1.2 Conceptualizando 6
1.3 Sistemas de Gestión de Flujo de Trabajo 7
1.4 Herramientas de transformación 11
1.6 Metodología de desarrollo de software. 12
1.7 Librerías y Marcos de trabajo 14
Zend Framework 15
ZendExt Framework 15
Sauxe 15
ezComponents 17
Doctrine 17
1.8 Herramientas de desarrollo 18
Apache 18
Visual Paradigm 18
PostgreSQL 18
PGAdmin III 19
Mozilla Firefox 19
Subversion 19
1.9 Lenguajes de desarrollo y de modelado 20
UML 20
PHP 21
1.10 Conclusiones parciales 22
CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA 23

2.1 Introducción 23
2.2 Modelo conceptual 23

2.3 Especificación de requisitos 24
2.3.1 Requisitos funcionales 24
2.3.2 Requisitos no funcionales 28
2.3.4 Validación de los requisitos funcionales 29
2.4 Diagrama de clases 30
2.5 Modelo de datos 32
2.6 Patrones de Diseño empleados 33
Patrones GRASP 34
2.7 Patrones arquitectónicos 34
Patrón de arquitectura por capas 35
2.8 Conclusiones Parciales 36
CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA 37
3.1 Introducción 37
3.2 Modelo de implementación 37
Estándares de codificación 37
Diagrama de componente: 40
Diagrama de despliegue: 41
Métricas de diseño 42
3.3 Resultados obtenidos de la aplicación de la métrica TOC 45
3.4 Resultados obtenidos de la aplicación de la métrica RC 48
3.5 Matriz de inferencia de indicadores de calidad 52
3.6 Pruebas estructurales o de caja blanca 53
3.7 Pruebas de rendimiento 57
Resultados de pruebas de estrés. 58
3.8 Conclusiones parciales 59
CONCLUSIONES 60
RECOMENDACIONES 61
REFERENCIAS BIBLIOGRÁFICAS 62

Índice de figuras

Figura 1: Integración de BPMS con sistemas empresariales. 3
Figura 2: Arquitectura de Sauxe. 16
. Figura 3 : Modelo Conceptual. 23
Figura 4: Diagrama de clases del diseño 31
Figura 5: Modelo de datos. 33
Figura 6: Ejemplo de comentario de clase 40
Figura 7: Diagrama de componentes 41
Figura 8: Modelo de despliegue 41
Figura 9: Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad. 47
Figura 10: Resultados de la evaluación de la métrica TOC para el atributo Reutilización 47
Figura 11: Resultados de la evaluación de la métrica TOC para el atributo Complejidad. 48
Figura 12: Resultados de la evaluación de la métrica RC para el atributo Acoplamiento. 50
Figura 13: Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento. 50
Figura 14: Resultados de la evaluación de la métrica RC para el atributo Reutilización. 51
Figura 15: Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas. 51
Figura 16: Resultados de la Matriz de inferencia de indicadores de calidad 53
Figura 17: Prueba de caja blanca 54
Figura 18: Grafo generado por la prueba de caja blanca. 55
Figura 19: Caso base de un flujo de trabajo. 56

INTRODUCCIÓN

En la actualidad, uno de los factores que garantizan la supervivencia de las empresas en el mercado, es la gestión de la información de los datos que disponen. La implantación de sistemas de gestión en busca de una mejor rentabilidad, efectividad y capacidad de adaptación, es una solución eficaz para mantener la competitividad. Debido al auge de estos sistemas dedicados a la gestión de procesos surge el término de Sistemas Consientes de Procesos de Información, (PAIS por sus siglas en inglés), sistemas de software que gestionan y ejecutan los procesos operativos relacionados con las personas, aplicaciones, y / o fuentes de información sobre la base de modelos de proceso.[1]

Los PAIS dan la posibilidad de una infraestructura que da soporte a un cambio de programación del sistema, ya que permite que las tareas, tanto automáticas como semiautomáticas, sean tratadas de tal forma que se pueda disponer de ellas sacando su mayor provecho.

Existen básicamente dos maneras de implementar los PAIS. Desarrollar sistemas que den soporte a procesos específicos, o configurando sistemas genéricos. En el primer caso es donde el sistema presenta una conciencia implícita del proceso de negocio, lo cual implica que el proceso sea conocido y manejado de forma interna. Este enfoque hacia los procesos asegura que el sistema resultante se ajuste a las necesidades de las empresas y a las peculiaridades de sus procesos. Sin embargo, el costo de inversión inicial de este puede ser demasiado alto para algunas, provocando que el sistema resultante no sea escalable. Una vez que se introducen nuevos procedimientos, los procesos existentes se hacen cada vez más sofisticados, y los usuarios tienden a desarrollar expectativas más altas, lo que provoca que se haga difícil adaptar el sistema para satisfacer las nuevas demandas.[1]

Entre las soluciones que se corresponden con esta definición podemos mencionar los Sistemas de Planificación de Recursos Empresariales (ERP por sus siglas en inglés), capaces de gestionar los procesos que se llevan a cabo en las empresas, en aras de alcanzar la eficiencia económica. Su integración y adaptabilidad a la realidad de cada institución, permiten aumentar la productividad y minimizar el tiempo para realizar y analizar tareas.

Los ERP facilitan los flujos de información, definidos como un conjunto de transferencias de información de acuerdo a un cierto análisis y en referencia a un período de tiempo, entre los procesos de negocios, que no son más que un conjunto de tareas relacionadas lógicamente, que utilizan recursos para transformar en un resultado de negocio definido los elementos de entrada que reciban.[1-3]

Ventajas de los ERP

- Estandarización e integración de la información en una base de datos centralizada.
- Mayor control organizacional.
- Minimiza el tiempo de análisis de la información.
- Optimización de los tiempos de producción y entregas.
- Disminución de costos.
- Se cuenta con información actualizada que permite la toma de decisiones.
- Evita duplicidad de información.
- Cuentan con módulos configurables de acuerdo a cada área de la empresa.
- Permite mejorar el ROI[1] de la empresa.

Desventajas de los ERP

- Costosos a primera vista.
- Mucho tiempo para su implementación.
- Adquisición o adaptación del hardware.
- Pocos expertos en los sistemas ERP.
- Algunos sistemas ERP pueden ser difíciles de utilizar.

Cuando se conoce de forma explícita los procesos de negocios el sistema es capaz de manipularlos de forma genérica, esto permite el cambio en los modelos de procesos de negocio sin necesidad de aplicar cambios sobre el sistema. Ejemplo de ello son los Sistemas de Gestión de Procesos de Negocio (BPMS, por sus siglas en inglés). BPMS consisten en la implementación de actividades por sistemas de software minimizando la participación de usuarios, permitiendo con esto, la posibilidad de que los procesos de negocio puedan ser modelados, interpretados y ejecutados con el objetivo de lograr un mejor entendimiento del proceso y en ocasiones optimizarlo. [1, 4]

Estos sistemas utilizan diferentes metodologías y procedimientos las cuales dan la posibilidad de gestionar los procesos. Utilizan estándares como la Notación de Modelado de Procesos de Negocio (BPMN por sus siglas en inglés) para modelar gráficamente, a través de diagramas, el flujo de información de los procesos de negocio, adoptados por las especificaciones oficiales del Grupo de Gestión de Objetos (OMG por sus siglas en inglés).

[1] Retorno de inversión

También existe otra estandarización para la definición e intercambio de procesos de negocio, sin ningún tipo de esquema de información y más orientada a persistir el modelo gráfico de BPMN en archivos XML[2], llamada Lenguaje de Definición de Procesos XML (XPDL, por sus siglas en inglés).

Las aplicaciones empresariales como ERPs son típicas candidatas para escenarios de integración con BPMS, buscando mayor validez en los procesos, surgiendo así sistemas híbridos, capaces de gestionar los modelos de procesos de negocios logrando una mayor robustez en cuanto a la gestión de los mismos.[4, 5]

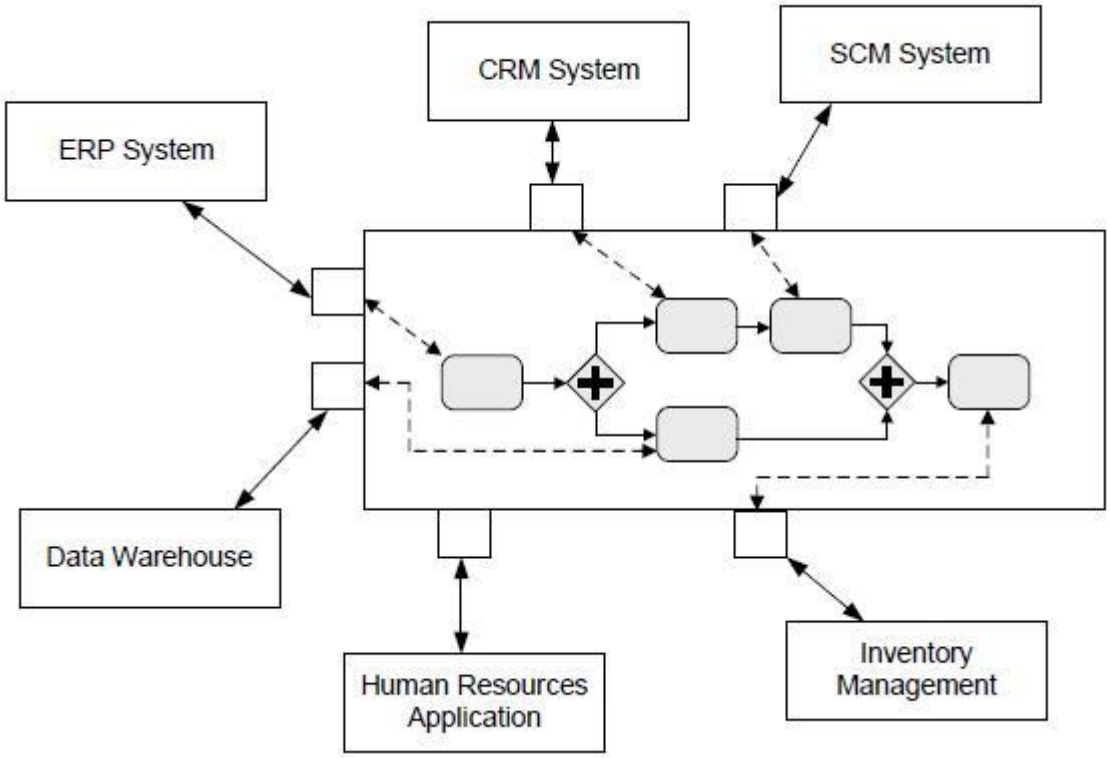


Figura 1: Integración de BPMS con sistemas empresariales.

Las empresas cubanas, en busca de gestionar sus recursos empresariales, auspician el desarrollo de un ERP denominado Cedrux, que sea adaptable a las características propias de cada empresa. Su desarrollo está enmarcado en mejorar la eficiencia económica, y cumplir con la premisa de independencia tecnológica, logrando eliminar las dependencias existentes de empresas extranjeras. Esto constituye un

[2] Del inglés Extensible Markup Language.

factor estratégico para el desarrollo en el ámbito de las nuevas tecnologías de la informática y las comunicaciones.

Se le da la tarea a la Universidad de las Ciencias Informáticas (UCI), específicamente en el Centro de Informatización para la Gestión de Entidades (CEIGE), el desarrollo de Cedrux, implementado sobre Sauxe, marco de trabajo para el desarrollo de aplicaciones web dinámicas, y que brinda soporte tecnológico para la elaboración de módulos de software determinados. El centro, con el objetivo de dar una mayor flexibilidad a Cedrux, toma la iniciativa de incorporar a Sauxe una herramienta de flujo de trabajo que se encargará de interpretar los flujos de información de los procesos de negocio. La misma poseerá un componente encargado de interpretar y ejecutar los modelos de procesos auxiliándose en datos almacenados en un servidor de base de datos.

Como entrada inicial del componente de flujo de trabajo se tiene el diagrama de procesos de negocio modelado con anterioridad en BPMN, en un segundo paso se realizará la transformación de dicho diagrama gráfico implementado en ExtJS[3] a su homologación en clases PHP para que puedan ser validadas, con el objetivo de reportar y mitigar posibles errores. Estas clases PHP serán traducidas a una definición en XPDL, logrando una primera persistencia del diseño del flujo de procesos de negocio que se desea interpretar y obtener un objeto que permita generar el diagrama inicial. En un último momento es necesario convertir el objeto XPDL en una especificación que pueda ser descifrada por el motor de flujo de trabajo de la herramienta.

El motor de flujo de trabajo de Sauxe (basado en la librería ezComponents de PHP), implementa un lenguaje de definición nativo descrito en sintaxis XML y con características propias, por lo que no puede interpretar directamente el flujo de procesos de negocio una vez definido en XPDL, por tanto se presenta el siguiente **problema a resolver**: ¿Cómo garantizar la especificación de un proceso de negocio, a partir de su definición en XPDL, para la interpretación de su flujo de trabajo en el motor de flujo de trabajo de Sauxe?

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

Teniendo en cuenta el problema, se define como **objetivo general**: Desarrollar un componente para la transformación de un proceso de negocio descrito en XPDL a una especificación que permita su ejecución en Sauxe.

[3] Del inglés Extended JavaScript. Es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas.

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

1. Realizar un estudio sobre los lenguajes de especificación y ejecución de flujo de trabajo para determinar posibles áreas de reutilización.
2. Identificar reglas de transformación.
3. Implementar reglas de transformación.
4. Validar la solución propuesta.

Se propone como **idea a defender**: Con la implementación de un componente para la transformación de un proceso de negocio definido en XPDL a una especificación de dicho proceso, se podrá garantizar la interpretación de un flujo de trabajo en el motor de Flujo de trabajo de Sauxe.

