Universidad de las Ciencias Informáticas
Facultad 3

Título: Componente para la validación de procesos de negocios modelados con BPMN en el marco de trabajo Sauxe

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autor(es): Gloria Caridad Raventós Ladrón de Guevara.

Fidel Arnulfo Ruiz Hechavarría.

Tutor(es): Ing. Yoriangel Rivero González.

Ing. Yuniel Cedeño Mendoza.

Consultante: Ing. Magdanis Galvan Rey.

Msc. Raykenler Yzquierdo Herrera.

Ing. Javier Ruiz Duran.

Ing. Pedro Manuel Nogales Cobas

Junio del 2012

DECLARACIÓN DE AUTORÍA

Declaro a Gloria Caridad Raventós Ladrón de Guevara y Fidel Arnulfo Ruiz Hechavarría como los únicos autores de este trabajo y autorizamos al Centro de Informatización y Gestión de Entidades (CEIGE), de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____ _____

Gloria Caridad Raventós Ladrón de Guevara
Autor

Fidel Arnulfo Ruiz Hechavarría
Autor

_____ _____

Ing. Yoriangel Rivero González

Ing. Yuniel Cedeño Mendoza

DATOS DE CONTACTO

Tutores:

Ing. Yuniel Cedeño Mendoza…… Correo: ycedenom. uci.cu
Ing. Yoriangel Rivero González…….Correo: yriverog.uci.cu

RESUMEN

En la actualidad la Gestión de procesos de negocio (BPM) juega un papel fundamental en la gestión que se lleva a cabo en cualquier empresa. La misma incluye conceptos, métodos y técnicas para apoyar el análisis, el diseño y seguimiento de los procesos de negocio.[1] Actualmente los sistemas de gestión de flujos de trabajo (WFMS) son utilizados para complementar o desarrollar con más efectividad la BPM en una entidad.

En aras de mejorar la gestión de los procesos de negocio se está desarrollando un WFMS para el Marco de Trabajo Sauxe, el mismo está dividido en varios componentes, uno de ellos es una herramienta que permite el modelado de procesos de negocio utilizando la Notación para el Modelado de Procesos de Negocio (BPMN). Esta notación ofrece un conjunto de reglas que especifican cómo se deben relacionar los objetos definidos por BPMN para que los modelos estén sintácticamente bien representados. La herramienta desarrollada para Sauxe no implementa estas reglas, por lo que pudiesen existir errores sintácticos en los modelos de procesos de negocio modelados en ella.

Este trabajo tiene como objetivo desarrollar un componente que permita la validación de los modelos de procesos de negocio modelados con BPMN en el Marco de Trabajo Sauxe. Además se describen las tecnologías a utilizar y las decisiones asociadas a la determinación de una técnica de validación con el fin de disminuir los errores en los modelos de procesos de negocio modelados con el Editor de procesos de negocio de Sauxe.

PALABRAS CLAVE

Gestión de Procesos de Negocios, Sistemas de Gestión de Flujo de Trabajo, validación de los procesos de negocio, reglas sintácticas.

Índice

Índice de Figuras

Índice de Tablas.

Introducción

El vertiginoso desarrollo de las tecnologías de la información ha conllevado a un crecimiento acelerado de la competencia entre las empresas por el apoderamiento del mercado mundial. Sistemas informáticos como los Sistemas de Planificación de Recursos Empresariales, por sus siglas en inglés (ERP) y los Sistemas de Gestión de Flujos de Trabajo, por sus siglas en inglés (WFMS) son algunas de las buenas prácticas que pueden implementar las empresas para lograr adentrarse con resultados satisfactorios en el sistema económico que hoy impera.[2]

Hoy en día las empresas han adoptado la optimización de los procesos para lograr mejoras en el cumplimiento de sus metas fundamentales. Es por eso que los estudios realizados a los WFMS van encaminados hacia soluciones que permitan la implementación, automatización y seguimiento de procesos administrativos en donde se involucren documentos, información o tareas que pasen de un participante a otro(s), para la realización de acciones específicas, de acuerdo a ciertas reglas de negocio pre-establecidas.[3] Los principales campos donde se han aplicado soluciones para gestionar los procesos son: planificación de recursos empresariales, en la integración de aplicaciones empresariales, por sus siglas en inglés (EAI), en la gestión de las relaciones con los clientes, por sus siglas en inglés (CRM) y en la gestión de flujos de trabajo.[2]

A nivel mundial la principal entidad reconocida en los temas de flujo de trabajo es la Workflow Managment Coalition (WFMC) que establece como una de sus principales misiones el promover y desarrollar el uso de flujos de trabajo estableciendo estándares en el proceso de desarrollo del software, en la interoperabilidad y la conectividad entre la gestión de procesos de negocio y los WFMS.[3] Dentro de los principales estándares se encuentra la notación de modelado de procesos de negocio o Business Process Modeling Notation (BPMN)[4], la cual fue creada y definida por el Grupo de Gestión de Objetos, por sus siglas en inglés (OMG) como una notación basada en diagramas de flujos para definir procesos de negocio.[5] Esta notación define un conjunto de reglas que especifican como debe ser realizado el modelado para que el proceso esté correctamente graficado. Soluciones tales como Oracle Workflow, IBM Lotus Workflow, SAP Business Workflow, Bonita Workflow y CuteFlow son algunas de las que podemos encontrar hoy en día. Muchas de estas herramientas están protegidas bajo licencias privativas y otras están desarrolladas para ejecutarse en plataformas privadas, y herramientas implementadas sobre plataformas privadas no cumplen con las políticas de migración hacia plataformas abiertas o de software

libre por lo que no son viables en busca de posibles áreas de reutilización.[6] Las herramientas de flujo de trabajo pueden ser de escritorio o web, estas últimas presentan la mayor proliferación debido al auge que existe hacia el uso de la Arquitectura Orientada a Servicios, por sus siglas en inglés (SOA) que permite el acceso a sistemas distribuidos, logrando con la implantación de un WFMS alcanzar una explotación al máximo de las prestaciones de cada uno de los sistemas distribuidos.[2]

Los WFMS gestionan los procesos de negocios de manera tal que permiten lograr una integración departamental de los datos y las aplicaciones, ayudando en la elaboración de planes y en la toma de decisiones. Manejan los procesos de negocios de forma peculiar, en primer lugar los WFMS están centrados en un modelo que está diseñado para representar específicamente la estructura de un proceso de negocio, el mismo puede estar ya previamente definido o se necesita optimizar, una vez que se termina el modelo, se crean instancias que van realizando los pasos que están descritos en el flujo de trabajo, ya en la ejecución las instancias pueden acceder a sistemas heredados[1], es decir, que a las aplicaciones y las bases de datos les es posible interactuar con el usuario. De esta manera estos sistemas incluyen la lógica necesaria para controlar los procesos organizacionales, este tipo de sistema brinda como ventaja que las aplicaciones no necesiten ser desarrolladas según las necesidades de la empresa. Es fundamental que los sistemas heredados a los cuales se accede sean lo más modulares[2] posibles para lograr un mejor funcionamiento de un sistema de flujos de trabajo.[2]

Las investigaciones realizadas por la WFMC demostraron como una buena práctica el establecimiento de un híbrido entre los sistemas ERP y los WFMS para lograr una complementación en cuanto a las deficiencias que poseen ambos sistemas individualmente.[3] Siguiendo este resultado en Cuba se está desarrollando un sistema ERP denominado Cedrux, el cual está soportado por el marco de trabajo Sauxe y este dentro de sus prestaciones desea incluir un WFMS. Dicho sistema está separado en distintos componentes, entre los cuales se encuentra: un motor de ejecución de procesos de negocio y un modelador de procesos de negocio, que usa BPMN como notación creada específicamente para brindar la estandarización necesaria entre todos los usuarios que deseen modelar procesos de negocio. Este componente denominado Editor de procesos de negocio, en una primera versión, permite modelar los procesos, pero no posibilita conocer si los mismos son correctos o no de acuerdo a las reglas sintácticas establecidas por la notación, las cuales definen cómo deben relacionarse los objetos de BPMN, por lo que

[1] *Sistema heredado:* Es un componente de un entorno de gestión de contenido.
[2] *Modulares:* Partes independientes de un sistema informático.

pudieran ocurrir diferentes errores sintácticos en el modelo de procesos de negocio; algunos de los que se pueden mencionar son: falta de transiciones entre objetos cuando un evento de inicio comienza el proceso, utilización de un flujo de secuencia para unir un artefacto a una tarea, uso de un flujo de mensaje para conectar dos objetos de flujos dentro de un mismo proceso, entre otros. Según el documento de las especificaciones de BPMN, documento publicado por la OMG en su sitio oficial, BPMN tiene más de 100 reglas de modelado de las cuales en una primera versión se cubrirán alrededor de 70 (ver ***¡Error! No se encuentra el origen de la referencia.***), las cuales corresponderán a los objetos de flujo: eventos, tareas, nodos de decisión; a los objetos de conexión: flujo de secuencia, flujo de mensaje; y a las piscinas.[7]

En la actualidad, el Editor de procesos de negocio no cuenta con un componente que utilizando las reglas de modelado de las especificaciones de BPMN valide los diagramas, es decir, que una vez realizado el modelo, pueden encontrarse errores en él, provocando que al llegar el mismo al resto de los componentes ocurran disímiles afectaciones como: la inexistencia de transiciones entre actividades, lo que puede provocar la interrupción del flujo normal del proceso o cambiar la forma correcta en la que se debía ejecutar el proceso para un escenario específico dentro de una organización, por otra parte, haciendo uso de las restricciones que imponen las especificaciones de la notación, se pudiesen corregir de forma sintáctica algunos patrones de flujos de trabajo ejecutados por el motor de ejecución de flujos de trabajo, entre otros. Como se mencionó anteriormente, el WFMS estará desarrollado para el marco de trabajo Sauxe y esto implica, que dicho sistema deba ser adaptado específicamente a la arquitectura del mismo, ateniéndose tanto a las restricciones de seguridad para el acceso a datos como a la ejecución de los procesos modelados por determinados usuarios.

De acuerdo a la problemática descrita anteriormente surge el siguiente **Problema a resolver**: ¿Cómo disminuir los errores en los modelos de procesos de negocio modelados con BPMN en el marco de trabajo Sauxe?.

En aras de resolver el problema planteado se seleccionó como **Objeto de estudio**: la Gestión de procesos de negocio.

Para darle solución al problema que se plantea se propone como **Objetivo general**: Desarrollar un componente que permita disminuir los errores en el modelo de procesos de negocio con BPMN en el marco de trabajo Sauxe.

El objetivo general se descompone en los siguientes **Objetivos específicos**:

1. Elaborar el marco teórico de la investigación para proponer un mecanismo que permita realizar la validación de los procesos de negocio.
2. Identificar reglas de validación de procesos de negocio.
3. Implementar reglas validación de procesos de negocio.
4. Validar la solución propuesta.

Se define como **Campo de acción**: la validación de procesos de negocio en el marco de trabajo Sauxe.

La **idea a defender**: Con el desarrollo de un componente de validación de procesos de negocio, se disminuirán los errores en el modelo de procesos de negocio con BPMN en el marco de trabajo Sauxe.

Para darle cumplimiento a los objetivos específicos antes expuestos se definieron un conjunto de **Tareas de la investigación**:

1. Realización de un marco teórico acerca de las técnicas existentes para la validación de modelos de procesos de negocios basándose en la notación BPMN.
2. Propuesta de un modelo de clases a partir de la revisión y análisis del meta-modelo de datos para la definición de flujos de trabajo y cada uno de sus elementos.
3. Identificación de las reglas de validación de procesos de negocio.
4. Realización del análisis de la solución generando los artefactos definidos por el modelo de desarrollo propuesto.
5. Realización del modelo conceptual.
6. Identificación de los requisitos funcionales y no funcionales de la solución.
7. Descripción de los requisitos funcionales y no funcionales de la solución.
8. Definición del conjunto de tokens[3] de la gramática.
9. Realización del diseño de la solución generando los artefactos definidos por el modelo de desarrollo propuesto.
10. Realización de los diagramas de clases.

[3] *Tokens:* Par formado por un nombre de símbolo y un atributo opcional.

11. Realización del diagrama de componentes.
12. Definición de la gramática según las reglas de modelado de procesos definidas según las especificaciones de BPMN.
13. Implementación del Analizador Léxico para el reconocimiento de los elementos:
    a. Todos los Eventos de Inicio.
    b. Todos los Eventos Intermedio.
    c. Todos los Eventos de Fin.
    d. Todas las Tareas.
14. Implementación del Analizador Léxico para el reconocimiento de los elementos.
    a. Todos los Nodos de Decisión.
    b. Todos los Objetos de Conexión.
        - Flujo de secuencia.
        - Flujo de mensaje.
15. Implementación del Analizador Sintáctico para la validación de los procesos de negocio.
16. Implementación del mecanismo de gestión de errores.
17. Realización de los casos de pruebas del componente para la validación de los procesos de negocio.
18. Realización de las pruebas para validar la solución propuesta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

En el presente capítulo, se presenta el marco teórico de la investigación, en el mismo se definen los principales conceptos relacionados con el tema, que constituyen el eslabón primario para la realización de la solución. Además, se hace alusión a las tendencias actuales de la problemática en cuestión, así como un estudio de las herramientas, lenguajes y tecnologías utilizadas.

1.1. Proceso de negocio.

Un proceso según la definición de la ISO 9000 es un conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados.[8]

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

Un proceso de negocio es un conjunto estructurado de actividades, diseñado para producir una salida determinada o lograr un objetivo. Los procesos describen cómo es realizado el trabajo en la empresa y se caracterizan por ser observables, medibles, mejorables y repetitivos.[10]

Un proceso de negocio consiste en un conjunto de actividades que se realizan en coordinación en un entorno organizativo y técnico. Estas actividades en conjunto cumplen un objetivo de negocio. Cada proceso de negocio es aprobado por una sola organización, pero que puede interactuar con los procesos de negocio realizado por otras organizaciones.[11] A continuación en la Figura 1se modela el ejemplo de un proceso de negocio.

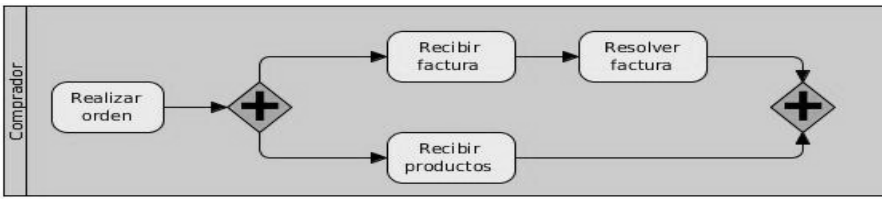

Figura 1: Ejemplo Representación gráfica de un proceso de negocio. [11].

1.1.1. Ciclo de vida de un proceso de negocio.

Según [11] en el ciclo de vida de un proceso de negocio las fases se organizan en una estructura cíclica, mostrando sus dependencias lógicas. Estas dependencias no implican un estricto orden temporal en el que las fases se ejecutarán. Muchos diseños y actividades de desarrollo se llevan a cabo durante cada una de estas fases, y enfoques graduales y evolutivos relacionados con las actividades concurrentes en múltiples fases no son infrecuentes. A continuación en la ***Figura* 2** se muestra el ciclo de vida de un proceso.

Figura 2: Ciclo de vida de un proceso de negocio. [11]

Según [11] los procesos de negocios se agrupan en cuatro clasificaciones fundamentales. A continuación se definen cada una de estas categorías.

