Universidad de las Ciencias Informáticas

Facultad 3

Título: Solución informática para el soporte de la transformación de modelos descritos en BPMN a objetos PHP

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Lian Amaurys Rojas Sosa

José Ignacio Saballo López

Tutores: Ing. Yoriangel Rivero González

Ing. Ariel Torres Gálvez

La Habana, junio 2012.

Declaración de autoría

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

Para que así conste firmamos los presentes a los __ días del mes de _____ del año _____

Lian Amaurys Rojas Sosa

_____

José Ignacio Saballo López

_____

Ing. Yoriangel Rivero González

_____

Ing. Ariel Torres Gálvez

_____

RESUMEN

Los procesos de negocios constituyen un esfuerzo para mejorar las operaciones y las funciones de las compañías. Son distinguidos por garantizar el funcionamiento de un negocio y facilitar el cumplimiento de las metas definidas en la estrategia de la empresa.

En aras de lograr la gestión eficiente de los procesos de negocio se desarrolla en el Centro de Informatización para la Gestión de Entidades de la Universidad de las Ciencias Informáticas, un Sistema de Gestión de Flujo de Trabajo (WFMS) para el marco de trabajo Sauxe, entre sus componentes se encuentra el modelador de procesos de negocio utilizando la notación BPMN, creada específicamente para brindar estandarización a todos los usuarios que deseen modelar.

Con el trabajo realizado se desarrolló una solución informática que permite la transformación de modelos descritos en BPMN a objetos PHP, para su posterior persistencia y validación en Sauxe.

La propuesta está respaldada por el estudio de los principales conceptos tratados en la investigación y el análisis de los WFMS. Se describen las herramientas y tecnologías a utilizar, el modelo de desarrollo a seguir durante el ciclo de vida de la solución, los requisitos funcionales y no funcionales, los patrones de diseño empleados y los artefactos generados durante el diseño y la implementación, así como las pautas de codificación, métricas de software y pruebas aplicadas al sistema.

CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las economías actuales y los mercados mundiales con su intensa competencia han impulsado a las empresas a desarrollar y adoptar nuevos modelos de trabajo. La mayoría han reconocido la necesidad de reorganización de los procesos de negocio y la gestión de alertas para aumentar la eficiencia. Los sistemas de información han sido una solución ampliamente adoptada para apoyar el rediseño de los procesos y la gestión de los mismos. En particular, dos sistemas se han distinguido por ofrecer la posibilidad de encontrar soluciones novedosas ante estos desafíos: Los Sistemas de Gestión de Flujo de Trabajo (WFMS por sus siglas en inglés) y los Sistemas de Planificación de Recursos Empresariales (ERP por sus siglas en inglés).

Las aplicaciones Workflow se encargan de la automatización de la secuencia de acciones, actividades o tareas en la ejecución del proceso, permiten realizar un seguimiento de cada etapa del mismo y aportan las herramientas necesarias para su control o gestión del flujo de trabajo. Utilizan la notación de modelado de procesos de negocio o por sus siglas en inglés (BPMN), para crear formularios electrónicos, flujos de trabajo y reportes con mínima programación. BPMN es una notación gráfica que describe la lógica de los pasos de un proceso de negocio. Además proporciona un lenguaje común para que las partes involucradas puedan comunicar los procesos de forma clara, completa y eficiente. (49)

Un ERP es un sistema integral de gestión empresarial que está diseñado para modelar y automatizar la mayoría de los procesos en la empresa. Su objetivo es facilitar la planificación de todos los recursos de la empresa. El software ERP planea y automatiza muchos procesos con la meta de integrar información a lo largo de la empresa y elimina los complejos enlaces entre los sistemas de las diferentes áreas del negocio. (44)

Cuba se esfuerza en alcanzar un desarrollo informático de avanzada, con la misión de lograr la informatización y automatización de sus empresas. Es por ello que surge la Universidad de las Ciencias Informáticas (UCI), entidad impulsora dentro del ámbito informático evolucionando la industria del software cubano. Debido a la necesidad de incrementar la eficiencia económica en cuanto a la planificación y el control de los recursos empresariales, se decidió crear un producto cubano, que se adapte a las particularidades de la economía, desarrollado en el Centro para la Informatización de la Gestión de Entidades (CEIGE); en el mismo se desarrolla, entre otros productos, un marco de trabajo para el desarrollo de aplicaciones Web de gestión denominado Sauxe, que soporta el desarrollo de los sistemas y responde a la independencia tecnológica por la que aboga el país.

Actualmente en Sauxe existe un componente que permite modelar procesos de negocio de manera visual, quedando por debajo sólo los Gráficos Vectoriales Escalables (SVG, por sus siglas en inglés) y no objetos reales que contengan la información; lo que provoca la pérdida de las referencias de los elementos modelados.

Por otra parte existe un modelo de datos que representa un objeto estructurado, sin embargo se eliminan las referencias y por ende toda la información del mismo. Además, no todos los elementos definidos en la paleta de componentes son transformados a objetos especificados por la Notación de Modelado de Procesos de Negocio, ni existe una forma de adicionar o establecer valores reales a los objetos. Toda la información se maneja del lado del cliente, ocasionando la pérdida de la información al recargar el navegador. La visualización de los procesos reduce errores, asegurando que estos se comporten siempre de la misma manera. Sin el elemento visual se dificulta la posibilidad de evaluar los procesos durante cada una de las etapas de análisis, modelación, implementación, validación, y ejecución de los procesos de negocio. Por cuanto se define como **problema a resolver:**

¿Cómo transformar el modelado de procesos de negocio de manera tal que se posibilite su posterior persistencia y validación en el marco de trabajo Sauxe?

Definiéndose como **objeto de estudio:** Modelado de procesos de negocio en BPMN. Para darle solución al problema planteado, se define como **objetivo general:** Desarrollar una solución informática que permita la transformación de modelos descritos en BPMN a objetos PHP, para su posterior persistencia y validación en el marco de trabajo Sauxe.

Para lograr el objetivo general se trazan los siguientes **objetivos específicos:**

- ✓ Realizar el marco teórico sobre las herramientas de descripción de procesos de negocios basadas en BPMN para identificar debilidades y puntos de reutilización
- ✓ Realizar el análisis y diseño de la solución
- ✓ Implementar la solución
- ✓ Validar la solución a partir de métricas y pruebas de software

Teniendo así como **campo de acción:** Herramientas informáticas para el modelado de procesos de negocio en BPMN.

También se definieron **tareas investigativas** para darle solución a la problemática en cuestión:

- ✓ Análisis de las herramientas de descripción de procesos de negocios basadas en BPMN para identificar debilidades y puntos de reutilización
- ✓ Obtención de los requisitos funcionales
- ✓ Elaboración del diseño de la solución
- ✓ Creación del artefacto Pool

- ✓ Creación del artefacto Lane
- ✓ Definición de roles por Pool
- ✓ Definición de usuarios por actividades
- ✓ Definición de estructuras por Pool o Lane
- ✓ Creación de procesos anidados
- ✓ Referencia entre procesos
- ✓ Relación entre las actividades automáticas y los servicios
- ✓ Definición de la estructura para la descripción de los servicios
- ✓ Relación entre las actividades de usuario y las acciones del sistema
- ✓ Creación de paquetes de procesos
- ✓ Creación de procesos por versiones
- ✓ Activación y desactivación de procesos
- ✓ Generación de forma dinámica del Split Join y el Join Split
- ✓ Diseño e implementación del modelo de clases para cada uno de los tipos de artefactos del modelo BPMN
- ✓ Transformación de un arreglo JSON a objetos PHP
- ✓ Transformación de objetos PHP a arreglos JSON
- ✓ Visualización de modelos existentes y permitir su modificación

Es por ello que se define como **Idea a defender** en la presente investigación:

El desarrollo de una solución informática que permita la transformación de modelos descritos en BPMN a objetos PHP, facilitará su posterior persistencia y validación en el marco de trabajo Sauxe.

El trabajo de diploma quedará estructurado en tres capítulos, que se definen a continuación:

Capítulo 1: Fundamentación teórica

El capítulo contiene la fundamentación teórica de la investigación, se describen los principales conceptos relacionados con el tema. Se incluye un estudio de software, tecnologías y herramientas involucradas en el desarrollo de la solución, analizando sus características, ventajas y desventajas con el propósito de obtener las bases para proponer la mejor solución al problema que se plantea.

Capítulo 2: Propuesta de solución

En este capítulo se analizan los temas de análisis y diseño de la solución, estudiándose las características del sistema para su posterior implementación. Se examinarán cada uno de los

requisitos funcionales, no funcionales, prototipos de interfaces, modelos, y diagramas que servirán de entrada en la etapa de desarrollo.

Capítulo 3: Implementación y prueba

Este último capítulo realiza una valoración de la solución. Expresa cómo se lleva a cabo la implementación del sistema y en aras de garantizar la calidad de la solución se validará el diseño mediante métricas y la implementación mediante pruebas de software.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo, se presenta el marco teórico de la investigación donde se definen los principales conceptos relacionados con el tema, así como un estudio de las tendencias actuales. Además se caracterizan las herramientas y tecnologías involucradas en el desarrollo de la solución. También se estudian algunos estándares de impacto internacional en el intercambio de modelos de procesos de negocio.

1.1 Bases conceptuales

Proceso

Según informa el diccionario de la **Real Academia Española (RAE)**, este concepto describe la acción de avanzar o ir para adelante, al paso del tiempo, al conjunto de etapas sucesivas advertidas en un fenómeno natural o necesarias para concretar una operación artificial y, desde la perspectiva del derecho, a la añadidura y valoración de documentación escrita en toda causa civil o criminal.

La Organización Internacional de Estandarización (ISO) lo define como el conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados con un valor añadido. (1)

Se denomina proceso al conjunto de acciones o actividades sistematizadas que se realizan o tienen lugar con un fin. Si bien es un término que tiende a remitir a escenarios científicos, técnicos y/o sociales planificados o que forman parte de un esquema determinado, también puede tener relación con situaciones que tienen lugar de forma más o menos natural o espontánea. (56)

En la informática, un proceso puede referirse a distintas combinaciones operativas que ocurren simultáneamente para alcanzar un resultado o un producto, como la instalación de un nuevo software, o la consecución de un análisis antivirus. (56)

Proceso de negocio

“Un conjunto estructurado, medible de actividades diseñadas para producir un producto especificado, para un cliente o mercado específico. Implica un fuerte énfasis en CÓMO se ejecuta el trabajo dentro de la organización, en contraste con el énfasis en el QUÉ, característico de la focalización en el producto”. (2)

Según Henry J. Johansson presidente del Consejo de. Normas de la industria Manufacturera un proceso de negocio es un conjunto de actividades relacionadas que permiten crear un producto o servicio final a través de la transformación de uno o varios productos o servicios iniciales. El desarrollo del proceso es el que debe aportar valor a las entradas iniciales. (3)

También Roger Burlton fundador del Grupo de Procesos de Renovación BPM, define un proceso de negocio como todas las actividades que deben realizarse para satisfacer las necesidades de los usuarios de una organización. Añadiendo, que un proceso de negocio estará correctamente ejecutado si durante el proceso se hace entrega de un determinado producto o servicio, o dicho proceso desencadena otro proceso. (4)

Modelado de procesos de negocio

El modelado de procesos de negocio es la base para comprender mejor la operación de una organización, documentar y publicar los procesos buscando una estandarización en la organización, buscar eficiencias en la operación e integrar soluciones en arquitecturas orientadas a servicios. (6)

Modelado de Procesos de Negocio (BPM) en ingeniería de sistemas, es la actividad de representación de los procesos de una empresa, de modo que el proceso actual puede ser analizado y mejorado. BPM se suele llevar a cabo por los analistas y gerentes de empresas que buscan mejorar la eficiencia de los procesos y la calidad. Las mejoras en los procesos identificados por BPM pueden requerir la participación de las Tecnologías de la Información. (54)

Programas de gestión de cambio se suelen utilizar para poner los procesos de negocio mejorados en práctica. Con los avances en la tecnología de los grandes proveedores de plataformas, la visión de convertirse en modelos BPM totalmente ejecutable (y capaz de simulaciones y de ingeniería de ida y vuelta) se acerca a la realidad todos los días. (5)

El modelado de procesos de negocio, es adoptado por diferentes empresas, con 3 objetivos fundamentales:

- ✓ Documentar: los procesos son parte fundamental de la organización de una empresa y un elemento primordial cuando se intenta implementar modelos de calidad como ISO
- ✓ Mejorar: las empresas que buscan una mayor eficiencia en sus procesos, localizar “cuellos de botella” en su gestión, identificar área de oportunidad o mejora, recurren al modelado y la simulación de procesos
- ✓ Agilizar: en un nivel de mayor sofisticación, las empresas requieren el modelado de procesos como articuladores de los servicios de Tecnologías de información, para poder

reaccionar con mayor agilidad a los constantes cambios que exige la competencia actual (6)

Flujo de trabajo

El término "**Workflow**", hace referencia a la gestión modelada y computarizada de todas las tareas que deben llevarse a cabo y de los distintos protagonistas involucrados en realizar el proceso de negocios. También puede traducirse como la gestión electrónica de procesos de negocio. (7)

Un Workflow o flujo de trabajo es el estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información (en muchos casos, documentos a cumplimentar para seguir las directrices de una normativa de calidad como la ISO: 9001 u otras normas estándares) que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas. Generalmente los problemas de flujo de trabajo se modelan con redes de Petri. (8)

Sistemas de gestión de flujos de trabajo (WFMS)

Es un sistema que define, crea y gestiona la ejecución de flujos de trabajo a través del uso del software, siendo capaz de interpretar la definición del proceso, interactuar con los participantes y, siempre que se quiera, invocar el uso de herramientas y aplicaciones. (9)

La ejecución del flujo de trabajo en un Sistema de gestión de flujo de trabajo consiste en la creación, manipulación y destrucción de instancias proceso, que representan al flujo de trabajo de acuerdo con la especificación previa. Cada instancia proceso tendrá una identidad visible externamente y un estado interno que representa su progreso hacia la finalización y su estado con respecto a las actividades que lo componen. Cada vez que la ejecución del proceso suponga la invocación de una actividad, según la definición del mismo, se crea una instancia actividad que la representa. Dicha instancia se encarga de ejecutar las acciones asignadas, accediendo a los datos que sean necesarios e invocando la aplicación externa correspondiente, si así lo requiere la actividad. (10)

1.2 Tendencias actuales

La selección de las herramientas analizadas en este epígrafe está basada en aquellas que la WfMC reconoce por implementar los estándares que establece. A continuación se describen algunas de las más representativas:

1.2.1 Bizagi

Bizagi es la solución líder de Business Process Management (BPMS) para una automatización de procesos más rápida y flexible. Poderosa y sencilla Suite de BPM, diseñada para resolver problemas de negocio reales. (11)

Bizagi es una suite ofimática con dos productos complementarios, un Modelador de Procesos (Bizagi Process Modeler) y una Suite de BPM (Bizagi BPM Suite).

Bizagi Process Modeler es utilizado para diagramar y documentar procesos usando la notación estándar BPMN.

Bizagi BPM Suite es una solución de Gestión de procesos de negocio (BPM) que le permite a las organizaciones ejecutar/automatizar procesos o flujos de trabajo. Bizagi BPM Suite se integra con aplicaciones como SAP, Documentum, SharePoint y correo electrónico.