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: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Este capítulo tiene el objetivo de realizar un estudio de algunos sistemas gestores de flujos de trabajo, enmarcándose 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 Conceptualizando

A continuación se exponen algunos conceptos necesarios para un mejor entendimiento del estudio realizado.

Flujo de trabajo: se refiere a la automatización de un procedimiento de trabajo donde intervengan documentos, información y tareas que interactúan entre los participantes siguiendo un conjunto de reglas definidas, con el objetivo de llegar o contribuir a un concepto de negocio más general.[6]

Sistema de gestión de flujo de trabajo: – (WFMS siglas en inglés) – no es más que un sistema informático encargado de la automatización de un proceso de negocio capaz de gestionar el mismo empleando tanto recursos humanos como tecnológicos. La principal tarea de un WFMS es la definición, gestión y ejecución de un flujo de trabajo.[4]

XPDL: es un lenguaje de definición de proceso de negocio establecido por la WFMC (Workflow Management Coalition) que usa una sintaxis basada en XML. El estándar es extensible y permite a cada implementación añadir funcionalidades adicionales siempre que cumpla las básicas. 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. Los principales elementos de XPDL son: Package, Application, Workflow Process, Activity, Transition, Participant, DataField, and DataType. Entre sus ventajas encontramos la interoperabilidad, la posibilidad de ser modelado en cualquier editor de texto y su fácil interpretación.[7]

BPEL4WS (Business Process Execution Lenguage for Web Services): 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 de sistemas de gestión de procesos de negocios. Las operaciones permiten recibir y enviar mensajes SOAP[4]. Se puede tener dos visiones de BPEL, por un lado puede verse como un archivo XML que una máquina de procesos de negocio ejecuta. Pero por otro lado puede verse como un lenguaje de intercambio, o sea la máquina de procesos de negocio permite convertir un lenguaje propietario a BPEL y viceversa. Este permite invocar múltiples servicios web al mismo tiempo y sincronizar los resultados, permitiendo ventajas como reusabilidad, eficiencia, bajo acoplamiento tecnológico y división de responsabilidades. No tiene soporte gráfico; es decir, no especifica cómo deben ser los diagramas interpretativos de los procesos que define. 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. BPEL como lenguaje se compone de los siguientes elementos básicos:[8, 9]

Traductor de lenguaje: es un programa que recibe como entrada código escrito en un cierto lenguaje y produce como salida código en otro. Estos tienen como elementos fundamentales: un lenguaje fuente que es aquel el cual se quiere transformar, un lenguaje objeto el cual será el elemento de salida y un lenguaje de implementación con el que se encuentra implementado el traductor. Ejemplos de traductores son los ensambladores y los compiladores.[10]

1.3 Sistemas de Gestión de Flujo de Trabajo

Los sistemas que se exponen a continuación fueron seleccionados teniendo en cuenta el criterio de la WfMC y el OMG, entidades rectoras en la gestión de procesos de negocios. También se hace una descripción de algunas herramientas dedicadas a la transformación de lenguajes de definición a ejecución de procesos.

ProcessMaker: es una solución de software de flujos de trabajo, de código abierto, simple y rentable. Permite diseñar, automatizar e implementar procesos de negocio. ProcessMaker permite a los usuarios de negocio crear formas y mapas de flujos de trabajo completamente funcionales. El software está

[4] Del inglés Simple Object Access Protocol. Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

completamente basado en web, lo que facilita la coordinación del flujo de trabajo entre los usuarios, departamentos y organizaciones. Como una aplicación de Arquitectura Orientada a Servicios (SOA por sus siglas en inglés) de gran alcance, ProcessMaker puede interconectarse con sistemas que incluyen la gestión de documentos, ERP, CRM[5] y aplicaciones de inteligencia empresarial. Es ligero, extremadamente eficiente, e implica los gastos generales más bajos de cualquier BPMS en la industria. 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.[11, 12]

Bonita Open Solution: es una solución de flujo de trabajo de código abierto para definir flujos de procesos orientados al usuario, desarrollado por la empresa ObjectWeb, esta herramienta está implementada sobre un marco de trabajo llamado JONAS, disponible bajo licencia LGPL[6]. De esta existen versiones para los sistemas operativos de Linux y Windows. En su versión 2.0 se quiso estandarizar la definición de procesos de negocio con el objetivo de que sus usuarios migraran sus modelos de procesos de negocios con un mínimo costo, definiéndose así la utilización de XPDL. Está compuesta por varios módulos, Bonita Execution Engine, que se encarga de la conexión de los procesos que existen en el sistema así como el despliegue y ejecución de los procesos. Ya que este motor es genérico y extensible da la posibilidad de añadir con mayor o menor dificultad nuevos estándares o servicios que puedan aparecer en el mundo de BPM con posterioridad. Otro módulo es Bonita Studio, aplicación gráfica cuya función es diseñar los procesos usando BPMN (en su versión 2.0) sobre un área de diseño de forma muy intuitiva, basada en "arrastrar" los elementos y en su configuración específica mediante una o varias pestañas habilitadas para ello. Bonita User Experience encarga de la gestión de todo lo relacionado con los procesos BPM desplegados. El soporte de XPDL en Bonita lo da el modulo Bonita XPDL el cual tiene la habilidad de parsear[7] este tipo de archivo. Una vez definido un diseño de flujo de trabajo el motor es el encargado de interpretarlo y transformarlo a lenguaje BPEL ya que está orientada a uso de servicios web.[13-15]

Intalio: es un software de código abierto basado en Java-J2EE, que implementa BPMS, y está basado en un conjunto de frameworks y arquitecturas muy conocidas en la industria del software y con una madurez

[5] Del inglés Customer Relationship Management. Es un modelo de gestión de toda la organización, basada en la orientación al cliente.
[6] Del inglés Lesser General Public License.
[7] Se refiere a la acción de cargar, leer e interpretar un XPDL.

aceptable. Es una plataforma basada en Apache ODE (motor BPEL),cuenta con un diseñador basado en Eclipse llamado Intalio|Designer, el cual al ser salvado el diseño del modelo de negocio modelado con las especificaciones de BPMN se genera automáticamente código BPEL, luego es ejecutado por el componente Intalio|Server. Este sistema tiene tres formas de implementar reglas de negocios:

- Data mapper.
- En BPMN como un proceso.
- En un motor de reglas de negocios, para casos más complejos.

El modelo de negocio de Intalio, está basado en una licencia doble. IntalioBPMS se distribuye en 3 ediciones: La edición abierta de IntalioBPMS, bajo una licencia pública de Mozilla (MPL[8]), una edición para la comunidad de IntalioBPMS, y la edición de IntalioBPMSEnteprise.

- La edición abierta incluye aproximadamente el 95% del código usado para la edición comunitaria y la de empresa. La edición abierta está desplegada sobre el servidor de Apache Gerónimo J2EE, y la base de datos de MySQL.
- La edición comunitaria se distribuye con el servidor de IBM WebSphere, junto con MySQL.
- La edición empresarial puede desplegarse en otros servidores y bases de datos, su mayor características es el manejo transaccional.

Sus componentes son:

- Una herramienta para el diseño de los procesos de negocio, basada en Eclipse (ambientes grafico para el desarrollo java).
- Un motor que ejecuta los artefactos de software generados por el diseñador de procesos.
- Un Servidor de Aplicaciones donde residirán los servicios de procesos de negocio que se despliegan.[13, 16]

JbossjBPM: esta herramienta no utiliza XPDL para la definición de procesos de negocios, pero tiene soporte para este estándar, en sustitución de este usan JPDL[9], 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 4[10],

[8] Del inglés Mozilla Public License.
[9] Del inglés jBPM Process Definition Language.
[10] Es un servidor de aplicaciones de código abierto implementado en Java puro.

completamente modificable, y siempre sobre Hibernate, eso nos permite correr el flujo de trabajo sobre cualquier base de datos.[17]

ADONIS: es un software de Gestión de Procesos de Negocios desarrollado por la compañía BOC Group, soportado sobre el marco de trabajo ADONIS Process Portal o APP, este complementa a ADONIS mediante acceso web online permitiendo con esto que cada rol tengan una interfaz definida y las funciones que requieran para el desempeño de sus tareas. Su concepto de estar basado en roles ofrece a los empleados exactamente aquella información y funciones que son relevantes para sus tareas. APP ofrece un acceso directo 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. 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 le permite visualizar modelos (procesos de negocio, procedimientos, mapas de procesos, organigramas, mapas de sistemas y paisajes TI). La reutilización de estructuras garantiza una modelación eficiente, mientras que las distintas vistas de los modelos resaltan las características del modelo en cada caso. Con la simulación de ADONIS puede investigar los efectos de cambios en sus procesos de negocio y/o la estructura de su organización y analizar las repercusiones que pueden producir. ADONIS le ofrece para tal propósito cuatro algoritmos de simulación que puede utilizar para la planificación estática y dinámica de los recursos y necesidades de personal. Al igual que en el componente de análisis, los resultados aparecen gráfica y tubuladamente o incluso reproducidos con una función de animación. El componente Import/Export le ofrece una interfaz abierta mediante la cual puede importar/exportar todos los modelos contenidos en ADONIS gracias a los formatos ADL [11] y XML. Esto garantiza la migración de datos entre instalaciones ADONIS distribuidas (independencia de la base de datos) y con otras aplicaciones.[13, 18]

Workflow Fundation (WF): esta herramienta de gestión de flujo de trabajo desarrollado por Microsoft, basado en librerías .Net integra varias herramientas que juntas proveen la infraestructura necesaria para ejecutar flujos de trabajos. Presenta dos formas de crear flujos de trabajo: mediante un diseñador gráfico

[11] Del inglés ADONIS Definition Language.

que utiliza como lenguaje de modelado XAML [12] o implementándolo en clases .NET como C # o VB.NET. Los flujos de trabajo pueden ser ejecutados utilizando uno de los siguientes métodos:

- WorkflowInvoker: ejecuta flujos de trabajo en el subproceso de llamada (es decir, que no se crea un nuevo hilo para el flujo de trabajo). Esto significa que el proceso de llamada esperará por este para completarlo.
- WorkflowApplication: ejecuta flujos de trabajo en un nuevo hilo (cuando la aplicación realiza la llamada no se detendrá su ejecución).
- WorkflowServiceHost, ejecuta el flujo de trabajo como un servicio WCF [13] implementado con el fin de permitir la conexión entre aplicaciones orientadas a servicios. El servicio de flujo de trabajo resultante por lo general utiliza los datos de la red como insumos para las actividades contenidas.

La persistencia de un flujo de trabajo se realiza guardando los datos en un modelo de datos (como SQL Server). Las instancias pueden ser recargadas después de un período de tiempo especificado, o cuando este recibe un mensaje. Al eliminarlos de la memoria, el motor puede aumentar el número de estos activos que un sistema pueda manejar, lo que aumenta la escalabilidad. Los datos son consumidos por las actividades que utilizan argumentos y variables, que mantiene el tiempo de ejecución. Utilizando argumentos y variables para almacenar datos para las actividades significa que el tiempo de ejecución tiene acceso a un estado completo de la actividad en el caso de una persistencia previa. En el tiempo de ejecución también se pueden correlacionar los mensajes entrantes y los datos a una instancia específica en el caso de que varios se están ejecutando al mismo tiempo.[19]

1.4 Herramientas de transformación

EnhydraJaWe: es un sistema gráfico de flujo de trabajo, código abierto implementado en Java. Este producto permite al usuario crear, administrar y revisar las definiciones de flujo de trabajo de procesos de negocios. Soporta como lenguaje nativo XPDL y también conexiones LDAP[14], está implementada sobre el marco de trabajo denominado Shark siendo utilizada por el motor de flujo de trabajo del mismo, llamado Enhydra Shark. Esta complementa tres metas fundamentales:

[12] Del inglés Extensible Application Markup Language.
[13] Del inglés Windows Communication Foundation.
[14] Del inglés Lightweight Directory Access Protocol. Protocolo a nivel de aplicación el cual permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

- Definición gráfica de procesos de negocios.
- La exportación de definiciones de proceso de negocios de un XPDL.
- La importación de cualquier XPDL válida y su representación gráfica.[20]

BusinessProcessIncubator: (BPI) es una herramienta online el cual tiene el propósito de optimizar los métodos llevados a cabo por los (BPMs). Su registro gratuito proporciona acceso a una gran cantidad de recursos relacionados con los BPMs, como pueden ser las mejores presentaciones de práctica, la verificación y conversión de varios lenguajes de definición, así como de ejecución relacionados con servicios web. Esta aplicación permite convertir, transformar, validar y visualizar cualquier archivo descrito en XPDL, o cualquier modelado de procesos de negocios descrito en BPMN.[21]

Una vez estudiadas las herramientas se descartan las mismas ya que aunque todas soportan XPDL para la definición de sus procesos de negocios, orientan su ejecución a servicios web utilizando BPEL, incumpliendo con la arquitectura orientada a servicios de Sauxe. Tampoco cumplen con las especificaciones del módulo de ejecución de su herramienta de flujo de trabajo ni se encuentran implementadas en el lenguaje de programación PHP.

1.6 Metodología de desarrollo de software.

Las Metodologías de Desarrollo de Software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto software. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software, pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que existan una gran variedad de metodologías para la creación del software. Se podrían clasificar en dos grandes grupos:

- Las metodologías orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Estas metodologías son llamadas Metodologías Pesadas.
- Las metodologías orientadas a la interactuación con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando, son llamadas Metodologías ágiles.