1.1.2. Clasificación de los procesos de negocio:[11]

- Procesos de negocios organizacionales:

Los procesos de negocios organizacionales son procesos de alto nivel que normalmente se especifican en forma textual formado por sus entradas, sus salidas, sus resultados esperados, y su dependencia de otros procesos de negocio de la organización. Estos procesos de negocio actúan como los procesos del proveedor o del consumidor. Un proceso de negocio de la organización para administrar las materias primas proporcionadas por un conjunto de proveedores es un ejemplo de un proceso de negocio de la organización.

- Procesos de negocio operacionales:

Los procesos de negocio operacionales son la base para el desarrollo de procesos de negocio implementados, los cuales contienen información sobre la ejecución de las actividades del proceso y el entorno técnico y organizativo en el que se ejecutará.

- Procesos de negocio intra organizacionales:

Los procesos de negocio intra organizacionales son aquellos procesos que no tienen interacción con procesos de negocio realizados por otras partes.

- Coreografías de proceso:

Las coreografías de proceso son aquellas que ocurren cuando los procesos tienen interacción con procesos que estén en otras organizaciones. Incluyen no solo las comunicaciones en aspectos relacionados a la estructura del proceso de negocio, sino también a los asuntos legales, en cuanto a procesos de negocio que necesitan ser protegidos en la interacción entre las organizaciones.

Por los conceptos antes mencionados se puede definir un proceso de negocio como un conjunto de actividades relacionadas entre sí de forma lógica, que se llevan a cabo con el objetivo de dar cumplimiento a un objetivo del negocio. Estas actividades reciben una entrada y generan una salida con un valor agregado para una empresa o cliente. Dichos procesos pueden tener relaciones con otros dentro de una misma organización y con otros que no estén en el marco de su misma empresa. Además tienen un ciclo de vida en el cual se realiza el análisis, diseño, seguimiento e implementación. Para las empresas lograr una mejora en sus procesos de negocios pueden apoyarse en distintas técnicas las cuales pueden ser efectivas a través de la BPM.

1.2. Gestión de procesos de negocio.

La BPM, se entiende como la aplicación de técnicas para modelar, gestionar y optimizar los procesos de negocio de la organización. Partiendo de que el proceso es la forma natural de organización, el modelado de los procesos permite establecer un flujo de trabajo dentro y entre funciones, para tratar de conseguir que, con la suma de los esfuerzos funcionales, se capturen los requerimientos del negocio para obtener un mejor entendimiento y facilitar la comunicación así como identificar las mejoras en los procesos con el objetivo de conseguir los objetivos de la organización y las expectativas y requerimientos de los clientes, de una forma eficaz y eficiente.[12]

La BPM incluye conceptos, métodos, y técnicas para apoyar el diseño, administración, configuración, la promulgación y el análisis de los procesos de negocio. La base de la gestión de procesos de negocio es la representación explícita de los mismos con sus actividades y las limitaciones de ejecución entre ellos. Una vez que los procesos de negocio se definen, pueden ser objeto de análisis, mejora, y promulgación. Tradicionalmente, los procesos de negocio se promulgan de forma manual, guiado por el conocimiento del personal de la empresa y con la asistencia de las normas de organización y procedimientos que están instalados. Las empresas pueden obtener beneficios adicionales si se utilizan sistemas de software de la coordinación de las actividades involucradas en los procesos de negocio. Estos programas se denominan sistemas de gestión de procesos de negocio.[11]

Según [13] BPM se enfoca en la administración de los procesos del negocio. A través del modelado de las actividades y procesos, se puede lograr un mejor entendimiento del negocio y muchas veces se presenta la oportunidad de mejorarlos. La aplicación de BPM trae consigo una serie de beneficios para las empresas:

- Visibilidad de los procesos de las empresas.
- Mayor flexibilidad y agilidad para adaptación al cambio.
- Posibilidad de integrar la información del negocio dispersa en diferentes sistemas.
- Dirigir los esfuerzos de la empresa de una manera planeada y alineada con los objetivos estratégicos.

- Adquirir la habilidad para diseñar, simular y monitorear procesos de manera automática y sin la participación de usuarios técnicos.

- Adquirir una ruta de mejoramiento y eficiencia continua al convertir actividades ineficientes en menores costos a través de uso de tecnología enfocada en procesos.

- Reducir costos futuros de integración y mantenimiento, al adquirir tecnología ya preparada para abordar el cambio.

Después de haber analizado las definiciones asociadas a la BPM se puede concluir que esta posibilita a las empresas o entidades un conjunto de técnicas, métodos, tecnologías que apoyan el control, seguimiento, análisis y diseño de los procesos de negocio. Como una de las tecnologías asociadas a la BPM se encuentran los Sistemas de Gestión de Procesos de Negocio (BPMS) los cuales son descritos a continuación.

1.3. Sistemas de Gestión de procesos de negocio.

Un Sistema de Gestión de procesos de negocio (BPMS) posibilita la definición, administración y ejecución de los procesos de negocio como un activo corporativo, según [14] dichos sistemas proveen la tecnología que implementa uno o más de estas funciones centrales. Muchos BPMS proveen una herramienta de modelado que permite que los procesos se definan como un grafo donde los nodos representan la tarea y los arcos el flujo de control y dependencias de tareas.

Un BPMS debe proveer:[14]

- Modelado de procesos: permite capturar los requerimientos de negocio en su etapa inicial y ponerla disponible durante el resto del proceso de desarrollo.

- Ejecución de procesos: la máquina de ejecución de procesos de un BPMS importa el proceso modelado (definido usando BPEL[4]) y luego ejecuta y administra las instancias de procesos para alcanzar los requerimientos operacionales.

[4] *BPEL:* Lenguaje para la ejecución de procesos de negocio

- Monitoreo de procesos: esta capacidad incluye ver el resumen de los procesos en ejecución, de los completados, ver estados de procesos, suspender y reanudar procesos, dar alertas y reasignar procesos.

- Monitorear la actividad de negocio: (BAM Business Activity Monitoring) analiza los eventos generados por la actividad de negocio y permite dar métricas.

Como una de las funciones fundamentales de un BPMS está el Modelado de procesos de negocio el cual permite modelar un proceso. Uno vez modelado un proceso se le pueden realizar cambios para optimizarlos. A continuación se profundizará acerca del Modelado de procesos de negocio.

1.4. Modelado de procesos de negocio.

Es la representación de un proceso de negocio en una forma que permite manipulación automatizada, tales como el modelado, o la promulgación de un sistema de gestión de flujo de trabajo. La definición del proceso consiste en una red de actividades y sus relaciones, los criterios para indicar el inicio y la terminación del proceso, así como información sobre las actividades individuales como: los participantes, aplicaciones informáticas asociadas y los datos.[15]

Un modelo de procesos de negocio describe cómo funciona el negocio, es decir, describe las actividades involucradas en el negocio y la manera en que se relacionan entre ellas e interactúan con los recursos necesarios para lograr la meta del proceso. El modelado de procesos de negocios se utiliza para capturar, documentar o rediseñar procesos de negocio. Para llevarlo a cabo se pueden emplear diversos lenguajes, con diferente naturaleza, características y objetivos.[16]

Un modelo de proceso de negocio consiste en un conjunto de modelos de actividad y las limitaciones de ejecución entre ellos. Una instancia de proceso de negocios representa un caso concreto en el negocio operativo de una empresa, que consiste en las instancias de actividad. Cada modelo de procesos de negocio actúa como un modelo para un conjunto de instancias de procesos de negocio, y cada modelo de actividad actúa como un modelo para un conjunto de instancias de actividad.[11]

Según [17] existen cuatros puntos de vista en cuanto al modelado de los procesos de negocio: vista funcional, la cual representa la dependencia funcional entre los elementos del proceso; vista

dinámica, que proporciona una secuenciación y control de la información sobre el proceso; vista informacional, que incluye la descripción y relación entre las entidades que son producidas, consumidas o incluso manipuladas por los procesos y la vista organizacional que describe quien desarrolla cada tarea o función y dónde se desarrolla dentro de la organización.

Los modelos de procesos de negocio se representan mediante lenguajes de definición de procesos de negocio. Dependiendo del uso que se le pretenda dar a la definición de un proceso, el modelo debe recoger distintas características del mismo. Esto influye en gran manera en la elección de una metodología de modelado y lenguaje de definición. Existen tres categorías que representan los motivos por los cuales se modela un proceso de negocio:[18]

- Para describirlo, definirlo, comunicarlo o negociarlo.
- Para analizarlo: implica un análisis, cuantitativo o cualitativo de un proceso para, por ejemplo, decidir cambios en la planificación de actividades, cambios de asignación de tareas a empleados, incremento o reducción del grado de paralelismo.
- Para ejecutarlo: a partir de un modelo (o especificación) formal del proceso, conseguir que un sistema de software (por ejemplo, un WFMS) coordine su ejecución, o sea, se gestione automáticamente las tareas necesarias para que los procesos puedan ser llevados a cabo.

1.4.1. Requisitos de un lenguaje de modelado.

A pesar de que los requisitos exigibles a una técnica de modelado de procesos dependen de la utilización que se pretenda dar a las especificaciones, a continuación se recogen una serie de requisitos deseables en la mayoría de los casos:

- Debe permitir el modelado de situaciones complejas. Los procesos de negocio no se pueden modelar, en general, sólo con estructuras jerárquicas (árboles).
- Los modelos no pueden ser ambiguos. Para ello, resulta conveniente que posean una semántica y sintaxis formalmente definidas.

- La notación empleada debe ser fácilmente asimilable por personas. Desde este punto de vista, los lenguajes gráficos resultan más adecuados.
- Debe permitir capturar tanto los aspectos funcionales del proceso como los no funcionales.
- Debe permitir la incorporación fácil de los patrones que más frecuentemente aparecen en los procesos, tales como planificación, delegación de responsabilidades, contratos, actividades periódicas. Por ello, debe también permitir la reutilización de modelos o partes de los modelos.[18]

Después de analizar las definiciones asociadas al modelado de procesos de negocio se puede definir que el modelado posibilita la representación grafica de los procesos de negocio, de esta manera los procesos serán mas fácil de entender y manipular. Durante el modelado se pueden asociar documentos, actividades y roles que intervengan en los procesos. Para realizar la representación gráfica es necesario utilizar una notación de modelado de procesos de negocios, los cuales deben cumplir con el requisito de contar con una sintaxis formalmente definida. A continuación se describe una notación para el modelado de procesos de negocio.

1.4.2. BPMN

BPMN es una notación basada en diagramas de flujo para definir procesos de negocio, desde los más simples hasta los más complejos y sofisticados, para dar soporte a la ejecución de procesos. El principal objetivo de BPMN es proveer una notación que sea fácilmente entendible por todos los usuarios del negocio, tanto para los analistas que crean el borrador inicial de los procesos como para los técnicos y desarrolladores responsables de la implementación tecnológica.[5]

La notación BPMN contiene un conjunto de objetos representados por símbolos y a través de reglas, especifica cómo los objetos deben relacionarse. A la definición gráfica de la simbología, su debido uso por reglas bien definidas se le llama Syntax o reglas sintácticas. Al significado de los símbolos y de los patrones que con ellos se pueden modelar se le llama semántica. La primera versión de la BPMN fue desarrollada por la llamada Iniciativa de Gestión de Procesos de Negocio (BPMI) principalmente bajo la tutela de Stephan A. White profesional de la IBM en 2004. Desde un principio su objetivo principal fue proporcionar una notación gráfica, estandarizada, que permitiera automatizar los procesos a partir del

diseño gráfico. En el año 2005 fue trasladado el proyecto al Grupo de Gestión de Objetos (OMG), debido a que el BPMI no era un instituto que administraba estándares.[19]

El modelado de procesos de negocio se utiliza para comunicar una amplia variedad de información a una amplia variedad de audiencias. BPMN está diseñado para cubrir muchos tipos de modelos y permite la creación de procesos de negocio extremo a extremo. Los elementos estructurales de BPMN permiten que el espectador sea capaz de diferenciar fácilmente entre las secciones de un diagrama BPMN.[7]

Hay tres tipos básicos de sub-modelos dentro de un modelo BPMN:[7]

• Privado (interno).

• Abstracto (público).

• Colaboración (global).

BPMN define un diagrama de procesos de negocio (DPN), que se basa en una técnica de diagramas de flujo adaptado para la creación de modelos gráficos y de esa manera poder representar las operaciones de un proceso de negocio.[20]

Un DPN se compone de un conjunto de elementos gráficos. Estos elementos permiten el fácil desarrollo de diagramas simples que le resultará familiar a la mayoría de los analistas de negocio (por ejemplo, un diagrama de flujo). Los elementos fueron elegidos para ser distinguibles entre sí y para utilizar las formas que son familiares para la mayoría de los modeladores. Por ejemplo, las actividades son de forma rectangular y las decisiones son los diamantes. Cabe destacar que uno de los controladores para el desarrollo de BPMN es crear un mecanismo simple para la creación de modelos de procesos de negocio y al mismo tiempo manejar la complejidad inherente a los procesos de negocio. El enfoque adoptado para manejar estos dos conflictivos requisitos, fue organizar los aspectos gráficos de la notación en categorías específicas. BPMN v1.2 agrupa sus elementos gráficos en 4 categorías básicas: Objetos de flujo, Objetos de conexión, Artefactos y Contenedores. Esta clasificación proporciona al lector de un DPN poder reconocer fácilmente los tipos básicos de elementos y entender el diagrama. En la ***Figura* 3** se pueden apreciar cada categoría asociada a sus elementos correspondientes:[20]

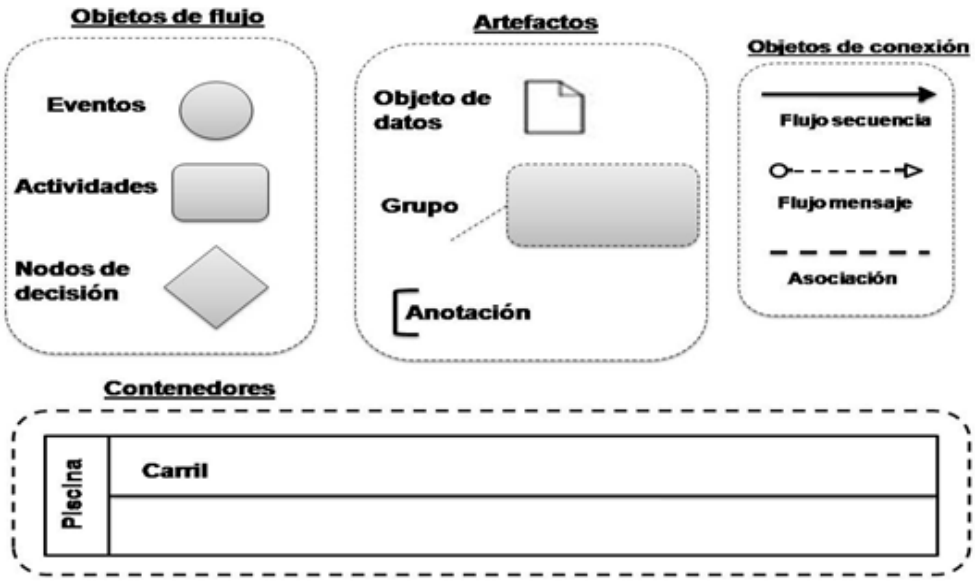

Figura 3: Categorías de los elementos en BPMN. [11]

El modelado de procesos de negocio como se menciona anteriormente es una de las fases del ciclo de vida de un proceso de negocio, la cual además es una etapa fundamental para iniciar la automatización de procesos de negocio, lo cual es posible a través de la utilización de los Flujos de trabajo.

