Universidad de las Ciencias Informáticas

Facultad 3

Título:”Biblioteca gráfica para el modelado de procesos de negocio usando BPMN.”

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor(es): Elizabeth Morales Macías.
Raidel Rosabal Herrera.

Tutor(es): Ing. Yoriangel Rivero González.
MSc. Pedro Manuel Nogales Cobas.

“La Habana, junio de 2013”

"Seamos realistas y hagamos lo imposible."

Ernesto Che Guevara de La Serna

DECLARACIÓN DE AUTORÍA

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

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

Elizabeth Morales Macías	Raidel Rosabal Herrera
_____	_____
Firma del Autor	Firma del Autor
Ing. Yoriangel Rivero González	MSc. Pedro Manuel Nogales Cobas
_____	_____
Firma del Tutor	Firma del Tutor

DEDICATORIA

Con mucho amor

A mi hermanito

A mami y papi

A mi abuelita Dolores

A mi abuelito Hilario en donde quiera que este

…Elizabeth Morales Macías…

Con mucho amor

A mi madre y a mi padre

A mi hijo y mis amigos

…Raidel Rosabal Herrera…

AGRADECIMIENTOS

Quisiera agradecer, primero a mi compañero de tesis Raidel Rosabal Herrera, que también fue mi compañero de grupo y de puesto por 4 años, por haberme aguantado y ayudado. Sé que sin ti nunca hubiese terminado este trabajo, que te costó mucho dolores de cabeza. Muchas gracias por haber compartido conmigo y no haberte rendido.

Agradezco a las dos personas más importantes para mí , las dos personas que me dieron la vida mi Mami y mi papi. Papi eres el hombre de mi vida, gracias por ser mi ejemplo y siempre he aprendido mucho de ti, gracias por confiar en mí. Mami que decirte... eres lo mejor de este mundo, gracias por ayudarme, por ser mi amiga, mi apoyo, por guiarme, por quererme tanto Te amo.

Agradezco a mi tutor Yoriangel Rivero González, por ayudarme con este trabajo y tener paciencia conmigo. A toda mi hermano y familia.

Mi compañero, mi amor gracias por apoyarme. Siempre serás mi Pipi.

Mis amigos, mi Janilla tu sabes.., gracias por todo eres como mi hermana. Mi Blankusa (Ley), imposible olvidarme de ti gracias por existir. A la gorda de arlis, a Yoandris, a Sahily.

Mis compañeros de grupo..., les agradezco a todos, son el mejor grupo que he tenido.

La gente del apartamento.. la flaca de Eimé, la otra mama, je poquito tiempo pero estas aquí.

A yoyi gracias por tenderme la mano. A la gente del sindicato. En fin a todos los que me ayudaron y conocí en estos 5 años.

Elizabeth Morales Macías

Primero quiero agradecer a mi mamá y mi papá. Por haber apoyado, en estos años y haber confiado en mí, los quiero mucho. Gracias por su ejemplo.

A mi hijo que es lo más lindo que tengo este mundo, gracias por existir y ser mi alegría.

Agradezco a mi hermano, y a toda mi familia, los quiero.

A mis amigos. Yasmani, Víctor , Diógenes, a Diana gracias por apoyarme.

A Laritza, por ser una buena madre.

A mi tutor… Yoriangel gracias por ayudarme.

A mis compañeros de grupo, Camilo, Luis Angel, Evelio por compartir estos 5 años y todos los demás.

A mi compañera de tesis Elizabeth.

En fin a todos me ayudaron y me apoyaron.

Raidel Rosabal Herrera

RESUMEN

El modelado de procesos de negocio (BPM[1]) es la base para comprender la operación de una organización; documentar y publicar los procesos que se llevan a cabo, buscando una estandarización, eficiencia en la operación e integración de soluciones en arquitecturas orientadas a servicios. Permite controlar todo el ciclo de vida de los procesos, teniendo así una visión de qué necesita la entidad para desarrollarse (White, 2009).