Existen también las híbridas, formadas a partir de la combinación de distintas metodologías, tomando de cada una los elementos que se necesiten y adaptándose al proceso de desarrollo de software que se realizará.[22]

Modelo de desarrollo de Software orientado a componentes del ERP-Cuba

Los elementos y relaciones de un proceso de desarrollo de software deben responder a, Quién debe hacer Qué, Cuándo y Cómo. Esto se logra modelando las interacciones y relaciones que suceden entre los roles, las actividades que estos desarrollan y los artefactos que se generan o actualizan durante el proceso.

Quién: Las personas participantes en el proyecto de desarrollo desempeñando uno o más roles específicos.

Qué: Un artefacto es producido por un rol como resultado del desarrollo de sus actividades, estos se especifican utilizando notaciones las cuales son elaboradas mediantes herramientas destinadas a este propósito.

Cómo y Cuándo: Las actividades son una serie de pasos que lleva a cabo un rol durante el proceso de desarrollo. El avance del proyecto está controlado mediante hitos que establecen un determinado estado de terminación de ciertos artefactos.

El CEIGE utiliza su propio modelo de desarrollo de software basado en componentes. El mismo está basado en principios y buenas prácticas de metodologías ágiles, fue elaborado teniendo en cuenta las características especiales que presenta la UCI y tiene un valor social representativo, ya que implica mejoría en aspectos importantes como la planeación, formación y satisfacción del equipo de trabajo. Entre sus principales características podemos mencionar:[23]

- Está orientado a la reutilización de componentes parametrizables.
- Se utilizan solamente los artefactos necesarios para documentar el producto.
- Se desarrollan partes pequeñas y se ensambla después el producto.
- Los flujos se integran a través de la arquitectura de software y de negocio. Todos los flujos de desarrollo de cada fase se integran siempre detrás de la arquitectura de software y de negocio.
- Existen áreas dedicadas a tareas específicas y especializadas en temas específicos.
- Se hacen pruebas continuas sobre los compones y/o productos y los cambios se hacen a tiempo, antes de poner un componente en el repositorio se hacen pruebas unitarias, y cuando se va a

utilizar como parte de otro producto se hacen pruebas de integración, al igual que antes de liberar el producto también. Todo esto demuestra que se está probando en todo el proceso de desarrollo.

- Las áreas de proceso están especializadas, en temas de apoyo a la producción que es el elemento fundamental de la Subdirección y llevan unido al proceso productivo procesos tales como Investigación, Formación, Gestión del capital humano y calidad.
- La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software.
- En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.
- Preocupación por el aprendizaje de los desarrolladores.

Contempla las fases principales del desarrollo de un producto de software, adecuándolas a las necesidades del centro, como son modelación de procesos de negocio, definición de requisitos, diseño de la arquitectura de software, análisis de reutilización y estrategia de desarrollo, diseño detallado, implementación e integración y pruebas; además de analizar los procesos de gestión de proyecto.[23]

1.7 Librerías y Marcos de trabajo

Librerías: son un conjunto de subprogramas utilizados para desarrollar software. Estas contienen código y datos, que proporcionan servicios a programas independientes, que son frecuentemente utilizados y que no necesitan ser modificados, es decir, pasan a formar parte de éstos. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular. Algunos programas ejecutables pueden ser a la vez programas independientes y librerías.

Marcos de trabajo (framework): en el desarrollo de software, un marco de trabajo o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, permite que otros proyectos de software se organicen y desarrollen fácilmente. Típicamente, puede incluir soporte de programas, librerías, lenguajes interpretados y otras herramientas, para beneficiar la unión y desarrollo de los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio, también provee una estructura y una metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio.[24]

Zend Framework

Concebido e implementado como una biblioteca robusta, componente rico en características para desarrolladores de PHP, que permite realizar de forma rápida y eficiente una gran variedad de tareas comunes de desarrollo de aplicaciones, incluyendo la creación y validación del formulario de entrada, procesamiento de XML, la generación de menús dinámicos, paginar los datos, todo esto en colaboración con los servicios web. 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.[25]

ZendExt Framework

Cumple con todas las características de Zend Framework ya que está construido a partir de este. Cuenta con un motor de reglas para las validaciones del servidor, un controlador vertical para el control de acciones realizadas desde las vistas, contiene un IoC [15]para la comunicación entre módulos mediante servicios internos, además, está integrado con el ORM Doctrine Framework para trabajo en la capa de abstracción a base de datos.[25]

Sauxe

Sauxe es un marco de trabajo el cual se implementa con una arquitectura en capas, que a su vez presenta en su capa superior un MVC[16]. Contiene un conjunto de mecanismos reutilizables que suministra la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Siguiendo el paradigma de independencia tecnológica por el cual apuesta el país, reutiliza las siguientes tecnologías libres

[15] Del inglés Inversion of Control. Componente para la comunicación entre subsistemas.
[16] (**MVC**) Modelo Vista Controlador. Es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

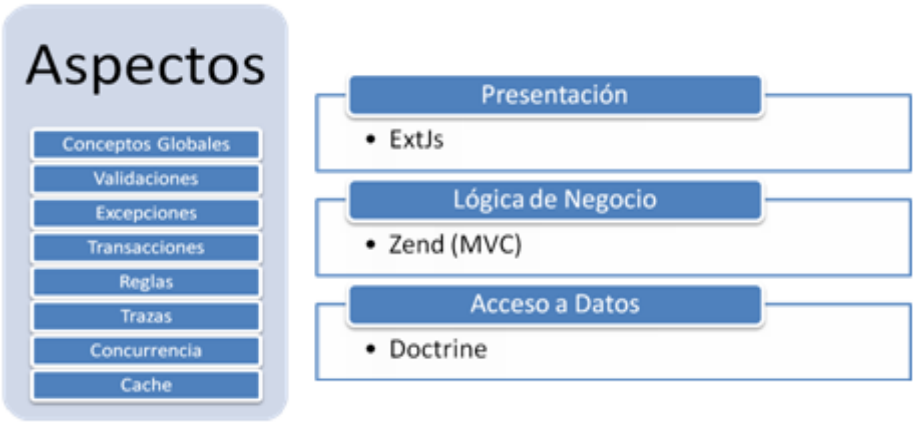


Figura 2: Arquitectura de Sauxe.

Sauxe presenta en su arquitectura tecnológica la capacidad de gestionar y configurar la cache de forma dinámica. Permite la integración de las aplicaciones o dominios de soluciones que se instancien o construyan con el mismo. Esta integración permite la comunicación entre componentes expresando sus contratos en servicios mediante el formato WSDL[17]. El soporte de integración permite de manera transparente para el usuario, consumir un servicios gestionado por la plataforma desde 3 alternativas específicas, una por vía de un servicio Web, otra por algún mecanismo de inversión de control que deberá proveer la arquitectura de integración del marco de trabajo, y la otra desde algún servidor de comunicación o servicio de mensajería. Provee un mecanismo de abstracción de la capa relacional de persistencia, este mecanismo de mapeo relacional de objeto, permite además contar con un mecanismo de SQL Helper, el cual permite persistencia compuesta, carga perezosa, actualización y modificación sincronizada tanto en el modelo mapeado como en el modelo relacional. Sauxe también permite la administración y configuración dinámica de trazas de la solución, permitiendo configurar la arquitectura de traza de un dominio de aplicación específico. Como mecanismo de seguridad provee formas de verificación de identidad mediante la autenticación, de protección hacia los recursos del sistema mediante la autorización, permitiendo con esto que solo sólo sean usados por aquellos consumidores a los que se les ha concedido autorización. También es capaz de garantizar la personalización de las aplicaciones de cualquier dominio a nivel de cada usuario así como la administración de conexiones a la base de datos.[26]

[17] Del inglés Web Services Description Language. Formato XML que se utiliza para describir servicios Web.

ezComponents

Es un marco de trabajo de código abierto desarrollado en el lenguaje PHP que facilita el desarrollo de aplicaciones web, este tiene la característica de que sus componentes son independientes permitiendo así que estos sean reutilizados e implantados a cualquier otra solución de forma independiente. Se encuentra bajo la licencia BSD [18] permitiendo así la modificación de su código fuente. La biblioteca ezComponents no dicta la forma de organizar su aplicación, sino que proporciona la flexibilidad para decidir cómo y cuándo usar cada componente. Además, todas las API [19]de los componentes están diseñados de una manera similar, y lo más coherente posible. Esta solución está compuesta por un módulo de gestión de flujo de trabajo que tiene el propósito de proporcionar la funcionalidad básica de un sistema de flujo de trabajo basado en actividades, incluyendo la definición y ejecución de las especificaciones de flujo de trabajo. Este implementa un lenguaje nativo de definición y ejecución basados en estructura XML o modelo de datos los cuales se gestionan mediante las clases ezcWorkflowDefinitionStorageXml y ezcWorkflowDatabaseDefinitionStorage estas proporcionan la funcionalidad para guardar las definiciones de flujo de trabajo y cargarlas.[27]

Doctrine

Es un mapeador de objetos relacional (ORM por sus siglas en inglés) escrito en PHP que proporciona persistencia para objetos de este lenguaje. Está por encima de la capa de abstracción a la base de datos, uno de sus características es la posibilidad de escribir consultas a la base de datos a partir del tratamiento con objetos en PHP llamado Doctrine Query Language (DQL). Puede generar clases a partir de una base de datos creada, también permite agregar funcionalidades, y especificar relaciones a las mismas. [28]
Entre las características de Doctrine podemos encontrar

- Soporte para datos jerárquicos.
- Soporte para hooks [20] y eventos para manejar la lógica de negocio relacionada.
- Herencia.
- Transacciones ACID[21].

[18] Del inglés Berkeley Software Distribution.
[19] Del inglés Application programming interface. Conjunto de funciones y procedimientos que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción.
[20] Métodos que pueden validar o modificar las escrituras y lecturas de la base de datos.
[21] Conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción (Atomicidad, Consistencia, Aislamiento y Durabilidad en español).

- 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.

1.8 Herramientas de desarrollo

A continuación se describen las herramientas propuestas para el desarrollo de la solución propuesta.

Apache

El servidor HTTP [22]Apache es un servidor web HTTP de código abierto multiplataforma que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. Al ser el servidor web por excelencia da la facilidad de encontrar ayuda y soporte para el mismo. 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.[29]

Visual Paradigm

Visual Paradigm para UML 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. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.[30]

PostgreSQL

Es un sistema gestor de base de datos basado en software libre y liberado bajo la licencia BSD[23]. Está considerado como el sistema de gestión de bases de datos de código abierto más potente del mercado,

[22] Del inglés Hypertext Transfer Protocol. Es el protocolo usado en cada transacción realizada en internet o cualquier red, sigue el esquema petición-respuesta entre un cliente y un servidor.
[23] Del inglés Berkeley Software Distribution.

ofrece nuevos conceptos como son: clases, herencia 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.[31]

PGAdmin III

Es el cliente más completo y popular desarrollado en C ++ [24]que utiliza la librería gráfica multiplataforma wxWidgets18, permitiendo que se pueda usar en sistemas operativos como: GNU/Linux, MacOS y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3, ejecutándose en cualquier plataforma. PGAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente su administración.[32]

Mozilla Firefox

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.[33]

Subversion

La herramienta es libre y de código fuente abierto, se emplea para manejar los ficheros y directorios a través del tiempo haciendo posible la recuperación en caso de una perdida inminente, la recuperación se lleva a cabo mediante la salva existente en el repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque guarda todos los cambios hechos a sus ficheros y directorios. Subversion puede acceder al repositorio a través de la red, por lo que es usado por desarrolladores que se encuentran en distintas estaciones de trabajo.

[24] Lenguaje de programación apreciado por la eficiencia del código que produce, es uno de los lenguajes de programación más utilizado para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Una característica importante de Subversión es que los archivos versionados 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 cierto punto del tiempo.
Presenta también las siguientes características.

- Mantiene versiones no sólo de archivos, sino también de directorios
- Mantienen 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.

Tener en cuenta que Subversion ayuda a medir la calidad y cantidad del trabajo realizado en una unidad de tiempo. Esta visibilidad instantánea nos permite observar la productividad del equipo de trabajo, así como los beneficios a escala administrativa para un líder de grupo.[34]

1.9 Lenguajes de desarrollo y de modelado

Llamamos lenguaje de modelado de objetos a un conjunto estandarizado de símbolos y formas disponibles para modelar un diseño de software o parte de este.[35]

UML

UML, por sus siglas en inglés, Unified Modeling Language: es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; respaldado por el OMG. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de

negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Utilizado para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir.[36]

Lenguajes del lado del servidor

Una aplicación en el lado del servidor es cualquier programa o conjunto de instrucciones diseñadas con la finalidad de que un servidor Web las procese para realizar alguna acción. Las aplicaciones del lado del servidor están escritas mediante un lenguaje de programación.

PHP

Lenguaje de programación interpretado, diseñado originalmente para la creación de Página web dinámicas. Es usado principalmente en interpretación del lado del servidor pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools). Publicado bajo la PHP License, es considerado como software libre.[37]

Ventajas

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

Desventajas

- No posee una abstracción de base de datos estándar, sino bibliotecas especializadas que permiten conectarse con bases de datos o a servicios web, parsear XML, enviar email, generar PDFs, generar imágenes, entre otras.
- Por sus características promueve la creación de código desordenado y complejo de mantener.

- Todo el trabajo lo realiza el servidor, por tanto puede ser más ineficiente a medida que aumenten las solicitudes.
- La orientación a objetos es aún muy deficiente en aplicaciones grandes.[37, 38]

1.10 Conclusiones parciales

Una vez realizado el estudio del entorno de desarrollo se descartan las herramientas estudiadas para darle solución al problema planteado, ya que no cumplen con las especificaciones establecidas por el módulo de ejecución de flujo de trabajo del marco de trabajo ezComponents ni las de Sauxe. Se establecen los requerimientos y herramientas necesarias a tener en cuenta para la implementación de un traductor de lenguaje que permita la transformación de XPDL a un lenguaje de ejecución de procesos, para la solución de flujos de trabajo en el marco de trabajo Sauxe.