1.5. Flujos de trabajo.

Un flujo de trabajo se define como la automatización completa o parcial de un proceso de negocio, que durante su ejecución transmite los documentos, información o tareas de un participante a otro para la acción, de acuerdo con un conjunto de reglas de procedimentales.[15]

En un workflow o flujo de trabajo, se estructuran las actividades que deben ser realizadas en un proceso de negocio, la realización de ellas y el orden en que deben realizarse, así como la sincronización entre ellas, el flujo de la información que les da soporte y el seguimiento de su cumplimento. En otras palabras, describe los posibles caminos por los cuales pueden transitar los procesos de una empresa, definiendo las reglas para tomar alguno de ellos y las acciones necesarias para transitarlos.[9, 21]

Después de analizar las definiciones asociadas a los flujos de trabajo se puede concluir que estos permiten la automatización de un conjunto de tareas o actividades, las cuales son ejecutadas de acuerdo

a un orden preestablecido, donde intervienen los roles que las realizan y los artefactos que interactúan en el proceso de negocio. Para que los flujos de trabajo puedan ser ejecutados es necesario contar con sistemas para poder alcanzar esta meta y son los WFMS los encargados de realizar esta acción.

1.6. Sistemas de gestión de flujos de trabajo.

Un WFMS define, crea y gestiona la ejecución de flujos de trabajo mediante el uso de software que se ejecuta en uno o más motores de flujo de trabajo, que es capaz de interpretar la definición del proceso, interactuar con los participantes del flujo de trabajo y, en su caso, invocar el uso de herramientas informáticas y aplicaciones.[15]

Un WFMS es un software genérico que es impulsado por las representaciones explícitas de proceso para coordinar la promulgación de los procesos de negocio.[11]

Según [22] se conoce como un WFMS a la automatización de procesos de negocio, este tiene como objetivo el apoyo a una determinada ruta de actividades realizadas en una empresa, para que estas se ejecuten de manera eficiente, en el momento requerido y por la persona específica que tiene la responsabilidad de ejecutarlas.

Según los conceptos mencionados con anterioridad se puede definir un WFMS como la automatización de los procesos de negocios, sistema en el cual es posible gestionar y ejecutar flujos de trabajo.

1.7. Tendencias Actuales

En la actualidad existen diversas herramientas que posibilitan modelar y definir los procesos de negocios de una organización apoyándose en distintas estrategias y metodologías. A continuación se muestran las características de algunas que utilizan como notación gráfica BPMN para modelar los procesos de negocios.

Together WorkFlow Editor.[23]

- Hace uso del lenguaje XPDL[5] para almacenar la definición de sus procesos.
- Posee un editor gráfico que permite diagramar el flujo de un proceso a través de un conjunto de herramientas de diagramado que están basadas en la notación gráfica BPMN
- Mantiene la información de un proceso a través de ventanas de diálogo y/o formularios.
- Genera un archivo XML con la definición de los procesos creados por los usuarios.
- Realiza una validación gráfica de los procesos, basándose en las reglas establecidas por los patrones de procesos.
- Permite importar la definición de un proceso a partir de un archivo XML que cumpla con el estándar definido por el lenguaje XPDL. Puede compartir fácilmente la definición de los procesos generados con otras herramientas afines al lenguaje utilizado.
- Tiene Licencia Pública General (GPL).

ObjectWeb Bonita.[23]

- 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. Estas deben valerse de 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.
- Permite importar la definición de un proceso a partir de un archivo XML que cumpla con el estándar definido por el lenguaje XPDL.

Bizagi.[11]

- Es un modelador de procesos que permite representar de forma esquemática todas las actividades y decisiones que se toman en el negocio.
- Cumple con el estándar BPMN.
- Posee un editor gráfico para definir reglas de asignación de trabajo. Incluye algoritmos de optimización de carga balanceada y maneja delegados y calendarios de trabajo.

[5] XPDL: Lenguaje de definición de procesos XML

- Admite seguimiento y monitoreo gráfico en tiempo real (quién hace qué, cuándo y en dónde).
- Permite la conectividad con aplicaciones como SAP[6], Documentum[7], Sharepoint[8] y correo electrónico.
- El uso del software es gratis.

Microsoft Visio. [24]

- Permite conectar los diagramas con más de una fuente de datos incluyendo Microsoft Excel, Microsoft Access, Microsoft SQL Server, SharePoint Foundation, OLEDB [9](Object Linking and Embedding Database) o ODBC [10](Open Database Connectivity).
- El programa está desarrollado para sistema operativo Windows.
- Usa BPMN como notación para el modelado de los procesos.
- Posee una licencia de tipo comercial.
- Cuando se ejecuta la validación, Visio comprueba el diagrama contra las reglas que describen procesos bien formados, y presenta una lista de todas las cuestiones para que los revise. Cuatro plantillas incluyen los conjuntos de reglas por defecto:
    - Diagrama de flujo básico.
    - Diagrama de flujo de funciones cruzadas.
    - Diagrama de flujo de trabajo de Microsoft SharePoint.
    - Diagrama de la notación de modelado procesos de negocio (BPMN).

A continuación se muestra una tabla con los principales aspectos de cada herramienta analizada:

Herramientas	Lenguaje de	Lenguaje de	Plataforma	Tipo de	Sistema Operativo

[6] *SAP:* Proviene de: Sistemas, Aplicaciones y Productos en Procesamiento de datos. Este sistema comprende muchos módulos completamente integrados, que abarca prácticamente todos los aspectos de la administración empresarial. Cada módulo realiza una función diferente, pero esta diseñado para trabajar con otros módulos.
[7] *Documentum*: Es una plataforma de administración de contenido. que tiene como objetivo ayudar a las organizaciones a administrar y explotar de manera eficiente y segura la información no estructurada generada por diversas fuentes y en distintos formatos.
[8] *Sharepoint:* Es un software que se utiliza para crear portales web internos (Intranet) para el intercambio y búsqueda de documentos, colaboración en equipo, blogs, wikis y noticias en una entidad.
[9] *OLEDB:* "Enlace e incrustación de objetos para bases de datos" es una tecnología desarrollada por Microsoft usada para tener acceso a diferentes fuentes de información, o bases de datos, de manera uniforme.
[10] *ODBC:* Estándar de acceso a bases de datos.

	modelado	proceso		licencia	
Together WorkFlow Editor	BPMN	XPDL	Stand Alone	Comercial	Windows
ObjectWeb Bonita	BPMN	XPDL	Web	GNU (LGPL)	Multiplataforma
Microsoft Office Visio	BPMN		Stand Alone	Comercial	Windows
Bizagi	BPMN	XPDL	Stand Alone	Gratis	Win2000/XP/2003/Vista/7

Tabla 1: Comparación entre distintas herramientas de modelado de procesos de negocio.

Las herramientas analizadas se basan en las especificaciones de BPMN para realizar la validación de los procesos de negocio que en ellas son modelados, pero en la documentación disponible no se expone información sobre el mecanismo que emplean para ejecutar la validación de los DPN.

Por lo planteado anteriormente se hace necesario realizar una investigación para identificar algún mecanismo que pueda ser adaptado para realizar la validación de los procesos de negocio modelados con BPMN.

1.8. Mecanismos propuestos.

A continuación se describen dos mecanismos que pueden ser transformados para realizar la validación a partir de un conjunto de reglas.

Gramática.

Una gramática es un juego de reglas formalizadas con precisión matemática que genera, sin necesidad de ninguna otra información que no esté representada en el sistema, las oraciones gramaticales del lenguaje que describe o caracteriza y asigna a cada oración una descripción estructural o análisis gramatical.[25]

Ventajas significativas que ofrecen las gramáticas:[26]

- Una gramática brinda una especificación sintáctica precisa y fácil de entender de un lenguaje de programación.

- A partir de algunas clases de gramáticas se puede construir automáticamente un analizador sintáctico eficiente que determine si un programa fuente está sintácticamente bien formado.

- El proceso de construcción del analizador sintáctico puede revelar ambigüedades sintácticas y otras construcciones difíciles de analizar, que de otro modo podrían pasar sin detectar en la fase inicial de diseño de un lenguaje y de su compilador.

- Una gramática diseñada adecuadamente imparte una estructura a un lenguaje de programación útil para la traducción de programas fuente a código objeto correcto y para la detección de errores. Existen herramientas para convertir descripciones de traducciones basadas en gramáticas en programas operativos.

- Los lenguajes evolucionan con el tiempo, adquiriendo nuevas construcciones y realizando tareas adicionales. Estas nuevas construcciones se pueden añadir con más facilidad a un lenguaje cuando existe una aplicación basada en la descripción gramatical del lenguaje.

En sentido general la gramática es la descripción de lo que caracteriza exactamente a un código, es necesaria para poder escribir un programa que determine si ese código es válido o no. El análisis gramatical o análisis sintáctico es la tarea de determinar la sintaxis o estructura de un programa. La sintaxis de un lenguaje de programación por lo regular se determina mediante las reglas gramaticales de una gramática libre de contexto, de manera similar como se determina mediante expresiones regulares la estructura léxica de los tokens reconocida como analizador léxico.[26]

Analizador léxico o scanner.

El analizador léxico es la primera fase de un compilador. Su función principal es leer los caracteres de entrada y producir como salida una secuencia de componente léxicos (tokens) que el parser luego utiliza en el proceso de análisis sintáctico.[27]

Tokens.

Un token es un par formado por un nombre de símbolo y un atributo opcional valor. El nombre del símbolo es abstracto y representa una especie de unidad léxica, por ejemplo, una palabra clave concreta,

o una secuencia de caracteres de entrada que denota un identificador. Los tokens son los símbolos de entrada del analizador sintáctico.[27]

Analizador sintáctico o parser.

La fase de análisis sintáctico resuelve la problemática de, a partir de una cadena fuente (programa escrito en un lenguaje de programación) y una gramática que genera cadenas válidas en el lenguaje, determinar si el programa (la cadena) está escrito correctamente desde el punto de vista sintáctico (puede ser generado por la gramática).[27]

1.8.1. Sistemas basados en reglas (SBR).

Arquitecturas basadas en reglas.

Hay dos estructuras básicas para organizar el conocimiento de un SBR: redes de inferencia y sistemas de pattern-matching.[19]

a) Redes de inferencia.

Una red de inferencia es un gráfico, en el cual los nodos representan parámetros que son datos iniciales o valores inferidos. Cada parámetro mide algún aspecto del problema que se considera y puede servir como consecuente o antecedente de una regla. Cada parámetro puede tener uno o más valores asociados con su correspondiente medida de incertidumbre. [19]

Las reglas en el sistema constituyen los enlaces en el gráfico. Este conocimiento es usado por el proceso de inferencia para programar los resultados a través de la red. Conocer todas las conexiones entre nodos de la red antes de la ejecución se minimiza la búsqueda de las reglas a aplicar en cada momento del proceso de inferencia. [19]

Ejemplo: [19]

Sean las reglas:

R1: IF la temperatura ambiente está alrededor 900 F THEN el tiempo está caluroso.

R2: IF la humedad relativa es mayor que 65% THEN la atmósfera está humedad.

R3: IF el tiempo está caluroso y la atmósfera está humedad THEN las tormentas son más probables que ocurran

La red correspondiente es la mostrada en la ***Figura* 4.**

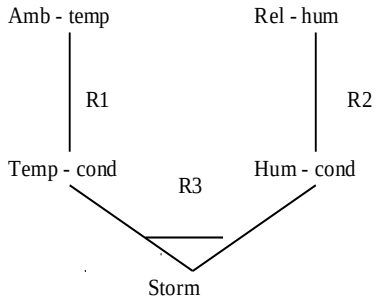

Figura 4: Red de Inferencia. [19].

Los sistemas MYCIN y PROSPECTOR son ejemplos de grandes SBR que usaron exitosamente las redes de inferencia.

b) Sistemas de pattern-matching.

Estos sistemas realizan la búsqueda extensiva para "machear'' y ejecutar las reglas, derivando nuevos hechos. Las relaciones entre las reglas y los hechos se forman en tiempo de ejecución. Un sistema de pattern-matching depende del matching (compatibilidad) de las premisas de una regla con los hechos existentes para determinar cuáles reglas tienen sus premisas satisfechas por los hechos y, por eso, pueden ser ejecutadas. Las premisas de una regla son patrones, estos usualmente se describen mediante expresiones lógicas complejas.[19]

Este esquema para determinar las reglas que se satisfacen con la información contenida en la base de datos, al requerir resolver problemas muy interesantes será necesario usar una gran cantidad de reglas. Explorar toda la BC resultará muy ineficiente. Si tenemos R reglas en la BC, f hechos en la base de datos y un promedio de *p* premisas en cada regla, serán necesarios Rxf^P comparaciones en cada ciclo para determinar cuáles reglas pueden ser ejecutadas; para 150 reglas, 20 hechos y 4 premisas se tendrían que realizar $150x20^4$ = 24,000,000 comparaciones en cada ciclo.[19]

Para enfrentar este problema la máquina de inferencia puede considerar dos términos: el indexamiento y el algoritmo RETE.[19]

La técnica de indexamiento consiste en usar el estado actual como un índice a las reglas. Desafortunadamente este esquema de indexamiento sólo trabaja cuando las precondiciones de las reglas y las configuraciones de la base de datos machean exactamente (se restringe a solo usar como operador relacional el =). Otra forma de indexamiento es el usado por PROLOG y otros demostradores de teoremas, en los cuales las reglas se indexan por los predicados que ellos contienen.[19]

El algoritmo RETE se basa en lo siguiente: Típicamente, una cantidad muy limitada de cambios ocurren en las bases de datos cuando se ejecuta una regla. Luego, las mismas que pueden ser potencialmente ejecutadas en un ciclo, cambian muy poco en el siguiente. En lugar de comparar las reglas a los hechos para ver cuáles reglas se satisfacen, se mantiene una lista de las reglas satisfechas y es necesario determinar cómo esta cambia debido a la adición o eliminación de hechos. Este algoritmo también tiene en cuenta que diferentes reglas pueden compartir un gran número de precondiciones, por ejemplo: [19]

IF mammal (x), feline (x), carnivorous (x), y has-spots (x) THEN jaguar (x)

IF mammal (x), feline (x), carnivorous (x), y has-stripes (x) THEN tiger (x)

Este algoritmo almacena las reglas de modo que ellas comparten estructuras en la memoria; las condiciones que aparecen en varias reglas se chequean una vez por ciclo.[19]

Evaluación de las arquitecturas.[19]

Los SBR que usan pattern-matching son extremadamente flexibles y poderosos. Ellos son más aplicables a dominios donde hay muchas soluciones posibles. En estos dominios (diseños, planificación) no hay relaciones predeterminadas entre las reglas y los hechos. Shells como OPS-5, ART y KEE usan esta arquitectura.[19]

Por otra parte, las redes de inferencia son bastante útiles para dominios donde la cantidad de alternativas diferentes es limitada, como clasificación y diagnóstico.[19]

Una red de inferencia es más fácil de implementar pero menos poderosa. Otra ventaja de esta arquitectura es que es más fácil explicar la solución encontrada. Es más eficiente porque las relaciones entre las reglas y los parámetros se conocen desde antes de la ejecución.[19]

1.9. Modelo de desarrollo

Según las políticas del Centro de Informatización de la Gestión de Entidades (CEIGE) el modelo de por el cual se debe guiar el proceso de desarrollo de software es el definido por el equipo de producción del centro.