- ✓ Es un modelador de procesos que permite representar de forma esquemática todas las actividades y decisiones que se toman en el negocio
- ✓ Con una interfaz que recuerda a Microsoft Office, Bizagi Process Modeler cumple con el estándar BPMN (Business Process Management Notation)
- ✓ El Modelador de Procesos BPMN Bizagi, permite compartir las ideas de mejoramiento con los otros miembros de tu equipo
- ✓ Presenta los procesos en un formato estándar de aceptación mundial, que ha sido conocido como BPMN: Business Process Modeling Notation
- ✓ Permite realizar diagramas y documentar los procesos de la manera más eficiente y buscando fomentar la colaboración en la organización (12)

1.2.2 WorkflowGen

Permite el enfoque de la función de flujo de trabajo, su diseño robusto facilita la rápida implementación de nuevas funciones, y aprovecha las tecnologías Web básicas (NET, XML, RSS, Web Services), el grado de flexibilidad de desarrollo por parte del usuario es alto, el rendimiento económico también es bueno. La herramienta es fácil de usar, el diseño y la configuración de WorkflowGen son intuitivos y comprensibles. La interfaz de usuario se refina, y la facilidad de uso es alta. WorkflowGen facilita la personalización de todos los aspectos de los flujos de trabajo (rutas, notificaciones por correo electrónico). (13)

- ✓ Hace uso del lenguaje XPDL para almacenar la definición de sus procesos
- ✓ Posee un diseñador gráfico
- ✓ Mantiene la información de un proceso a través de ventanas de diálogo y/o formularios

- ✓ Posee un sistema de seguridad en base a perfiles, lo que permite tener un mejor control sobre el nivel de acceso de los usuarios que acceden a la herramienta
- ✓ Permite la administración de versiones de los procesos definidos

1.2.3 ObjectWeb Bonita

Bonita es un sistema flexible de flujo de trabajo de cooperación, conforme a la especificación WFMC, basada en el modelo de flujo de trabajo, incorpora la previsión de actividades como un mecanismo más flexible de la ejecución del flujo de trabajo. Es de código abierto y se puede descargar en la licencia LGPL.

- ✓ Hace uso del lenguaje XPDL para almacenar la definición de sus procesos
- ✓ Permite definir el flujo de un proceso de forma gráfica, sin embargo las herramientas de diagramado que utiliza son limitadas. La herramienta utiliza BPMN como notación gráfica
- ✓ Mantiene la información de un proceso a través de ventanas de diálogo y/o formularios
- ✓ Posee un sistema de seguridad basado en roles, los cuales poseen diferentes niveles de acceso que permiten desde la visualización de un proceso, hasta la aprobación o rechazo del mismo (14)

Permite importar la definición de un proceso a partir de un archivo XML que cumpla con el estándar definido por el lenguaje XPDL. (XML Process Definition Language) es un lenguaje para la definición de un Flujo de trabajo. Fue creado por WFMC en el año 2001 y el 3 de octubre del 2005 se liberó la versión 2.0)

1.2.4 Business Process Visual ARCHITECT (BP-VA)

Es la herramienta de modelado de procesos que permite visualizar, entender, analizar, mejorar y documentar los procesos de negocio, el flujo de documentos y la información dentro de la organización. BP-VA soporta el modelado de procesos de negocio con la notación BPMN (Business Process Modeling Notation), el modelado del Flujo de Datos o Documentos con la notación DFD tradicional (Data Flow Diagram), así como el Modelado de Datos mediante los requisitos de ERD (Entity Relationship Diagram). Esta herramienta reduce significativamente el esfuerzo a realizar en el modelado mientras la documentación ordenada muestra en qué forma se ejecutan los procesos dentro de su empresa. (59)

- ✓ Permite definir el procedimiento de trabajo de procesos empresariales
- ✓ Genera XML Process Definition Language (XPDL) de forma automática a partir del diagrama de procesos de negocio BPMN
- ✓ Genera procesos jPBMN (lenguaje de definición JPDL) de forma automática a partir del diagrama de procesos de negocio BPMN

- ✓ Permite especificar las propiedades del proyecto de gestión, a los objetos de flujo de procesos de negocio (59)

BP-VA se integra con otras herramientas que cumplen con los estándares BPEL como Oracle Workflow o JBoss Workflow, además permite la importación y exportación usando formatos XML. (60)

1.2.5 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 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[1]), 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 (ambiente gráfico para el desarrollo en 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. (57)

[1] Del inglés Mozilla Public License.

Luego de un estudio exhaustivo de las herramientas presentadas, se obtuvieron claras conclusiones, todas poseen las especificaciones para lograr el modelado de procesos y generan un estándar XPDL definido por el Grupo de Administración de Objetos (OMG por sus siglas en inglés) y adoptado por la WfMC. Algunas de ellas presentan el inconveniente del pago de licencias pues son sistemas privativos y la implantación de una herramienta de flujo de trabajo debe estar, basada en los principios de independencia tecnológica por los que aboga el país. Otras son implementadas en lenguajes de programación como Java y .NET y el marco de trabajo está desarrollado en PHP, es por ello que se hace necesario el desarrollo de una Solución informática para el soporte de la transformación entre modelos descritos en BPMN a objetos PHP.

1.3 Modelo de Desarrollo orientado a componentes del proyecto ERP -CUBA

El modelo de desarrollo a utilizar durante todo el ciclo de vida del producto que se propone fue elaborado por el equipo de producción del centro CEIGE valorando los riesgos y características propias del mismo. Está orientado a la reutilización de componentes parametrizables, donde se simplifican las actividades y se eleva el nivel de especialización de los implicados. (49)

Basado en los principios de buenas prácticas y elementos relevantes de las metodologías RUP, XP y SCRUM, es un modelo basado en componentes, iterativo e incremental y utiliza la técnica de prototipado con el objetivo de desarrollar una definición mejorada de los requisitos del usuario para el sistema. (49)

1.3.1 Características

Centrado en la arquitectura

La arquitectura determina la línea base, los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades del desarrollo y resuelve las necesidades tecnológicas y de soporte para el desarrollo. (61)

Basado en componentes

Las iteraciones son orientadas por el nivel de significancia arquitectónicas de los componentes, los mismos son abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan, el componente es la unidad de medición y ordenamiento de las iteraciones. (61)

Iterativo e incremental

Las iteraciones son planificadas y coordinadas con el equipo de arquitectura, los clientes y la alta gerencia. Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo de esta manera la evolución incremental del producto. (61)

Ágil y adaptable al cambio

El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados. (61)

1.3.2 Artefactos y roles

El modelo de desarrollo utilizado define roles y artefactos, la propuesta de solución evidencia el uso de ellos, los cuales se muestran a continuación: (49)

Roles	Artefactos asociados a los roles
Arquitecto de Sistema	Diagrama de componentes Prioridad de los componentes
Analista	Modelo Conceptual Prototipo de IU Especificación de requisitos Casos de Prueba
Especialista de Calidad	Registro de No Conformidades
Desarrollador	Implementación de componentes Descripción de los componentes
Diseñador de Lógica del Negocio	Diagrama de Clases Descripción del Diseño de Clase

1.4 Herramientas y tecnologías empleadas

La presente investigación forma parte de un proceso productivo guiado por el CEIGE, en el cual se tomaron decisiones tecnológicas partiendo de la utilización de tecnologías de código abierto para el desarrollo de sus productos. A continuación se detallan brevemente las que se utilizaron.

Apache 2.0

Apache es el servidor Web que se ejecuta en gran cantidad de sistemas operativos, lo que lo hace prácticamente universal. Es una tecnología gratuita, código abierto y altamente configurable de diseño modular por lo que resulta muy sencillo ampliar sus capacidades. Actualmente existen muchos módulos para Apache que son adaptables al mismo y están disponibles para su instalación cuando sean necesarios. Permite personalizar la respuesta ante los posibles errores

que se puedan dar en el servidor y es posible configurarlo para que ejecute un determinado script cuando esto suceda. (22)

Servidor Web seleccionado para el desarrollo de la solución, con el fin de previsualizar y probar el código durante la implementación.

NetBeans 7.1

NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extenderlo. El entorno de desarrollo integrado (IDE) NetBeans es un producto libre y gratuito sin restricciones de uso. (23)

Herramienta de desarrollo utilizada para la implementación de la solución en los lenguajes PHP y JavaScript.

Visual Paradigm 8.0

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones con calidad, y a un menor coste. 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. Soporta UML versión 2.1, permite modelado colaborativo con CVS y Subversion, generación de código, ingeniería inversa, generación de bases de datos (transformación de diagramas entidad-relación en tablas de la base de datos), importación y exportación a ficheros XML, distribución automática de diagramas, entre otras características. (24)

Herramienta para modelar los diagramas UML generados (modelo conceptual, diagrama de clases, diagrama de componente y diagrama de despliegue), durante el proceso de desarrollo de la solución propuesta.

Control de versiones RapidSVN 2.7

Es un cliente gratuito de código abierto para el sistema de control de versiones RapidSVN, maneja ficheros y directorios que se almacenan en un repositorio el cual recuerda todos los cambios que se hayan escrito en él, permitiendo obtener versiones antiguas de documentos, saber cuándo y cómo fue la última vez que se realizó el cambio y por quién fue realizado, entre las características se encuentran su integración con el Shell de Windows: esto permite continuar trabajando con dichas herramientas sin necesidad de instalar aplicaciones nuevas cuando se necesiten las funciones del control de versiones. (25)

También facilita el acceso a los comandos de Subversion, versionado de carpetas, confirmaciones atómicas, metadatos versionados, lección de capas de red, manejo de datos consistente, etiquetado y creación de ramas eficiente y extensibilidad. (25)

Es la aplicación seleccionada para la comunicación entre los desarrolladores a través del repositorio del proyecto. Así como para manejar el control de versiones en el proceso de desarrollo y permitir el acceso a la versión más actualizada.

Navegador Mozilla Firefox 12

Mozilla Firefox es un navegador de código abierto, multiplataforma, basado en el código base de Mozilla que proporciona una navegación rápida, segura, y más eficiente que otros navegadores. Entre sus principales características se encuentran:

- ✓ Velocidad: las páginas se abren en menor tiempo y se navega con comodidad en ellas
- ✓ Seguridad: es software de código abierto, permite que las fallas de seguridad se corrijan al instante. Al mismo tiempo Mozilla Firefox usa su propio motor de dibujado de páginas Gecko, lo que lo hace inmune a las fallas de seguridad del Internet Explorer, que también tienen todos los navegadores basados en él
- ✓ Desde sus inicios Mozilla Firefox procuró evitar las molestas ventanas emergentes de publicidad, de un modo sencillo y eficaz
- ✓ Navegación por pestañas: permitiendo visualizar varias páginas a la vez, en una misma ventana, logrando una navegación mucho más cómoda
- ✓ Multiperfil: se pueden usar varios perfiles de usuario, con configuraciones diferentes para cada uno
- ✓ Posee numerosas extensiones y temas (apariencia), lo que posibilita la configuración de acuerdo a las necesidades (26)

En el desarrollo de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP se utiliza un complemento del Mozilla Firefox denominado Firebug, que permite analizar las peticiones AJAX (Asynchronous JavaScript And XML) que se realizan al servidor apreciando datos relevantes de la petición al servidor así como de la respuesta que brinda el mismo. Posibilita no solo analizar las peticiones AJAX sino también cualquier petición que se realice al servidor ya sea una imagen, un archivo con extensión css (Cascading Style Sheets) o js (JavaScript), además permite explorar las respuestas JSON que brinde el servidor. Esta herramienta también permite depurar el código JavaScript acelerando la productividad. Permite inspeccionar cualquier elemento HTML de la página y acceder a sus estilos, los cuales se pueden modificar y ver los cambios directamente sin la necesidad de refrescar la página. Este complemento incorpora una consola en la cual se puede escribir código JavaScript que se

ejecuta directamente, permitiendo así obtener el valor de variables, explorar objetos del DOM, agregar eventos a determinados elementos HTML (HyperText Markup Language). Esta consola conjuntamente con la herramienta de depuración constituye un factor importante en la resolución de determinados errores de ejecución o cualquier otro comportamiento erróneo que se presente en el cliente.

1.4.1 Lenguajes de programación

PHP 5.3

Preprocesador de Hipertexto PHP (por sus siglas en inglés) es un lenguaje de programación de alto nivel orientado al desarrollo de aplicaciones Web, que es interpretado del lado del servidor. Sus sintaxis son muy similares a lenguajes como C y PERL. Puede ser utilizado en casi todos los sistemas operativos existentes, permitiendo migrar las aplicaciones de un sistema a otro sin necesidad de realizar cambios en el código. Su rapidez en la ejecución y los bajos requerimientos de consumo en los sistemas donde es desplegado lo hacen uno de los preferidos por los desarrolladores. Se integra perfectamente a la mayoría de los Sistemas Gestores de Bases de Datos. Su mayor ventaja radica en ser un lenguaje libre, por lo que se convierte en una alternativa de muy fácil acceso, además de poseer una comunidad de desarrolladores que intercambian experiencias, de esta forma cuando se presenta un problema, es muy fácil obtener documentación para darle solución de forma rápida y sin costo alguno. Quizás una de sus mayores desventajas radica en que promueve la creación de código desordenado, por lo que lo hace muy complejo de mantener. (18)

En la solución propuesta se utiliza el lenguaje de programación PHP 5.3 para codificar los mecanismos de respuesta del servidor; a las peticiones del cliente a través de acciones. Permite la especificación de la identidad, el estado y el comportamiento, encapsulados en objetos PHP de los elementos que componen BPMN; que se modelan en la interfaz. Así como la lógica, capaz de transformar en objetos PHP el JSON que viaja del cliente al servidor.

JavaScript 3.0

JavaScript es un lenguaje de scripting basado en objetos, que se utiliza principalmente para crear páginas Web dinámicas y permite el desarrollo de interfaces de usuario mejoradas. Una página Web dinámica es aquella que permite la interacción entre el contenido de la misma y el usuario. JavaScript permite incorporar a dichas páginas efectos, ejemplo: texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con

JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. A pesar de su nombre, este no guarda relación directa con el lenguaje Java, sino que simplemente la compañía dueña del mismo lo adoptó por una cuestión de mercado. (19)

Para construir una estructura de objetos del lado del cliente, asociada a cada uno de los elementos modelados en el área de trabajo, en la solución propuesta se utiliza JavaScript 3.0. Dicha estructura está fundamentada en la definición dada a cada elemento en BPMN.

1.4.2 Librerías y marcos de trabajo

Sauxe 2.0

Se implementa con una arquitectura en capas, que a su vez presenta en su capa superior un MVC. Está basado en Zend framework, utiliza en la capa de acceso a datos el Lenguaje de Consulta de Datos (DQL) que implementa Doctrine. Utiliza ExtJS en la capa de presentación, por la gran gama de componentes que se pueden reutilizar y para mostrarle al usuario una interfaz más amigable. (53)

Marco de trabajo para el cual se propone la solución. Anteriormente poseía un modelador muy primitivo (*ver introducción*), utilizado como punto de partida para agregarle las nuevas funcionalidades definidas. El componente propuesto consume un conjunto de librerías y servicios que brinda el marco de trabajo.

Zend Framework 1.4

Zend Framework es un framework MVC (Modelo-Vista-Controlador) código abierto para el desarrollo de aplicaciones Web con PHP. Brinda soluciones para construir sitios Web modernos, robustos y seguros. Posee un bajo acoplamiento entre sus componentes, lo que posibilita la utilización de los mismos a conveniencia, aunque todos estos en conjunto conforman un potente y extensible framework para aplicaciones Web. Brinda una alta abstracción de bases de datos, haciendo extremadamente simple la interacción con estas, sin necesidad de escribir ninguna consulta SQL. Cuenta con módulos para el manejo de ficheros PDF, canales RSS, entre otros. Posee clientes para el acceso a servicios Web y robustas clases para la autenticación y el filtrado de entrada; completa documentación y test de alta calidad. (20)

ExtJS 3.3.1

Es una librería JavaScript, código abierto, de alto rendimiento para la creación y desarrollo de aplicaciones Web dinámicas. Provee interfaces gráficas de usuario que brindan experiencias parecidas o iguales a las que se tienen con aplicaciones de escritorio. Permite la creación de

aplicaciones complejas utilizando componentes predefinidos. Es extensible para la gran mayoría de los navegadores, evitando el problema de validar el código para cada uno. Entre sus principales ventajas se encuentra el balance entre Cliente-Servidor, distribuyendo la carga de procesamiento en el último, y este al tener menor carga, maneja los clientes de manera más eficiente. La comunicación asíncrona permite el intercambio de información con el servidor sin necesidad de pedirle una acción al usuario, dando la libertad de cargar la información sin que este lo note. (19)

Es importante destacar que Sauxe donde se desea implantar la solución, utiliza la versión 2.2 de esta librería para inicializar su portal y el modelador se comenzó a desarrollar con la versión 3.3.1. La solución que se propone utiliza el anterior modelador como base, añadiéndole nuevas interfaces y la lógica a todas las partes visuales, tanto las construidas durante el proceso de esta investigación como las creadas anteriormente, es por ello que se desarrolla sobre la versión 3.3.1.