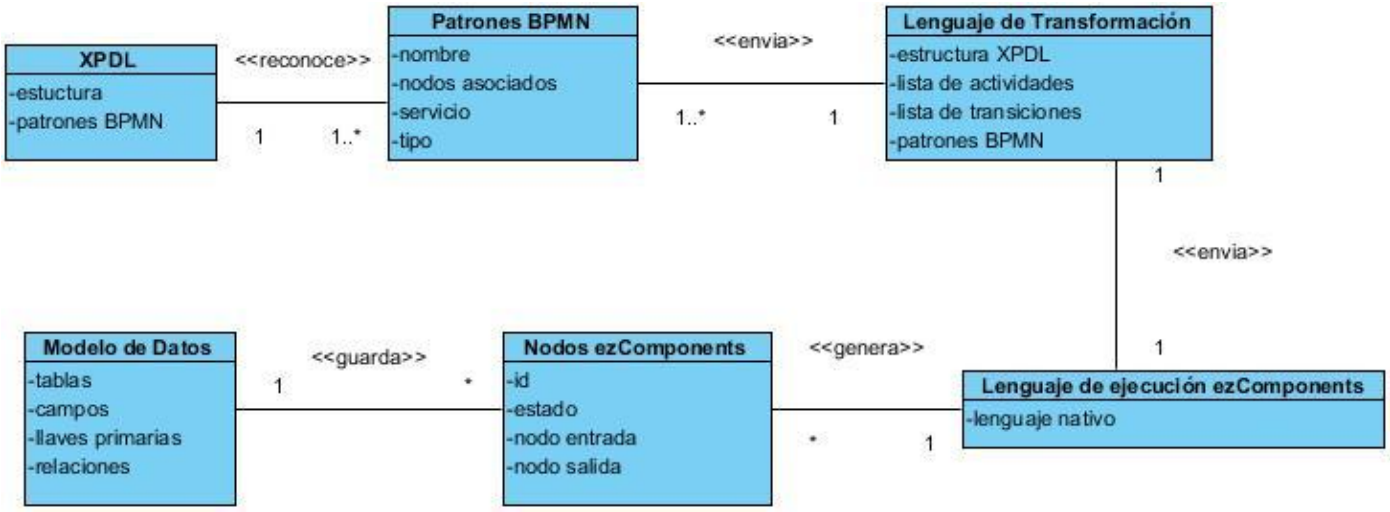
CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

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 Modelo conceptual

Para obtener un mejor dominio del sistema se realiza el siguiente modelo conceptual en el que se describen aquellos objetos o conceptos reales que son significativos para el problema.



. Figura 3 : Modelo Conceptual.

En el diagrama se muestra gráficamente como a partir de un archivo de tipo XPDL se reconocen los patrones BPMN, estos una vez reconocidos son recibidos por un lenguaje de ejecución nativo el cual los

transforma en nodos de tipo ezcWorkflowNode, y al mismo tiempo persistiéndolos en un modelo de datos para su posterior ejecución.

2.3 Especificación 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 utilizó la técnica de tormenta de ideas que no es más que reuniones en grupos cuyo objetivo es que cada participante muestre su idea libremente. Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. 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.

2.3.1 Requisitos funcionales

Los requisitos funcionales identificados se enuncian a continuación:

.

- ✓ RF1 Cargar XPDL
- ✓ RF2 Identificar patrones BPMN.
- ✓ RF3 Transformar patrones BPMN a los nodos correspondientes utilizados por el motor de workflow del marco de trabajo ezComponents.
- ✓ RF4 Persistir flujo de trabajo en modelo de datos.

A continuación se describen cada uno de los requisitos funcionales

RF1 Cargar archivo XPDL

Precondiciones	Que exista el XPDL.
Flujo de eventos	
Flujo básico Cargar XPDL	

1.	Invocar a la función loadfile de la clase SimpleXML.
2.	El sistema devuelve un arreglo de objeto SimpleXML.
Pos-condiciones	
1.	Se guarda el arreglo de objetos SimpleXML en una variable.
Flujos alternativos	
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>><Condición que dio lugar a la extensión>	
1	N/A
2	N/A
Pos-condiciones	
1	N/A
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual 0127_Modelo conceptual

Conceptos	**XPDL**	Utilizados internamente: etiquetas
Requisitos especiales	Lenguajes de definición de procesos de negocio	
Asuntos pendientes	-	

RF2 Identificar patrones BPMN.

Precondiciones	Se cargó el XPDL
Flujo de eventos	
Flujo básico Identificar patrones BPMN	
1.	Invocar la función children de la clase SimpleXML.
2.	El sistema devuelve un arreglo de objetos SimpleXML que contiene los nombres de los hijos de la etiqueta correspondiente.
3.	Invocar la función attributes de la clase SimpleXML.
4.	El sistema devuelve un string con el nombre del atributo de la etiqueta.
5.	Invocar la función getName de la clase SimpleXML.
6.	El sistema devuelve un string de la petición realizada.
7.	La funcionalidad children solicita el nombre de la etiqueta.
8.	La funcionalidad attributes solicita los datos de la etiqueta seleccionada.
9.	El sistema devuelve el nombre y los datos de la etiqueta seleccionada.
10.	La función crearActividades de la clase de XPDLParserActividades construye objetos de tipo actividad.
11.	Se guarda los objetos actividades en el atributo $lista_actividades.

12.	La función crearTransiciones de la clase de XPDLParserTransiciones construye objetos de tipo transición.
13.	Se guarda los objetos transiciones en el atributo $lista_transiciones.
14.	Se invoca a la función identificador de la clase XPDLParserController.
Pos-condiciones	
1.	Se identificaron los patrones BPMN existentes en el XPDL.
Flujos alternativos	
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>><Condición que dio lugar a la extensión>	
Pos-condiciones	
1.	
Validaciones	
1.	Se validan los datos según lo establecido en el Modelo conceptual 0127_Modelo conceptual.

Conceptos	XPDL	*Utilizados internamente:* etiquetas
Requisitos especiales	Lenguajes de definición de procesos de negocio Patrones BPMN	
Asuntos pendientes	N/A	

RF3 Transformar patrones BPMN a los nodos correspondientes utilizados por el motor de workflow del framework ezComponents.

Precondiciones	Requisito Transformar patrones BPMN a sus nodos correspondientes en el lenguaje del framework ezComponents.
Flujo de eventos	
Flujo básico Transformar patrones	
1.	Se invoca a la función identificador de la clase XPDLParser Controller.
2.	El sistema devuelve un objeto de tipo ezcWorkflowNode.
3.	Se añade el objeto de tipo ezcworkflowNode a la clase ezcWorkflow.
3.	El sistema devuelve un objeto de tipo ezcWorkflow.
Pos-condiciones	
1.	Quedan transformados los patrones BPMN a nodos de tipo ezcWorkflowNode.
Flujos alternativos	
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>><Condición que dio lugar a la extensión>	

Pos-condiciones		
1.	N/A	
Validaciones		
1.	Se validan los datos según lo establecido en el Modelo conceptual 0127_Modelo conceptual.	
Conceptos	**ezcWorkflowNode**	Utilizados internamente: -rol #_contruct() #activeNode() +_toString() +accept() +activate() +addInNode() +addOutNode() +execute() +getActivatedFrom() +getConfiguration() +GetId() +getInNodes() +getOutNodes() +getThreatId() +getState()
Requisitos especiales	Patrones BPMN	
Asuntos pendientes	N/A	

RF4 Persistir flujo de trabajo en modelo de datos.

Precondiciones	Que este creado el objeto ezCworkflow
Flujo de eventos	
Flujo básico <<Nombre del flujo básico>>	
1	Se invoca a la función save de la clase ezCworkflowDataBaseDefinitionStorage
Pos-condiciones	
1.	El objeto se persiste en el modelo de datos.
Flujos alternativos	
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>><Condición que dio lugar a la extensión>	
Pos-condiciones	
1.	N/A
Validaciones	
1.	Se validan los datos según lo establecido en el Modelo conceptual 0127_Modelo

	conceptual.	
Conceptos	**Flujo de trabajo**	nombre Id
Requisitos especiales	Clases de Doctrine. Libreria Zend Postgres SQL	
Asuntos pendientes		

2.3.2 Requisitos no funcionales

Ya que la solución que se desea implementar esta propuesta para el marco de trabajo SAUXE, desarrollado en el proyecto Paquete de herramientas, 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

Software

Para el cliente:

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

Para el servidor:

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

Rendimiento

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

Seguridad

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

Hardware

Para el servidor:

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

Para el cliente:

- Requerimientos mínimos: procesador Pentium III a 1GHz con 256Mb de memoria RAM.
- Tarjeta de red el mismo del anterior.[39]

2.3.4 Validación de los requisitos funcionales

Una de los procesos de la Ingeniería de requisitos es la validación, la cual tiene como objetivo demostrar que los requisitos definidos cumplan con las necesidades y expectativas del cliente o usuario. Los costos de errores en los requerimientos son altos, por lo cual, la validación es muy importante. Fijar un error de requerimiento después del desarrollo puede resultar en un costo 100 veces mayor que fijar un error en la

implementación. 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. [40]

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 utilizó la técnica de Revisión técnica formal.

- 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.

2.4 Diagrama de clases

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases, contiene información como asociaciones, atributos, métodos y dependencias.[35]

A continuación se muestra el diagrama de clases del diseño basado en estereotipos web que se genera durante el proceso de desarrollo.

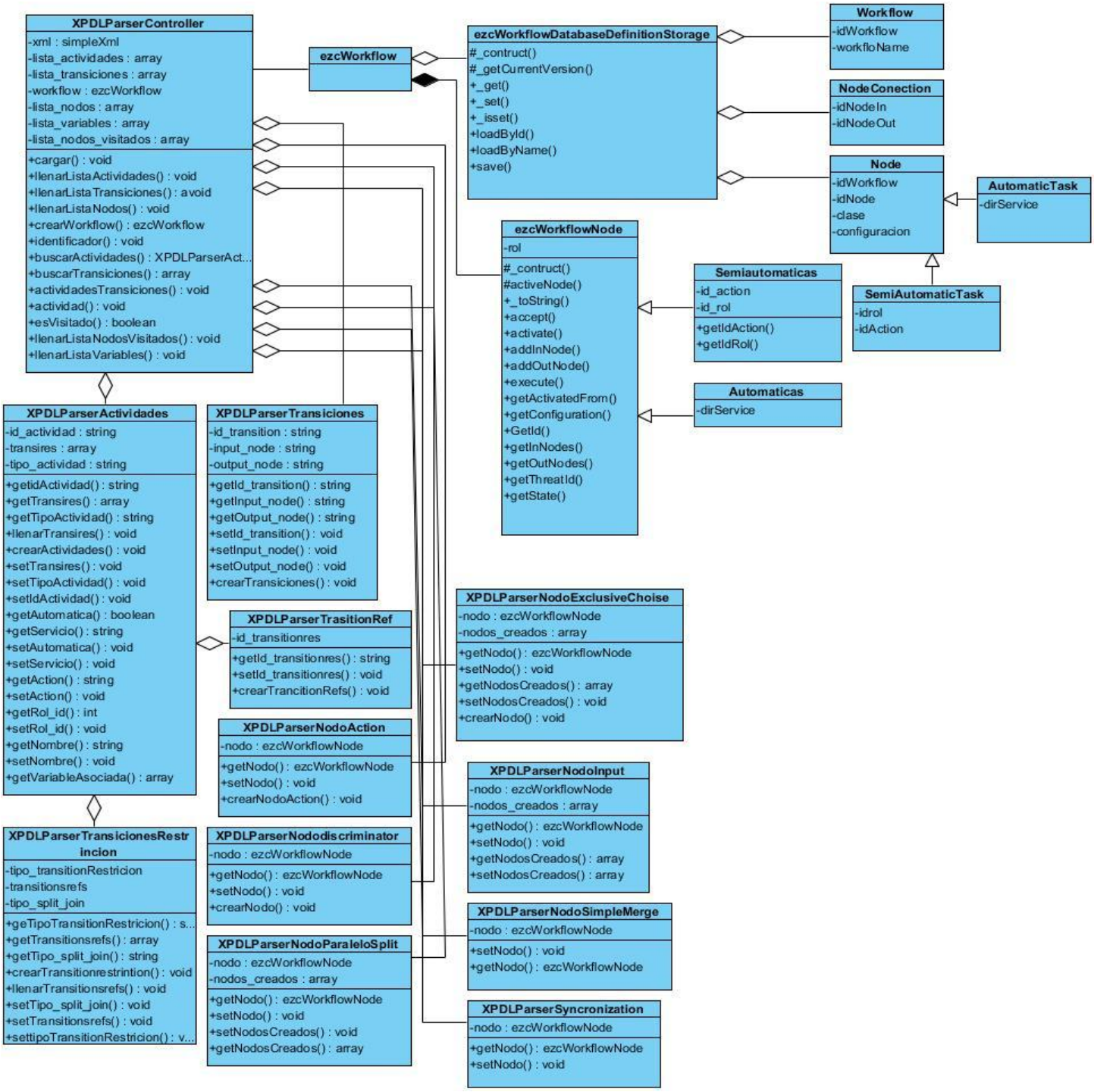


Figura 4: Diagrama de clases del diseño

En el diagrama se muestra la forma en que se identifican los patrones BPMN definidos en un archivo XPDL, estos a su vez se transforman a nodos de tipo ezcWorkflowNode utilizando las clases correspondientes ezComponents y de acceso a datos quedando como resultado final un objeto de tipo ezcWorkflow guardado en una base de datos.