El modelo de desarrollo define un conjunto de actividades, acciones, tareas, fundamentos y productos de trabajo que se requieren para desarrollar software de alta calidad. Debe llenar cada actividad del marco de trabajo con un acumulado de acciones de ingeniería de software, y definir cada acción en cuanto a un grupo de tareas que identifique el trabajo que deben completarse para alcanzar las metas de desarrollo.[28]

1.9.1. Modelo de desarrollo propuesto

Este modelo está orientado a la reutilización de componentes parametrizables, donde se simplifican las actividades y se eleva el nivel de especialización de los involucrados. Proporciona una guía para regir el proceso de desarrollo de software tecnológico, centrado en la arquitectura.[28]

Dicho modelo está basado en principios, prácticas propuestas y algunos elementos de las metodologías SCRUM y RUP, fue elaborado teniendo en cuenta las características especiales que presenta la UCI.[28]

En general propone una solución sencilla y novedosa, que se centra en el desarrollo de componentes como base tecnológica, con una mayor calidad y en menor tiempo, para su posterior uso en la construcción de productos concretos; además propone dividir el trabajo y el equipo de desarrollo para lograr mayor especialización. Su buena aplicación proporcionará en gran medida la independencia tecnológica de los sistemas finales.[28]

Características. [28]

- **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.

- **Orientado a 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.

- **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.

- **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.

Luego de haber examinado las principales características del modelo de desarrollo propuesto por el centro se analizarán los artefactos que serán necesarios generar para darle solución al problema planteado. A continuación se analiza el modelo conceptual, requisitos funcionales, diagrama de clases, patrones de diseño, estándares de codificación, diagrama de componentes, diagrama de despliegue, las métricas para validar el diseño y las pruebas que se emplearan para validar el software.

1.10. Modelo conceptual.

El modelo conceptual es una representación gráfica de los principales conceptos en el dominio del problema, muestra las relaciones entre estos conceptos y los atributos de la solución.[29]

1.11. Requisitos de software.

Los requisitos cumplen un papel primordial en el proceso de producción de software, ya que se enfocan en un área fundamental: la definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento del sistema; de esta manera, se pretende minimizar los problemas relacionados al desarrollo de sistemas.[29]

Los requisitos de software se clasifican en funcionales y no funcionales. Los requisitos funcionales describen qué es lo que el sistema debe hacer para dar soporte a las funciones y objetivos del usuario. Los requisitos no funcionales imponen restricciones de cómo los requisitos funcionales deben ser implementados.[29]

Según [30] algunas de las técnicas de captura de requisitos son:

Entrevistas: la técnica consiste en la realización de preguntas relacionadas con varios aspectos del sistema. La misma cuenta con tres fases: preparación, realización y análisis.

Tormenta de ideas es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios.

Casos de Uso: un caso es la descripción de una secuencia de interacciones entre el sistema y uno o más actores en la que se considera al sistema como una caja negra y en la que la que los actores obtienen resultados observables. Los casos de uso presentan ciertas ventajas sobre la descripción meramente textual de los requisitos funcionales, ya que facilitan la elicitación de requisitos y son fácilmente comprensibles por los clientes y usuarios.

Prototipo: Es un modelo del software que debe ser construido: se capturan requerimientos, se definen objetivos globales, se diseña rápidamente centrándose en el parte visible al usuario, se construye el prototipo y los clientes y usuarios evalúan el prototipo refinando los requerimientos.

Una vez realizado el modelo conceptual y descrito lo que debe realizar el sistema, a través de los requisitos funcionales, se procede al modelado del diagrama de clases, el cual utiliza algunos conceptos presentes en el modelo conceptual que estén más enfocados a lo que el sistema debe ser capaz de hacer.

1.12. Diagrama de clases.

El diagrama de clase describe la estructura del sistema, mostrando las clases, relaciones entre clases, atributos, métodos y asociaciones.[29, 31, 32].

Una vez realizado el diagrama de clases se pueden determinar los patrones de diseño que pueden utilizarse para darle solución al problema.

1.13. Patrones de diseño.

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Identifican: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades.[22, 33, 34]

Objetivos de los Patrones:

Proporcionar catálogos de elementos reusables en el diseño de sistemas software.

Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.

Formalizar un vocabulario común entre diseñadores.

Estandarizar el modo en que se realiza el diseño.

Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Asimismo, no pretenden:

Imponer ciertas alternativas de diseño frente a otras.

Eliminar la creatividad inherente al proceso de diseño.

El libro Design Patterns escrito por el grupo **Gang of Four (GoF)** compuesto por Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides, recoge 23 patrones diseño comunes , clasificados en:

1. Creacionales.

2. Estructurales.

3. De comportamiento.

Los patrones de diseño hacen más fácil reutilizar con éxito los diseños de clases, hacen a un sistema reutilizable y evitan alternativas que comprometen la reutilización. Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software.[22]

Otra clasificación para los patrones según [33, 34] son:

Patrones GRASP: Asignar correctamente las responsabilidades es muy importante en el diseño orientado a objetos. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Dentro de este grupo de patrones se pueden encontrar los siguientes: Experto, Creador, Bajo Acoplamiento, Alta Cohesión, Controlador, Fabricación Pura, Indirección, Variaciones Protegidas, No hables con extraños y Polimorfismo.

Una vez escogidos los patrones a utilizar se procede a la implementación, en la que hay que tener en cuenta los estándares de codificación.

1.14. Estándares de codificación.

Se define como estándares de codificación a un estilo de programación en un proyecto determinado, permitiendo que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenido. Debido a la gran cantidad de personal involucrado en la implementación del sistema Cedrux, su complejidad y el alto nivel de integración, el grupo arquitectónico del proyecto definió normas de codificación con el fin de obtener un estándar en la implementación por el equipo de desarrollo que permitiera asegurar la calidad del software y de esta forma obtener un código más legible y reutilizable.[29]

1.14.1. CamelCasing.

El estándar CamelCasing es parecido al PascalCasing con la particularidad de que la letra inicial del identificador no comienza con mayúscula. Esta notación se utilizó para el nombre de funciones y atributos.[29, 35]

1.14.2. Notación húngara.

Esta convención, también conocida como notación: REDDICK por el nombre de su creador, se basa en definir prefijos para cada tipo de datos según el ámbito de las variables con el fin de brindar mayor información al nombre de la variable, método o función. La notación húngara fue utilizada para la definición de variables de acuerdo con los siguientes prefijos que se muestran en la Tabla 2:[29, 35]

Tipo de datos	Prefijos
Arreglos	arr
Objetos	obj
Enteros	int
Cadena	str
Float	flt
Boolean	bool

Tabla 2: Prefijos para la creación de variables. [28]

Después de analizados los estándares de codificación se puede proceder a realizar el diagrama de componentes.

1.15. Diagrama de componentes.

Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento de los servicios que estos componentes proporcionan y utilizan a través de las interfaces.[28[31, 36]

Otro de los artefactos que deben generarse es el del diagrama de despliegue.

1.16. Diagrama de despliegue.

Un diagrama de despliegue es un tipo de diagrama UML con el cual queda representado un modelado del hardware utilizado en la elaboración del software. En la **Figura 13** se describen los dispositivos necesarios para la utilización del producto, es decir, desde dónde se encuentra instalado, dónde se puede utilizar y 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. [28, [31, 36]

Una vez realizado los diagramas correspondientes se procede a validar el diseño propuesto haciendo uso de las métricas que a continuación se describen.

1.17. Métricas de software.

En el libro "Ingeniería del software, un enfoque práctico", Pressman plantea: "El proceso del software y las métricas del producto son una medida cuantitativa que permite a la gente del software tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software".[36]

Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad[11]:

- **Responsabilidad:** Se le asigna a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación:** Grado de dificultad en la implementación de un diseño de clases determinado.
- **Reutilización:** Nivel de reutilización que tiene una clase o estructura de clase, dentro de un diseño de software determinado.
- **Acoplamiento:** Valor de dependencia de una clase o estructura de clase con otras. Este atributo está muy ligado al de Reutilización.

[11] *Atributo de calidad:* Es una medida cuantitativa, el grado en que un sistema, componente o proceso, posee dicho atributo.

- **Complejidad del mantenimiento:** Categoría de esfuerzo para realizar un arreglo, mejora o rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- **Cantidad de pruebas:** Número de esfuerzos para realizar las pruebas de calidad (Unidad) del producto (componente, clase, conjunto de clases) diseñado.

Debido a que este software se realizó bajo la programación orientada a objetos (POO) y las clases constituyen la unidad básica y fundamental de un sistema orientado a objetos, es indiscutible que la validación del mismo se centró en la aplicación de métricas dirigidas a sus clases de forma individual, sus jerarquías y colaboraciones.[29] A continuación se encuentra la evaluación desarrollada a las métricas Tamaño operacional de clase y Relaciones entre clases.

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

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

Tabla 3: Tamaño operacional de clase (TOC).[29]

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

Atributos	Categorías	Criterios
Responsabilidad.	Baja	<=Promedio
	Media	Entre promedio y 2 * Promedio

	Alta	>2*Promedio
Complejidad implementación.	Baja	<=Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	>2*Promedio
Reutilización.	Baja	>2*Promedio
	Media	Entre Promedio y 2 * Promedio
	Alta	<=Promedio

Tabla 4: Rango de valores para los criterios de evaluación de la métrica Tamaño Operacional de Clase (TOC).[29]

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:

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

Tabla 5: Atributos de calidad evaluados por la métrica RC.[29]

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

Atributos	Categorías	Criterios
Acoplamiento.	Ninguna	0
	Baja	1
	Media	2
	Alta	>2

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

Tabla 6: Criterios de evaluación para la métrica RC.[29]

1.18. Matriz de cubrimiento o matriz de inferencia de indicadores de calidad.

La matriz de cubrimiento o matriz inferencia de indicadores de calidad es una representación estructurada de los atributos de calidad y métricas utilizadas en los epígrafes anteriores para evaluar la calidad del diseño de los componentes que integran la solución propuesta. La misma permite conocer si el resultado obtenido de la relación atributos/métricas para cada componente es positivo o negativo. Llevando estos resultados a una escala numérica donde, si los resultados son positivos tendrá un valor de 1, si son negativos de 0 y si no existe relación alguna se tomará como nula (-). Una vez completado los datos de dicha relación se realiza un cálculo donde se promedia la sumatoria de los valores obtenidos de un atributo por cada métrica evaluada, y la división de dicha sumatoria por la cantidad de métricas evaluadas (solo se promedian las que arrojan un resultado, las nulas no). Este valor es el que va a tener el atributo dentro de una tabla que medirá si los atributos fueron buenos, regulares o malos.[29]

Categorías	Rango de valor
Malo	<=0.4
Regular	>0.4 y <0.7
Bueno	>=0.7

Tabla 7: Rango de valores para la evaluación de la relación Atributo/Métrica. [29]

1.19. Pruebas de software.

Las pruebas de software forman parte de una etapa imprescindible durante el proceso de desarrollo del software, pues permiten detectar y corregir el máximo de errores posibles antes de ser liberado el producto final. Por lo que el éxito de las mismas incide directamente sobre la percepción de calidad del usuario final.[36] [29]

1.19.1. 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 primeramente se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado.[29, 37]

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Para calcular la complejidad ciclomática se utilizan tres fórmulas que se describen a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad sea correcto.[29, 37]

Fórmula 1. V (G) = (A – N) + 2.

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

Fórmula 2. V (G) = P + 1.

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

Fórmula 3. V (G) = R.

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

Para cada camino se realiza un caso de prueba, y es preciso cumplir con las siguientes exigencias:[29]

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

1.19.2. Pruebas de rendimiento.

Según la IEEE: “Las pruebas de rendimiento es el grado en que un sistema o componente realiza sus funciones designadas dentro de las limitaciones dadas, tales como la velocidad, precisión o el uso de la memoria”.[38]

Existen diferentes tipos de pruebas de rendimiento que ayudarán a mejorar las capacidades de una aplicación observando y evaluando las respuestas del sistema ante todas las posibles situaciones que se puedan dar. Cada una de estas pruebas tiene sus objetivos y características.[38] A continuación se describen dos de estos tipos de prueba.

Las pruebas de estrés son utilizadas normalmente para someter a la aplicación al límite de su funcionamiento mediante la ejecución de un número de usuarios muy superior al esperado. Esta prueba tiene como finalidad el determinar la robustez de una aplicación cuando la carga es extrema.[38]

Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. El resultado de esta prueba mostrará el tiempo de respuesta de todas las transacciones críticas.[38]

1.20. Tecnologías.

Debido a que este trabajo forma parte de un proceso productivo iniciado en el CEIGE las tecnologías que ahora serán descritas han sido definidas y adoptadas por el grupo de desarrolladores del mismo, de acuerdo con las características del software.[29]

1.20.1. Lenguajes de programación.

- **PHP5 5.3.3.**

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y el lenguaje está orientado al desarrollo de aplicaciones web, que son interpretadas del lado del servidor. Sus sintaxis son muy similares a lenguajes como C y PERL. [39]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.[40]

- **JavaScript.**

JavaScript es un lenguaje de scripting basado en objetos, que se utiliza principalmente para crear páginas web dinámicas y permite el desarrollo de interfaces de usuario mejoradas. Una página web dinámica es aquella que permite la interacción entre el contenido de la misma y el usuario. JavaScript permite incorporar a dichas páginas efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. 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.[41]

1.20.2. Librerías y marcos de trabajo.

El desarrollo de la solución se realizará utilizando el marco de trabajo Sauxe, desarrollado por el Departamento de Tecnología del CEIGE, el cual contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo.[42]

Sauxe está compuesto por varios marcos de trabajo, los cuales serán descritos a continuación, estructurados en niveles o capas como se puede apreciar en la siguiente figura.