El marco de trabajo Sauxe cuenta con un modelador de procesos de negocio que implementa el estándar BPMN[2], el cual permite representar un conjunto de objetos de flujo, de datos, de conexión y contenedores que conforman la notación. Pero este conjunto de elementos viola ciertas restricciones de modelado, lo cual puede resultar en un diagrama de procesos de negocio mal formado y consecuentemente incorrecto, por lo que no se ajusta al propósito principal de BPMN, que es el de proporcionar un entendimiento común de los procesos a los principales involucrados en los mismos. Para suplir esta necesidad se implementa una biblioteca gráfica que permitirá definir el comportamiento de los elementos gráficos del modelador en el marco de trabajo Sauxe. Se realizó un estudio de los principales conceptos, y sistemas que modelan procesos de negocio en el mundo, posteriormente se evalúan los resultados obtenidos en la implementación mediante las diferentes pruebas de software.

Palabras Claves: biblioteca gráfica, modelado, procesos de negocio.

[1] **BPM:** por sus siglas en inglés Business Process Modeling.
[2] **BPMN:** por sus siglas en inglés Business Process Modeling Notation.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las denominadas Tecnologías de la Información y las Comunicaciones (TIC) ocupan un lugar central en la sociedad y la economía, impulsando la competitividad y el desarrollo económico, propiciando así que las organizaciones busquen medios para mejorar la eficiencia de los procesos de negocio que impactan positivamente en el desempeño financiero.

Los procesos de negocio son un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. Cada proceso de negocio tiene sus entradas, funciones y salidas. Las entradas son requisitos que deben tenerse antes de que una función pueda ser aplicada. También pueden ser vistos como una guía para hacer funcionar un negocio y alcanzar las metas definidas en la estrategia de negocio de la empresa. Las dos formas principales de visualizar una organización, son la vista funcional y la vista de procesos (Barros, 1994).

La informatización de los procesos de negocio permite que la información sea consumida por una o más personas al mismo tiempo en distintos lugares. Esto permite mejorar los circuitos administrativos internos de la organización, ya que la información puede tratarse simultáneamente en más de un área de la empresa (Zaratiegui, 1999).

El modelado de procesos de negocio (BPM) es un método sistemático para identificar, levantar, documentar, diseñar, ejecutar, medir y controlar tanto los procesos manuales, como los automatizados (Ruiz, 2006).

La Notación de Modelado de Procesos de Negocio (BPMN), proporciona una forma estándar de representar procesos de negocio, tanto para propósitos descriptivos de alto nivel, como para detallados y rigurosos entornos de software orientados a procesos (Zaratiegui, 1999).

La adopción cada vez mayor de la notación BPMN como estándar, ayudará a unificar la expresión de conceptos básicos de procesos de negocio; así como conceptos avanzados de modelado (Pantiagozo, 2002).

El correcto modelado de un proceso garantiza que quede representada la estructura del mismo, así como la manera en que se comportará un sistema; permitiendo identificar las interrelaciones existentes entre las actividades, comprender el papel de cada elemento que lo conforma y sus relaciones, buscar oportunidades de simplificación y reutilización o encontrar problemas existentes.

Cuba, al igual que el resto del mundo, no queda exenta de las ventajas que representan el modelado y gestión de los procesos de negocio en una empresa, y en consecuencia, adopta políticas con el fin de

lograr que los procesos de negocio se realicen más eficientemente cada día. Un ejemplo de ello se observa en la construcción de un marco de trabajo Sauxe para el desarrollo de aplicaciones web de gestión, que desarrolla entre sus componentes un modelador de procesos de negocios. Sauxe es desarrollado por el Centro de Informatización de la Gestión de Entidades (CEIGE), de la Universidad de las Ciencias Informáticas (UCI).

En este punto es importante resaltar algunos aspectos. BPMN impone un conjunto de reglas y restricciones que se deben cumplir a fin de obtener un correcto modelado de los procesos de negocio. Aunque en ocasiones algunas de estas reglas se puedan violar durante el modelado, existe un conjunto de reglas que no se deben violar, puesto que resultarían en un diagrama sintácticamente incorrecto. Las que pueden ser violadas desde el modelado son detectadas en una fase posterior de la gestión de los procesos, específicamente en la simulación, donde se detectan abrazos fatales, cuellos de botellas y ciclos infinitos. El objetivo de este trabajo no se centra en esta fase, por lo que queda fuera del alcance de este trabajo. Las reglas que no pueden ser violadas incluyen que se utilicen eventos de inicio sin flujos de secuencia entrantes, eventos de fin sin flujos de secuencia salientes, que los objetos de flujos solo sean modelados dentro de piscinas (pools) y carriles (lanes), que los conectores no atraviesen los límites de la piscina, por solo citar algunos ejemplos. Adicionalmente, el modelado de los procesos de negocio debería permitir que los objetos de flujos y los contenedores se puedan redimensionar, cambiar y modificar los orígenes de coordenadas, además de actualizar los objetos de conexión cada vez que un objeto de flujo cambia las coordenadas de origen.