2.5 Modelo de datos

Un modelo de datos es un lenguaje orientado a describir una base de datos. Este permite describir las estructuras de datos de la base, es decir el tipo de los datos que hay en la base y la forma en que se relacionan. Las restricciones de integridad que no son más que un conjunto de condiciones que deben cumplir los datos para reflejar correctamente la realidad deseada y las operaciones de manipulación de los datos las cuales pueden ser, operaciones de agregado, borrado, modificación y recuperación de los datos de la base.[41]

El componente para la ejecución de flujos de trabajo de Sauxe realiza la ejecución de los mismos auxiliándose en datos guardados en un modelo de datos, este se elaboró teniendo en cuanta la reducción a la mínima expresión de los campos nulos.

El modelo utilizado consta de 10 tablas. Teniendo en cuenta el requisito funcional “Persistir flujo de trabajo en modelo de datos” se accede a este mediante la clase ezcWorkflowDatabaseDoctrineDefinitionStorage, la cual recibe un objeto ezcWorkflow inicializado anteriormente. Esta clase tiene el objetivo de persistir un flujo de trabajo atendiendo a su configuración, donde las tareas semiautomáticas y automáticas son guardadas en las tablas semiautomatic_task, y automatic_task, respectivamente. Las mismas contienen los campos node_id, action_id, rol_id y service_dir y heredan de la tabla node. Con el propósito de conocer el estado de las instancias de los flujos de trabajo se obtiene la tabla task_list, la cual contiene los campos, action_name y rol_id y se encuentra relacionada con la tabla node y workflow.

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

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

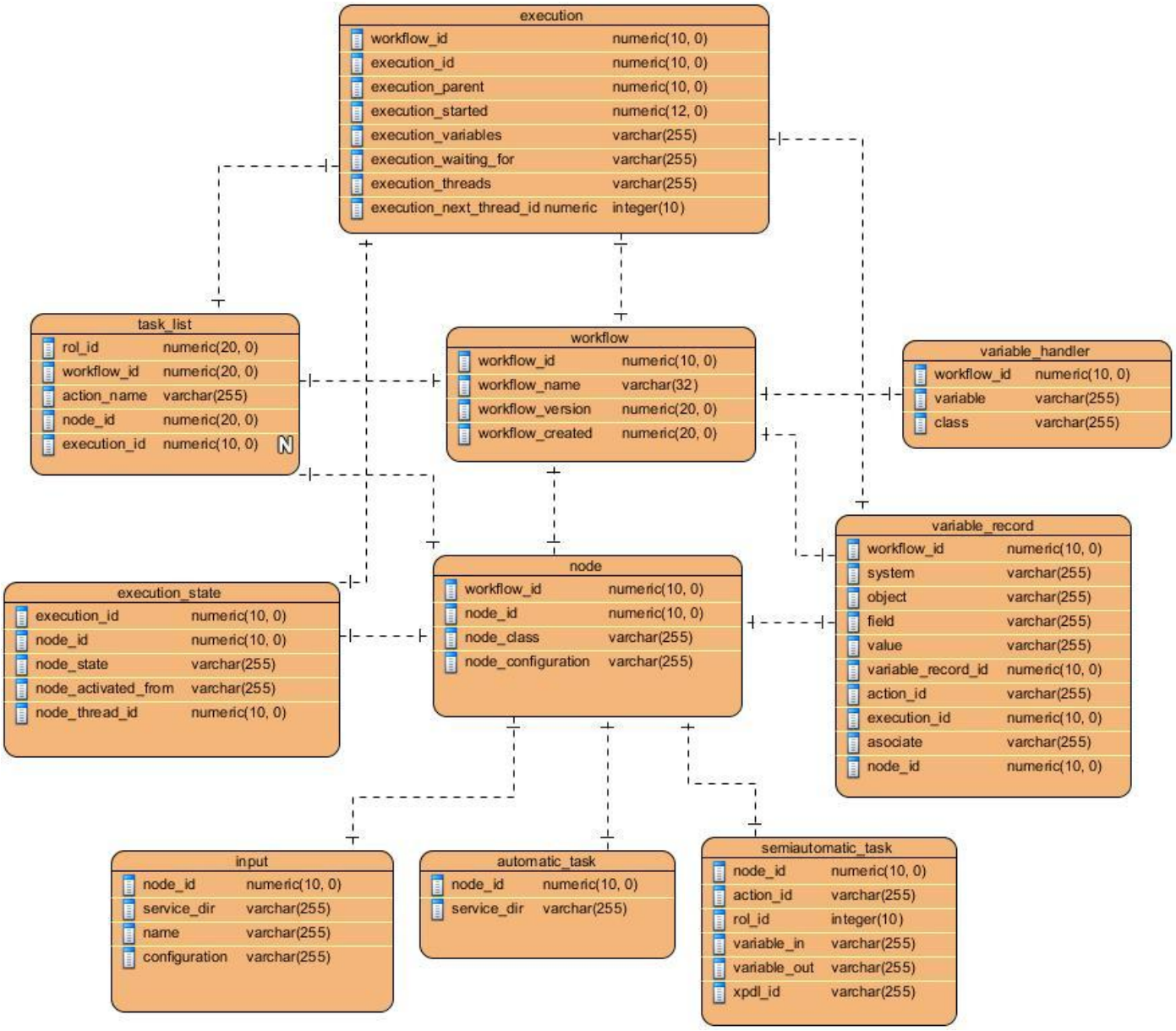


Figura 5: Modelo de datos.

2.6 Patrones de Diseño empleados

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.

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

Singlenton: garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Dicho patrón se utiliza en la clase XPDLParserController para lograr una única instancia.

Creador: este patrón se evidencia en las clases encargadas de crear objetos, por ejemplo en las clases XPDLParserActividades, XPDLParserTransiciones y ezcWorkflow especializada en la creación de los objetos de tipo actividad, transiciones y de flujo de trabajo respectivamente.

Visitador: Es un patrón de comportamiento, que permite definir una operación sobre objetos de una jerarquía de clases sin modificar las clases sobre las que opera implementado en la clase XPDLParserController verifica la existencia de objetos ezcWorkflowNode.

2.7 Patrones arquitectónicos

La arquitectura de software de un sistema de programa o computación es la estructura o estructuras del sistema, que comprende elementos de software, las propiedades externamente visibles de esos elementos, y las relaciones entre ellos. Es el proceso de definición de una solución estructurada que cumple con todos los requisitos técnicos y operativos, mientras que la optimización de los atributos comunes de calidad, tales como rendimiento, seguridad y manejabilidad. Se trata de una serie de decisiones basadas en una amplia gama de factores, y cada una de estas decisiones pueden tener un impacto considerable en la calidad, rendimiento, mantenimiento, y el éxito global de la aplicación. Estas decisiones significativas se toman teniendo en cuenta:

- La organización del sistema de software.
- La selección de los elementos estructurales y sus interfaces a través de los cuales se constituye el sistema.
- Su comportamiento, según resultados de las colaboraciones entre esos elementos.

- El estilo arquitectónico que guía esta organización: los elementos estáticos y dinámicos; sus interfaces, su colaboración y su composición. [42]

Patrón de arquitectura por capas

El Patrón de arquitectura por capas es una de las técnicas más comunes que los arquitectos de software utilizan para dividir sistemas de software complicados. Al pensar en un sistema en términos de capas, se imaginan los principales subsistemas de software ubicados de la misma forma que las capas de un pastel, donde cada capa descansa sobre la inferior. En este esquema la capa más alta utiliza varios servicios definidos por la inferior, pero la última es inconsciente de la superior. Además, normalmente cada capa oculta las capas inferiores de las siguientes superiores a esta.

Los beneficios de trabajar un sistema en capas son

- Se puede entender una capa como un todo, sin considerar las otras.
- Las capas se pueden sustituir con implementaciones alternativas de los mismos servicios básicos.
- Se minimizan dependencias entre capas.
- Las capas posibilitan la estandarización de servicios.
- Luego de tener una capa construida, puede ser utilizada por muchos servicios de mayor nivel. [5, 35, 43]

A continuación se describen las capas definidas para la solución propuesta siguiendo el patrón de arquitectura por capas.

1. Capa controladora: referente a la intermediaria entre la capa del negocio y entidades.

2. Capa de Reglas de Negocio: también denominada Lógica de Dominio, esta capa contiene la funcionalidad que implementa la aplicación. Involucra cálculos basados en la información dada, datos almacenados y validaciones. Controla la ejecución de la capa de acceso a datos y servicios externos.

3. Capa de entidades: esta capa contiene la lógica de comunicación con otros sistemas que llevan a cabo tareas de la aplicación siendo esta una base de datos, responsable del almacenamiento persistente de información.

2.8 Conclusiones Parciales

El desarrollo del capítulo estuvo enfocado al análisis y diseño del componente para la transformación de XPDL a un lenguaje de ejecución de procesos para la solución de flujos de trabajo en el marco de trabajo Sauxe. Ilustrando el proceso con los principales artefactos que propone el modelo orientado a componentes propuesto por el centro CEIGE. Haciendo uso de la herramienta CASE[25]- Visual Paradigm- se generó, el modelo conceptual que representa los conceptos más importantes para la solución que se desea desarrollar. 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.

[25] Del ingles Computer Aided Software Engineering.

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 Modelo de implementación

El modelo de implementación es una colección de componentes y los subsistemas que los contienen. Estos componentes incluyen: ficheros de código fuente, y otros tipos de ficheros necesarios para la implantación y despliegue del sistema. Describe también, como se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y como dependen de los componentes unos de otros.[35]

Estándares de codificación

Los estándares de codificación son reglamentos de la programación que están enfocadas a la estructura y apariencia física del programa con el objetivo de facilitar la lectura, comprensión y mantenimiento del código. Un estándar de codificación comprende todos los aspectos de la generación de código. Los programadores deben implementar un estándar de forma prudente que tienda siempre a lo práctico. La legibilidad del código fuente influye directamente en el entendimiento que puedan tener los desarrolladores del equipo de trabajo, pues todo software tiene que someterse constantemente a mantenimiento y mejora de sus funcionalidades. El mejor método para lograr que un grupo de desarrolladores mantenga un código de calidad es establecer un estándar de codificación sobre el cual se realizarán revisiones rutinarias.

Los Estándares utilizados en la codificación fueron los siguientes

- Notación Pascal-Casing: los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula. Esta se evidencia al nombrar las clases del sistema. Ejemplo XPDLParserActividades.

- Notación Camel-Casing: es parecida al Pascal-Casing con la excepción que la letra inicial del identificador debe estar en minúscula. Esta hace su uso para declarar funcionalidades: Ejemplo: llenarListaActividades ().[44]

Nomenclatura de las clases controladoras

Para las clases controladoras se definió que después del nombre se les colocará como sufijo la palabra:"Controller". Ejemplo: XPDLParserController.php.

Nomenclatura de las clases de los modelos

Business (Negocio): para las clases que se encuentran dentro de Business se define que después del nombre se les colocará como sufijo la palabra:"Model" Ejemplo: workflowModel.

Bases del Dominio (Generated): para las clases que se encuentran dentro de Generated se define que después del nombre se les colocará como sufijo la palabra"Base". Ejemplo: workflowBase.

Nomenclatura de las variables

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

Nomenclatura para los comentarios

Con el propósito de lograr una mayor claridad y reutilización del código se da la necesidad de hacer comentarios en todo el código que se implemente dentro del desarrollo, esto facilita que a lo largo del tiempo el código pueda ser interpretado.

Se definió el uso de comentarios claros y precisos que ayudarán a una mejor comprensión del código implementado para que se entiendan sus objetivos específicos.

Nomenclatura para los comentarios en clases

Al implementar una clase se escribe una pequeña descripción en la cual se expliquen de forma general el propósito de la misma, así como su autor y sistema al que pertenece. Dicha descripción se escribe de la siguiente forma:

/**

* Nombre de la clase *

* Descripción *

* @author *

* @package *(módulo)

* @subpackage *(sub módulo)

* @copyright *

* @version (versión - parche)

/

```php
<?php
/**
 * Nombre de la clase * XPDLParserController
 * Descripción * Clase controladora del sistema
 * @author * Maximo E. Marin Lopez
 * @package * Herramientas
 * @subpackage * workflow
 * @copyright * CEIGE
 * @version 1.0
 */
```

Figura 6: Ejemplo de comentario de clase

Diagrama de componente:

El diagrama de componentes representa la estructura física del sistema, su agrupación por paquetes y las dependencias entre estos. En el diagrama que se expone a continuación se representa gráficamente la relación de EZSEL con un conjunto de librerías que permiten su interacción con Sauxe. El componente Portal invoca a Zend, este a su vez brinda una funcionalidad que permite el acceso a las clases ezComponents y el uso de funcionalidades relacionadas con flujos de trabajos, como nodos correspondientes a patrones BPMN y su configuración. Estas clases son utilizadas para la transformación del lenguaje de definición XPDL a uno de ejecución comprensible para el motor de flujos de trabajo de Sauxe el cual se persiste en un modelo de datos interactuando con Doctrine.