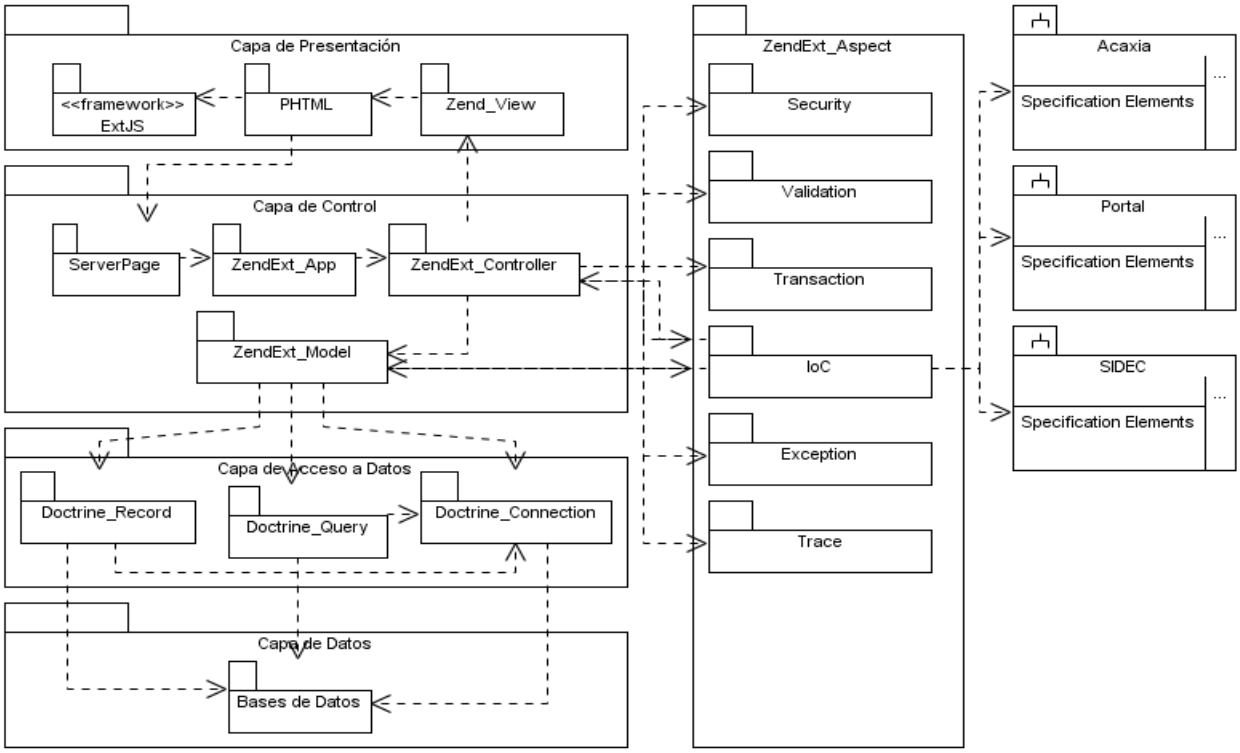

Figura 5: Estructura Del Marco de Trabajo Sauxe. [42]

- **Zend Framework 1.9.5.**

Zend Framework no es más que un framework MVC (Modelo-Vista-Controlador) open-source 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. Cuenta con clientes para el acceso a WS y robustas clases para la autenticación y el filtrado de entrada; completa documentación y tests de alta calidad.[43]

- **Doctrine 0.11.**

Doctrine es un potente y completo sistema ORM[12] para PHP con un DBAL[13] incorporado que permite exportar una base de datos a sus clases correspondientes y viceversa, o sea, a partir de las clases creadas y siguiendo las especificaciones de ORM, generar las tablas de la base de datos. Una de sus principales características es la opción de escribir las consultas de base de datos en un objeto con una propiedad orientada al dialecto SQL llamada Doctrine Query Language (DQL), inspirada en Hibernate HQL. Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria.[44]

- **ExtJS 3.2.**

Es una librería Java Script open-source 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 tedioso problema de validar el código para cada uno de estos. Entre sus principales ventajas se encuentra el balance que logra para la arquitectura Cliente-Servidor, distribuyendo la carga de procesamiento en el último, y 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.[45]

1.20.3. Herramientas de desarrollo.

De las herramientas que se proponen en el centro para desarrollar a continuación se describen las que se utilizan para el desarrollo del componente.

[12] *ORM:* Por sus siglas en inglés de "Mapeador Relacional de Objetos".
[13] *DBAL:* Por sus siglas en inglés de "Capa de Abstracción de Bases de Datos"

- **Apache 2.2.16.**

Apache es el servidor web por excelencia, su facilidad de configuración, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Se ejecuta en gran cantidad de sistemas operativos, lo que lo hace prácticamente universal. Es una tecnología gratuita, open-source y altamente configurable de diseño modular por lo que resulta muy sencillo ampliar sus capacidades. Actualmente existen muchos módulos para Apache que son adaptables a este, y están disponibles para su instalación cuando sean necesarios. Además 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.[46]

- **PostgreSQL 8.3.**

Es un sistema de gestión de bases de datos libre basado en el proyecto Postgres, perteneciente a la Universidad de Berkeley. Es un sistema objeto-relacional, que incluye características como la herencia, valores no atómicos (atributos basados en vectores y conjuntos), funciones, disparadores, entre otras. Es altamente extensible, permitiendo el uso de operadores, funciones y tipos de datos definidos por el usuario. Soporta la integridad referencial garantizando la integridad de los datos en la base de datos. PostgreSQL permite realizar múltiples conexiones desde procesos clientes, existiendo un proceso maestro en el servidor que siempre se ejecuta y que está a la espera de nuevas conexiones clientes, de forma tal que cuando alguien se conecta, se inicia un nuevo proceso, asegurando que el cliente y la nueva conexión no necesiten del proceso postgres original. Una de sus principales características es la alta concurrencia. Esto permite que mientras se realizan cambios en una tabla, otros procesos accedan a la misma sin la necesidad de bloqueos, además de que cada usuario tiene visión de la última modificación. Presenta el inconveniente de que para bases de datos pequeñas su velocidad de respuesta no es muy eficiente en comparación con otras relativamente grandes.[44]

- **Visual Paradigm 4.3.**

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: abordando el análisis y diseño orientado a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de

aplicaciones de calidad, mejores 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 de UML, demostraciones interactivas de UML y proyectos UML. Soporta UML versión 2.1, permite modelado colaborativo con CVS[14] 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.[47]

- **Mozilla Firefox 3.0**

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

Entre sus principales características se encuentran:[48]

- Mejora en la velocidad de carga de las páginas: Un manejo más eficiente de objetos JavaScript que permiten el uso de ECMAScript5 y que están basados en el motor de JavaScript JägerMonkey.
- Sincronización entre diferentes dispositivos: Permite a Firefox conectar al navegador de escritorio con dispositivos móviles para acceder al historial de navegación y navegar desde un móvil con las mismas pestañas abiertas y contraseñas guardadas en la aplicación de escritorio.
- Amplia personalización:
    - Administrador de complementos rediseñado para permitir descubrir e instalar complementos sin dejar de usar Firefox.
    - Administrador de descargas que permite también ver, administrar y deshabilitar plugins de terceros.

[14] *CVS:* es un Sistema de Control de Versiones que permite que varios programadores trabajen de forma colaborativa en un mismo proyecto llevando un control de las versiones de los ficheros. De esta forma se permiten cambios concurrentes en un mismo fichero sin perder lo cambios realizados.

- Gran número de temas para cambiar la apariencia del navegador.
- Interfaz modificable para reordenar organizar, agregar o quita botones o campos.

- Barra de navegación: con la función de autocompletar, que incluye las posibles coincidencias del historial de búsquedas, sitios marcados y pestañas abiertas.
- Interfaz mejorada: las pestañas están en la parte superior de la Barra de navegación. Se incluye además el Botón Firefox con los elementos del menú agrupados en un único botón para facilitar el acceso.

Otras características generales incluyen navegación por pestañas, corrector ortográfico, búsqueda progresiva, marcadores dinámicos, un administrador de descargas, navegación privada, navegación con georreferenciación, aceleración mediante GPU, e integración del motor de búsqueda que desee el usuario. Además de poseer barra de direcciones inteligente, Identificación instantánea del sitio web, anti-malware, anti-phishing, control de contenido, programas antivirus, gestor de contraseñas, actualización automática, bloqueador de ventanas emergentes, gestor de descargas, corrector ortográfico, restauración de sesiones, sugerencias de búsqueda y búsqueda en la web integrada.[27]

1.20.4. Control de Versiones.

- **RapidSVN 0.12.0.**

RapidSVN es una plataforma visual para el sistema Subversion escrito en C++, con código abierto y software libre bajo la licencia GNU General Public License (GPL) versión 3. Está disponible en varios idiomas diferentes y actúa de cliente gráfico para el acceso al repositorio SVN, tanto si este es remoto como si es local. Es de fácil manejo para los usuarios principiantes pero lo suficientemente potente y con herramientas interesantes para los usuarios avanzados.

Este programa funciona en cualquier plataforma y se puede ejecutar en Linux, Windows, Mac OS / X, Solaris.[49]

1.21. Conclusiones parciales del capítulo 1.

Luego de haber realizado el marco teórico de la investigación se concluye lo siguiente:

- Los SBR que usan pattern-matching son extremadamente flexibles y poderosos. Ellos son más aplicables a dominios donde hay muchas soluciones posibles, pero tienen como desventaja que para resolver problemas muy complejos será necesario usar una gran cantidad de reglas y explorar toda la BC resulta muy ineficiente, por lo que esta arquitectura no se ajusta, porque en futuras versiones se hace necesario incluir las demás reglas que tiene la notación de modelado y provocaría un resultado ineficiente a la hora de realizar la validación, debido a la demora en los tiempos de respuesta del sistema.

- Los SBR con una arquitectura de red de inferencia son más fáciles de implementar y de explicar la solución encontrada, pero menos poderosos. Es más eficiente porque las relaciones entre las reglas y los parámetros se conocen desde antes de la ejecución. Por otra parte, las redes de inferencia son bastante útiles para dominios donde la cantidad de alternativas diferentes es limitada, como clasificación y diagnóstico. El conjunto de alternativas diferentes no es limitado por lo que en este caso sucede lo mismo que con la arquitectura pattern-matching, el tiempo de respuesta del sistema sería lento.

- Las gramáticas brindan una especificación sintáctica precisa y fácil de entender de un lenguaje, por lo que cuando sea necesario agregar nuevas reglas, estas se pueden añadir con más facilidad a un lenguaje cuando existe una aplicación basada en la descripción gramatical del mismo.

Luego de haber analizado los mecanismos propuestos para realizar la validación, se concluyó, hacer uso de la gramática sobre los SBR, debido a que estos últimos no son recomendados para sistemas que necesiten un tiempo de respuesta relativamente rápido. Por lo antes mencionado se propone la elaboración de un componente de validación de procesos de negocio para el marco de trabajo Sauxe.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

En el siguiente capítulo se muestran los resultados obtenidos durante el proceso de desarrollo de la solución y algunos de los artefactos generados. Posteriormente se describen, el modelo conceptual, los requisitos funcionales y no funcionales, el diagrama de clases y los patrones de diseño utilizados.

2.1. Modelo conceptual.

A continuación se describen los principales conceptos relacionados a la solución, los cuales se encuentran representados en la **Figura 6** a través del modelo conceptual.

El **Editor de procesos de negocio** es una herramienta desarrollada sobre el marco de trabajo Sauxe que permitirá a los usuarios que utilicen este sistema, modelar procesos de negocio. Esta herramienta forma parte de un conjunto de componentes que darán como resultado un Sistema de Gestión de Flujos de trabajo, por lo tanto los diagramas de procesos de negocios que se modelen en ella tienen que ser basados en la notación BPMN.

El **paquete de objetos** es el conjunto de clases creadas para darle persistencia a los elementos del modelado, a través de él se pueden obtener todos los objetos del modelado.

El **proceso** (BPM) contiene todos los objetos de la notación BPMN, el cual debe ser verificado según las reglas definidas.

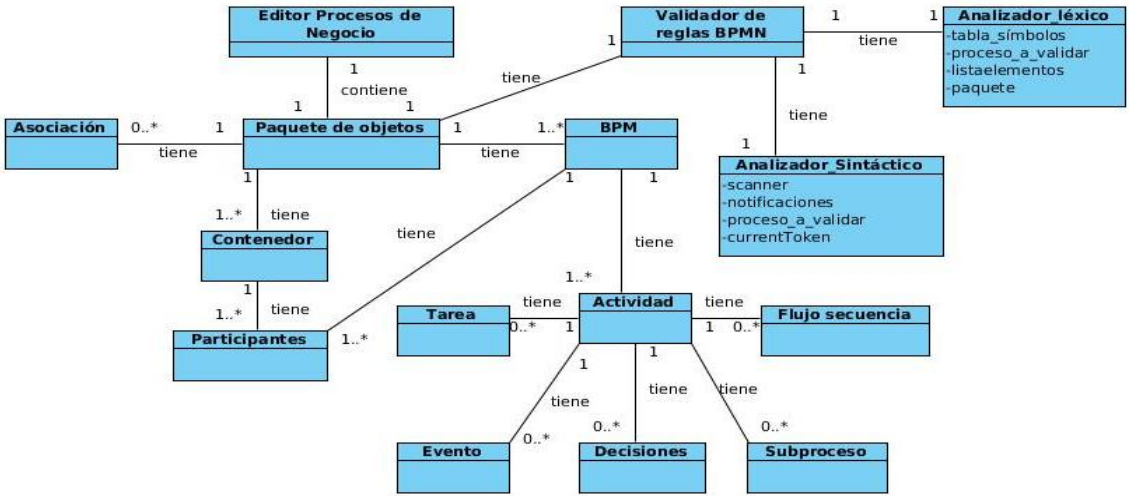

Figura 6: Modelo conceptual.

2.2. Requisitos de software.

2.2.1. Requisitos funcionales.

Después de realizado el modelo conceptual y a través de la técnica de captura de requisitos, tormenta de ideas se determinó que:

El sistema debe ser capaz de:

R1: Verificar reglas BPMN en los procesos de negocio.

R2: Verificar reglas BPMN del objeto de flujo: Evento de inicio.

R3: Verificar reglas BPMN del objeto de flujo: Evento intermedio.

R4: Verificar reglas BPMN del objeto de flujo: Evento de fin.

R5: Verificar reglas BPMN del objeto de flujo: Nodo de decisión.

R6: Verificar reglas BPMN del objeto de flujo: Tarea.

1. Requisito funcional: Verificar reglas BPMN en los procesos de negocio.

Precondiciones	*El sistema selecciona la opción de Guardar ó Validar.*
Flujo de eventos	
Flujo básico validar proceso de negocio	
1	*Seleccionar el proceso a validar.*
2	*Verificar que no haya un flujo de secuencia entre el proceso a validar y los otros procesos.*
3	*Verificar que el proceso a validar sea privado.*
4	*Verificar que los objetos de flujos contenidos en el proceso no tengan flujo de mensaje saliente.*
5	*Verificar que los objetos de flujos contenidos en el proceso no tengan flujo de mensaje entrante.*
6	*Buscar los primeros elementos del proceso a validar.*
7	*Buscar los siguientes de los elementos obtenidos del proceso a validar.*
8	*Validar sintácticamente los elementos obtenidos según las reglas BPMN.*
9	*Verificar si hay más procesos.*
10	*Ejecutar la validación de los procesos restantes.*
11	*Terminar acción.*
Pos-condiciones	
1	*Se ha validado el diagrama de proceso de negocio.*
Flujos alternativos	
Flujo alternativo 2.a: Hay un flujo de secuencia entre el proceso a validar y los otros procesos.	
1	*Crear y almacenar el error.*
2	*Terminar acción.*
Flujo alternativo 3.a: El proceso a validar no es privado.	
1	*Buscar los primeros elementos del proceso a validar.*
2	*Buscar los siguientes de los elementos obtenidos del proceso a validar.*

3	*Validar sintácticamente los elementos obtenidos según las reglas BPMN.*	
4	*Verificar si hay más procesos.*	
5	*Ejecutar la validación de los procesos restantes.*	
Flujo alternativo 3.4.a: No hay más procesos.		
1	*Terminar acción.*	
Flujo alternativo 4.a: Los objetos de flujos contenidos en el proceso tiene flujo de mensaje saliente.		
1	*Crear y almacenar el error.*	
2	*Terminar acción.*	
Flujo alternativo 5.a: Los objetos de flujos contenidos en el proceso tiene flujo de mensaje saliente.		
1	*Crear y almacenar el error.*	
2	*Terminar acción.*	
Flujo alternativo 8.a: El elemento no está correcto sintácticamente.		
1	*Crear y Guardar los errores encontrados.*	
	Terminar acción.	
Flujo alternativo 9.a: No hay más procesos.		
1	*Terminar acción.*	
Pos-condiciones		
	N/A	
Validaciones		
	Se validan los datos según lo establecido en el Modelo conceptual CSG-CNP Modelo Conceptual del componente para la validación de procesos de negocio modelados con BPMN en el Marco de Trabajo Sauxe.	
Conceptos	**Validar**	*Utilizados internamente:* *proceso a validar* *currenttoken* *notifications*
Conceptos	**proceso**	*Utilizados internamente:* *id*
Requisitos especiales	*Reglas BPMN*	
Asuntos pendientes	*Posibles mejoras al requisito.*	

Tabla 8: Verificar reglas BPMN en los procesos de negocio.