1.5 Conclusiones parciales

Al terminar este capítulo se arriba a las siguientes conclusiones

- ✓ Las herramientas de flujo de trabajo estudiadas en este epígrafe se descartan ya que no se adaptan a la arquitectura del marco de trabajo Sauxe. Además no son implementadas sobre el mismo lenguaje de programación que utiliza el marco de trabajo
- ✓ El modelador que existe actualmente pierde las referencias de los elementos modelados pues sólo modela los procesos de negocio de manera visual, quedando por debajo sólo los Gráficos Vectoriales Escalables (SVG, por sus siglas en inglés) y no objetos reales que contengan la información por lo que se hace necesario extender el modelador del Sauxe para que soporte las transformaciones de los modelos descritos en BPMN a objetos PHP
- ✓ Basado en las decisiones tecnológicas del CEIGE, se seleccionaron las herramientas y tecnologías a usar durante el proceso de desarrollo

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

Introducción

En este capítulo se presentarán los artefactos generados como parte del proceso de desarrollo, para dar solución al problema planteado. Entre los artefactos se encuentran las descripciones de los requisitos funcionales y no funcionales, el modelo conceptual y el diagrama de clases del diseño, todos ellos según el modelo de desarrollo propuesto por el CEIGE.

2.1 Modelo Conceptual

Con la construcción del modelo conceptual se logra una mejor comprensión del dominio del problema; el modelo conceptual no es más que una representación visual de los conceptos u objetos del mundo real que son significativos para el problema o el área que se analiza; representando las clases conceptuales, no los componentes de software. Puede verse como un modelo que comunica a los interesados, cuáles son los términos importantes y cómo se relacionan entre sí. (27)

El modelador de flujo de trabajo se integrará como una de las herramientas que provee el marco de trabajo Sauxe, su función es permitir a los usuarios el modelado de procesos de negocios, regido por la notación BPMN. En la *ilustración 1* Modelo conceptual, se presenta una perspectiva de todo aquello que interviene ya sea como concepto u objeto y sus relaciones entre sí.

El editor de proceso de negocio (EPN) brindará las herramientas necesarias para modelar procesos de negocios, definidas por un conjunto de clases (Paquetes Objetos) especificadas en la definición de procesos de negocios (BPM). Este último contiene actividades (Actividad) y de ellas se derivan componentes tales como Eventos (Eventos), subprocesos (Subprocesos), Tareas (Tareas), secuencias (Secuencias) y condicionales (Condicionales). Por otro lado las actividades son relacionadas con las asociaciones (Asociaciones) y estas a su vez se componen de artefactos (Artefactos). El editor posee uno o más participantes (personas o sistemas) encargados de ejecutar el flujo de trabajo.

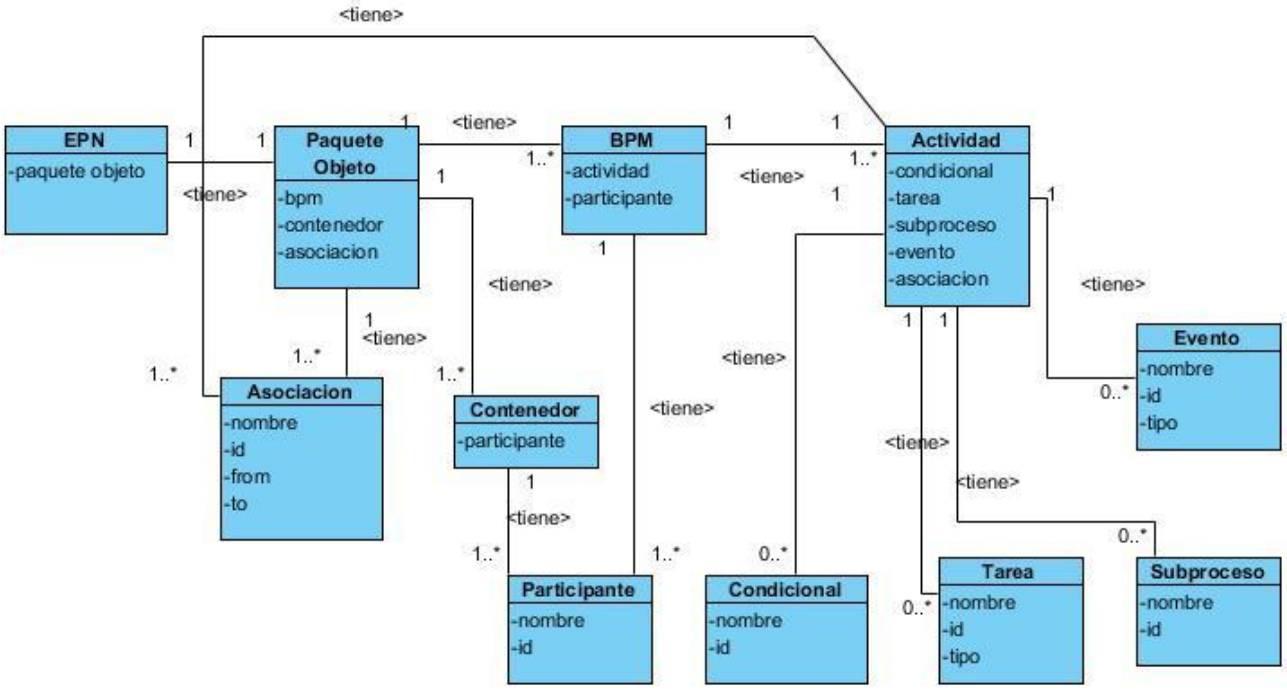

Ilustración 1: Modelo conceptual

2.2 Requisitos de software

La especificación de requisitos es una descripción del comportamiento del sistema. Que incluye tanto requisitos funcionales (definen funcionalidades del sistema así como, el contenido o la forma de un servicio o producto) como requisitos no funcionales (o complementarios). Los requisitos no funcionales son requisitos que imponen restricciones en el diseño o la implementación (Como por ejemplo restricciones en el diseño o estándares de calidad). (28)

2.2.1 Requisitos funcionales

Los requisitos funcionales comprenden todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta las diversas necesidades de los clientes. Existen varias técnicas para la captura de requisitos, entre las que se encuentran, las entrevistas, los talleres, los prototipos y los casos de uso, por mencionar algunas. Los requisitos se pueden dividir en funcionales y no funcionales. Los funcionales son los que el usuario necesita que efectúe el software y los no funcionales son los "recursos" necesarios para que trabaje el sistema e imponen restricciones al diseño o funcionamiento del mismo.

Para capturar los requisitos funcionales, se empleó la técnica: *Tormenta de ideas*, que consiste en la reunión de varios interesados para expresar sus ideas sobre un problema en cuestión y valorar una posible solución. Cada participante expresa su criterio sin ser interrumpido por ningún otro. Al finalizar la sesión de lluvia de ideas se puede hacer una recolección de ideas sin duplicidad. (29)

Para el sistema propuesto fueron identificados los siguientes requisitos funcionales:

RF1: Crear procesos de negocio

RF2: Crear actividades

RF3: Modificar actividades

RF4: Crear Eventos

RF5: Modificar Eventos

RF6: Crear nodos de decisión

RF7: Modificar nodos de decisión

RF8: Crear Pool

RF9: Modificar Pool

RF10: Crear Lane

RF11: Modificar Lane

RF12: Adicionar objetos de conexión

RF13: Guardar procesos de negocio

RF14: Eliminar elementos

A continuación se listan las descripciones de los requisitos funcionales identificados:

Requisito funcional: Crear procesos de negocio

Conceptos tratados	**Conceptos**	**Atributos**
	Proceso de Negocio	id, nombre
Precondiciones	**Precondiciones**	**Pre-requisito**
	El usuario debe tener acceso al modelador de procesos de negocio	N/A
Descripción	*1-El usuario accede al modelador* *2-El sistema crea un paquete de procesos de negocio* *3- El sistema visualiza el área de modelado*	

Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*
Post-condiciones	*Se ha creado un proceso de negocio*
Post-requisito	*Crear Pool*

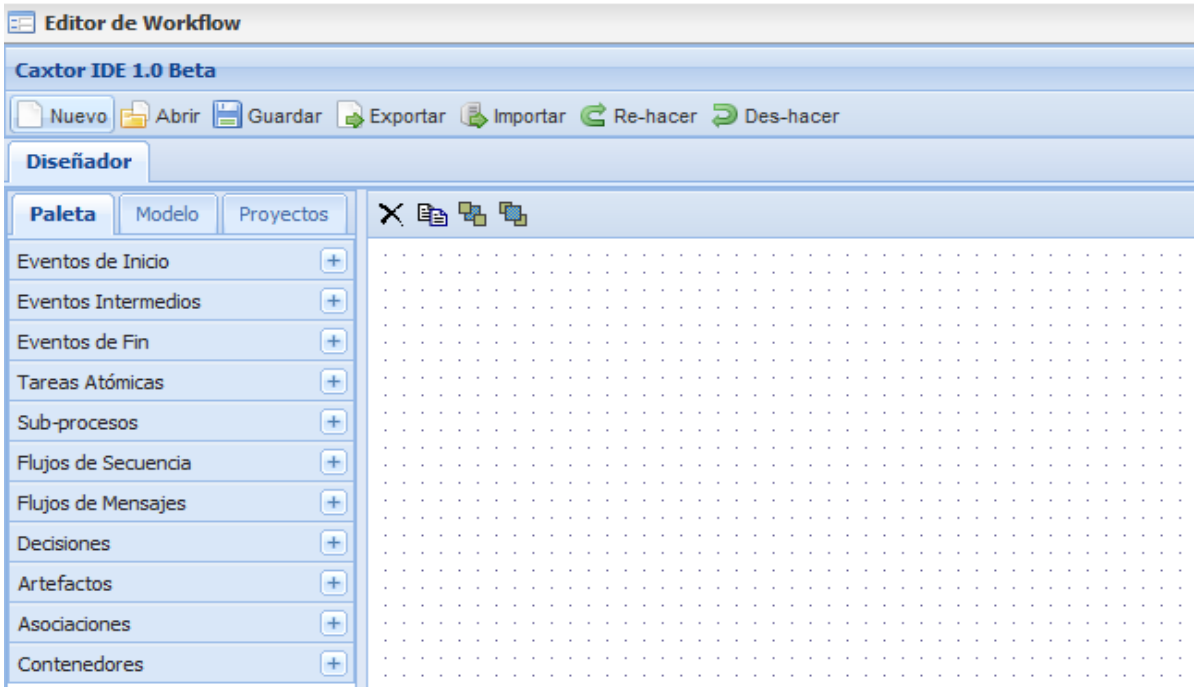

Ilustración 2: Prototipo de interfaz Crear procesos de negocio

Requisito funcional: Crear Tareas

Conceptos tratados	**Conceptos**	**Atributos**
	Tarea Usuario	*variable de entrada, variable de salida, rol asociado, controladora, acción, nombre, id*
	Tarea Servicio	*dirección de servicio, variable de entrada, nombre, id*
	Tarea Manual	*nombre, id*
Precondiciones	**Precondiciones**	**Pre-requisito**

	Debe existir un elemento Pool o Lane creado	*Crear Pool*
Descripción	*1-El usuario debe seleccionar en la paleta de componente las Tareas atómicas* *2- Dar clic en el elemento Tarea (Usuario, Servicio, Manual)* *3- El usuario debe arrastrar el elemento al área de modelado de proceso de negocio sobre un elemento Pool o un elemento Lane* *4-El sistema visualiza el elemento seleccionado*	
Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*	
Post-condiciones	*Se ha creado un elemento Tarea (Usuario, Servicio, Manual)*	
Post-requisito	N/A	

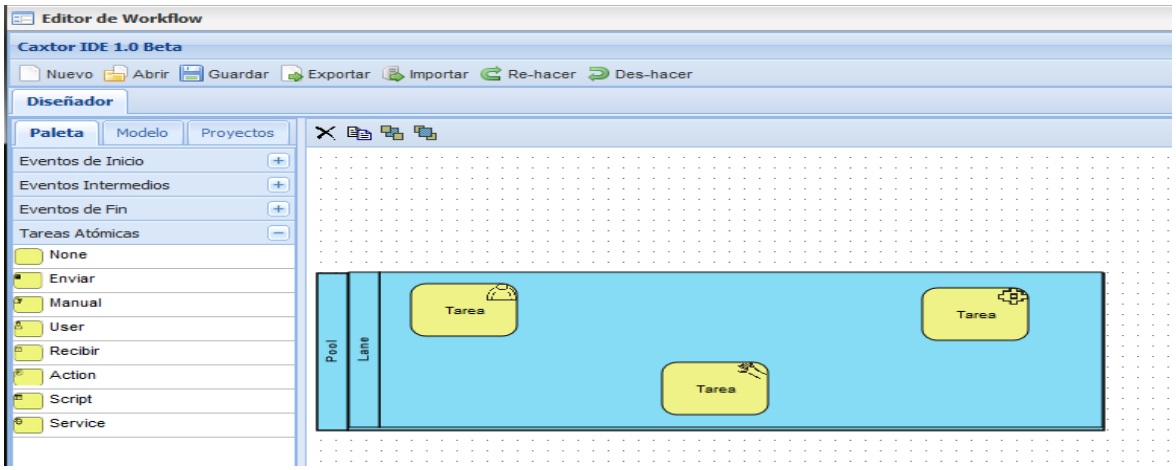

Ilustración 3: Prototipo de interfaz Crear Tareas

Requisito funcional: Modificar Tareas

Conceptos tratados	Conceptos	Atributos
	Tarea Usuario	*nombre, tipo, id, variable de entrada, variable de salida, rol asociado, controladora, acción*
	Tarea Servicio	*nombre, tipo, id, variable de entrada, dirección de servicio*

	Tarea Manual	*nombre, tipo, id*
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir un elemento Tarea creado	*Crear Tareas*
Descripción	1- *El usuario debe dar doble clic sobre el elemento* 2- *El usuario debe modificar los campos en el formulario* 3- *El usuario debe accionar el botón **Aplicar** del formulario* 4- *El sistema modifica los valores*	
Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*	
Post-condiciones	*Se ha creado un elemento Tarea*	
Post-requisito	N/A	