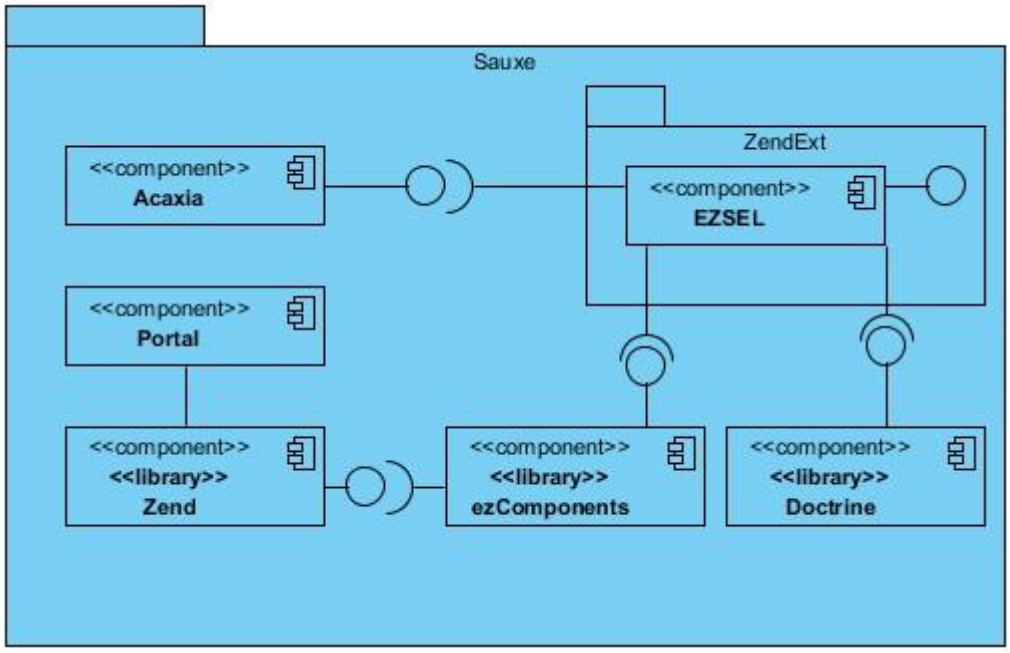


Figura 7: Diagrama de componentes

Diagrama de despliegue:

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. El desarrollador desde su estación de trabajo podrá acceder al sistema que estará desplegado en el servidor web. Dicho servidor se conectará al servidor de bases de datos que es necesario para el funcionamiento interno del marco de trabajo Sauxe.

Figura 8: Modelo de despliegue

Métricas de diseño

Según Pressman [’98], las métricas son la maduración de una disciplina, que van a ayudar a la evaluación de los modelos de análisis y de diseño, en donde proporcionarán una indicación de la complejidad de diseños procedimentales y de código fuente, y ayudarán en el diseño de pruebas más efectivas; Es por eso que propone un proceso de medición, el cual se puede caracterizar por cinco actividades:

- Formulación: la obtención de medidas y métricas del software apropiadas para la representación de software en cuestión.
- Colección: el mecanismo empleado para acumular datos necesarios para obtener las métricas formuladas.
- Análisis: el cálculo de las métricas y la aplicación de herramientas matemáticas.
- Interpretación: la evaluación de los resultados de las métricas en un esfuerzo por conseguir una visión interna de la calidad de la representación.
- Realimentación: recomendaciones obtenidas de la interpretación de métricas técnicas trasmitidas al equipo de software.

Las métricas del software permiten medir de forma cuantitativa la calidad de los atributos internos del producto, esto permite al ingeniero evaluar la calidad durante el desarrollo del sistema. Son varios los puntos de vista relacionados con la calidad del software. El objetivo de este epígrafe es desarrollar una evaluación del diseño propuesto para el componente de configuración visual para las excepciones en el marco de trabajo Sauxe. Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software con medidas que pueden ayudar al desarrollador a juzgar la calidad de un diseño a nivel de componente.

Atributos de la calidad que abarcan.

A continuación se detallan los atributos que se miden en la validación de la solución:

- **Responsabilidad**: consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

- **Complejidad del diseño**: consiste en la complejidad que posee una estructura de diseño de clases.
- **Complejidad de implementación**: consiste en el grado de dificultad que tiene que implementar un diseño de clases determinado.
- **Reutilización**: consiste en el grado de reutilización que presente una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento**: consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- **Complejidad de mantenimiento**: consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- **Cantidad de pruebas**: consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, módulo, clase, conjunto de clases, etc.) diseñado.
- **Eficiencia**: se refiere al esfuerzo que un usuario tiene que hacer para conseguir un objetivo
- **Efectividad**: variables que permiten medir la exactitud y la plenitud con la que se alcanzan los objetivos de una tarea concreta.

Las **métricas** aplicadas al diseño descrito en el capítulo anterior para la evaluación de la solución propuesta son las siguientes:

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

Tabla 1: Atributos de calidad evaluados por la métrica TOC.

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

Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

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

Tabla 2: Criterios de evaluación para la métrica TOC.

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

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

Tabla 3: Atributos de calidad evaluados por la métrica RC.

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

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

Tabla 4: Criterios de evaluación para la métrica RC.

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

3.3 Resultados obtenidos de la aplicación de la métrica TOC

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 69% de las clases empleadas en el sistema posee 5 operaciones o menos lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (responsabilidad, complejidad de implementación y reutilización). A continuación se muestran los resultados obtenidos.

Tabla 5: Instrumento de evaluación de la métrica TOC.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
XPDLParserController	13	Alta	Alta	Baja
XPDLParserTrasitionRef	3	Baja	Baja	Alta
XPDLParserTransicionesRestrincion	8	Media	Media	Media
XPDLParserActividades	19	Alta	Alta	Baja
XPDLParserTransiciones	7	Baja	Baja	Alta
XPDLParserNodoAction.php	3	Baja	Baja	Alta
XPDLParserNododiscriminator	3	Baja	Baja	Alta

XPDLParserNodoExclusiveChoise	5	Baja	Baja	Alta
XPDLParserNodoInput.php	5	Baja	Baja	Alta
XPDLParserNodoParaleloSplit	5	Baja	Baja	Alta
XPDLParserNodoSimpleMerge	3	Baja	Baja	Alta
XPDLParserSyncronization	3	Baja	Baja	Alta
XPDLParserVariablesAsociadas	19	Alta	Alta	Baja

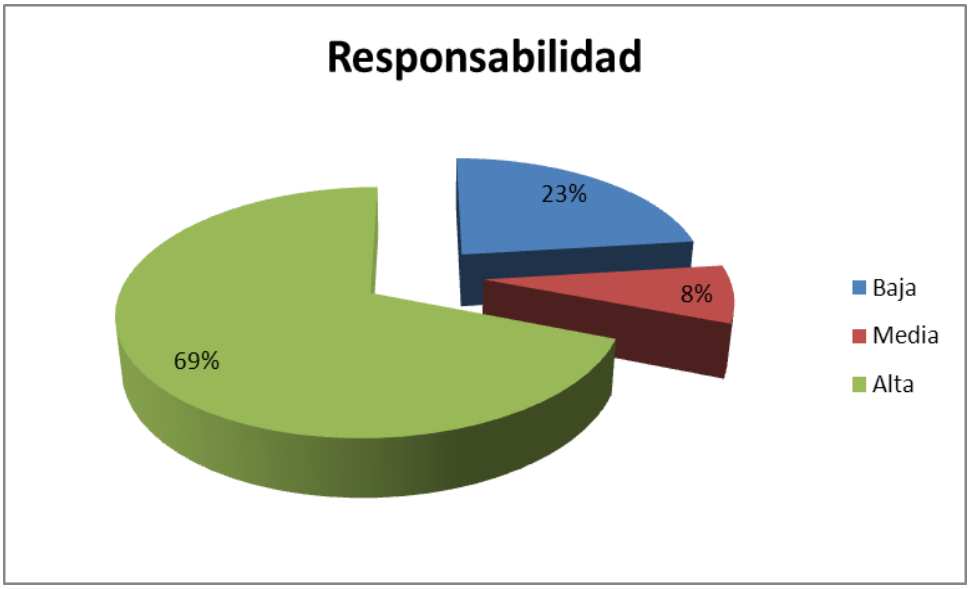


Figura 9: Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad.

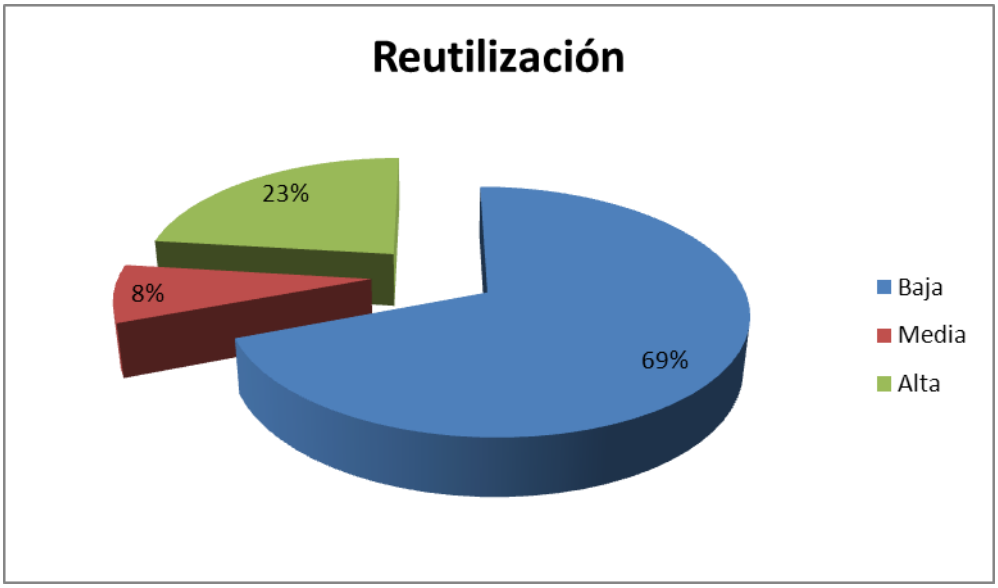


Figura 10: Resultados de la evaluación de la métrica TOC para el atributo Reutilización

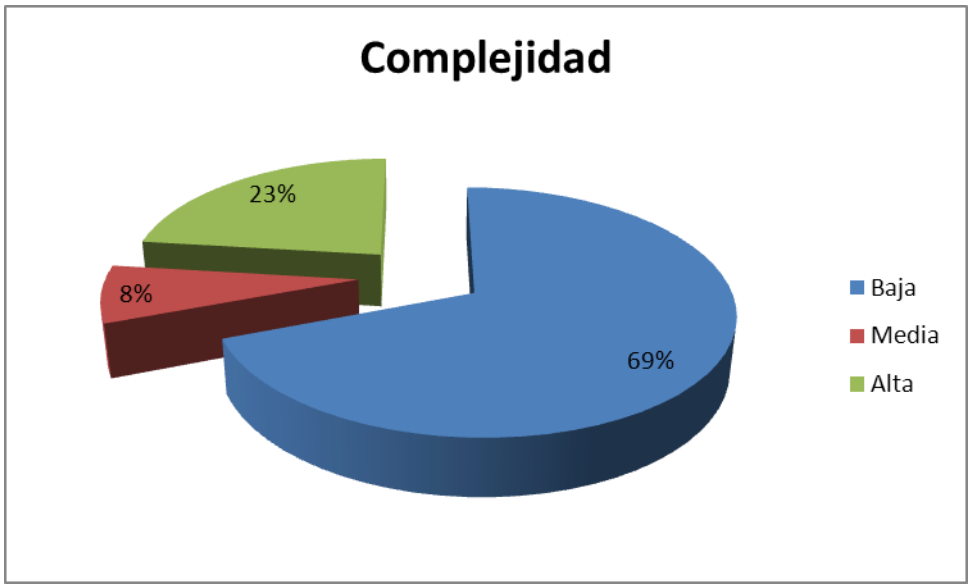


Figura 11: Resultados de la evaluación de la métrica TOC para el atributo Complejidad.

3.4 Resultados obtenidos de la aplicación de la métrica RC

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 92% de las clases empleadas posee menos de 3 dependencias de otras clases lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización). A continuación se muestran los resultados obtenidos.

Tabla 6: Instrumento de evaluación de la métrica RC.

Clase	Cantidad de relaciones de usos	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
XPDLParserController	**10**	Alta	Alta	Baja	Alta
XPDLParserTrasitionRef	**1**	Baja	Baja	Alta	Baja
XPDLParserTransicionesRestrincion	**1**	Baja	Baja	Alta	Baja
XPDLParserActividades	**2**	Media	Media	Media	Media
XPDLParserTransiciones	**1**	Baja	Baja	Alta	Baja
XPDLParserNodoAction.php	**1**	Baja	Baja	Alta	Baja
XPDLParserNododiscriminator	**1**	Baja	Baja	Alta	Baja
XPDLParserNodoExclusiveChoise	**1**	Baja	Baja	Alta	Baja
XPDLParserNodoInput.php	**1**	Baja	Baja	Alta	Baja
XPDLParserNodoParaleloSplit	**1**	Baja	Baja	Alta	Baja

XPDLParserNodoSimpleMerge	**1**	Baja	Baja	Alta	Baja
XPDLParserSyncronization	**1**	Baja	Baja	Alta	Baja
XPDLParserVariablesAsociadas	**1**	Baja	Baja	Alta	Baja

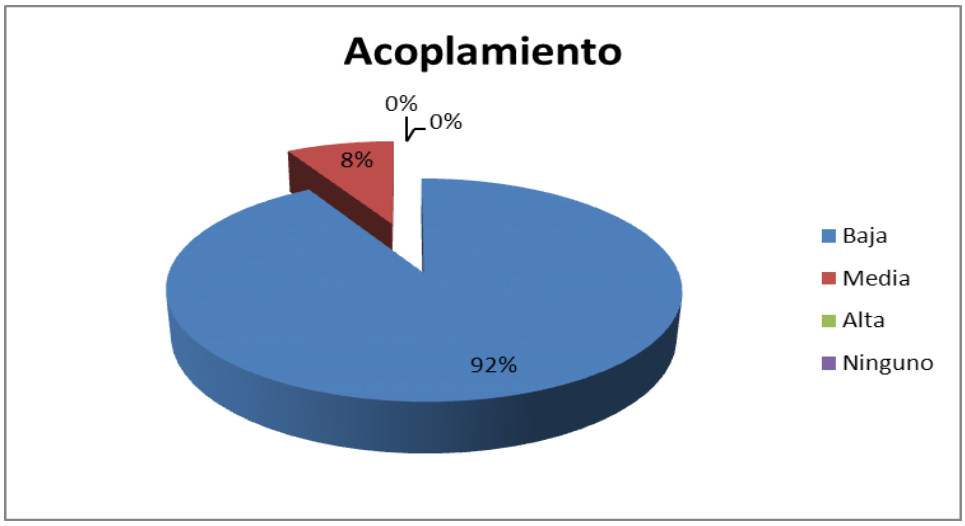


Figura 12: Resultados de la evaluación de la métrica RC para el atributo Acoplamiento.