2. Requisito funcional: Verificar reglas BPMN del objeto de flujo: Evento de inicio.

Precondiciones	*El elemento actual debe ser un Evento de Inicio.*
Flujo de eventos	
Flujo básico validar proceso de negocio	
1	*Buscar los elementos siguientes al evento de inicio.*
2	*Verificar si los elementos siguientes al evento de inicio son flujos de secuencia.*
3	*Verificar si los elementos siguientes al evento de inicio no son flujos de mensaje salientes.*
4	*Verificar que el evento de inicio no tiene flujo de secuencia entrante.*
5	*Terminar acción.*
Pos-condiciones	
	Se verificó el evento de inicio de acuerdo a las reglas BPMN.
Flujos alternativos	

Flujo alternativo 1.a: El evento de inicio no tiene elementos siguientes.		
1	*Crear y almacenar el error.*	
2	*Terminar acción.*	
Flujo alternativo 2.a: Los elementos siguientes al evento de inicio no son flujos de secuencia.		
1	*Crear y almacenar el error.*	
2	*Terminar acción.*	
Flujo alternativo 3.a: Los elementos siguientes al evento de inicio son flujos de mensaje salientes.		
1	*Crear y almacenar el error.*	
2	*Terminar acción.*	
Flujo alternativo 4.a: el evento de inicio tiene flujo de secuencia entrante.		
1	*Crear y almacenar el error.*	
2	*Terminar acción.*	
Pos-condiciones		
	N/A	
Validaciones		
	Se validan los datos según lo establecido en el Modelo conceptual CSG-CNP Modelo Conceptual del componente para la validación de procesos de negocio modelados con BPMN en el Marco de Trabajo Sauxe.	
Conceptos	**Error**	*Utilizados internamente:* *currenttoken* *notifications*
Conceptos	**Evento de inicio**	*Utilizados internamente:* *token*
Requisitos especiales	*Reglas BPMN*	
Asuntos pendientes	*Posibles mejoras al requisito.*	

Tabla 9: Verificar reglas BPMN del objeto de flujo: evento de inicio.

3. Requisito funcional: Verificar reglas BPMN del objeto de flujo: Evento intermedio.

Precondiciones	*El elemento actual debe ser un evento intermedio.*
Flujo de eventos	
Flujo básico validar proceso de negocio	
1	*Buscar los elementos siguientes al evento intermedio.*
2	*Verificar que los elementos siguientes al evento intermedio sea solamente un flujo de secuencia.*
3	*Verificar que los elementos siguientes al evento intermedio no sea un flujo de mensaje.*
4	*Verificar que el evento intermedio tenga solamente un flujo de secuencia entrante.*
5	*Terminar acción.*
Pos-condiciones	
	Se verificó el evento intermedio de acuerdo a las reglas BPMN.
Flujos alternativos	
Flujo alternativo 1.a: El evento intermedio no tiene siguientes	
1	*Verificar si hay un evento de inicio.*
2	*Terminar acción.*
Flujo alternativo 1.1.a: Hay evento un evento de inicio	

1	*Crear y almacenar el error.*	
2	*Terminar acción.*	
Flujo alternativo 2.a: Verificar que los elementos siguientes al evento intermedio no sea solamente un flujo de secuencia.		
1	*Crear y almacenar el error.*	
2	*Terminar acción.*	
Flujo alternativo 3.a: Verificar que los elementos siguientes al evento intermedio sea un flujo de mensaje.		
1	*Crear y almacenar el error.*	
2	*Terminar acción.*	
Flujo alternativo 4.a: Verificar que el evento intermedio no tenga solamente un flujo de secuencia entrante.		
1	*Verificar si hay un evento de inicio.*	
2	*Terminar acción.*	
Pos-condiciones		
	N/A	
Validaciones		
	Se validan los datos según lo establecido en el Modelo conceptual CSG-CNP Modelo Conceptual del componente para la validación de procesos de negocio modelados con BPMN en el Marco de Trabajo Sauxe.	
Conceptos	**Error**	*Utilizados internamente:* *currenttoken* *notifications*
Conceptos	**Evento intermedio**	*Utilizados internamente:* *token*
Requisitos especiales	*Reglas BPMN*	
Asuntos pendientes	*Posibles mejoras al requisito.*	

Tabla 10: Verificar reglas BPMN del objeto de flujo: Evento intermedio.

4. Requisito funcional: Verificar reglas BPMN del objeto de flujo: Evento de fin.

Precondiciones	*El elemento actual del proceso es un evento de fin.*
Flujo de eventos	
Flujo básico validar proceso de negocio	
1	*Buscar los elementos siguientes del evento de fin.*
2	*Verificar que el evento de fin no tenga flujo de mensaje entrante.*
3	*Verificar que haya un evento de inicio iniciando el proceso*
4	*Terminar acción.*
Pos-condiciones	
	Se verificó el evento de fin de acuerdo a las reglas BPMN.
Flujos alternativos	
Flujo alternativo 1.a: El evento de fin tiene elementos siguientes.	
1	*Crear y almacenar el error.*
2	*Terminar acción.*
Flujo alternativo 2.a: El evento de fin tenga flujo de mensaje entrante.	
1	*Crear y almacenar el error.*
2	*Terminar acción.*

Pos-condiciones		
	N/A	
Validaciones		
	Se validan los datos según lo establecido en el Modelo conceptual CSG-CNP Modelo Conceptual del componente para la validación de procesos de negocio modelados con BPMN en el Marco de Trabajo Sauxe.	
Conceptos	**Error**	*Utilizados internamente:* *currenttoken* *notifications*
Conceptos	**Evento de fin**	*Utilizados internamente:* *token*
Requisitos especiales	*Reglas BPMN*	
Asuntos pendientes	*Posibles mejoras al requisito.*	

Tabla 11: Descripción del requisito funcional Verificar reglas BPMN del objeto de flujo: Evento de fin.

5. Requisito funcional: Verificar reglas BPMN del objeto de flujo: Nodo de decisión.

Precondiciones	*El elemento actual del proceso es un nodo de decisión.*
Flujo de eventos	
Flujo básico validar proceso de negocio	
1	*Buscar los siguientes del nodo decisión.*
2	*Verificar que los elementos siguientes del nodo de decisión no son flujos de mensaje.*
3	*Verificar que los elementos anteriores del nodo de decisión no son flujos de mensaje.*
4	*Verificar si el nodo de decisión es basado eventos.*
5	*Terminar acción.*
Pos-condiciones	
	Se verificó el nodo de decisión de acuerdo a las reglas BPMN.
Flujos alternativos	
Flujo alternativo 1.a: El nodo decisión no tiene siguientes.	
	Verificar si hay un evento de inicio.
	Terminar acción.
Flujo alternativo 1.1.a: Hay un evento de inicio.	
1	*Crear y almacenar el error.*
2	*Terminar acción.*
Flujo alternativo 2.a: Los elementos siguientes del nodo de decisión son flujos de mensaje.	
1	*Crear y almacenar el error.*
2	*Terminar acción.*
Flujo alternativo 3.a: Los elementos anteriores del nodo de decisión son flujos de mensaje.	
1	*Crear y almacenar el error.*
2	*Terminar acción.*
Flujo alternativo 4.a: El nodo de decisión es basado en eventos.	
1	*Verificar que los elementos siguientes sean todos objetos de flujo de tipo tarea recibir y/o eventos intermedios (tiempo, mensaje, regla o enlace).*
2	*Terminar acción.*
Flujo alternativo 4.1.a: Los elementos siguientes son combinaciones de objetos de flujo de tipo tarea recibir	

y/o eventos intermedios de tipo mensaje.		
1	*Crear y almacenar el error.*	
2	*Terminar acción.*	
Pos-condiciones		
	N/A	
Validaciones		
	Se validan los datos según lo establecido en el Modelo conceptual CSG-CNP Modelo Conceptual del componente para la validación de procesos de negocio modelados con BPMN en el Marco de Trabajo Sauxe.	
Conceptos	**Error**	*Utilizados internamente:* *currenttoken* *notifications*
Conceptos	**Nodo de decisión**	*Utilizados internamente:* *token*
Requisitos especiales	*Reglas BPMN*	
Asuntos pendientes	*Posibles mejoras al requisito.*	

Tabla 12: Descripción del requisito funcional Verificar reglas BPMN del objeto de flujo: Nodo de decisión.

6. Requisito funcional: Verificar reglas BPMN del objeto de flujo: Tarea.

Precondiciones	*El elemento actual debe ser una tarea.*	
Flujo de eventos		
Flujo básico validar proceso de negocio		
1	*Buscar los elementos siguientes a la tarea*	
2	*Terminar acción.*	
Pos-condiciones		
1	*Se verificó la tarea de acuerdo a las reglas BPMN.*	
Flujos alternativos		
Flujo alternativo 1.a: La tarea no tiene elementos siguientes		
1	*Verificar si hay un evento de inicio*	
2	*Terminar acción.*	
Flujo alternativo 1.1.a: Hay un evento de inicio		
1	*Crear y almacenar el error*	
2	*Terminar acción.*	
Pos-condiciones		
	N/A	
Validaciones		
	Se validan los datos según lo establecido en el Modelo conceptual CSG-CNP Modelo Conceptual del componente para la validación de procesos de negocio modelados con BPMN en el Marco de Trabajo Sauxe.	
Conceptos	**Error**	*Utilizados internamente:* *currenttoken* *notifications*

Conceptos	**Tarea**	*Utilizados internamente:* *token*
Requisitos especiales	*Reglas BPMN*	
Asuntos pendientes	*Posibles mejoras al requisito.*	

Tabla 13: Descripción del requisito funcional: Verificar reglas BPMN del objeto de flujo: Tarea.

2.2.2. Requisitos no funcionales.

La solución propuesta fue iniciada en el proyecto Marco de Trabajo Sauxe. De esta manera los requisitos no funcionales a los que se debe acoger el componente a desarrollar son los establecidos al inicio del proceso de desarrollo. A continuación son descritos algunos de estos requisitos.[29]

Software

Para el cliente:

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

Para el servidor:

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

Rendimiento

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

Seguridad

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

Hardware

Para el servidor:

- Requerimientos mínimos: Procesador Pentium IV a 3GHz de velocidad de procesamiento y 1Gb de memoria RAM.
- Disco duro: 160Gb.
- Tarjeta de red.
- UPS.
- Lector CD.

Para el cliente:

- Requerimientos mínimos: Procesador Pentium III a 1.40GHz con 256MB (recomendado 512 MB) de memoria RAM.
- Tarjeta de red: 1.

2.3. Diagrama de clases.

A continuación en la ***Figura* 7** se muestra el diagrama de clases diseñado para la solución, donde se representan las clases que tendrán el sistema y las relaciones entre ellas.

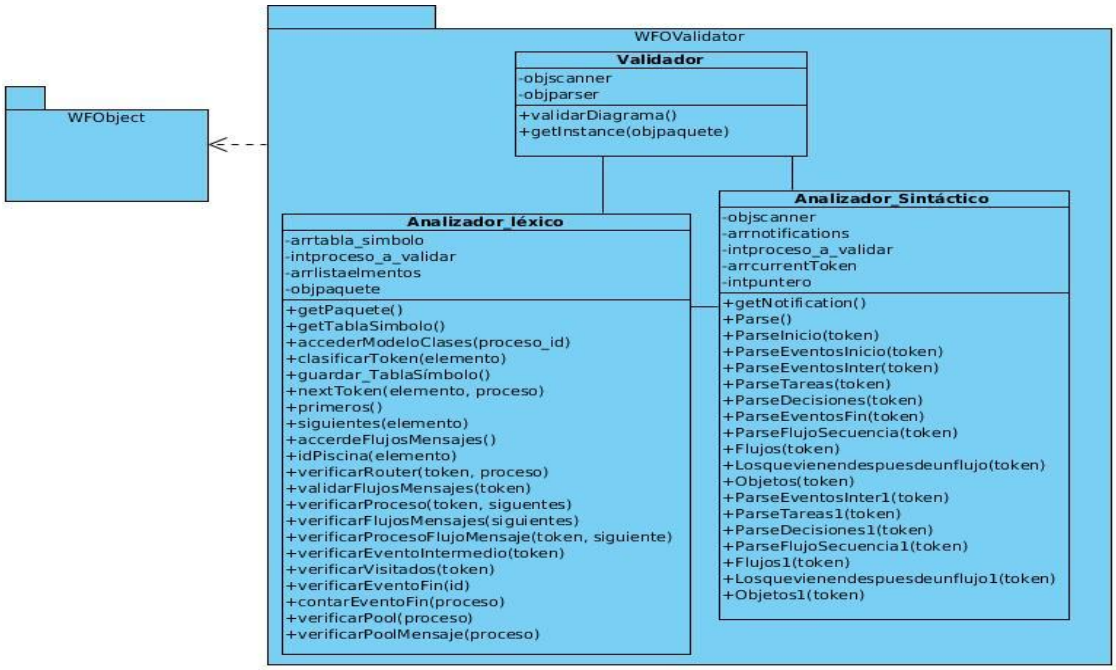

Figura 7: Diagrama de clases.

En el caso de la solución el diagrama está formado por tres clases, una de ellas actúa como controladora, la clase Validador, dirigiendo en que momento deben utilizarse las otras clases del componente. La clase Analizador Léxico es la encargada de acceder al paquete con los objetos que son representados en la herramienta de modelado y clasificar los token, que luego serán la entrada a la clase Analizador para que este se encargue de verificar al token que llega. Para realizar algunas validaciones el Analizador Sintáctico se apoya de las funcionalidades que se encuentran implementadas en la clase Analizador Léxico.

La clase Analizador Léxico cuenta con 21 procedimientos, el Analizador Sintáctico con 19 y la validador con 2.

2.4. Patrones de diseño utilizados.

Después de haber analizado los patrones de diseño propuestos en [50] a continuación se describen los que serán utilizados para el desarrollo del componente.

- **Patrón de diseño: Fachada**

El patrón de diseño estructural Fachada tiene como propósito simplificar el acceso a un conjunto de objetos proporcionando uno que todos los clientes pueden usar para comunicarse con el conjunto. Tiene como objetivo minimizar las comunicaciones y dependencias entre componentes. Normalmente sólo hace falta un objeto Fachada, por lo cual suele implementarse como Singleton. A continuación en la ***Figura* 8** se muestra un ejemplo de cómo funciona el patrón fachada.[51, 52]

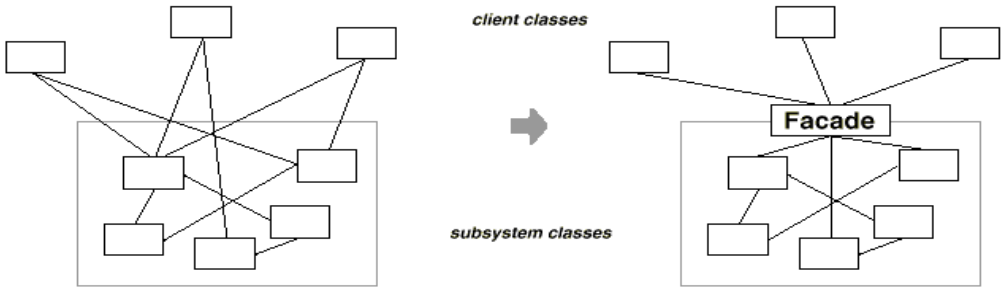

Figura 8: Patrón de diseño Fachada. [51, 52]

Descripción del patrón Fachada en la solución.