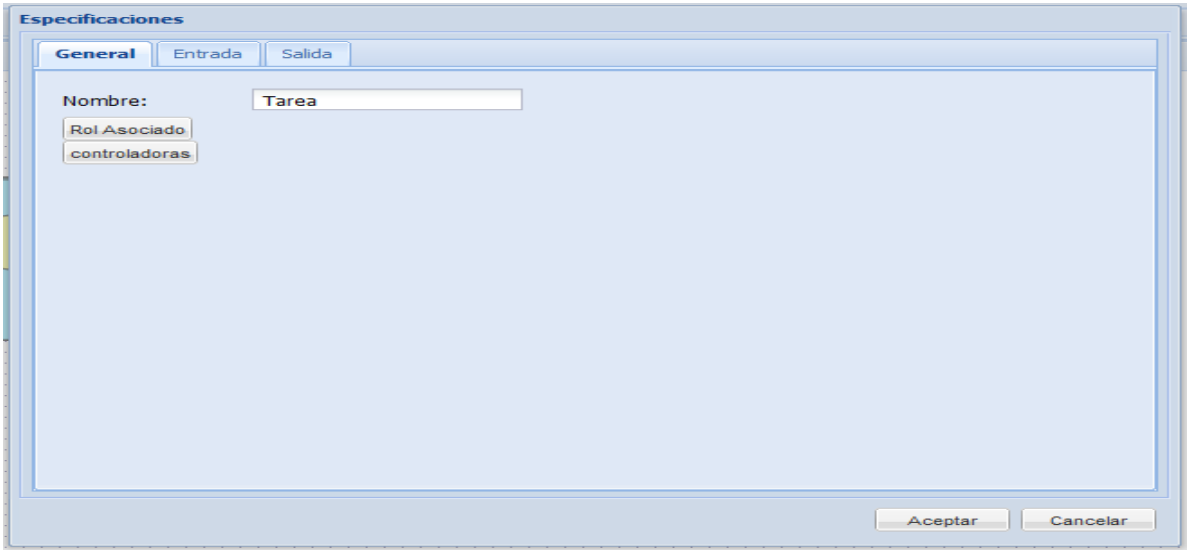

Ilustración 4: Prototipo de interfaz Modificar Tareas

Requisito funcional: Crear Eventos

Conceptos tratados	**Conceptos**	**Atributos**
	Evento inicio Evento intermedio Evento fin	*nombre, tipo, id*
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir elemento Pool o Lane creado	*Crear Pool*

Descripción	1- *El usuario debe seleccionar en la paleta de componentes, dentro de los Eventos(inicio, intermedio, fin) el tipo de elemento Evento(mensaje, condicional, error, temporizador, señal, cancelar)* 2- *El usuario debe arrastrar el elemento al área de modelado de proceso de negocio sobre un elemento Pool o un elemento Lane* 3- *El sistema visualiza el Evento seleccionado*
Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*
Post-condiciones	*Se ha creado un elemento Evento*
Post-requisito	N/A

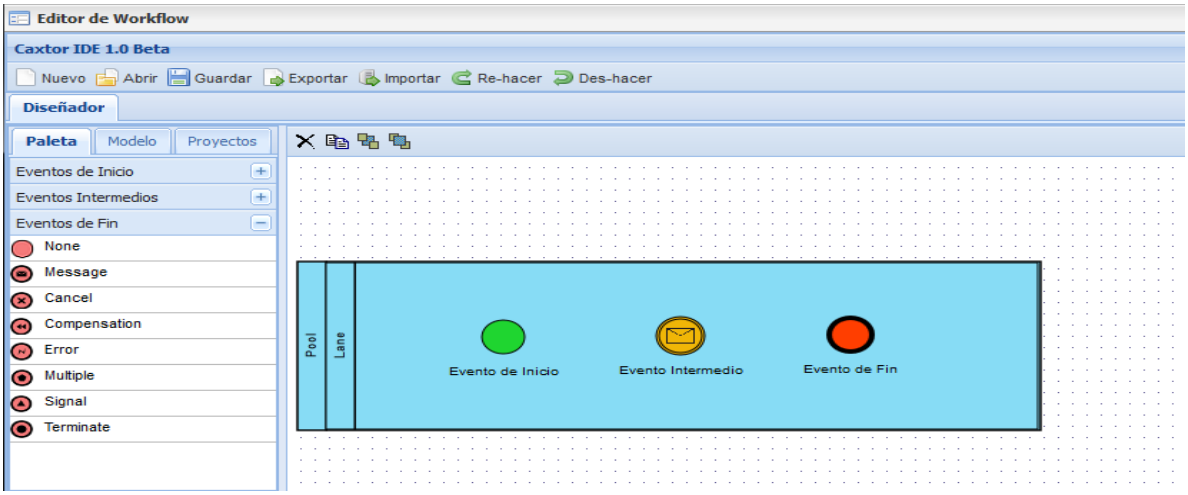

Ilustración 5: Prototipo de interfaz Crear Eventos

Requisito funcional: Modificar Eventos

Conceptos tratados	**Conceptos**	**Atributos**
	Evento inicio Evento intermedio Evento fin	*Nombre, tipo, Id*
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir elemento Evento creado	*Crear Eventos*
Descripción	1- *El usuario debe dar doble clic sobre el elemento* 2- *El usuario debe modificar los campos en el formulario* 3- *El usuario debe accionar el botón **Aplicar** del formulario* 4- *El sistema modifica los datos*	
Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*	

Post-condiciones	*Se ha modificado un elemento Evento*
Post-requisito	N/A

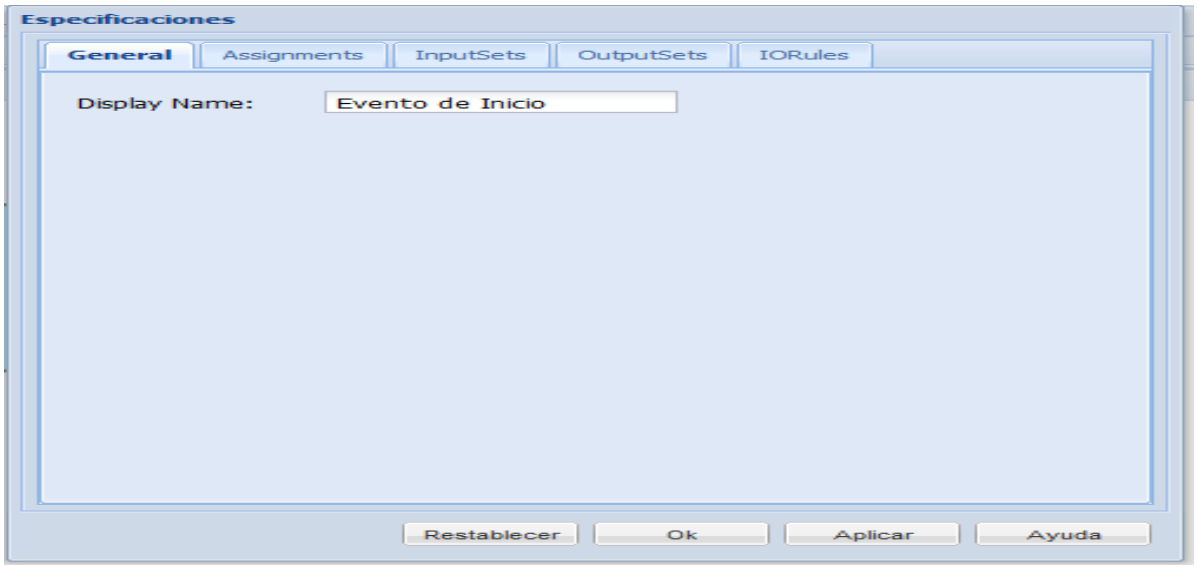

Ilustración 6: Prototipo de interfaz Modificar Eventos

Requisito funcional: Crear nodos de decisión

Conceptos tratados	**Conceptos**	**Atributos**
	Nodo de decisión	*nombre, id*
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir elemento Pool o Lane creado	Crear Pool
Descripción	1- *El usuario debe seleccionar en la paleta de componentes dentro de las decisiones, el elemento nodo de decisión* 2- *El usuario debe arrastrar el elemento al área de modelado de proceso de negocio sobre un elemento Pool o un elemento Lane* 3- *El sistema visualiza el elemento seleccionado*	
Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*	
Post-condiciones	*Se ha creado un elemento nodo de decisión*	
Post-requisito	N/A	

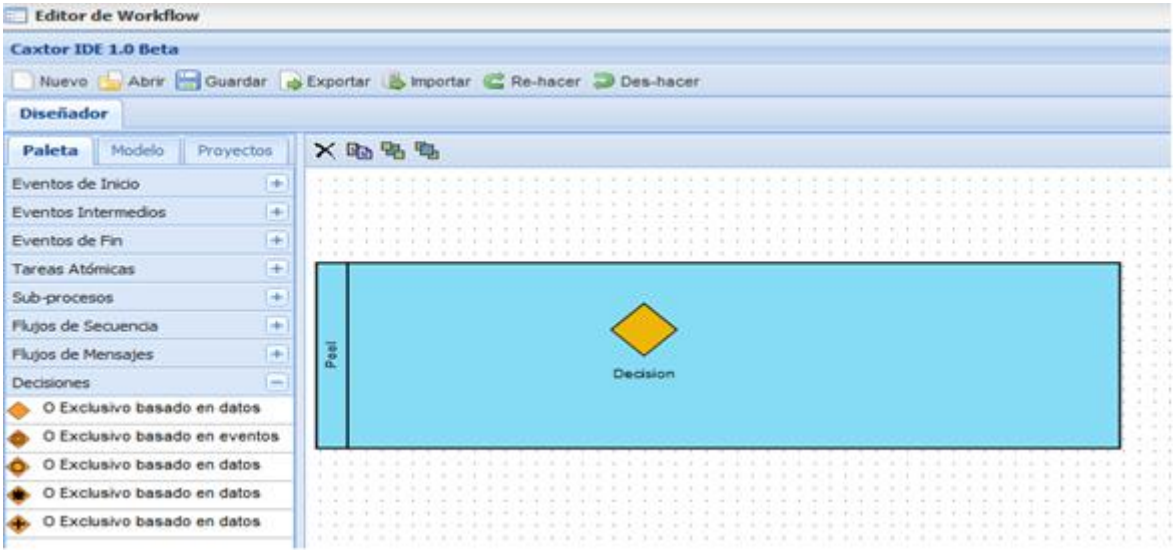

Ilustración 7: Prototipo de interfaz Crear nodos de decisión

Requisito funcional: Modificar nodos de decisión

Conceptos tratados	**Conceptos**	**Atributos**
	Nodo de decisión	*nombre, id*
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir un elemento nodo de decisión creado	*Crear nodos de decisión*
Descripción	1- *El usuario debe dar doble clic sobre el elemento* 2- *El usuario debe modificar los campos en el formulario* 3- *El usuario debe accionar el botón **Aplicar** del formulario* 4- *El sistema modifica los datos*	
Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*	
Post-condiciones	*Se ha modificado un elemento nodo de decisión*	
Post-requisito	N/A	

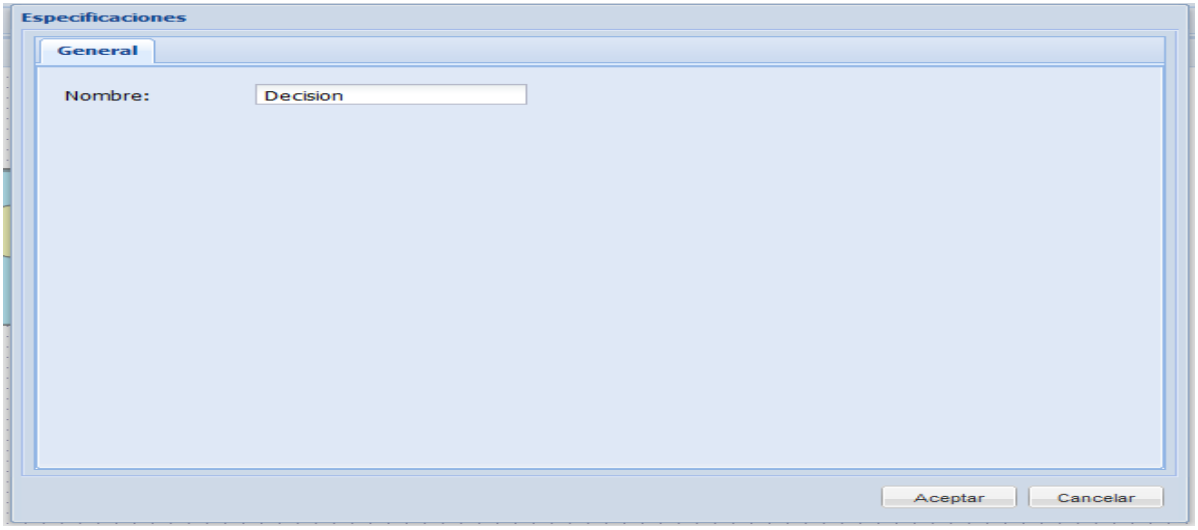

Ilustración 8: Prototipo de interfaz Modificar nodos de decisión

Requisito funcional: Crear Pool

Conceptos tratados	**Conceptos**	**Atributos**
	Pool	*proceso, id, nombre*
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir un paquete de proceso de negocio creado	*Crear proceso de negocio*
Descripción	1- *El usuario debe seleccionar en la paleta de componente dentro de los contenedores, el elemento Pool* 2- *El usuario debe arrastrar el elemento al área de modelado de proceso de negocio* 3- *El sistema visualiza el elemento seleccionado*	
Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*	
Post-condiciones	*Se ha creado un elemento Pool*	
Post-requisito	N/A	

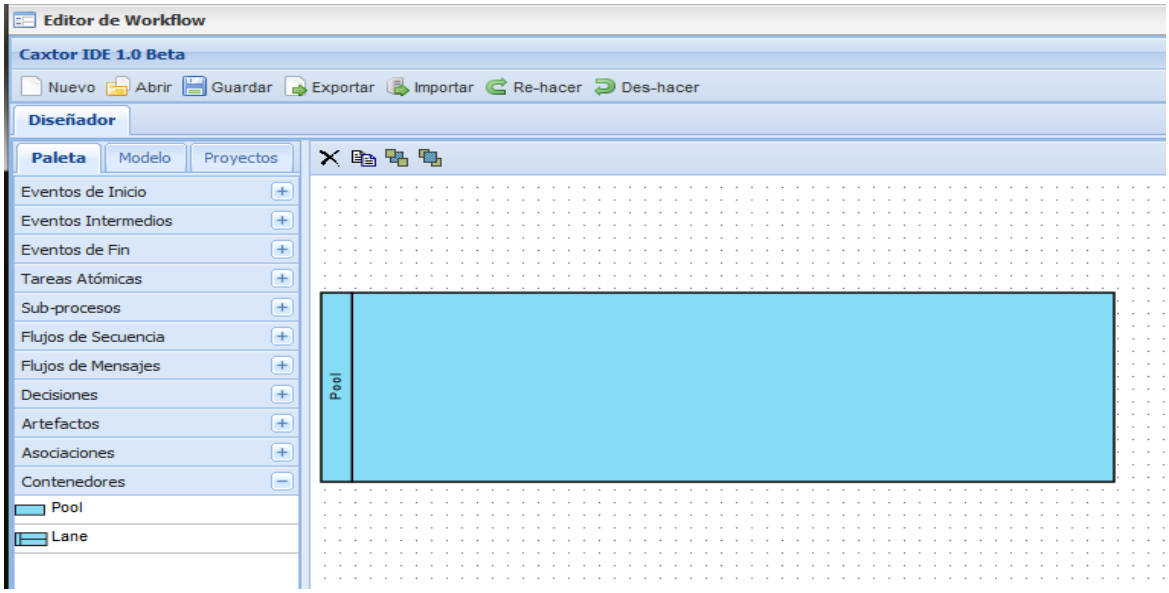

Ilustración 9: Prototipo de interfaz Crear Pool

Requisito funcional: Modificar Pool

Conceptos tratados	**Conceptos**	**Atributos**
	Pool	*nombre, proceso, id*
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir un elemento Pool creado	*Crear Pool*
Descripción	1- *El usuario debe dar doble clic sobre el elemento* 2- *El usuario debe modificar los campos en el formulario* 3- *El usuario debe accionar el botón aplicar del formulario* 4- *El sistema modifica los datos*	
Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*	
Post-condiciones	*Se ha modificado un elemento Pool*	
Post-requisito	N/A	