Los objetos de flujo limitan su movimiento a un área determinada por el elemento contenedor, y cada objeto de flujo por separado define atributos, responsabilidades y operaciones que definen su comportamiento como instancia de elemento gráfico. El modelador de procesos de negocio de Sauxe en su estado actual viola estos aspectos mencionados, lo que trae consigo la imposibilidad de obtener una definición de procesos de negocio sintácticamente correcta. Con el objetivo de dar solución a estos problemas se define como **problema a resolver:** ¿Cómo contribuir a la definición del comportamiento de los elementos gráficos en el marco de trabajo Sauxe, cumpliendo con las especificaciones de BPMN?

Como **objetivo general:** Elaborar una biblioteca gráfica que permita la definición del comportamiento de los elementos gráficos en el marco de trabajo Sauxe, usando las especificaciones de BPMN.

Definiéndose como **objeto de estudio**: Sistemas de gestión de procesos de negocio. Investigando en **el campo de acción**: Modelado de procesos de negocio utilizando la notación BPMN.

Objetivos específicos:

- Elaborar el marco teórico de la investigación a partir de un análisis crítico de los sistemas de gestión de procesos de negocio existentes y su modelado con la notación BPMN.
- Diseñar la biblioteca gráfica para el modelado de procesos de negocio utilizando BPMN.
- Implementar la biblioteca gráfica para el modelado de procesos de negocio utilizando BPMN.
- Validar el resultado obtenido a través de la aplicación de pruebas de caja negra.

Métodos Teóricos

- Histórico – Lógico: Es usado para sistematizar las tendencias históricas y actuales del modelado de procesos de negocios y determinar su influencia en el problema actual de la investigación.
- Analítico – Sintético: Es utilizado para realizar un análisis de la información empleada para la investigación, así como las especificaciones según BPMN del modelado de procesos de negocio, y realizar una valoración de las diferentes características del marco de trabajo Sauxe y poder obtener la solución centrada en los objetivos de la investigación.
- Modelación: Para modelar los nuevos requisitos funcionales que se proponen en la solución, su relación con los procesos actuales y el diseño de los nuevos componentes visuales, las entidades relacionales y sus características.

Métodos Empíricos

- Entrevistas: Para realizar el estudio de los procesos actuales que se desarrollan en el marco de trabajo de Sauxe, específicamente en el modelador de procesos de negocio (Workflow[3]) existente, sus características y determinar los problemas que presenta la aplicación.
- Observación: Para determinar el resultado final del producto y su influencia en la integración con el actual sistema.
- Experimento: Para establecer las diferentes pruebas de calidad con el propósito de determinar la fiabilidad del sistema y el mejoramiento de la usabilidad del mismo, a través de las pruebas de caja blanca y negra.

Es por ello que se define como **Idea a defender** en la presente investigación: El desarrollo de una biblioteca gráfica, permitirá definir el comportamiento de los elementos gráficos en el marco de trabajo Sauxe.

[3] **Workflow:** Flujo de trabajo

Para una mejor comprensión de cada fase de la investigación, se decidió estructurar el trabajo en tres capítulos:

Capítulo 1: “Fundamentación Teórica”.

Capítulo 2: “Análisis y diseño de la Biblioteca Gráfica”.

Capítulo 3: “Implementación y Prueba”.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se realiza la fundamentación teórica del trabajo, donde se describen los principales conceptos relacionados con el modelado de procesos de negocio. Además se analizan dentro del estado del arte sistemas que modelan procesos de negocio. Incluye también la descripción de los lenguajes, herramientas y framework que serán usados en el desarrollo de la solución.

1.1 Bases Conceptuales

1.1.1 Proceso

Un proceso se define como un conjunto de actividades o una secuencia de las mismas, que se realizan o ejecutan bajos determinadas circunstancias con un fin definido que transcurre con el paso del tiempo. Proceso en informática se describe como “distintas combinaciones operativas” que se realizan para lograr un resultado o un producto (ABC, 2013).