En el caso de la solución, el Editor de procesos de negocio se comunica con el componente WFValidator a través de la clase Validador que funciona como Fachada, la cual es la encargada de realizar peticiones a las clases que conforman el componente. Con la utilización de este patrón no se realizan invocaciones directas a las clases Analizador léxico o Analizador Sintáctico, puesto que la clase Validador conoce las responsabilidades de cada clase y controla las acciones de ellas de acuerdo a cada petición. Este patrón es utilizado en conjunto al patrón Singleton, el cual se encuentra implementado en la clase Validador.

A continuación en la ***Figura* 9** se muestra como es utilizado el patrón Fachada en la solución propuesta.

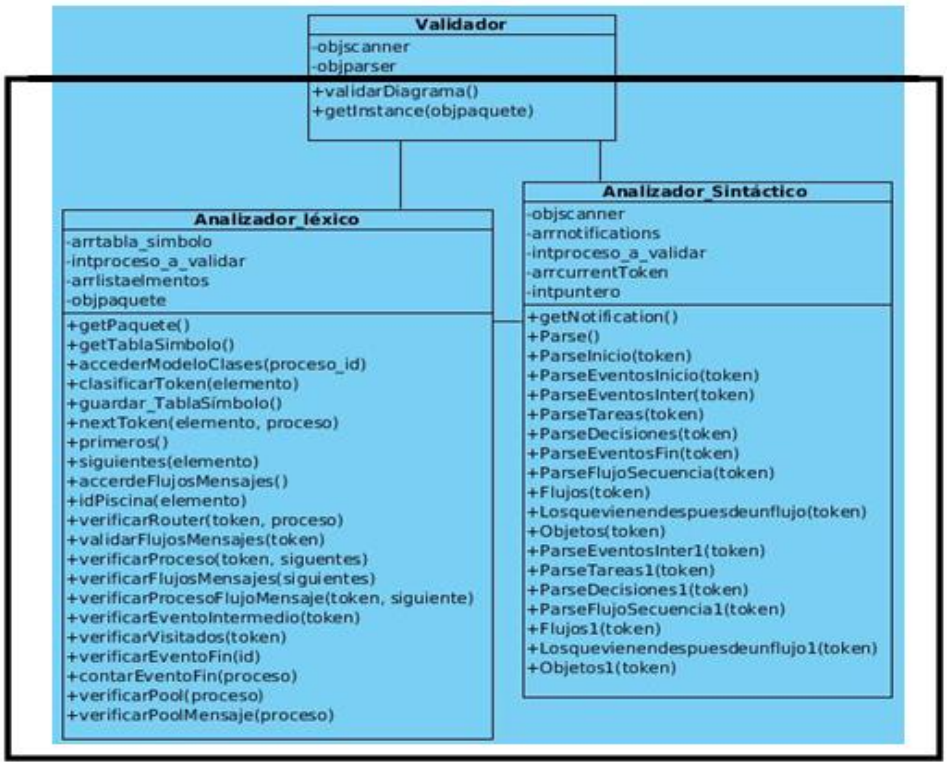

Figura 9: Patrón Fachada.

- **Patrón de diseño: Singleton.**

El patrón de diseño Singleton tiene como propósito asegurar que una clase sólo tiene un ejemplar, y proporcionar un punto de acceso global a este. El patrón de creación Singleton se usa en casos en los que algunas clases sólo necesitan exactamente un ejemplar y entonces en vez de tener una variable global para acceder a ese ejemplar único, la clase se encarga de proporcionar un método de acceso.[51, 52]

Descripción del patrón Singleton en la solución.

Para el caso del componente, el sistema accede desde la clase AplicationControler del Editor de procesos de negocio al ejemplar de Singleton, únicamente a través del método getInstance de la clase Validador. Con la utilización de este patrón se evita la creación de una variable global para realizar una llamada a la función Validar () de la clase Validador.

A continuación en la ***Figura* 10** se muestra cómo en la clase AplicationControler se realiza una petición al componente de validación a través de la función getInstance().

AplicacionController.php

```php
<?php

class AplicacionController extends ZendExt_Controller_Secure {

    public function init() {
        parent::init();
    }

    public function aplicacionAction() {
        $this->render();
    }

    public function salvarAction() {
        $paquete = $this->_request->getPost('paquete');
        $objPaquete = ZendExt_WF_WFObject_Parser::createObject($paquete);
        $validador=ZendExt_WF_WFValidator_Validador::getInstance($objPaquete);
        $errores=$validador->validarDiagrama();

    }
```

Figura 10: Función donde se hace uso del patrón Singleton en la solución.

A continuación en la ***Figura* 11** se muestra la clase donde ha sido implementado la función getInstance().

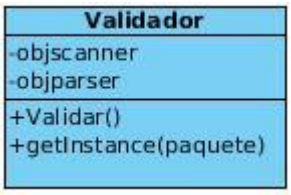

Figura 11: Patrón Singleton.

- **Patrón de diseño Experto.**

El patrón de diseño GRASP Experto asigna una responsabilidad al experto en información o sea a la clase que cuenta con la información necesaria para cumplir la responsabilidad.[33]

Un modelo de clase puede definir docenas y hasta cientos de clases de software, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades. Durante el diseño orientado a objetos, cuando se definen las interacciones entre los objetos, se toman decisiones sobre la asignación de responsabilidades a las clases. Si se hacen en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se presenta la oportunidad de reutilizar los componentes en futuras aplicaciones.[53]

Descripción del patrón Experto en la solución

En el caso del componente, la clase Analizador Léxico es la responsable de clasificar el token que recibirá la clase Analizador Sintáctico como entrada, la cual a su vez valida que este sea el correcto de acuerdo a las reglas BPMN que han sido representadas a través de la gramática que describe la clase.

- **Patrón de diseño Controlador.**[53]

El patrón de diseño GRASP Controlador asigna la responsabilidad del manejo de un mensaje de eventos de un sistema a una clase que represente una de las siguientes opciones:

- El "sistema" global (controlador de fachada).
- La empresa u organización global (controlador de fachada).

Un Controlador es un objeto de interfaz no destinada a un usuario que se encarga de manejar un evento del sistema. Define además el método de su operación.

¿Quién debería encargarse de atender un evento del sistema?

Un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que emite en respuesta a los eventos del sistema.

Descripción del Patrón Controlador en la solución.

En el caso de la solución la clase Validador hace función de Controlador porque es la encargada de manejar los eventos del sistema que involucren al componente de validación.

2.5. Conclusiones parciales del capítulo 2.

En el capítulo se realizó el análisis y diseño de la solución propuesta, se generaron los artefactos que forman parte del modelo de desarrollo propuesto, tales como: el modelo conceptual, la descripción textual de los requisitos funcionales, el modelo de diseño, los requisitos no funcionales, el diagrama de clases. Se identificaron además los patrones que serán utilizados: el patrón Fachada, Singleton y el

Experto. Una vez concluida esta fase de la solución se procederá a la realización de los flujos de implementación y prueba.

CAPÍTULO 3: IMPLEMENTACION Y PRUEBA.

En el siguiente capítulo se muestran los artefactos generados en la fase de implementación, que utilizando el modelo de desarrollo serían: el diagrama de componentes y el de despliegue. También se especifica cuáles son los estándares de codificación utilizados y el conjunto de validaciones y pruebas que evalúan la calidad del componente desarrollado.

3.1. Estándares de codificación.

De los estándares de codificación propuestos para la solución se utilizarán el CamelCasing y la Notación húngara.

Del estándar **CamelCasing** se utilizará la Nomenclatura de las funciones.
Ejemplo en la solución: **getInstance()**, **guardarTablaSimbolo()**.

También se utilizará el estándar **Notacion húngara.**
Ejemplo en la solución: **arrelementos**, **objpaquete**.

3.2. Diagrama de componentes.

La solución consta de un solo componente denominado componente WFValidator que interactúa con el componente WFObject del cual toma información, estos dos componentes forman parte del marco de trabajo Sauxe. A continuación en la ***Figura* 12** se muestra el diagrama de componentes.

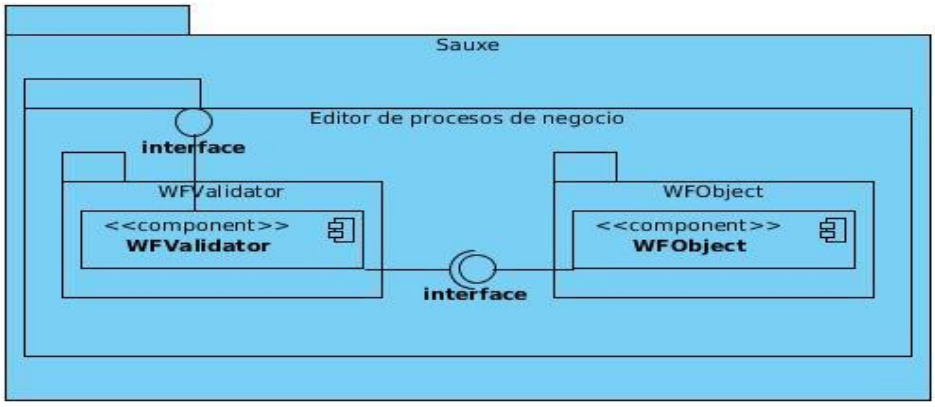

Figura 12: Diagrama de componentes.

3.3. Diagrama de despliegue.

Para realizar el despliegue del componente de validación es necesario la utilización de las siguientes tecnología: en la estación de trabajo para el cliente, tener instalado el navegador web Morzilla Firefox 3.0 o superior con conexión http al servidor Web el cual deberá tener instalada Apache en su versión 2.2 y este a su vez tiene una conexión PDO con la base de datos que utiliza PostgreSQL en su versión 8.3. A continuación en la ***Figura* 13** se muestra el diagrama de despliegue.

Figura 13: Diagrama de Despliegue.

3.4. Métricas de software.

A continuación se describen los resultados obtenidos luego de haber aplicado la métrica TOC y RC para validar el diseño propuesto.

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

Al aplicar la métrica TOC se determinó que la solución está compuesta de 3 clases y 42 procedimientos, reflejados en la siguiente tabla junto a los atributos de calidad de Responsabilidad, Complejidad de Implementación, Reutilización. Se obtuvo, como promedio de procedimientos por clase 14, y los siguientes datos se utilizaron para categorizar los atributos por cada clase:

Clases	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
Validador	2	Baja	Baja	Alta
AnalizadorSintactico	19	Media	Media	Media
AnalizadorLexico	21	Media	Media	Media

Tabla 14: Instrumento de evaluación de la métrica TOC.

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

Tabla 15: Resultados obtenidos de la evaluación de la métrica TOC para el atributo Responsabilidad.

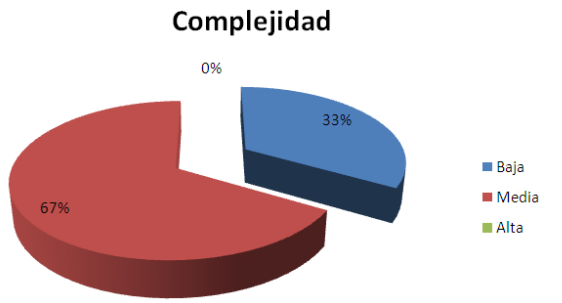

Tabla 16: Resultados obtenidos de la evaluación de la métrica TOC para el atributo Complejidad.

Tabla 17: Resultados obtenidos de la evaluación de la métrica TOC para el atributo Reutilización.

Los valores entre Media y Baja de esta métrica indican que las clases no tienen una gran responsabilidad, ni una gran complejidad de implementación. Algo parecido sucede con la reutilización, aunque su valor está entre medio y alto, lo cual posibilita de cierta forma su reutilización. Para todos los atributos el umbral más significativo es el indicador de Media. En general esto indica realmente el nivel de utilización para la responsabilidad, reutilización, y complejidad de implementación.

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

El componente está formado por 3 clases, las cuales tienen 4 relaciones de uso entre ellas, a continuación en la siguiente tabla se refleja la relación de uso existente entre las clases junto a los atributos de calidad de Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas. Se obtuvo un promedio de relaciones de dependencia por clase de 1,33333.

Clases	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
AnalizadorLexico	1	Bajo	Baja	Alta	Baja
AnalizadorSintactico	1	Bajo	Baja	Alta	Baja
Validador	2	Medio	Media	Media	Media

Tabla 18: Instrumento de evaluación de la métrica RC.

Después de obtener estos resultados se generaron una serie de gráficas en correspondencia a cada atributo de calidad analizado, las mismas brindan un mejor entendimiento de los resultados obtenidos:

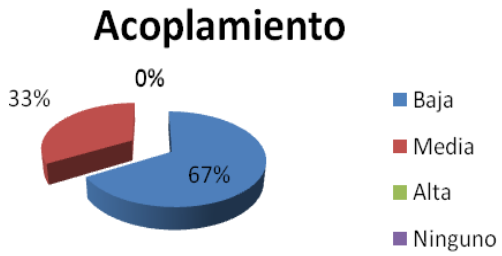

Tabla 19: Resultados de la evaluación de la métrica RC para el atributo Acoplamiento.

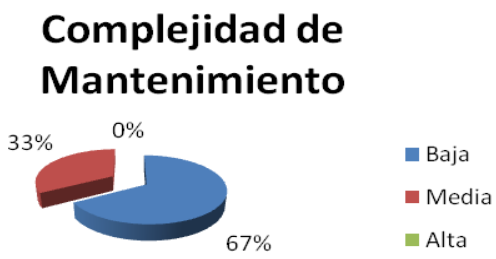

Tabla 20: Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento.

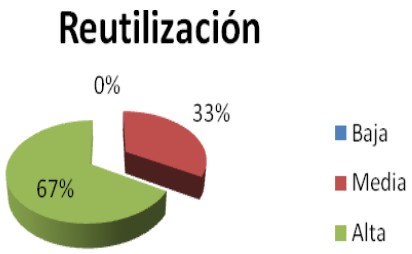

Tabla 21: Resultados de la evaluación de la métrica RC para el atributo Complejidad de Reutilización.

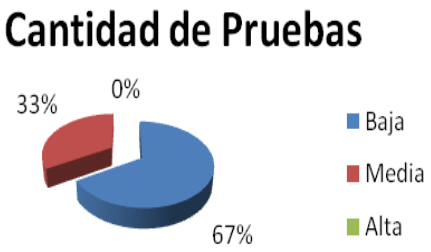

Tabla 22: Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas.

Los valores altos para esta métrica indican que las clases tienen buena reutilización, lo cual reducirá el acoplamiento de las clases, la complejidad de mantenimiento y la cantidad de pruebas.

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

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

Tabla 23: Resultados de la evaluación de la relación Atributo/Métrica.

Luego de haber aplicado estas métricas se obtuvo como resultado que las clases no tendrán mucha Responsabilidad, su implementación no es muy complicada, tendrán bajo acoplamiento, la complejidad de mantenimiento será baja así como la cantidad de pruebas a realizar, aunque si tendrá una

alta reutilización. Todos estos resultados demuestran que el diseño no es complejo por lo que confirman la calidad del diseño.

3.5. Pruebas de software.

En aras de obtener un producto favorable se determinó aplicar al componente una serie de pruebas para comprobar su correcto funcionamiento. Las pruebas a las que será sometida la aplicación son: pruebas del camino básico, pruebas de cajas negras y pruebas de rendimiento.