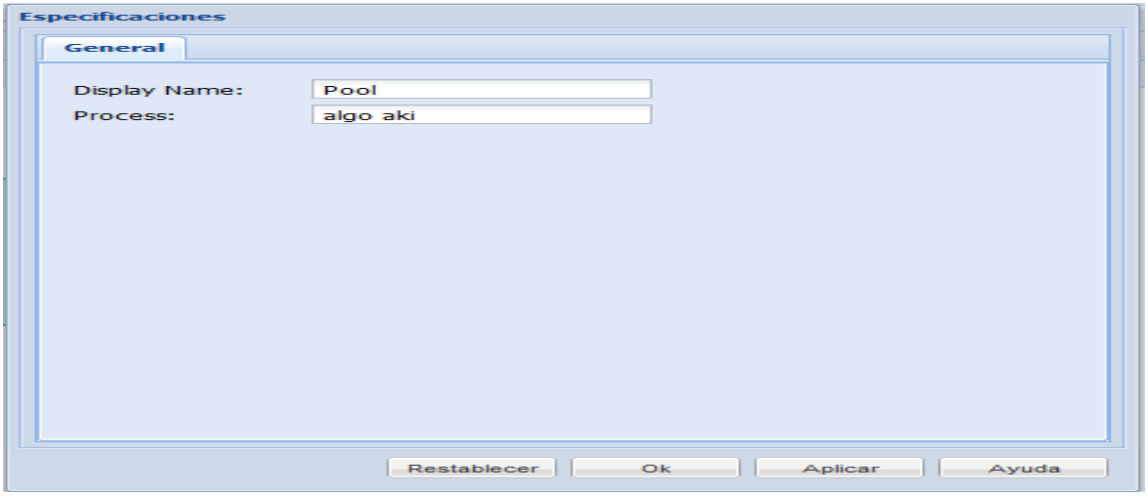

Ilustración 10: Prototipo de interfaz Modificar Pool

Requisito funcional: Crear Lane

Conceptos tratados	**Conceptos**	**Atributos**
	Lane	*nombre, participante, id*
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir un elemento Pool creado	Crear Pool
Descripción	1- *El usuario debe seleccionar en la paleta de componente dentro de los contenedores, el elemento Lane* 2- *El usuario debe arrastrar el elemento al área de modelado de proceso de negocio sobre un elemento Pool* 3- *El sistema visualiza el elemento seleccionado*	
Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*	
Post-condiciones	*Se ha creado un elemento Lane*	
Post-requisito	N/A	

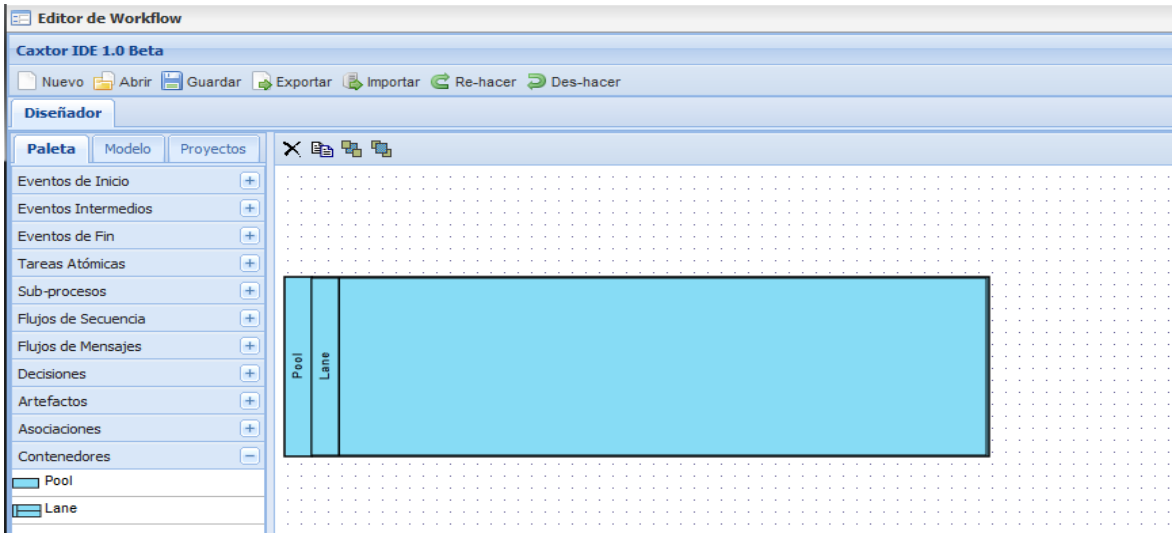

Ilustración 11: Prototipo de interfaz Crear Lane

Requisito funcional: Modificar Lane

Conceptos tratados	**Conceptos**	**Atributos**
	Lane	*nombre, participante, id*
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir un elemento Lane creado	*Crear Lane*
Descripción	1- *El usuario debe dar doble clic sobre el elemento* 2- *El usuario debe modificar los campos en el formulario* 3- *El usuario debe accionar el botón aplicar del formulario* 4- *El sistema modifica los datos*	
Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*	
Post-condiciones	*Se ha modificado un elemento Lane*	
Post-requisito	N/A	

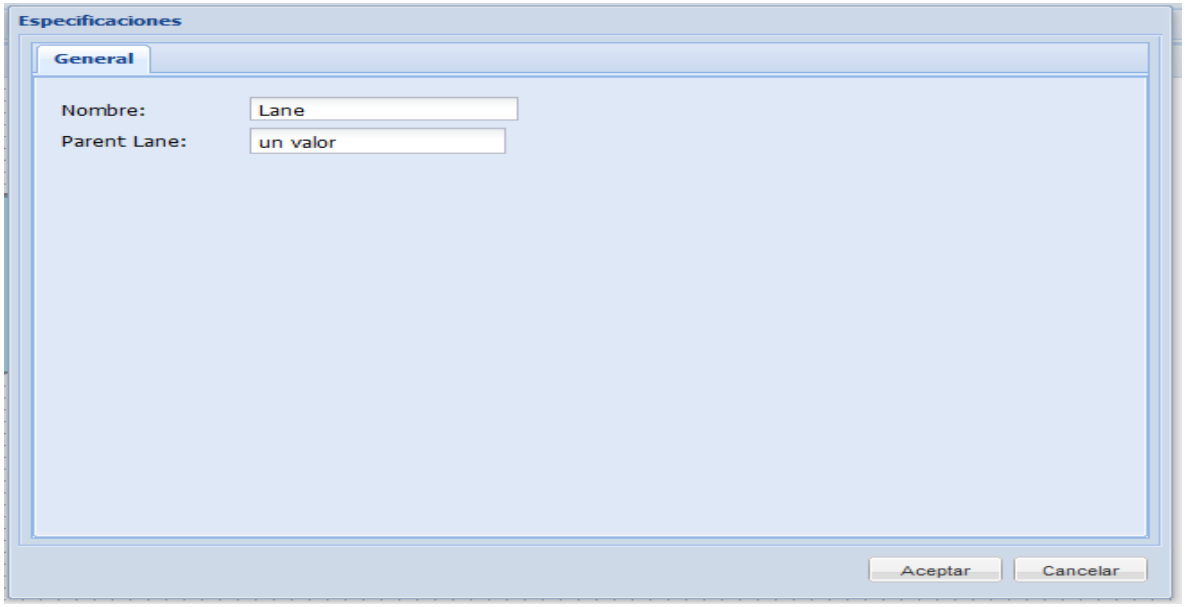

Ilustración 12: Prototipo de interfaz Modificar Lane

Requisito funcional: Crear objetos de conexión

Conceptos tratados	**Conceptos**	**Atributos**
	Objeto conexión	id, inicio, fin
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir un elemento Pool o Lane creado y al menos dos elementos del tipo Tarea, Evento o nodo de decisión	*Crear Tareas*
Descripción	1- *El usuario debe seleccionar en la paleta de elementos dentro de flujos de secuencia, el elemento objeto de conexión* 2- *El usuario debe seleccionar los elementos a conectar dentro del área de modelado de proceso de negocio* 3- *El sistema dibuja un objeto conexión entre los elementos seleccionados*	
Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*	
Post-condiciones	*Se ha creado un elemento objeto de conexión*	
Post-requisito	N/A	

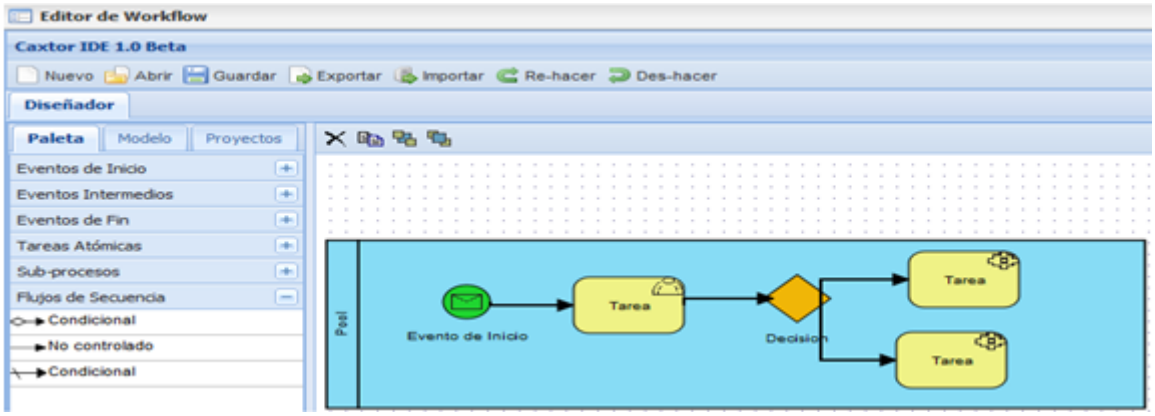

Ilustración 13: Prototipo de interfaz Crear objetos de conexión

Requisito funcional: Guardar procesos de negocio

Conceptos tratados	**Conceptos**	**Atributos**
	Proceso de negocio	N/A
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir un proceso de negocio modelado	*Crear procesos de negocio*
Descripción	1- *El usuario debe seleccionar en el menú la opción **Guardar*** 2- *El sistema guarda el proceso de negocio*	
Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*	
Post-condiciones	*Se ha guardado un proceso de negocio*	
Post-requisito	N/A	

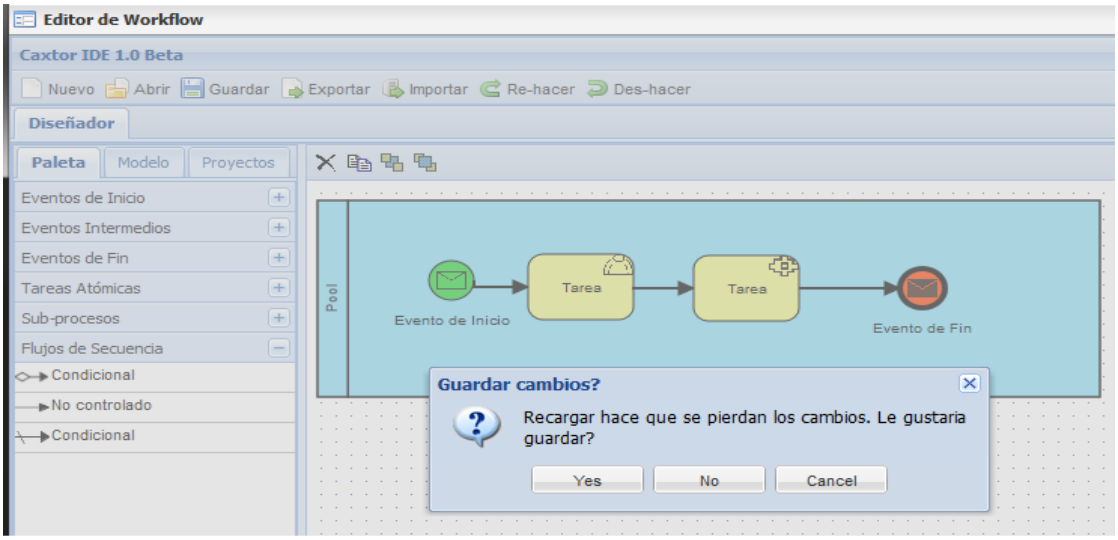

Ilustración 14: Prototipo de interfaz Guardar procesos de negocio

Requisito funcional: Eliminar elementos

Conceptos tratados	**Conceptos**	**Atributos**
	Elemento	N/A
Precondiciones	**Precondiciones**	**Pre-requisito**
	Debe existir al menos un elemento creado	N/A
Descripción	*1-El usuario debe seleccionar el elemento que desee eliminar* *2-El usuario debe dar clic derecho sobre el elemento y seleccionar la opción* ***Eliminar este elemento*** *3-El sistema borra el elemento seleccionado*	
Validaciones	*Se validan los datos según lo establecido en el CSG-CNP Modelo Conceptual de la Solución informática para el soporte de la transformación entre modelos BPMN y objetos PHP*	
Post-condiciones	*Se ha eliminado un elemento*	
Post-requisito	N/A	

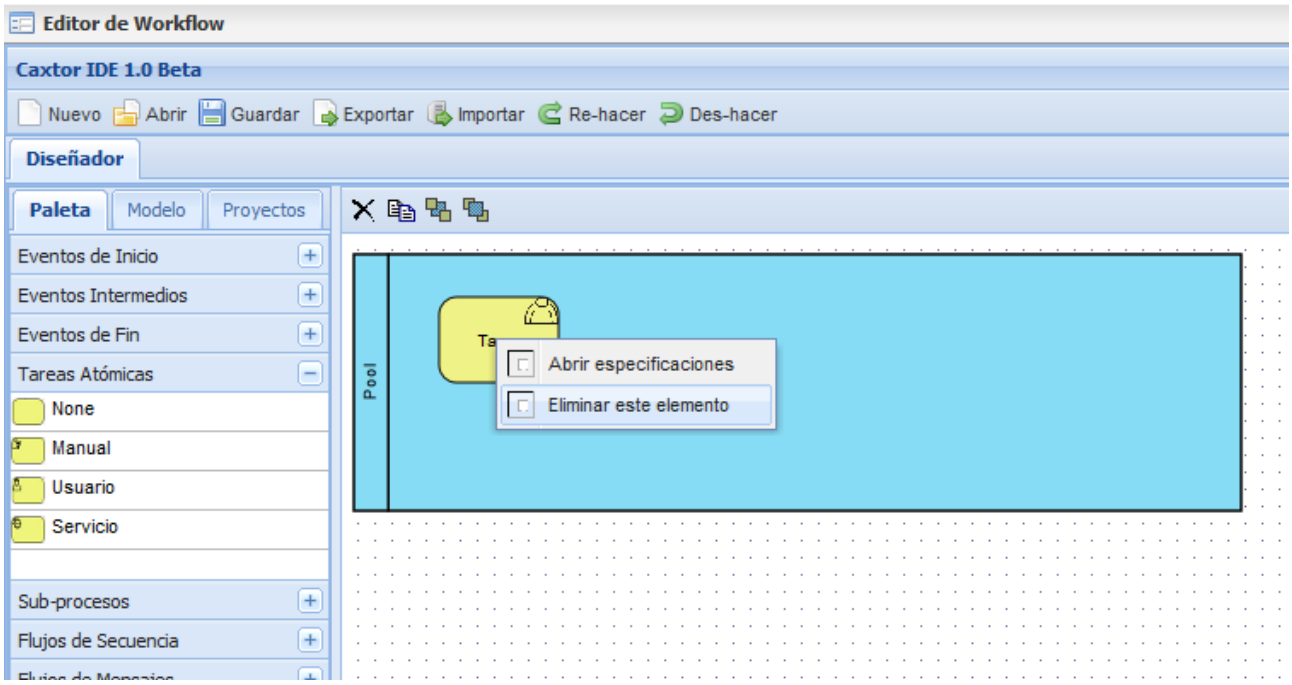

Ilustración 15: Eliminar elementos

Uno de los procesos de la Ingeniería de requisitos es la validación, con el objetivo de demostrar que los requisitos definidos cumplen con las necesidades y expectativas del cliente o usuario. Examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones y 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. (58)

Para la validación de los requisitos se aplican técnicas, entre ellas 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 funcionales propuestos se utilizó la técnica de Revisión técnica formal.