Los autores de esta investigación hacen mayor enfoque en el concepto que describe la norma internacional ISO-9001, que define un proceso como “una actividad que utiliza recursos, y que se gestiona con el fin de permitir que los elementos de entrada se transformen en resultados” (ISO-9001, 2005).

Este concepto se ajusta mejor a la definición de un proceso en la informática, ya que al ejecutar actividades, como por ejemplo la instalación de un nuevo software o la consecución de un análisis antivirus, consume recursos y convierten entradas en salidas.

1.1.2 Enfoque basado en proceso

El enfoque basado en proceso es un término que se define cuando una entidad utiliza un sistema de procesos aparejado a su selección, interacción y gestión para lograr el resultado esperado. Este enfoque permite el control continuo de la relación, combinación e interacción “entre los procesos individuales dentro del sistema de procesos” (ISO-9001, 2008).

1.1.3 Proceso de negocio

Después del análisis de varias bibliografías se puede definir un proceso de negocio como:

- Un grupo o conjunto estructurado de actividades que puede ser medido con el fin de diseñar un producto especificado o solicitado por un cliente determinado (Benghazi, 2009).

- Una colección de actividades que tomando una o varias clases de entradas, crean una salida que tiene valor para un cliente (Ruiz, 2006).

Según Roger Burlton, fundador del Grupo de Procesos de Renovación BPM, se puede definir como todas las actividades que deben realizarse para satisfacer las necesidades de los usuarios de una organización. Los autores de esta investigación, concluyen que un proceso de negocio son todas las actividades que tiene un objetivo o producto final, que para lograrlo convierten entradas en salidas, consumen recursos, proporcionado información para un cliente.

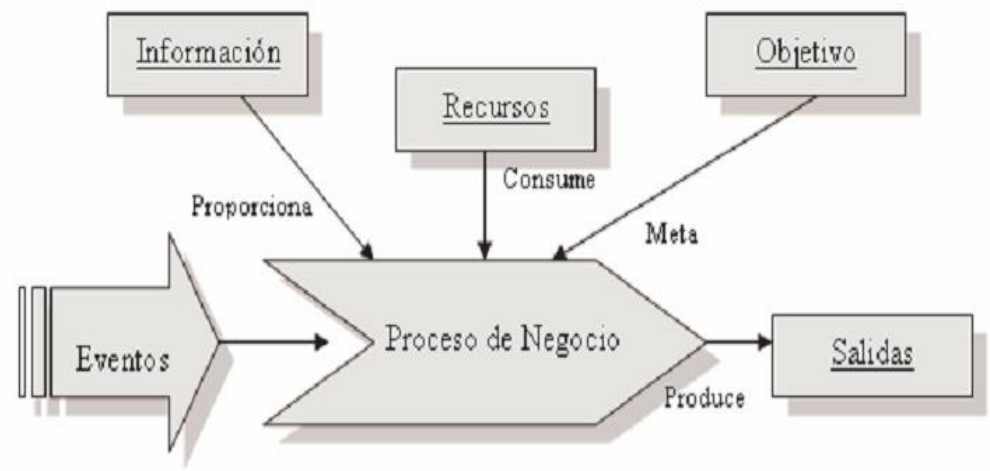


Figura 1: Proceso de Negocio

Tomado de Tecnología para la Gestión de Procesos de Negocio, Francisco Ruiz, 2006

1.1.4 Modelado de procesos de negocio

Modelado de Procesos de Negocio (BPM, por sus siglas en inglés), retoma todas las tecnologías, herramientas y técnicas desarrolladas durante las etapas de su evolución en un todo unificado. Para mejorar los procesos de negocio se suelen utilizar programas de gestión. Con los avances en la tecnología de los grandes proveedores de plataformas, la visión de convertirse en modelos BPM totalmente ejecutable (y capaz de simulaciones y de ingeniería de ida y vuelta) se acerca a la realidad todos los días (Ruiz, 2006).

Los modelos de negocio “son un conjunto de técnicas y representaciones gráficas plasmadas sobre una base de datos orientada a objetos y basados en estándares” que permiten representar y entender cuáles

son las perspectivas, los problemas, oportunidades e indicadores de gestión de la calidad, siendo la satisfacción del cliente el propósito fundamental (Oroya, 2010).