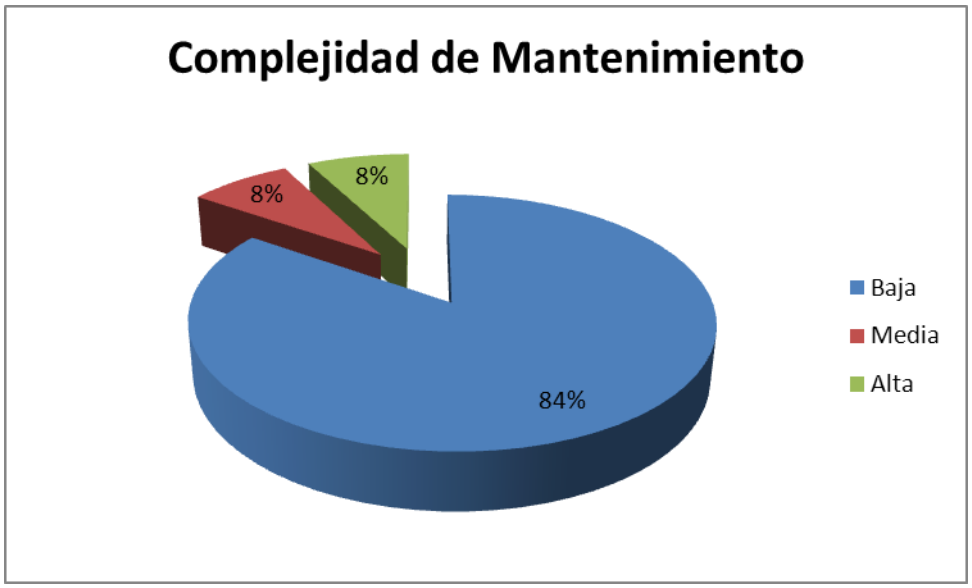


Figura 13: Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento.

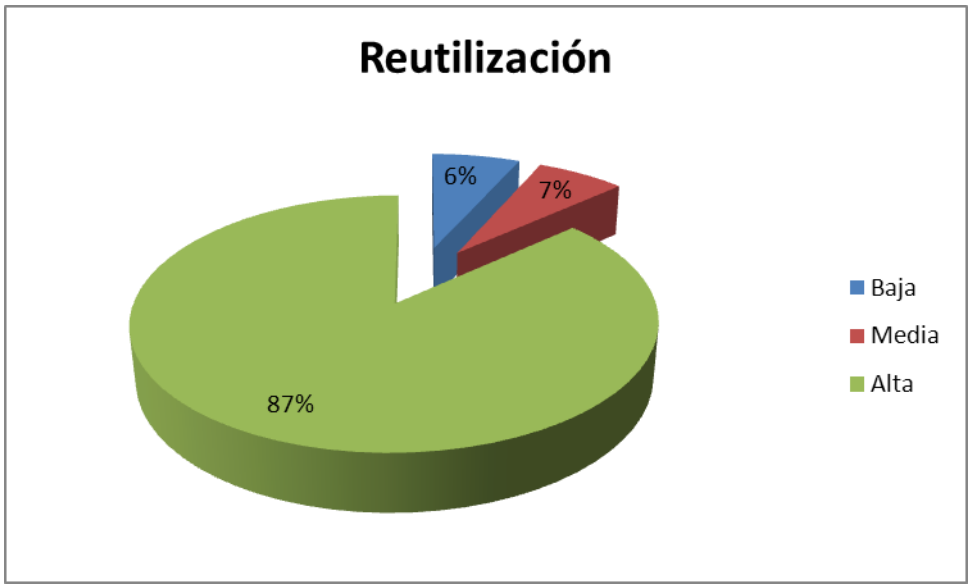


Figura 14: Resultados de la evaluación de la métrica RC para el atributo Reutilización.

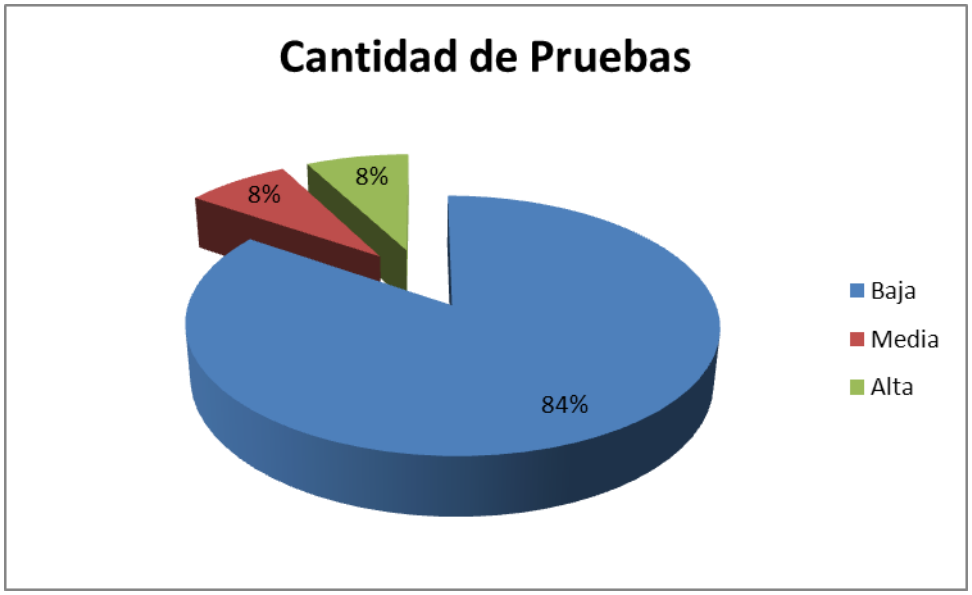


Figura 15: Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas.

3.5 Matriz de inferencia de indicadores de calidad

La matriz inferencia de indicadores de calidad, también llamada matriz de cubrimiento permite conocer si los resultados obtenidos de las relaciones atributo/métrica es positivo o no, llevando estos resultados a una escalabilidad numérica donde, si los resultados son positivos se le asigna el valor de 1, si son negativos toma valor 0 y si no existe relación es considerada como nula y es representada con un guión simple (-). A continuación se muestran los resultados obtenidos.

Tabla 7: Resultados de la Matriz de inferencia de indicadores de calidad.

Atributos\Métricas	TOC	RC	Promedio
Responsabilidad	0	-	0
Complejidad de Implementación	0	-	0
Reutilización	1	1	1
Acoplamiento	-	0	0
Complejidad de Mantenimiento	-	0	0
Cantidad de pruebas	-	0	0

Tabla 8: Rango de valores para la evaluación de la relación atributo/métrica

Categoría	Rango de valores
Malo	<=0.7
Regular	>0.7 y <1.4
Bueno	>=1.4

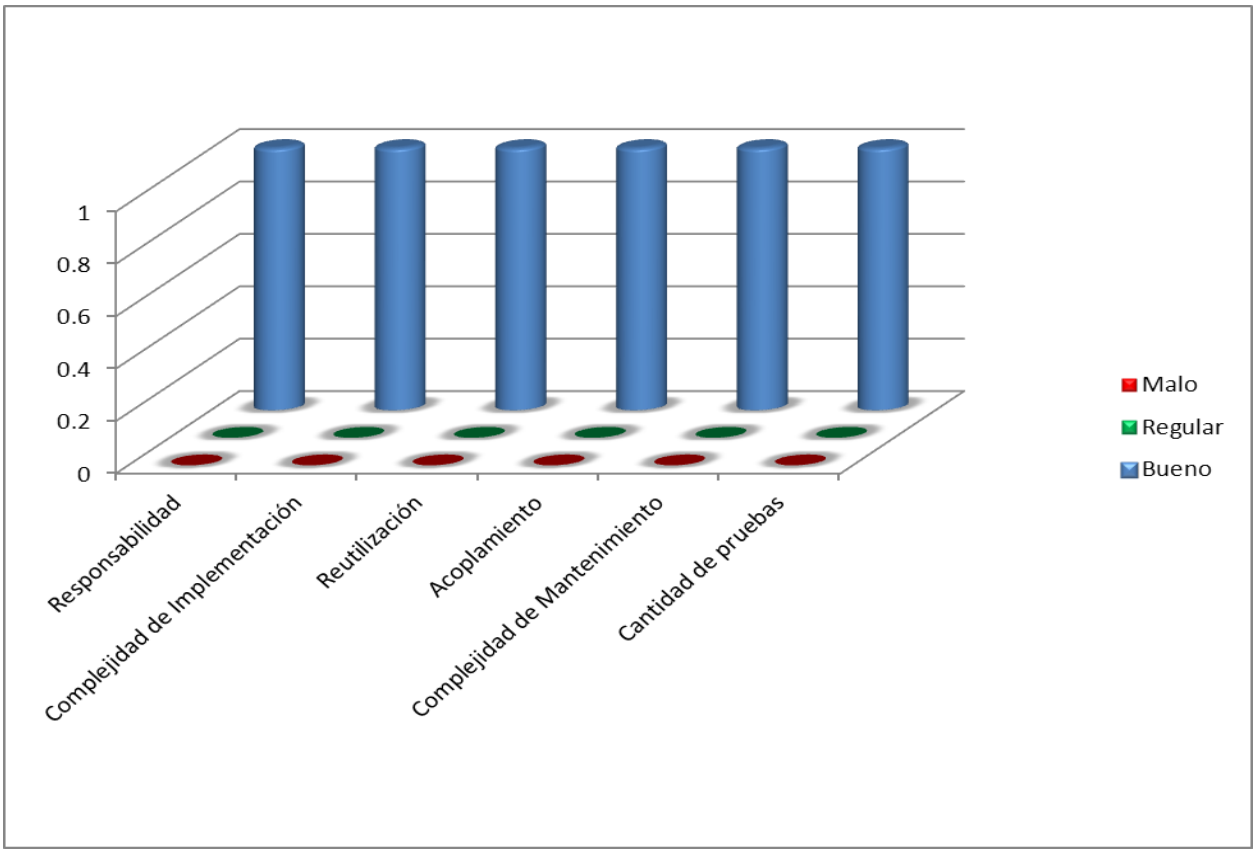


Figura 16: Resultados de la Matriz de inferencia de indicadores de calidad

Se llegan a los siguientes valores en los atributos de calidad mostrados en la Tabla 7 ya que el componente esta formulado por 14 clases, las cuales tienen, para realizar la transformación de un lenguaje de definición a uno de ejecución, una gran cantidad de procedimientos por clase lo que provoca un aumento del promedio de procedimientos por estas, pero no así en cuanto a las relaciones entre clases lo que provoca que los indicadores de acoplamiento, responsabilidad, complejidad de mantenimiento e implementación, cantidad de pruebas y reutilización obtengan valores positivos.

3.6 Pruebas estructurales o de caja blanca

Este método de casos de prueba usa los detalles procedimentales del programa. Se busca obtener casos de prueba que

- Garanticen que se ejecuta por lo menos una vez todos los caminos independientes de cada módulo.

- Verificar las decisiones lógicas (V/F).
- Ejecutar las condiciones en sus límites.
- Ejecutar las estructuras internas de datos para asegurar su validez.