2.2.2 Requisitos no funcionales

Los requisitos no funcionales especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad, y fiabilidad. (62)

Los requisitos no funcionales de hardware y software que debe poseer el componente son los establecidos por el centro CEIGE al inicio del proceso de desarrollo, a continuación se describen: (49)

	Cliente	Servidor (1)	Servidor (2)
Software	• Mozilla Firefox 2.2	• Ubuntu Server. • Apache 2.0 • PHP 5	• Ubuntu Server. • PostgreSQL 8.3
Hardware	• Procesador: 1.40 GHZ • RAM: 256 MB (recomendado 512 Mb) • Tarjeta de Red: 1	• Procesador: 3.00 GHZ • RAM: 1GB • Disco duro: 160 GB • UPS: 1 • Lector de CD: 1 • Tarjeta de Red: 1	• Procesador: 3.00 GHZ • RAM: 1GB • Disco duro: 160 GB • UPS: 1 • Lector de CD: 1 • Tarjeta de Red: 1

Ilustración 16: Requisitos no funcionales

2.3 Patrones

Se denominan patrones de software a los patrones que son aplicados durante el desarrollo de un sistema de software o aplicación.

La utilización de patrones de software en el desarrollo de una aplicación permite el ahorro de tiempo al programador, quien enfocará sus esfuerzos en el desarrollo de la lógica de la aplicación. Brinda una arquitectura uniforme a la misma, facilitando así su mantenimiento, modificación y expansión. (30)

Un patrón presenta una estrategia definida en busca de solucionar un problema específico, debido a esto es posible la utilización de uno o más patrones durante el desarrollo de una aplicación.

En la actualidad la opción más aceptable y recomendable en el desarrollo de aplicaciones Web es el patrón MVC (Modelo Vista Controlador). (31)

2.3.1 Patrón arquitectónico Modelo-Vista-Controlador

El patrón modelo-vista-controlador divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

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

Cada vista tiene asociado un componente controlador

- ✓ **Controlador** (Controller): reciben las entradas, usualmente como Eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, entre otras. Los Eventos son traducidos a solicitudes de servicio ("servicerequests") para el modelo o la vista
  La arquitectura MVC fue diseñada para que los cambios realizados en una aplicación afecten lo menos posible a la programación que ya se encuentra implementada. Esto es posible gracias a que su arquitectura desacopla en diferentes capas los datos, la lógica del negocio y la lógica de presentación. Todo esto hace posible la actualización y desarrollo de cada uno de los componentes de forma independiente. (32)

Ilustración 17: MVC

2.3.2 Patrones de diseño

"Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software." (33)

Un patrón de diseño es:

- ✓ Una solución estándar para un problema común de programación
- ✓ Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios
- ✓ Un proyecto o estructura de implementación que logra una finalidad determinada
- ✓ Un lenguaje de programación de alto nivel
- ✓ Una manera más práctica de describir ciertos aspectos de la organización de un programa

En el presente epígrafe se relacionan los patrones de diseño que se utilizan en la solución en aras de lograr mayor flexibilidad y encapsulación, así como menor acoplamiento:

Solitario (Singleton)

Tipo: creación, a nivel de objetos.

Propósito: garantizar que una clase sólo tiene una única instancia, proporcionando un punto de acceso global a la misma.

Ventajas

- ✓ El acceso a la "InstanciaÚnica" está más controlado
- ✓ Se reduce el espacio de nombres (frente al uso de variables globales)
- ✓ Permite refinamientos en las operaciones y en la representación, mediante la especialización por herencia de "Solitario"
- ✓ Es fácilmente modificable para permitir más de una instancia y, en general, para controlar el número de las mismas (incluso si es variable)
- ✓ Es más flexible que la alternativa de las "operaciones de clase", además de que éstas (en C++), al ser funciones miembro estáticas, no son virtuales, luego no pueden ser especializadas mediante herencia ni redefinidas polimórficamente.(51)

La utilización de patrones de diseño surgió a partir de la aparición de problemas a nivel de clases. Se necesitaba tener una única instancia del objeto JavaScript que se iba generando a partir del modelo de proceso creado en la vista, por tanto se decidió utilizar una variante del patrón singleton que se encarga fundamentalmente de que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella.

Esto mismo ocurría con los objetos PHP que se construyeran a partir del JSON que se envía al servidor. De igual manera se necesita una sola instancia de la estructura de objetos que se creaba en PHP.

Acción (Command)

Tipo: comportamiento, a nivel de objetos

Contexto: este patrón, al encapsular en un objeto la acción, promueve una separación entre dicha acción y la petición que resuelve, lo cual redunda en una mayor flexibilidad en todo lo relativo a la acción. Esto tiene las siguientes consecuencias:

Diferentes objetos pueden ejecutar la misma acción sin necesidad de repetir su declaración e implementación. Basta con que la "compartan".

- ✓ Se puede cambiar con facilidad (incluso dinámicamente) la acción que realiza o está asociada a un objeto
- ✓ Se pueden añadir nuevas acciones sin tocar las clases ya existentes
- ✓ Se puede especificar, encolar y ejecutar una acción en momentos diferentes
- ✓ Se puede contemplar la posibilidad de deshacer una operación. Para ello es necesario disponer de una "lista de históricos" en la que se reflejen los diferentes estados por los que

pasa el sistema o el objeto afectado por las sucesivas operaciones. Igual que se deshacen una o varias operaciones, se puede dar la posibilidad de volver a hacer ("rehacer") operaciones ya deshechas

- ✓ Se pueden ensamblar acciones, para formar una acción compuesta (macroacción). Esto puede resultar interesante en sistemas de información que dan soporte a transacciones (51)

Se necesita hacer constantes variaciones en las interfaces visuales y estas incluían los botones de manejo de eventos por lo que incluir el código de las operaciones dentro de la implementación del botón significa una repetición continua del código. Por lo que se decidió separar el manejo de las solicitudes de los eventos, de la implementación de los botones.

2.4 Modelo del diseño

El modelo del diseño es el encargado de modelar el sistema teniendo en cuenta las especificaciones de requisitos descriptas previamente. Este facilita la implementación en gran medida ya que en él participan las clases, las interfaces, los conceptos y sistemas necesarios para la solución.

2.4.1 Diagrama de clases del diseño

Un diagrama de clases del diseño representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Permite visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de convencimiento. Está compuesto por los siguientes elementos: Clase: atributos, métodos y visibilidad. Relaciones: Herencia, Composición, Agregación, Asociación y Uso con sus respectivos estereotipos. Ver *ilustración 18* Modelo de clases del diseño.

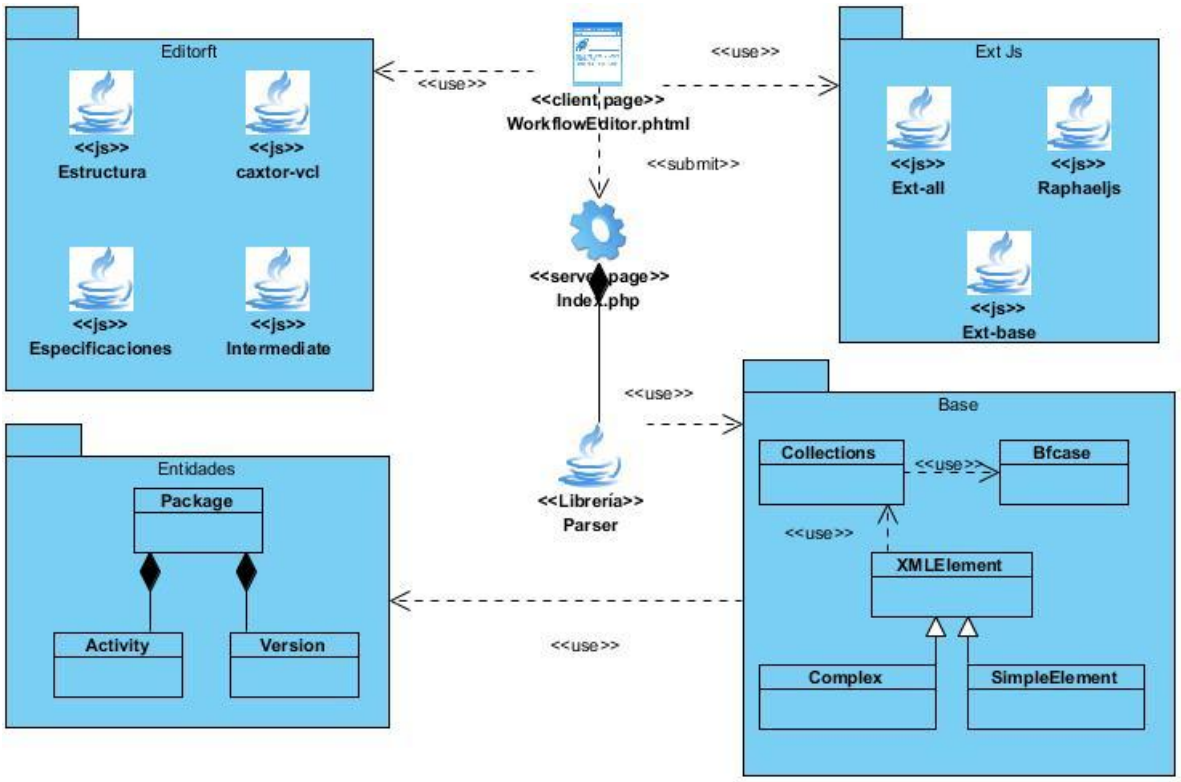

Ilustración 18: Diagrama de clases del diseño

Conclusiones parciales

Durante el desarrollo del capítulo se realizaron los artefactos generados durante el análisis y diseño de la solución, guiados por el modelo de desarrollo orientado a componentes propuesto por el CEIGE. Se representó el modelo conceptual, el diagrama de clases del diseño y la arquitectura base que lo rige. También se evidenciaron los patrones de diseño utilizados y la lista de requerimientos que debe cumplir el sistema. Una vez concluida esta etapa del ciclo de desarrollo, se procederá a las posteriores fases: implementación y prueba.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

En el presente capítulo se generan los artefactos de las fases de implementación y prueba, incluye los estándares de codificación, los diagramas del modelo de implementación: Diagrama de componentes y Diagrama de despliegue así como el conjunto de pruebas realizadas a la solución.

3.1 Estándar de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, establezca un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. (56)

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas, y, posteriormente, se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener. (56)

Durante la implementación del sistema se utilizaron los siguientes estándares de codificación:

Notación Húngara: esta convención se basa en definir prefijos para cada tipo de datos y según el ámbito de las variables. También es conocida como notación REDDICK (por el nombre de su creador). La idea de esta notación es la de dar mayor información al nombre de la variable, método o función definiendo en ella un prefijo que identifique su tipo de dato y ámbito.

Notación PascalCasing: es como la notación húngara pero sin prefijos. En este caso, los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.

Notación CamelCasing: es parecido al Pascal-Casing con la excepción que la letra inicial del identificador debe estar en minúscula.

Nomenclatura de las clases

Los nombres de las clases comenzarán con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing.

El nombre de las clases se basa según el XSD utilizado para definir los elementos que componen un BPMN, es decir se le agregará al nombre original en el XSD un prefijo o sufijo que describe la ubicación dentro del marco de trabajo Sauxe. A continuación se listan los prefijos y sufijos determinados para cada tipo de clase.

A las clases CONTROLADORAS se le adiciona el sufijo: “ZendExt_WF_WFObject_”.

Ejemplo: ZendExt_WF_WFObject_Parser.

A las CLASES BASES se le adiciona el sufijo “ZendExt_WF_WFObject_Base_”.

Ejemplo: ZendExt_WF_WFObject_Base_Collections

A las CLASES ENTIDADES pertenecientes al modelo de negocio se le adiciona el sufijo “ZendExt_WF_WFObject_Entidades_”.

Ejemplo: ZendExt_WF_WFObject_Entidades_Activities.

Nomenclatura de las funciones

Los nombres a emplear para las funciones se escribirán con minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing.

Nomenclatura de las variables

Los nombres a emplear para las variables se escribirán con la notación húngara a excepción de los elementos que componen un BPMN definidos en el XSD, que permanecerán invariablemente con el nombre original, en caso de que sea un nombre compuesto se empleará notación CamelCasing.

El nombre de las variables estará dado según la notación húngara, es decir se le agregará al nombre original un prefijo que describa el tipo de dato. A continuación se listan los prefijos a utilizar para cada tipo de dato.

Arreglos:	“arr, aa”
Objetos:	“obj, aa”
Enteros:	“int”
Cadena:	“str”
Float:	“flt”

Boolean:	"boo"

Nomenclatura de los comentarios

Los comentarios deben ser claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando.

Es de gran importancia que tanto clases como métodos sean correctamente comentados para obtener un código fuente más legible y reutilizable, de manera que se pueda dar mantenimiento de manera más fácil a lo largo del tiempo. Antes de la declaración de una clase o método se escribirá una breve descripción donde se explique la intención del mismo. Este comentario tendrá la siguiente estructura:

En las clases:	**En los métodos:**
/* **** Nombre de la clase **** **** Descripción **** **** @author **** **** @package *(módulo)*** **** @subpackage *(sub módulo)*** **** @copyright **** **** @version (versión - parche)*** ****/***	/** * Nombre de la función * * Descripción * * @author * (en caso de que no sea el autor de la clase) * @param *(los parámetros que se le pasan a la función con su descripción) * @throws *(en caso de que dispare una excepción) * @return *(se pone lo que devuelve la función y un comentario) /*

3.2 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño, se implementan en términos de componentes. Representan la organización de los mismos de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje de programación utilizado, así como las dependencias y los recursos necesarios para poder ejecutar el sistema desarrollado.

3.2.1 Diagrama de componentes

Un diagrama de componentes muestra las dependencias lógicas entre componentes de software, incluye desde componentes fuentes y binarios hasta ejecutables y librerías, ilustran las piezas del software. Prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema, es decir para describir la vista de implementación estática de un sistema. Los diagramas de componentes son utilizados para modelar sistema de software de cualquier tamaño y complejidad. (35)

La *Solución informática para el soporte de la transformación de modelos descritos en BPMN a objetos PHP* estará implementada sobre el marco de trabajo Sauxe, se divide en el Editor de Flujo de Trabajo(Editor FT) que compone las interfaces con las que va interactuar el usuario para realizar el modelado de procesos del negocio, las mismas a su vez se dividen en el área de trabajo para el modelado de procesos de negocios contenido en **canvas.js** y en un conjunto de librerías disponibles para modelar dichos procesos, contenido en **especificaciones.js, estructura.js y intermediate.js**. El editor depende directamente del componente **WFObject** encargado de brindarle soporte PHP al diseño realizador en la interfaz visual. Ver *ilustración 19* Diagrama de componentes.