El modelado de procesos es adoptado por diferentes empresas con tres objetivos fundamentales:

- **Documentar:** los procesos son parte fundamental de la organización de una empresa y un elemento primordial cuando se intenta implementar modelos de calidad como ISO.
- **Mejorar:** las empresas que buscan una mayor eficiencia en sus procesos, localizar cuellos de botella en su gestión, identificar área de oportunidad o mejora, recurren al modelado y la simulación de procesos.
- **Agilizar:** en un nivel de mayor sofisticación, las empresas requieren el modelado de procesos como articuladores de los servicios de Tecnologías de Información, para poder reaccionar con mayor agilidad a los constantes cambios que exige la competencia actual (León, 2013).

1.3 Lenguaje de modelado

1.3.1 BPMN

BPMN permite el empleo de una forma estandarizada gráfica para representar los procesos de negocio. Tiene como principal objetivo proporcionar una notación que sea fácilmente comprensible por todos los usuarios de negocios, crea un puente estandarizado para la brecha entre el diseño de procesos de negocio y la implementación de procesos. BPMN proporcionará un medio sencillo de comunicación de la información del proceso a los usuarios empresariales, ejecutores de otros procesos, clientes y proveedores (Hitpass, 2011).

Como principales características de BPMN:

- Proporciona un método normalizado para representar procesos de negocio.
- Facilita su entendimiento debido a la poca complejidad de su notación.
- Proporciona un lenguaje común entre los usuarios de negocio y los técnicos.
- Facilita la diagramación de los procesos de negocio (White, 2009).

1.4 Herramientas para el modelado de proceso de negocio

El objetivo del análisis de algunas herramientas que modelan procesos de negocio, es utilizar algunas de sus características de diseño en el modelador del marco de trabajo Sauxe. En la actualidad existen varias

aplicaciones que permiten el modelado de procesos. Durante la investigación se encontraron herramientas que son difundidas y utilizadas en distintos niveles por la OMG[4] y la WFMC[5].

1.4.1 jBPM

Es una suite de administración de procesos de negocios flexible (BPM). Esto hace el puente entre los analistas de negocio y desarrolladores. Tradicionalmente los Motores BPM tienen un enfoque que se limita únicamente a personas sin conocimientos técnicos. Es una excelente herramienta para entender muchos de los conceptos básicos que hay detrás de BPM (Cummunity, 2013).

1.4.2 Bizagi

Bizagi es la solución líder de Business Process Management (BPMS) para una automatización de procesos más rápida y flexible. Poderosa y sencilla Suite de BPM, diseñada para resolver problemas de negocio reales. BizAgi es una herramienta BPM gratuita que controla el ciclo de vida completo de la administración de procesos de negocios a través de la realización de tres pasos. BizAgi permite diseñar gráficamente los procesos mediante la utilización de BPMN. Permite importar procesos modelados previamente en otras herramientas que soporten dicho lenguaje (Bizagi, 2013).

1.4.3 Intalio

Es un software de código abierto basado en Java-J2EE, que implementa BPMS, y está basado en un conjunto de frameworks y arquitecturas muy conocidas en la industria del software y con una madurez aceptable. Es una plataforma basada en Apache ODE (motor BPEL), cuenta con un diseñador basado en Eclipse llamado Intalio Designer, el cual al ser salvado el diseño del modelo de negocio con las especificaciones de BPMN se genera automáticamente código BPEL[6], luego es ejecutado por el componente Intalio Server (Mayora, 2010).

1.4.4 Bonita

Se utiliza para modelar gráficamente un proceso de negocio con la notación BPMN y generar procesos que permitan automatizar los procesos de la organización. Bonita Studio contiene un modelador que permite dibujar la gráfica de flujo del proceso, y aplicar conectores para conectar el proceso a sistemas de

---

[4] OMG: Object Management Group.
[5]WFMC: Workflow Management Coalition.
6 BPEL: Business Process Execution Language.

información externos, como base de datos, correo electrónico, calendario, LDAP[7], ERP[8], etc. Varias formas de datos se pueden definir, los cuales son persistentes durante toda la ejecución del proceso. La Licencia del producto es GPL[9]. Se puede ejecutar bajo plataforma Linux o Windows (bonitasof, 2013).