Estas pruebas proponen una manera de diseñar los casos de prueba centrándose en el comportamiento interno y la estructura del programa. De esta manera se examina solo la lógica interna del programa, dejando fuera los aspectos de rendimiento del mismo.

```
//Crea y guarda un workflow de tipo ezComponent en un modelo de datos
public function Workflow()
{
    $work = new ezcWorkflow ('Test');//1
    $this->workflow = $work;//1
    if(count($this->lista_nodos) == 1)//2
     {
      $this->workflow->startNode->addOutNode($this->lista_nodos[0]->get_Nodo());//3
      $$this->lista_nodos[0]->get_Nodo()->addOutNode($this->workflow->endNode);//3
     }
    else
    {
    foreach($this->lista_nodos as $num =>$nodo)//4
    {
        if ($num == 0)//5
        {
            $this->workflow->startNode->addOutNode($nodo->get_nodo());//6
        }
        elseif($num == count($this->lista_nodos)-1)//7
        {
            $nodo->get_nodo()->addInNode($this->lista_nodos[$num-1]->get_nodo());//8
            $nodo->get_nodo()->addOutNode($this->workflow->endNode);//8
        }
        else
        {
          $nodo->get_nodo()->addInNode($this->lista_nodos[$num -1]->get_nodo());//9
        }

    }
    $definition = new ezcWorkflowDatabaseDoctrineDefinitionStorage();//10
    $definition->save($this->workflow);//10
}
```

Figura 17: Prueba de caja blanca

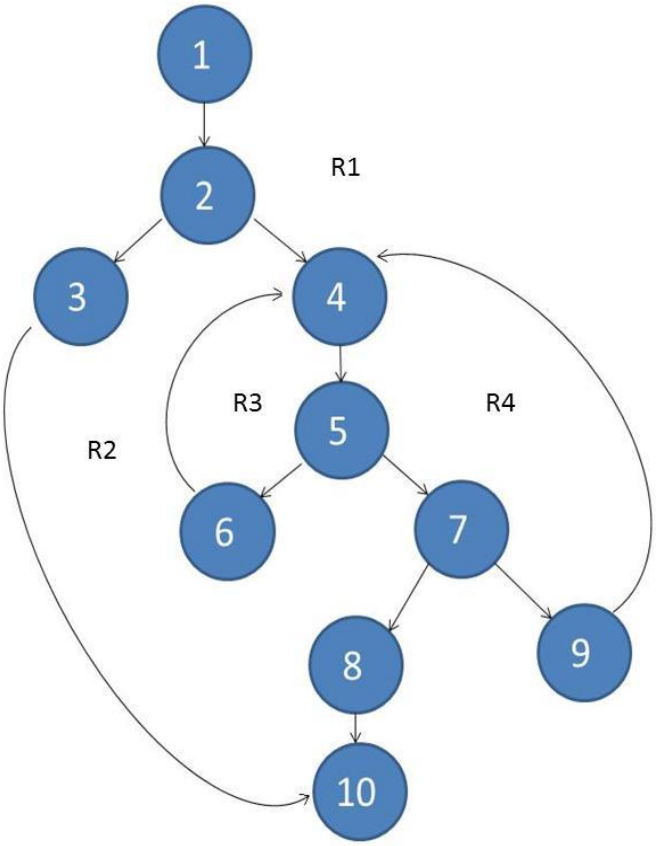


Figura 18: Grafo generado por la prueba de caja blanca.

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) = (12 - 10) + 2

V (G) = 4

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) = 3 + 1

V (G) = 4

2. **V (G) = R**

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del cálculo.
V (G) = 4

El cálculo efectuado anteriormente sobre las fórmulas ha dado el mismo valor, dando como resultado 4, esto indica que existen 4 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**: (1-2-3-10)
- **Camino básico #2**: (1-2-4-5-6-4-5-7-8-10)
- **Camino básico #3** : (4-5-7-8-10)
- **Camino básico #4** : (4-5-7-9-4-5-7-8-10)

Para cada camino se realiza un caso de prueba.

- Caso de prueba para el Camino básico #1

Descripción: controla, crea y guarda el caso básico de un flujo de trabajo ezcWorkflow donde se tiene un solo patrón BPMN transformado en nodo ezComponents en el atributo $lista_nodos.

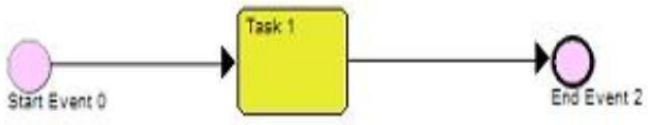


Figura 19: Caso base de un flujo de trabajo.

Condición de ejecución: el arreglo $lista_nodos tiene un solo elemento.

Resultados esperados: un caso base de flujo de trabajo persistido en un modelo de datos.

- Caso de prueba para el Camino básico #2

Descripción: se tienen dos patrones BPMN transformados en nodos ezComponents en el atributo de tipo arreglo $lista_nodos. Se efectúa una primera iteración, se verifica si esta corresponde al primer elemento de la lista y se le acopla como nodo de entrada el nodo startNode, se realiza una segunda iteración donde se controla que el elemento sea el último del arreglo y se le añade como salida el nodo endNode y como entrada el anterior a él. Los nodos startNode y endNode son creados automáticamente una vez que se inicializa un objeto ezcWorkflow.

Condición de ejecución: dos elementos en el arreglo $lista_nodos.

Resultados esperados: un flujo de trabajo ezcWorkflow persistido en un modelo de datos.

- Caso de prueba para el Camino básico #3

Descripción: se tienen más de dos elementos en el atributo $lista_nodos, y se realizan iteraciones en las cuales se identifican primer y último elemento con uno intermedio al cual se le agrega como nodo de entrada el anterior a él.

Condición de ejecución: más de dos elementos en el atributo de tipo arreglo $lista_nodos.

Resultados esperados: un flujo de trabajo ezcWorkflow persistido en un modelo de datos.

- Caso de prueba para el Camino básico #4

Descripción: se crea, controla y guarda un flujo de trabajo ezComponents en donde intervengan más de dos patrones BPMN transformados en objetos ezcWorkflowNode.

Condición de ejecución: más de dos elementos en el arreglo.

Resultados esperados: un flujo de trabajo ezcWorkflow persistido en un modelo de datos.

La realización de las pruebas de caja blanca permitieron validar la solución satisfactoriamente pues evidencian que se ejecuta por lo menos una vez todos los caminos y las decisiones lógicas fueron ejecutadas de manera positiva así como las estructuras internas de datos.

3.7 Pruebas de rendimiento

Las pruebas de rendimiento están diseñadas para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. La prueba de rendimiento se da durante todos los pasos del proceso de la prueba. Incluso al nivel de unidad, se debe asegurar el rendimiento de los módulos individuales a medida que se llevan a cabo las pruebas de caja blanca. Sin embargo, hasta que no estén completamente integrados todos los elementos del sistema no se puede asegurar realmente el rendimiento del sistema. Las pruebas de rendimiento, a menudo, van emparejadas con las pruebas de resistencia y, frecuentemente, requieren instrumentación tanto de software como de hardware. Es decir, muchas veces es necesario medir la utilización de recursos (por ejemplo, ciclos de procesador), de un modo exacto. La instrumentación externa puede monitorizar los intervalos de ejecución, los sucesos ocurridos (por ejemplo, interrupciones) y muestras de los estados de la máquina en un funcionamiento normal. Instrumentando un sistema, el encargado de la prueba puede descubrir situaciones que lleven a degradaciones y posibles fallos del sistema. [45]

Resultados de pruebas de estrés.

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

Las pruebas de estrés tienen como objetivo someter la aplicación al límite de su funcionamiento simulando la ejecución con un número de instancias superior a lo esperado. Esta prueba tiene el propósito de determinar la robustez de una aplicación ante una sobrecarga extrema. Para realizar la prueba de estrés, la aplicación se sometió a la ejecución de 50 usuarios.

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

Con la prueba de Rendimiento realizada al Editor de procesos de negocio se logró medir el tiempo de respuesta del sistema para el caso de 50 usuarios utilizando la aplicación. Para comprobar si el sistema relevó resultados positivos es necesario que el valor no sobrepase los 5 segundos, ya que este es el tiempo de repuesta límite definido en el requisito no funcional de Rendimiento para el marco de trabajo Sauxe.

Después de obtener los resultados generados en JMeter, el Editor de procesos de negocios relevó un tiempo de respuesta de 1,9 segundos., resultado positivo para el componente. Los resultados de las pruebas se muestran en el anexo 1.

3.8 Conclusiones parciales

En este capítulo fueron expuestos los artefactos generados como parte del modelo de implementación de la solución propuesta, tales como el diagrama de componentes y los estándares de codificación. Se realizó una validación del diseño expuesto en el capítulo anterior mediante la aplicación de métricas para la evaluación de atributos de calidad, lo cual permitió valorar el diseño propuesto Además se describieron las pruebas estructurales y funcionales realizadas que permitieron comprobar el correcto funcionamiento del sistema.

CONCLUSIONES

El desarrollo de los objetivos posibilitó arribar a las siguientes conclusiones:

- Se realizó el estudio de diferentes herramientas destinadas a la gestión de procesos de negocios, determinando que ninguna de estas cumplían con las especificaciones de Sauxe ni de su componente para la ejecución de flujo de trabajos.
- Se identificaron las reglas para la transformación de XPDL a un lenguaje de ejecución de procesos permitiendo su posterior implementación utilizando las herramientas propuestas.
- Se validó el diseño mediante las métricas (TOC y RC), arrojando resultados satisfactorios para cada uno de los atributos evaluados.
- El componente fue probado por el grupo de calidad del CEIGE donde se verificó que el sistema cumple con las características de rendimiento esperadas demostrando así la calidad del software.
- Se aplicaron pruebas estructurales al componente, examinándose la lógica interna del programa, para asegurar que se ejecuta cada sentencia al menos una vez.

Siendo así una mejor opción para la ejecución de procesos de negocios, pues cumple con los lineamientos de producción de software de la UCI, logrando la libertad tecnológica del producto.

RECOMENDACIONES

Se considera importante que una vez concluido el proceso de desarrollo se lleven a cabo las siguientes recomendaciones.

- La construcción de una segunda versión donde se incorporen las siguientes funcionalidades.
- Incorporar al componente la implementación de patrones BPMN para lograr un mayor control y generalización de los flujos de trabajo.
- Utilizar la documentación del componente como material de estudio en futuras investigaciones.
- Utilizar el componente de transformación para lograr la ejecución de flujos de trabajo del Sauxe.

REFERENCIAS BIBLIOGRÁFICAS

1. Marlon Dumas, W.M.v.d.A., Arthur H, *Process Aware Information Systems: Bridging People and Software Through Process Technology.* 2005.
2. al., T.e., *Information Technology for Management, Transforming Organizations in the Digital Economy.* 2008.
3. Christopher Koch, D.S., E. Baatz, *The ABCs of ERP.* 1999.
4. Weske, M., *Business Process Management. Concepts, Languages, Architectures.* . 2007.
5. Buschmann F., M.R., Rohnert H. & Sommerlad P. & Stal M, *Pattern-Oriented Software Architecture: A System of Patterns.* 1996.
6. Pedro Solana Gonzalez, M.A.M., Daniel Perez Gonzalez, *Analisis y modelado con redes de workflow del proceso de tratamiento de experiencias operativas.*
7. Aalst, W.M.P.v.d., *Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language.* 2003.
8. Sánchez, P.G., *Introducción a BPEL y OpenESB.* 2010.
9. Díaz, I.J.S.T., *BPEL y Orquestación de Servicios*
10. Wilson, L.B., *Comparative Programming Languages, Second Edition.* 1993.
11. *http://wiki.processmaker.com/index.php/Main_Page [Sitio oficial].*
12. Team, P.D., *ProcessMaker manual.* 2006.
13. José Nelson Pérez Castillo, J.M.C.L., Rubén González Crespo,Oscar Sanjuán Martínez, *OpenSource BPMS.* 2009.
14. *BonitaSoft's Bonita Open Solution 5.2: An Essential Toolkit for BPM.* 2009.
15. *http://www.bonitasoft.com [Sitio oficial].*
16. Antti Seppälä, P.H., Ossi Syd, *Evaluation of Intalio BPM Tool.*
17. *http://www.jboss.org [Sitio oficial].*
18. Harmon, P., *BPM Software Tools Report onBOC's Adonis Version 4.0.* 2010.
19. Zhu, A., *Microsoft Windows Workflow Foundation 4.0 Cookbook.* September 24, 2010,.
20. Richkov, K.a.E.T., *Modeling and Execution of Web Service in Internet using Enhydra workflow platform.* 2006.
21. *http://www.businessprocessincubator.com [Sitio oficial].*
22. Chin, G., *Agile Project Management: How to Succeed in the Face of Changing Project Requirements.* 2004.
23. Brito, I.H.R.G., *ERP cubano, un paso estratégico para la consolidación delSoftware Libre en Cuba.* 2006.
24. Gutiérrez, J.J., *¿Qué es un framework web?*
25. Vaswani, V., *Zend Framework: A Beginner's Guide.* 2010.
26. Ing. Oiner Gómez Baryolo, I.Y.M.B., Ing. Darien García Tejo, *ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE.* 2006.
27. Schlitt, T., *Status of Zeta Components.* 2008.
28. *http://www.doctrine-project.org [Sitio oficial].*
29. Foundation., A.S., *About the Apache HTTP Server Project.* 2008.

30. Pressman, R.S., *Ingeniería de Software, un enfoque práctico. Quinta edición. S.l. : McGraw-Hill Companies.* 2002.
31. http://www.postgresql-es.org/sobre_postgresql., *Portal en español sobre PostgresSQL. [En línea] Citado el: 2 de 12 de 2011.]*
32. Project, p., *About phpPgAdmin.* 2008.
33. *Firefox., M.M.F.*
34. Collins-Sussman, B., W. Fitzpatrick, Brian y Pilato, C. Michael. , *Version Control with Subversion. [En línea] 2002. [Citado el: 17 de 01 de 2011.] http://svnbook.red-bean.com/nightly/es/svn-book.pdf.* 2011.
35. Pressman, R.S., *Software Engineering, a practitioner's approach. (7th edición).* 2007.
36. Systems Popkin, S., *Modelado de Sistemas con UML.* 2008.
37. *www.php.net [Sitio oficial].*
38. Addison-Wesley, *Comparative Programming Languages, Second Edition.* 1993.
39. Pupo, Y.C., *Libro de Ayuda del Marco de Trabajo Sauxe, En su versión 2.0.* 2010.
40. 830, I.S., *IEEE Recommended Practice for Software Requirements Specifications -Description.* 1998
41. Lemus, J.M., *Modelado de datos e implementación de la base de datos.* 2009.
42. Bass L., C.P., Kazman R. , *Software Architecture in Practice: Second Edition. Addison-Wesley.* 2005.
43. Avgeriou, P.U.Z., *Architectural patterns revisited:a pattern language». 10th European Conference on Pattern Languages of Programs (EuroPlop 2005), Irsee, Germany, July. .* 2005.
44. ERP, P., *Normas y Estandares de codificación. Habana.* 2008.
45. Pressman, R.S., *Ingeniería de software. Un enfoque práctico. [quinta edicion].*