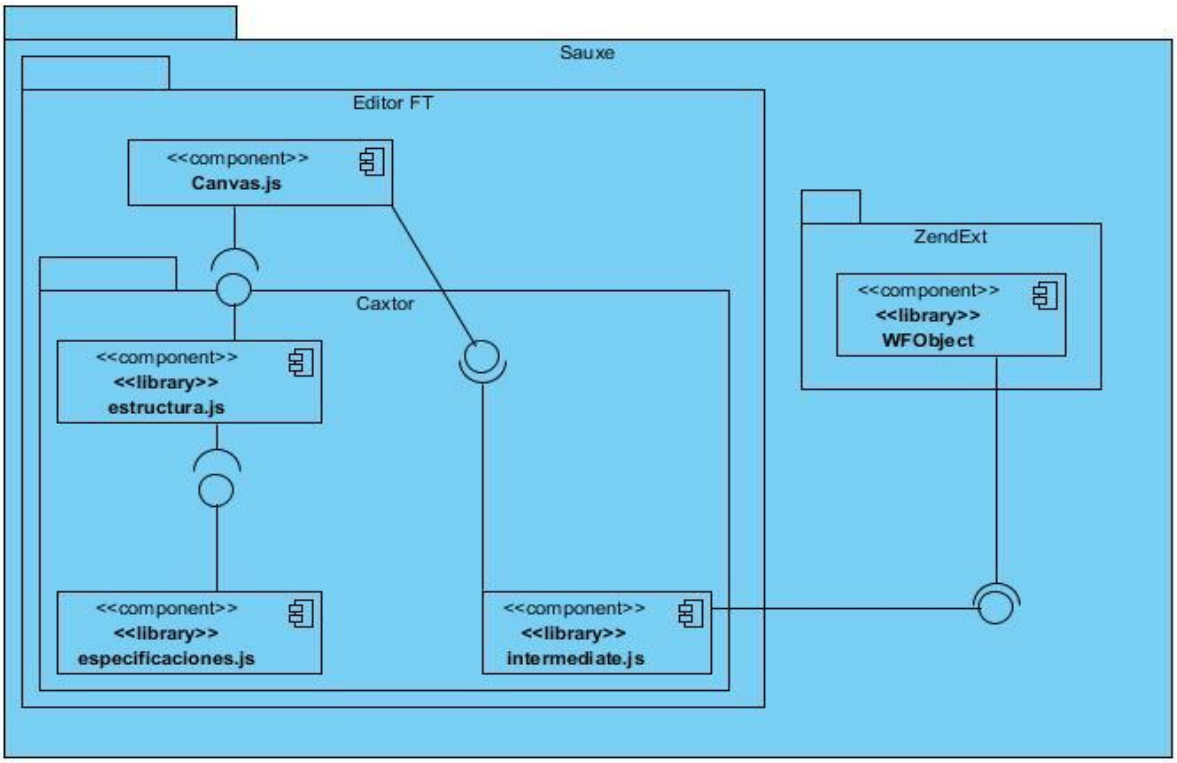

Ilustración 19: Diagrama de componentes

3.2.2 Diagrama de Despliegue

Los diagramas de despliegue muestran a los nodos procesadores la distribución de los procesos y de los componentes. (27)

El diagrama de despliegue describe los dispositivos necesarios para la utilización del producto, es decir, desde dónde se encuentra instalado, dónde se puede utilizar, hasta dónde se conecta para tomar los datos. Es una vista panorámica que describe los requisitos de hardware y software mínimos, necesarios para que esta herramienta funcione. Ver *ilustración 20* Diagrama de despliegue.

Ilustración 20: Diagrama de despliegue

3.3 Métricas de software

Las métricas del software se refieren a un amplio elenco de mediciones para el software de computadora. La medición se puede aplicar al proceso del software con el intento de mejorarlo sobre una base continua. Se puede utilizar en el proyecto del software para ayudar en la estimación, el control de calidad, la evaluación de productividad y el control de proyectos. Finalmente, el ingeniero de software puede utilizar la medición para ayudar a evaluar la calidad de los resultados de trabajos técnicos y para ayudar en la toma de decisiones tácticas a medida que el proyecto evoluciona. (29)

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

Complejidad de implementación: es la complejidad de implementación que posee una estructura de diseño de clases.

Reutilización: es el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

Acoplamiento: es el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.

Complejidad del mantenimiento: es 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 indirectamente, pero fuertemente en los costes y la planificación del proyecto.

Cantidad de pruebas: es el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, módulo, clase, conjunto de clases, entre otros) diseñado.

Las métricas que se seleccionaron para evaluar la calidad del diseño 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	Aumento del TOC provoca aumento en la responsabilidad asignada a la clase
Complejidad de implementación	Aumento del TOC provoca aumento en la complejidad de implementación de la clase
Reutilización	Aumento del TOC provoca disminución del grado de reutilización de la clase

Se definieron los siguientes criterios y categorías de evaluación para los atributos de calidad anteriores:

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 de 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	Aumento del RC provoca aumento del Acoplamiento de la clase
Complejidad de mantenimiento	Aumento del RC provoca aumento en la complejidad del mantenimiento de la clase
Reutilización	Aumento del RC provoca disminución en el grado de reutilización de la clase
Cantidad de pruebas	Aumento del RC provoca aumento en la Cantidad de pruebas de unidad necesarias para probar una clase

Se definieron los siguientes criterios y categorías de evaluación para los atributos de calidad anteriores:

Tabla 4: Criterios de evaluación para la métrica RC

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de	Baja	<=Promedio

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

3.3.1 Resultados obtenidos de la aplicación de la métrica TOC

Antes de analizar los resultados obtenidos se debe tener en cuenta que la solución posee además de las clases que componen la lógica del negocio; un gran número de clases que definen cada definición de objetos BPMN en el sistema, por lo que se decidió agruparlas en lo que serán Entidades (Complex) para las definiciones complejas y Entidades (SimpleElement) para las entidades simples. Dichas entidades aunque no se comportan de la misma forma si guardan relación entre ellas, pues se implementan a través de interfaces.

Partiendo de lo anterior luego de aplicarse la métrica TOC se obtuvieron resultados que permiten evaluar el diseño propuesto de calidad aceptable según los atributos de calidad involucrados (responsabilidad, complejidad de implementación y reutilización)

Tabla 5: Instrumento de evaluación de métricas TOC

	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
Parser2	9	Alta	Alta	Media
Choice	9	Media	Media	Media
BFCase	5	Media	Baja	Media
Collection	6	Media	Baja	Alta

Complex	10	Baja	Baja	Alta
SimpleElement	4	Baja	Baja	Alta
ComplexChoice	3	Baja	Baja	Alta
SimpleChoice	3	Baja	Baja	Alta

A partir de la siguiente tabla se utilizaron gráficas de pastel para obtener una mejor apreciación de los resultados.

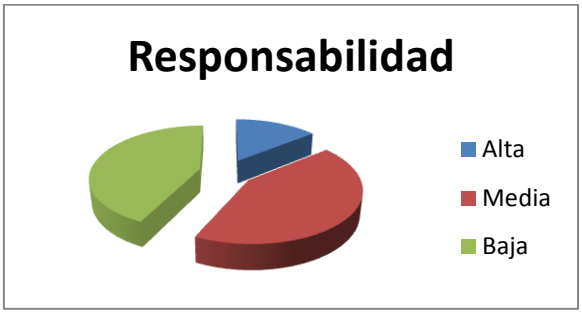

Ilustración 21: Resultados obtenidos según las Métricas TOC (responsabilidad)

Ilustración 22: Resultados obtenidos según las Métricas TOC (reutilización)

Ilustración 23: Resultados obtenidos según las Métricas TOC (complejidad)

3.3.2 Resultados obtenidos de la aplicación de la métrica RC

Luego de aplicarse la métrica RC se obtuvieron resultados que permiten evaluar el diseño propuesto de calidad aceptable de los atributos de calidad involucrados (acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización).

Tabla 6: Instrumento de evaluación de métricas RC

	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad de Mantenimiento	Reutilización	Cantidad de pruebas
Parser2	5	Alta	Alta	Media	Alta
Choice	9	Media	Media	Media	Media
BFCase	5	Media	Baja	Media	Media
Collection	18	Baja	Baja	Alta	Media
Complex	21	Media	Baja	Alta	Baja
SimpleElement	19	Media	Baja	Alta	Baja
ComplexChoice	11	Media	Baja	Alta	Baja

Ilustración 24: Resultados obtenidos según las Métricas RC (acoplamiento)

Ilustración 25: Resultados obtenidos según las Métricas RC (complejidad de mantenimiento)

Ilustración 26: Resultados obtenidos según las Métricas RC (reutilización)

Ilustración 27: Resultados obtenidos según las Métricas RC (cantidad de pruebas)

3.3.3 Matriz de inferencia de indicadores de calidad

Con la matriz inferencia de indicadores de calidad se representan los atributos de calidad y las métricas que se utilizaron para medir la calidad del diseño del componente. La matriz demuestra si los resultados de las relaciones entre atributos y métricas son positivos o negativos a partir de una escala numérica. En la escala 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 “-”. Luego se puede obtener un resultado general para cada atributo promediando todas sus relaciones no nulas.

Tabla 7: Rango de valores para la evaluación de la relación Atributo/Métrica

Categoría	Rango de valor

Malo	<=0.4
Regular	>0.4 y <0.7
Bueno	>=0.7

Tabla 8: Resultados de la evaluación de la relación Atributo/Métrica

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

Resultados obtenidos de la evaluación de los atributos de calidad.

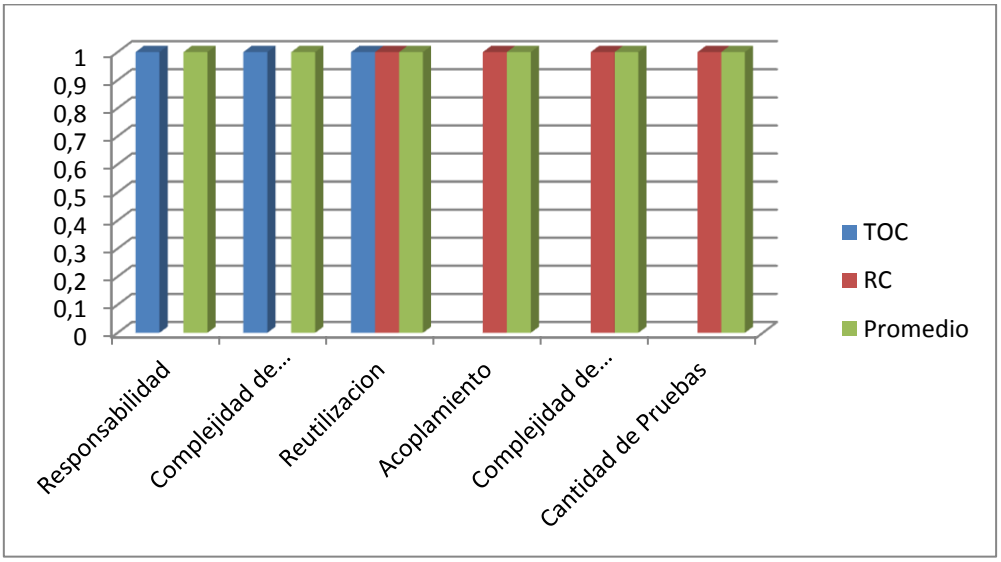

Ilustración 28: Matriz de cubrimiento

3.4 Pruebas de software

Las pruebas permiten corregir y eliminar errores en la fase final antes del producto ser liberado al cliente mediante un conjunto de actividades dentro del desarrollo de software. El objetivo de las pruebas es brindar información sobre la calidad del producto a los responsables del mismo. Para

su aplicación no existe un mecanismo específico, varía en dependencia del contexto en el cual se aplique por lo que debe estar condicionada por los elementos a los cuales está enfocada en dicho momento.

3.4.1 Pruebas estructurales o de caja blanca

Las pruebas estructurales o de caja blanca permiten diseñar casos de prueba que se centren en obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. También garantiza que en los casos de prueba obtenidos a través del camino básico se ejecute cada sentencia del programa al menos una vez. Para esto primero se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado.

```
private function assembleObject($asArray, $asObject, $objectToAssemble) {

    if($objectToAssemble instanceof SendExt_WP_WPObject_Base_ComplexChoice){
        self::fillChoiceElements($asArray, $asObject, $objectToAssemble);
    }else
    if(array_key_exists('Type', $asArray) && array_key_exists($asArray['Type'], $asArray)){
        $type = $asArray['Type'];
        $asArray = $asArray[$type];
        $asObject = $asObject->$type;
        self::assembleObject($asArray, $asObject, $objectToAssemble);
    }
    else{
        foreach ($asArray as $key -> $value) {
            if (is_array($value)) {
                $funcPrefix = 'get';
                $funcSuffix = $key;
                $fullFuncName = $funcPrefix.$funcSuffix;
                $newObject = $objectToAssemble->$fullFuncName();
                if($newObject instanceof SendExt_WP_WPObject_Base_Collections){
                    self::fillCollectionElement($value, $asObject->$key, $newObject);
                }
                else{
                    if($newObject instanceof SendExt_WP_WPObject_Base_Choice){
                        self::fillChoiceElements($value, $asObject->$key, $newObject);
                    }
                    else{
                        self::assembleObject($value, $asObject->$key, $newObject);
                    }
                }
            }
            else{
                $funcPrefix = 'set';
                $funcSuffix = $key;
                $fullFuncName = $funcPrefix.$funcSuffix;
                $newObject = $objectToAssemble->$fullFuncName($value);
            }
        }
    }
}
```

Ilustración 29: Código fuente de la funcionalidad assembleObject

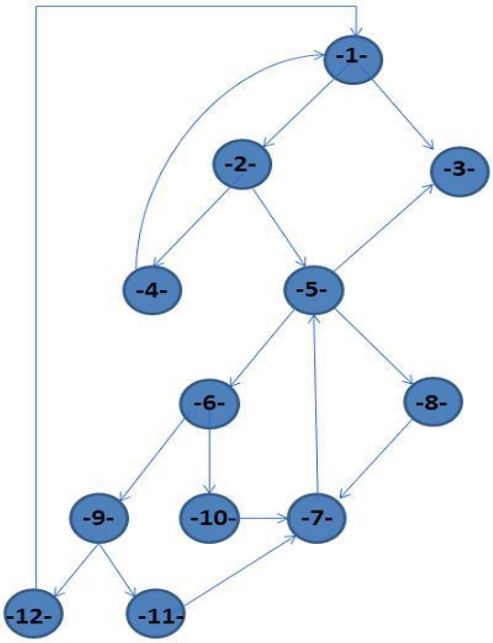

Ilustración 30: Grafo de flujo asociado a la funcionalidad assembleObject

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.

1. V (G) = (A – N) + 2

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

V (G) = (17 - 13) + 2

V (G) = 6

2. 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) = 5 + 1

V (G) = 6

3. V (G) = R

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

V (G) = 6

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, dando como resultado 6, lo que indica que existen 6 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 #1:1-3
- Camino básico #2:1-2-5-3
- Camino básico #3:1-2-5-8-7-5-3
- Camino básico #4:1-2-5-6-10-7-5-3
- Camino básico #5: 1-2-5-6-9-11-7-5-3
- Camino básico #6: 1-2-5-6-9-12-1

Para cada camino se realiza un caso de prueba.