1.4.6 Tibco Business Studio

Software de modelado de negocio basado en los estándares que permite a los expertos en negocios modelar, implementar, desplegar y manejar procesos de negocios. Primer producto de modelado para usuarios de negocio con funcionalidad completa y basada en estándares que se ofrece sin coste alguno (Orozco, 2010).

1.4.7 Herramientas WebSphere

Herramienta premier de IBM de análisis y modelado de procesos de negocios para usuarios de negocios. Este ofrece modelado de procesos, simulación y capacidades de análisis para ayudar a los usuarios a entender, documentar y desplegar procesos de negocios para mejoramiento continuos. Transforma los modelos para optimizar el comportamiento del tiempo de ejecución y comparte el modelo durante todo el ciclo de vida del proceso (Orozco, 2010).

1.5 Modelo de desarrollo

El modelo de desarrollo a emplear como guía de este producto es el Modelo de desarrollo del Centro de Informatización de la Gestión de Entidades (CEIGE) versión 1.1. En dicho modelo se incluye la especificación de las actividades de cada una de las fases del ciclo de vida de los proyectos del CEIGE, teniendo en cuenta los procesos de CMMI (Capability Maturity Model Integration) nivel 2 para la Universidad de las Ciencias Informáticas (UCI). Se detallan por tanto, los artefactos a generar en cada momento independientemente de las herramientas o métodos que se utilicen para ello. El modelo describe el ciclo de vida de los proyectos el cual está compuesto por fases:

[7] **LDAP**: Lightweight Directory Access Protocol.
[8] **ERP:** Enterprise Resource Planning.
[9] **GPL**: Licencia Pública General de GNU.

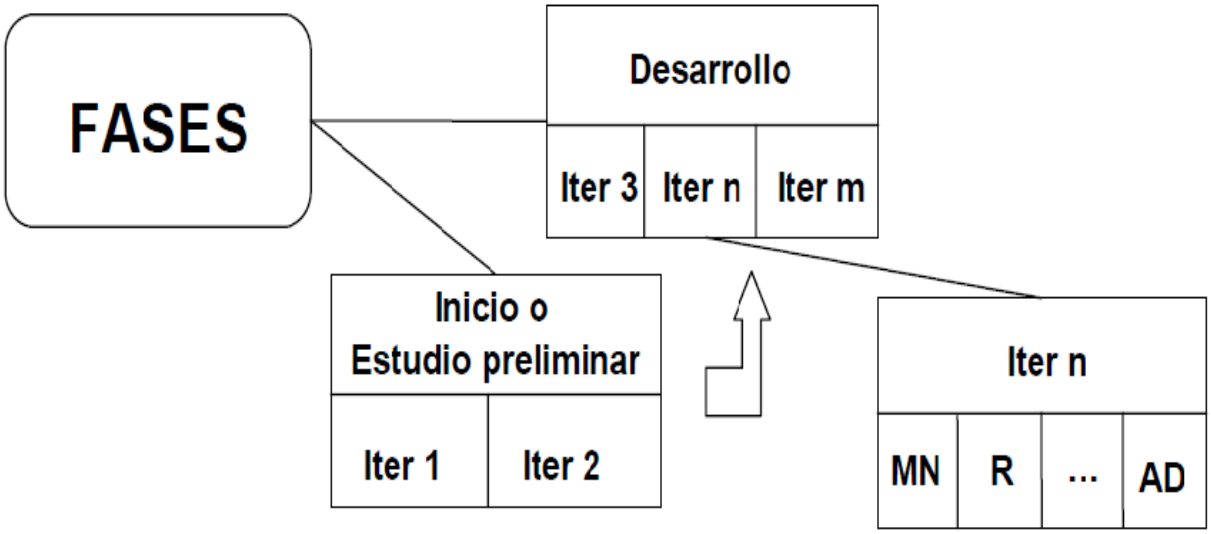


Figura 2: Fases del ciclo de vida de proyectos del CEIGE.

Tomado de Modelo de desarrollo del centro CEIGE

Inicio o Estudio preliminar: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel, la evaluación de la factibilidad del proyecto y el registro de este. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, y decidir si se ejecuta o no el proyecto. Los objetivos de la fase son: (Entidades, 2012)

- Asegurar la factibilidad del proyecto.
- Establecer un plan para la ejecución del proyecto.