3.5.1. Pruebas estructurales o de caja blanca.

A continuación en la ***Figura* 14** se muestra el código de una de las funcionalidades a la que se le aplicó la prueba del camino básico, esta funcionalidad es **guardarTablaSimbolo()**.

```
public function Guardar_TablaSimbolo() {
  //contar cantidad de procesos
  for ($i = 0; $i < $this->objpaquete->getWorkflowProcesses()->count(); $i++) //1
  {
$this->arrlistaelmentos = self::AccederModeloClases($this->objpaquete->getWorkflowProcesses()->get($i)->getId());//2

    $this->arrlistaelmentos = self::AccederFlujosMensajes();//2
    foreach ($this->arrlistaelmentos as $elemt)//3
    {
      //print_r($elemt);
      $tipotoken = self::ClasificarToken($elemt);//4
      $proceso = $this->objpaquete->getWorkflowProcesses()->get($i)->getId();//4 //Buscar el proceso

      $this->arrtabla_simbolo[] = array(
          'proceso' => $proceso,
          'objeto' => $elemt,
          'tipotoken' => $tipotoken);//4
    }
  }
  //print_r(count($this->arrtabla_simbolo));//die;
  return $this->arrtabla_simbolo;//5
}
```

Figura 14: Código fuente de la funcionalidad Guardar Tabla de Símbolo.

En la ***Figura* 15** se muestra el grafo correspondiente al código de la funcionalidad **guardarTablaSimbolo().**

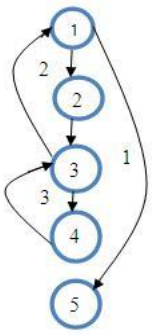

Figura 15: Grafo de flujo asociado a la funcionalidad Guardar Tabla de Símbolo.

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

Fórmula 1. V (G) = (A – N) + 2.

V (G) = (6-5) + 2

V (G) = 3

Fórmula 2. V (G) = P + 1.

V (G) = 2+1

V (G) = 3

Fórmula 3. V (G) = R.

V (G) = 3

Con la correcta realización del cálculo de complejidad se determinó el número de posibles caminos por donde el flujo debe circular y el número de casos pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. El cálculo arrojó como valor 3 por lo que seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo:

Camino #1: (1-2-3-1-5).

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

Camino #3: (1-5).

Para cada camino básico se realizó un caso de prueba, los cuales son mostrados a continuación:

1. **Caso de prueba para el camino básico #1:**

   Descripción: No necesita la entrada de parámetros para ejecutarse.

   Condición de ejecución: Hay un workflowprocess y no hay modelado ningún objeto de flujo.

   Resultados Esperados: Se obtiene la tabla símbolo vacía.

   Evaluación de los resultados: Al realizar las pruebas de caja blanca al caso guardar tabla de símbolo se obtuvieron los resultados esperados.

2. **Caso de prueba para el camino básico #2:**

   Descripción: No necesita la entrada de parámetros para ejecutarse.

   Condición de ejecución: Hay un workflowprocess y un objeto de flujo modelado.

   Resultados Esperados: Se obtiene la tabla símbolo con un elemento.

   Evaluación de los resultados: Al realizar las pruebas de caja blanca al caso guardar tabla de símbolo se obtuvieron los resultados esperados.

3. **Caso de prueba para el camino básico #3:**

   Descripción: No necesita la entrada de parámetros para ejecutarse.

   Condición de ejecución: No hay ningún workflowprocess

   Resultados Esperados: Devuelve la tabla de símbolos vacía.

   Evaluación de los resultados: Al realizar las pruebas de caja blanca al caso guardar tabla de símbolo se obtuvieron los resultados esperados.

3.5.2. Pruebas de caja negra.

El componente de validación está integrado a la herramienta de modelado Editor de procesos de negocio del Marco de trabajo Sauxe, por lo que las pruebas de caja negra del componente se realizarán en conjunto al Editor. Los resultados obtenidos se podrán apreciar en el Anexo 14.

3.5.3. Pruebas de rendimiento.

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

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

Después de obtener los resultados generados en JMeter, el Editor de procesos de negocios mostró un tiempo de respuesta entre 3,8 sec y 4,9 sec; resultado positivo para el componente ya que estos valores no sobrepasan los 5 segundos de tiempo de respuesta del sistema. A continuación en la ***Figura* 16** se muestran los resultados obtenidos por la herramienta JMeter.

Label	# Mue...	Media	Media...	Linea ...	Mín	Máx	% Error	Rendimiento	Kb/sec
/Herramientas/editorft/index.php/aplicacion/aplicacion	50	1107	1109	1922	531	2172	0,00%	3,8/sec	11489,7
/Herramientas/editorft/views/js/aplicacion/config.json	50	5	0	16	0	62	0,00%	4,8/sec	9542,0
/Herramientas/editorft/views/js/aplicacion/entry-points.j...	50	5	0	16	0	16	0,00%	4,8/sec	24,2
/Herramientas/editorft/views/js/components/toolbar/cax...	50	5	0	16	0	62	0,00%	4,8/sec	30626,4
/Herramientas/editorft/views/js/components/gui-design...	50	4	0	16	0	16	0,00%	4,8/sec	58678,4
/Herramientas/editorft/views/js/components/canvas/cax...	50	3	0	16	0	16	0,00%	4,8/sec	56134,4
/Herramientas/editorft/views/js/components/tool-box/ca...	50	3	0	16	0	16	0,00%	4,8/sec	21429,3
/Herramientas/editorft/views/js/components/tool-box/ca...	50	4	0	16	0	16	0,00%	4,8/sec	9988,4
/Herramientas/editorft/views/js/components/object-insp...	50	3	0	16	0	16	0,00%	4,9/sec	55569,6
TOTAL	1150	840	31	2047	0	6860	4,35%	49,8/sec	182523,3

Figura 16: Resultados obtenidos por la herramienta JMeter.

3.6. Prueba realizada por el departamento

Además de estas pruebas, por el departamento de tecnología se le realizaron también pruebas al componente. Estas pruebas fueron realizadas con el objetivo de validar la solución y verificar si cumplía con el objetivo de su desarrollo. Los profesionales encargados de realizar esta revisión emitieron un acta de liberación la cual puede ser consultada en el Anexo 4.

3.7. Conclusiones parciales del capítulo 3.

En el capítulo fueron definidos los artefactos generados durante la fase de implementación tales como el diagrama de componentes y el de despliegue, además de especificarse los estándares de calidad a tener en cuenta en esta fase.

También se realizaron pruebas para validar el diseño propuesto, para lo cual se utilizaron métricas basadas en diferentes atributos de calidad.

Para poder probar el componente de validación se realizaron algunas pruebas como las de caja blanca para poder determinar los caminos básicos, las cuales arrojaron resultados satisfactorios sobre el funcionamiento del código implementado.

Al componente también le fueron realizadas pruebas de carga y estrés de las cuales se obtuvieron resultados satisfactorios para el componente ya que el tiempo de respuesta del sistema no excedió de 5 segundos.

CONCLUSIONES

Con el desarrollo de los objetivos propuestos se puede arribar a las siguientes conclusiones:

- Con la realización del estudio del arte se pudo determinar una técnica para realizar la validación, la cual permitirá continuar agregando reglas en fututas versiones del editor de procesos de negocios del marco de trabajo Sauxe.

- También al identificar las reglas asociadas a la notación BPMN se pudieron determinar las que formarán parte de la primera versión y las que serán agregadas en versiones posteriores.

- Se desarrolló el componente haciendo uso de las herramientas y tecnologías propuestas por el centro obteniendo como resultado que el componente tenga acoplamiento con el marco de trabajo. La implementación realizada permite que el sistema tenga un tiempo de respuesta rápido y brinda la información esperada. El código fue comprobado a través de las pruebas de cajas blancas para determinar los caminos básicos asociadas a cada funcionalidad.

- Para validar que el tiempo de respuesta rápido del sistema al Editor de procesos de negocios se le realizaron las pruebas de carga y estrés las cuales arrojaron resultados satisfactorios.

Siendo así el componente para la validación de procesos de negocios modelados con BPMN soluciona al problema que existía, disminuyendo los errores de los modelos de procesos de negocios modelados en el Editor de procesos de negocios del marco de trabajo Sauxe.

RECOMENDACIONES

- Luego de haber concluido el proceso de desarrollo se recomienda utilizar la documentación generada por el componente como material de estudio para futuras versiones.
- Se recomienda el uso de la gramática definida en el Anexo 3, para adicionar el resto de las reglas BPMN que no se aplicaron en la versión 1.0.

BIBLIOGRAFÍA

1. Kiran Garimella, M.L.y.B.W., *Introducción a BPM para Dummies*. 2008: Wiley Publishing, Inc.
2. Jorge Cardoso, R.P.B.a.A.S., *Workflow Management Systems vs. ERP Systems: Differences, Commonalities, and Applications.* . 2004.
3. WFMC. Available from: www.wfmc.org/.
4. OMG. Available from: www.bpmi.org.
5. Albornoz, M.G., *Medidas tempranas para procesos tipo WorkFlow.* . 2011: Departamento de Computación e Informática, Universidad de Playa Ancha.
6. González, M.A.R., *Herramientas para la modelación de los procesos de negocio en el sistema Cedrux.* 2011.
7. (BPMI), B.P.M.I., *Business Process Modeling Notation (BPMN).* Vol. 1. 2004.
8. 9000, I.a.c. 2010; Available from: http://arpcalidad.com/definicin-de-proceso/.
9. González, P.S., M.A. Martínez, and D.P. González., *Análisis y modelado con redes de workflow del proceso de tratamiento de experiencias operativas.* . . 2003.
10. Quintana., C.J., *Indicadores de Alineamiento entre Procesos de Negocios y Sistemas Informáticos*. 2002, Universidad de Concepción.
11. Weske, M., *Business Process Management: Concepts, Languages and Architecture.* 2007.
12. Ivan Markovic, A.C.P., *Towards a Formal Framework for Reuse in Business Process Modeling. Business Process Management Workshops*. 2007.
13. Orantes, S.D.G., Agustín F.; López, Máximo, *Arquitecturas empresariales: gestión de procesos de negocio vs. arquitecturas orientadas a servicios ¿se relacionan?* Vol. 13. 2009, Universidad Distrital Francisco José de Caldas Bogotá, Colombia.
14. SoaAgenda. 2010; Available from: http://soaagenda.com/journal.
15. Coalition, W.M., *Workflow Management Coalition Terminology & Glossary* . .
16. Beatriz Mora, F.R., Félix García, Mario Piattini., , *Experiencia en transformación de modelos de procesos.* . 2006,: España : Universidad de Castilla-La Mancha, Escuela Superior de Informática.
17. Romero, M., *Tópicos selectos de TI* 2010, Universidad tecnológica: Fidel velázquez
18. Marlon Dumas, W.v.d.A., Arthur terHofstede, *: Process-aware information systems: bridgingpeople and software through process technology.* . 2005.
19. Jakob Freund, B.R., Bernhard Hitpass *BPMN 2.0 Manual de Referencia y Guía Gráfica* ISMB-13:978-1460-903933 ed. 2011, Santiago de Chile.
20. Stephen A. White, I.C., *Introduction to BPMN*. 2004.
21. Buenosvinos, C.a.C.C., *Motores de Workflow, Más allá de las Aplicaciones CRUD.* . 2008.
22. DUMAS, M., W.v.d. AALST, and A.H.M.t. HOFSTEDE, , *PROCESS-AWARE INFORMATION SYSTEMS Bridging People and Software Through Process Technology.* . 2005.
23. Meylin Cinthia Camarena Gil, J.M.P.N., Sandro Salvador Rondón Suasnabar. , *Análisis, Diseño y Construcción de una Herramienta para Modelado de Procesos: MJS Process Designer.* . 2008: Lima, Perú
24. Microsoft.; Available from: http://office.microsoft.com/en-us/visio/.
25. Díaz, E.L., *Análisis y Diseño de Compiladores.* . 2002: Mexico

26. Pena, M.D., *Desarrollo de un editor de código JavaScript para el IDE Caxtor*. 2010, Universidad de las ciencias informáticas, UCI C. de la Habana.
27. Acevedo Martínez, D.L.a.O.R., Msc. Karel. , , *Técnicas de Compilación: Manual Práctico para estudiantes.* . 2011, C. de la Habana
28. Pérez, M.S., *Propuesta de modelo de desarrollo de software tecnológico del Centro de Soluciones de Gestión.* 2009: C. de la Habana
29. González., O.O., *Interoperabilidad del proceso inventario en el sistema Cedrux.* 2011.
30. Firefox., M.M.F. 2008; Available from: http://www.mozilla-europe.org/es/firefox/3.0/releasenotes/.
31. Ivar Jacobson, G.B., James Rumbaugh, *El proceso unificado de desarrollo de software*. Edición en español ed. 200.
32. Pressman, R., *Ingenieria de software, Un enfoque práctico*. 2007-2008.
33. Camejo, R.R.B., *Desarrollo de una herramienta generadora de ficheros de mapeo, para la persistencia de objetos en esquemas relacionales basada en Doctrine.* 2008.
34. Mestras, J.P., *Patrones de diseño orientado a objetos*. 2004, Facultad de Informática UCM.
35. Parsons, D., *Data Types, Arithmetic, and Arrays.* . 2012.
36. Pressman, R.S., *Un Enfoque Practico.*, ed. 8448132149. 2002.
37. Software, P.d. 2012; Available from: http://eva.uci.cu/file.php/158/Documentos/Recursos_didacticos/Presentaciones_digitales_UD_2/Pruebas_de_Software.pdf.
38. Mustelier, S.C., *Propuesta de Procedimiento y Herramienta para el desarrollo de Pruebas de Rendimiento de Carga y Estrés en el Grupo de Calidad de FORTES.* 2011, Universidad de las Ciencias Informáticas (UCI): Ciudad de la Habana.
39. Manuales, T.y.H., 2008. Available from: http://max-alva.webs.com/javascript.htm. .
40. PHP, G.d.d.d., *Manual de PHP*. . 2007.
41. NT, C.d.W. *Manuales, Tutoriales y Herramientas*. 2008; Available from: http://max-alva.webs.com/javascript.htm.
42. Gómez Baryolo, O., Tenrero Cabrera, Marianela and Silega Martínez, Nemuris., , *Plantilla Registro de la Propiedad intelectual (Sauxe).* . 2008: La Habana
43. Esser, S., *Secure Programming with the Zend Framework.* . 2009: Amsterdam : Dutch PHP Conference.
44. Aquino, A.a.L., Yasser.,, *Implementación de módulo de Contabilidad General del sistema integral de gestión Cedrux.* . 2009: C. de la Habana : Universidad de la Ciencias Informáticas.
45. Frederick, S., Ramsay, Colin y Blades, Steve 'Cutter. , Learning Ext JS 2008: Birmingham - Mumbai
46. Ciberaula, Á.t.d.L.P.d.L.-. 2011; Available from: http://linux.ciberaula.com.
47. Prada Nicot, H.y.S.G., Kenner, *Desarrollo de los componentes Puesto de Trabajo y Pagos Adicionales del subsistema Capital Humano integrado al sistema integral de gestión CEDRUX.* 2009: C. de la Habana
48. EcuRed, E.c. 2012; Available from: http://www.ecured.cu/index.php/Mozilla_Firefox.
49. Alejandro Alvarez Hernández, H.R.P., *Plugin para la gestión de pruebas en la herramienta REDMINE*

2011.

50. Cooper, J.W., *Introduction to Design Patterns in C#*. 2002.
51. Mestras, J.P., *Patrones de diseño orientado a objetos*. 2004, Madrid.
52. Frank Buschmann, R.M., hans Rohnert, Peter Sommerlad, Michael Stal, *Pattern-Oriented Software Architecture: A system of patterns*. Vol. 1. 2001.
53. Larman, C., *UML y patrones: Introducción al análisis y el diseño orientado a objetos* 1999.