- **Caso de prueba para el Camino básico #1**

  Descripción: los datos de entrada cumplirán con los siguientes requisitos: JSON como arreglo, JSON como objetos y objeto a ensamblar

  Condición de ejecución: el elemento a ensamblar sea una opción compleja

  Entrada: N/A

  Resultados esperados: llamada a la función encargada de los elementos de opción compleja

- **Caso de prueba para el Camino básico #2**

  Descripción: los datos de entrada cumplirán con los siguientes requisitos: JSON como arreglo, JSON como objetos y objeto a ensamblar

  Condición de ejecución: arreglo de opciones complejas, se llama a la misma función recursivamente esperando ensamblar el correspondiente con el type específico

  Entrada: N/A

  Resultados esperados: ensamblar los elementos del arreglo de entidades definidas por un type de forma recursiva

- **Caso de prueba para el Camino básico #3**

  Descripción: los datos de entrada cumplirán con los siguientes requisitos: JSON como arreglo, JSON como objetos y objeto a ensamblar

  Condición de ejecución: arreglos de elementos simples a ensamblar

  Entrada: N/A

Resultados esperados: todos los elementos simples ensamblados

- **Caso de prueba para el Camino básico #4**

  Descripción: los datos de entrada cumplirán con los siguientes requisitos: JSON como arreglo, JSON como objetos y objeto a ensamblar

  Condición de ejecución: array de elementos de una colección

  Entrada: N/A

  Resultados esperados: array de elementos de una colección creados

- **Caso de prueba para el Camino básico #5**

  Descripción: los datos de entrada cumplirán con los siguientes requisitos: JSON como arreglo, JSON como objetos y objeto a ensamblar

  Condición de ejecución: array de elemento de opción compleja a ensamblar

  Resultados esperados: array de elemento de opción compleja a ensamblados

- **Caso de prueba para el Camino básico #6**

  Descripción: los datos de entrada cumplirán con los siguientes requisitos: JSON como arreglo, JSON como objetos y objeto a ensamblar

  Condición de ejecución: elemento a ensamblar si no son una colección u opción compleja

  Entrada: N/A

  Resultados esperados: elemento a ensamblar que no son una colección u opción compleja

La realización de las pruebas de caja blanca brindaron la medida de la complejidad del diseño procedimental validando la solución satisfactoriamente pues cada camino se ejecuta al menos una vez y las decisiones lógicas se ejecutaron de manera positiva así como las estructuras internas de datos.

3.4.2 Pruebas funcionales o de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las

técnicas de prueba de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca. (29)

La prueba de caja negra intenta encontrar errores de las siguientes categorías:

- Funciones incorrecta o ausente, errores de interfaz
- Errores en estructuras de datos o en accesos a bases de datos externas
- Errores de rendimiento y errores de inicialización y de terminación (29)

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

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

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

La aplicación fue probada por el Departamento de Calidad del CEIGE (Ilustración 1 y 2 de los Anexos), donde se comprobó el correcto funcionamiento de la misma a partir de los diseño de caso de prueba (Tablas 1 – 13 de los Anexos).

3.5 Conclusiones parciales

- ✓ Podemos concluir este capítulo con resultados satisfactorios, pues la aplicación de las métricas de calidad del diseño arrojaron resultados positivos para los atributos evaluados
- ✓ Las pruebas de caja blanca y caja negra, validaron el correcto funcionamiento de la solución
- ✓ La aplicación de los estándares de codificación posibilitaron que todos los programadores de la solución trabajarán de forma coordinada

CONCLUSIONES

Con la finalización de la investigación se alcanzan las siguientes conclusiones:

- La aplicación de las métricas de calidad del diseño arrojaron resultados esperados para cada uno de los atributos evaluados
- Las pruebas estructurales y funcionales realizadas a la aplicación, validaron el correcto funcionamiento de la misma
- La Solución informática para el soporte de la transformación de modelos BPMN a objetos PHP concederá al CEIGE, contar con una herramienta que permita al usuario construir visualmente un flujo de trabajo, permitiendo que el sistema transforme la definición a su homóloga en lenguaje PHP; utilizando como notación gráfica BPMN

Todo lo anterior posiciona al Modelador de procesos de negocio, como una herramienta útil para la gestión de procesos de negocios pues permite la transformación de modelos descritos en BPMN a objetos PHP, para su posterior persistencia y validación en el marco de trabajo Sauxe.

RECOMENDACIONES

Debido a la constante evolución de los negocios y las tecnologías se recomienda:

- ✓ Utilizar el presente trabajo como material de estudio en futuras investigaciones referentes al tema
- ✓ Se recomienda que el resultado obtenido en la investigación sea implantado en el marco de trabajo Sauxe para la gestión de procesos de negocio
- ✓ También se exhorta al perfeccionamiento de la herramienta a partir de los nuevos requisitos que puedan surgir como resultado de su puesta en funcionamiento

TRABAJOS CITADOS

1. **ISO 9000.** Procesos de negocio. *Procesos de negocio.* [En línea] 2010. [Citado el: 13 de 03 de 2012.] http://arpcalidad.com/definicin-de-proceso/.

2. **Kawtar Benghazi, José Luis Garrido Bullejos.** Introducción al Modelado de Procesos de Negocio. *Introducción al Modelado de Procesos de Negocio.* [En línea] 2009. [Citado el: 14 de 02 de 2012.] http://www.ugr.es/~mnoguera/collaborative_systems-business_processes_10-11.pdf.

3. **Johansson, Henry J.** *Reingeniería de Procesos de Negocios.* s.l. : Limusa, 2002.

4. **Burlton, Roger T.** Software. *Software.* [En línea] 08 de abril de 2008. [Citado el: 04 de febrero de 2012.] http://www.softwareag.com/latam/company/latestnews/20080204_bpm_advisory_council_page.asp. .

5. *MANUAL DE PLANIFICACIÓN, SEGUIMIENTO Y EVALUACIÓN DE LOS RESULTADOS DE DESARROLLO.* New York: A.K.Office Supplies (NY), 2009.

6. **L, Carlos Ponce de León.** Modelado De Procesos De Negocio (Bpm). *Modelado De Procesos De Negocio (Bpm).* [En línea] febrero de 2011. http://www.buenastareas.com/ensayos/Modelado-De-Procesos-De-Negocio-Bpm/1581374.html.

7. **Soto, Fernando Salazar.** *Trabajo de compilación bibliográfica Auditoria sistemas.* Universidad De Caladas, Manizales: s.n., 2010.

8. **RODRIGUEZ, SANDRA MELYNA LOPEZ.** FLUJOS DE TRABAJO. *FLUJOS DE TRABAJO.* [En línea] 2009. [Citado el: 23 de abril de 2012.] http://es.scribd.com/doc/42813565/Flujo-de-trabajo.

9. Generador de Aplicaciones Orquestadoras. *Generador de Aplicaciones Orquestadoras.* [En línea] 2005. [Citado el: 12 de febrero de 2012.] http://www.willydev.net/descargas/SOA/WillyDev_PG-P2005_0026-EstadoDelArte.pdf.

10. **Septiembre, Manuel Llavador Campos.** *UN MARCO FLEXIBLE PARA LA ESPECIFICACIÓN DE PROCESOS FLEXIBLES.* Valencia : s.n., 2007.

11. **Oroya, Víctor Paúl Ávila.** *Mejora continua de procesos con BPM.* LIMA – PERÚ : s.n., 2010.

12. **Hantelmann, Alejandro Gutierrez.** *Trabajo de investigación sobre BPMN.* s.l. : MPN-GENERISHON, 2012.

13. **ADVANTYS.** Workflowgen. *Workflowgen.* [En línea] 2011. http://www.workflowgen.com/overview/.

14. **Valdés-Faura, Miguel.** BonitaSoft open your processes. *BonitaSoft open your processes.* [En línea] 2011. [Citado el: 12 de abril de 2012.] http://www.bonitasoft.com/.

15. **Bazán, Patricia.** *BPEL: una propuesta para el uso de Web Services.* La Plata, Buenos Aires, Argentina : s.n., 2008.

16. **Group, YAWL.** YAWL Leading the world in process innovation. *YAWL Leading the world in process innovation.* [En línea] 2012. [Citado el: 02 de mayo de 2012.] http://www.yawlfoundation.org/.

17. **Workflow Management Coalition.** Workflow Management Coalition. *Workflow Management Coalition.* [En línea] 2012. [Citado el: 25 de mayo de 2012.] http://www.wfmc.org/xpdl.html.

18. **PHP Group.** PHP. [En línea] 2001. [Citado el: 20 de 10 de 2010.] http://www.php.net/manual/es/intro-whatis.php.

19. **Frederick, Shea, Ramsay, Colin y Blades, Steve 'Cutter'.** *Learning Ext JS.* Birmingham : Packt Publishing Ltd, 2008. 978-1-847195-14-2 -14-2 14-.

20. **Esser, Stefan.** *Secure Programming with the Zend-Framework.* Amsterdan : s.n., 2009.

21. **Baranovskiy, Dmitry.** Raphaël—JavaScript Library. *Raphaël—JavaScript Library.* [En línea] 2011. [Citado el: 13 de enero de 2012.] http://raphaeljs.com/.

22. **Kabir, Mohammed J.** *La biblia Servidor Apache2.* s.l. : Anaya.

23. **NetBeans Community.** NetBeans. *NetBeans.* [En línea] 2005. [Citado el: 10 de mayo de 2012.] http://netbeans.org/.

24. **Prada Nicot, Héctor y Sánchez González, Kenner.** *Desarrollo de los componentes Puesto de Trabajo y Pagos Adicionales del subsistema Capital Humano integrado al sistema integral de gestión CEDRUX.* La habana : s.n., 2009.

25. **Rellum, Xela.** RapidSVN. *RapidSVN.* [En línea] noviembre de 2011. http://www.rapidsvn.org.

26. MozillaEs. [En línea] [Citado el: 18 de 01 de 2011.] http://www.mozilla-europe.org/es/firefox/features/.

27. **Larman, Craig.** *UML y Patrones.*

28. *Journal of Logic and Computation,.* **Turner., Raymond.** 5, Colchester UK : s.n., October 2005, Vol. 15.

29. **Pressman, Roger S.** *Ingeniería de Software, un enfoque práctico.* 2002.

30. **Buschmann, F., Henney, K. y Schmidt, D.** *Pattern-oriented software architecture. .* s.l. : John Wiley&Sons, Ltd., 2007.

31. **Touris, Alberto Molpeceres.** *Arquitectura empresarial y software libre, J2EE.* Alemania : s.n., 2002.

32. **Mestras, Juan Pavón.** *Estructura de las Aplicaciones Orientadas a Objetos.El patrón Modelo-Vista-Controlador (MVC).* Madrid : s.n., 2009.

33. **Ralph E. Johnson, Erich Gamma.** *Design Patterns: Elements of Reusable Object-Oriented Software.* Illinois : Addison-Wesley Professional, 1994. 0201633612.

34. **Microsoft.** Msdn Microsoft. *Msdn Microsoft Revisiones de código y estándares de codificación.* [Online] 2003. http://msdn.microsoft.com/es-es/default.aspx.

35. **Ramírez, Lic. Elisa Arizaca.** *Artefacto:Diagrama de componente.* Bolivia: s.n., 2009.

36. **PHP Company.** Zend The PHP Company. [En línea] [Citado el: 06 de 10 de 2010.] http://www.zend.com/en/community/framework.

37. *Secure Programming with the Zend-Framework.* **Esser, Stefan.** Amsterdam: s.n., 2009. Dutch PHP Conference.

38. **Fronckowiak, John.** IBM. [En línea] 01 de 07 de 2008. [Citado el: 07 de 01 de 2011.] http://www.ibm.com/developerworks/web/library/wa-aj-extjs/.

39. **Equipo Arquitectura del ERP Cedrux.** 2008.

40. **Casas Rescalvo, Esther.** *Framework de desarrollo de Código Abierto.* [En línea] [Citado el: 12 de enero de 2011.] http://www.opensourceworldconference.com/papers/Dia22/Sala%204/Casas_235.pdf.

41. **Boda, Pedro, y otros, y otros.** *Zend Framework Manual en español.* 2009.

42. Visual Paradigm. [En línea] [Citado el: 24 de 11 de 2010.] http://www.visual-paradigm.com/product/vpuml/.

43. [En línea] http://www.techtastico.com/.../zend-framework-una-introduccion/.

44. **Producción, Equipo de.** Modelo de Desarrollo orientado a componentes del proyecto ERP-CUBA.

45. **Gracia, Joaquin.** IngenieroSoftware. [En línea] 27 de mayo de 2005. [Citado el: 25 de 3 de 2011.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

46. **Marca Huallpara, Hugo Michael y Quisbert Limachi, Nancy Susana.** Universidad Salesiana de Bolivia. [En línea] [Citado el: 19 de 03 de 2011.] http://virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc.

47. **Pérez Ramírez, Oilede y García López, Yonisley.** Desarrollo del Componente Trabajador del Subsistema Capital Humano del sistema Cedrux. Habana: s.n., 2009.

48. **Pupo, Yanisleydi Cañete.** Libro de Ayuda del marco de trabajo Sauxe, En su versión 2.0. Ciudad de la Habana: s.n., 2010.

49. **Linux Group.** Programación Web. *Programación Web.* [En línea] 2010. [Citado el: 09 de febrero de 2012.] http://www.ciberaula.com/.

50. **Microsoft.** Msdn Microsoft. *Msdn Microsoft Revisiones de código y estándares de codificación.* [Online] 2003. http://msdn.microsoft.com/es-es/default.aspx.

51. **Gamma, E, et al., et al.** *Design Patterns: Elements of Reusable Object Oriented Software.* s.l. : Addison Wesley, 1995.

52. **Jacobson, Ivar, Booch, Grady and Rumbaugh, James.** *El proceso unificado de desarrollo de software.* España : Addison Wesley, 1999. 84-7829-036-2.

54. **Chapman, Alan.** Business Balls. *Business Balls.* [Online] 1999. http://www.businessballs.com/business-process-modelling.htm.

55. **León, Carlos Ponce de.** Buenas Tareas. *Universidad Autónoma de Ciudad Juárez.* [Online] Febrero 2011. http://www.buenastareas.com/ensayos/Modelado-De-Procesos-De-Negocio-Bpm/1581374.html.

56. **autores, Coleccion de.** Definicion ABC. *Definicion ABC.* [Online] 2007. [Cited: 5 25, 2012.] http://www.definicionabc.com/general/proceso.php.

57. **Mayora, Alex.** MANUAL PARTICIPANTE DE LA HERRAMIENTA INTALIO DESIGNER V6.0 PARA EL MODELADO DE PROCESOS DENEGOCIO. *Scribd.* [Online] Marzo 6, 2010. http://es.scribd.com/doc/80123331/Manual-de-Participante-de-La-Herramienta-Intal.

58. **IEEE.** *Recommended Practice for Software Requirements Specifications -Description.* 1998. 830.

59. **Visual Paradigm International .** Business Process Visual ARCHITECT 3.2 Release Notes. *Business Process Visual ARCHITECT 3.2 Release Notes.* [Online] junio 2010. [Cited: junio 02, 2012.] http://www.visual-paradigm.com/support/bpva/releasenotes/320.jsp.

60. **Fernandez, Rodrigo.** Evoluciona, Consulting Technology Outsourcing . *Evoluciona, Consulting Technology Outsourcing .* [Online] mayo 2008. http://evoluciona.cl/vinculos/que_software.html.

GLOSARIO DE TÉRMINOS

AJAX: (Asynchronous JavaScript And XML) JavaScript asíncrono y XML

BPM: (*B*usiness Process Management) Gestión de procesos de negocio

CSS: (Cascading Style Sheets) Hojas de estilo en cascada

Complex: definición de un elemento complejo para la solución, basado en la especificación de un elemento complejo definido por BPMN

DOM: (Document Object Model) Modelo de Objetos del Documento

ezComponents: marco de trabajo de código abierto, implementado en PHP

HTML: (HyperText Markup Language) Lenguaje de marcado de hipertexto

Java: es un lenguaje de programación orientado a objetos

JSON: (JavaScript Object Notation) Formato ligero para el intercambio de datos

Lane: definición de carril para la solución, basado en la especificación de un carril definido por BPMN

marco de trabajo: es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos

PHP: (PHP Hypertext Preprocessor) lenguaje de programación

Pool: definición de piscina para la solución, basado en la especificación de una piscina definida por BPMN

SimpleElement: definición de un elemento simple para la solución, basado en la especificación de un elemento simple definido por BPMN

SVG: (Scalable Vector Graphics) gráficos vectoriales escalables

WfMC: (Workflow Management Coalition) es una organización global de los adoptantes, desarrolladores, consultores, y analistas, que investigan y desarrollan sobre flujo de trabajo y BPM

Workflow: gestión electrónica de procesos de negocio

.css: extensión para ficheros de tipo de CSS

.js: extensión para ficheros de tipo JavaScript

.NET: es un framework de Microsoft para el desarrollo de aplicaciones