Desarrollo: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se refinan los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. El objetivo de esta fase es obtener un sistema que satisfaga las necesidades de los clientes y usuarios finales. En esta fase se ejecutan las disciplinas Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas y Pruebas de liberación (Entidades, 2012).

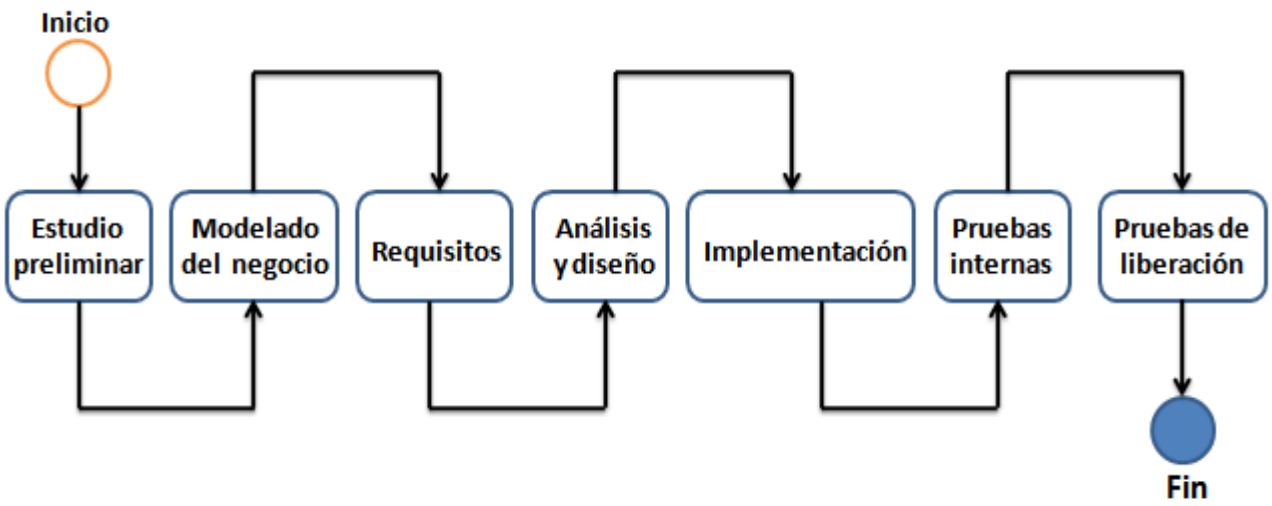


Figura 3: Ciclo de vida de proyectos del CEIGE.

Elaboración propia a partir de Modelo de desarrollo del centro CEIGE

1.6 Herramientas y tecnologías empleadas para el desarrollo

1.6.1 Lenguajes de programación

PHP 5.3

Preprocesador de Hipertexto PHP (por sus siglas en inglés) es un lenguaje de programación de alto nivel orientado al desarrollo de aplicaciones Web, que es interpretado del lado del servidor. Presenta gran rapidez en la ejecución y los bajos requerimientos de consumo en los sistemas donde es desplegado. Se integra perfectamente a la mayoría de los Sistemas Gestores de Bases de Datos.

JavaScript 3.0

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. 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.

1.6.2 Librerías y marcos de trabajo

Sauxe 2.0

Contiene un conjunto de componentes reutilizables que provee una estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor agilidad en el proceso de desarrollo. Se implementa con una arquitectura en capas, que utiliza Ext[10] para implementar la capa de presentación, se apoya en Zend, una extensión de Zend Framework para el desarrollo de la lógica del negocio y para la gestión de los datos, utiliza Doctrine[11]. En la capa de Control o Negocio se emplea el patrón de arquitectura Modelo Vista Controlador (MVC).

Zend Framework 1.4

Se trata de un framework para desarrollo de aplicaciones Web y servicios Web con PHP, brinda soluciones para construir sitios web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5. Este framework está formado por una serie de métodos estáticos y componentes que usarán estos métodos. Entre los componentes que se encuentran tienen vital importancia: Zend_Config para temas de configuración de aplicaciones web, Zend_Db para tratar con bases de datos, Zend_Search o Zend_Feed entre otros (Esser, 2009).

ExtJS 3.3.1

Es una librería JavaScript, código abierto, de alto rendimiento para la creación y desarrollo de aplicaciones Web dinámicas. Provee interfaces gráficas de usuario que brindan experiencias parecidas o iguales a las que se tienen con aplicaciones de escritorio. Permite la creación de aplicaciones complejas utilizando componentes predefinidos. Es extensible para la gran mayoría de los navegadores, evitando el problema de validar el código para cada uno. Entre sus principales ventajas se encuentra el balance entre Cliente-Servidor, distribuyendo la carga de procesamiento en el último, y este al tener menor carga, maneja los clientes de manera más eficiente.

Raphael JS 2.0

Biblioteca implementada en JavaScript que simplifica el trabajo con gráficos vectoriales en la WEB, permite crear gráficos vectoriales usando SVG (Scalable Vector Graphics).

Gráficos Vectoriales Escalables (SVG por sus siglas en inglés) son una especificación para describir gráficos vectoriales bidimensionales, tanto estáticos como animados en formato XML. El SVG permite tres tipos de objetos gráficos: elementos geométricos vectoriales por ejemplo: caminos consistentes en rectas y curvas, y áreas limitadas por ellos, imágenes de mapa de bits o digitales y texto.

---

[10] **Ext:** Extended file system.

[11] **Doctrine**: Mapeador de objetos-relacional.

1.6.3 Herramientas

Apache 2.0

Es un servidor web gratuito, potente y que ofrece un servicio estable y sencillo de mantener y configurar. El mismo incluye entre sus características el ser multiplataforma, incluye amplias librerías de PHP y Perl a disposición de los programadores, contiene diversos módulos que permiten incorporarle nuevas funcionalidades, los cuales son muy simples de cargar.

NetBeans 7.1

Es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Presenta entre sus características que cuenta con soporte para procesadores de anotaciones en el editor, configurable en las propiedades del proyecto, auto-completado de código y links para atributos de CSS, soporte para PHP Zend Framework (NetBeans, 2005).

Visual Paradigm 8.0

Es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. El mismo permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación, proporcionando abundantes tutoriales. Soporta UML versión 2.1, permite modelado colaborativo con CVS y Subversión, 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.

Conclusiones parciales

Después de la culminación de este capítulo se pudo arribar a las siguientes conclusiones:

- La construcción del marco teórico permitió conocer los diferentes conceptos relacionado con la investigación, destacándose el concepto de BPM como principal concepto, ya que es propósito de los autores que se logre comprender que la representación de los procesos de una organización, va a lograr satisfacer las necesidades de un cliente y un mejor entendimiento del negocio.
- Se identificó como modelo de desarrollo a emplear el modelo de desarrollo del centro CEIGE en su versión 1.1.

- A partir del análisis y el estudio realizado de los sistemas de modelado de procesos de negocio, se utilizará para la implementación de la biblioteca gráfica algunas de las características visuales de Bizagi, por ser la herramienta a la que se tuvo acceso.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA BIBLIOTECA GRÁFICA

Introducción

La descripción de los principales conceptos relacionados con la investigación, el estudio de los lenguajes y herramientas de modelado, así como la metodología de desarrollo a utilizar; permite el análisis y diseño de la solución, teniendo en cuenta los artefactos que se generan en cada fase del ciclo de vida de los proyectos de CEIGE.

2.1 Modelado del Negocio

Es la fase destinada a comprender los procesos de negocio y comprender cómo funciona el negocio que se desea automatizar para tener garantías de que el software a desarrollar va a cumplir su propósito. Se deben identificar y analizar los procesos que se llevan a cabo en el negocio que se desea automatizar u obtener (Entidades, 2012).

2.1.1 Modelo conceptual

En la etapa del modelado de negocio según el modelo de desarrollo del centro CEIGE uno de los artefactos que se generan es el Modelo conceptual. El modelo conceptual es una representación visual de los conceptos u objetos que son significativos para el problema. En la **Figura 4: Modelo Conceptual**, se representan las relaciones entre los conceptos Proceso Negocio y Elementos Gráficos como principales concepto del problema planteado. La descripción detallada de las clases del modelo conceptual se puede encontrar en el **Anexo 1.**

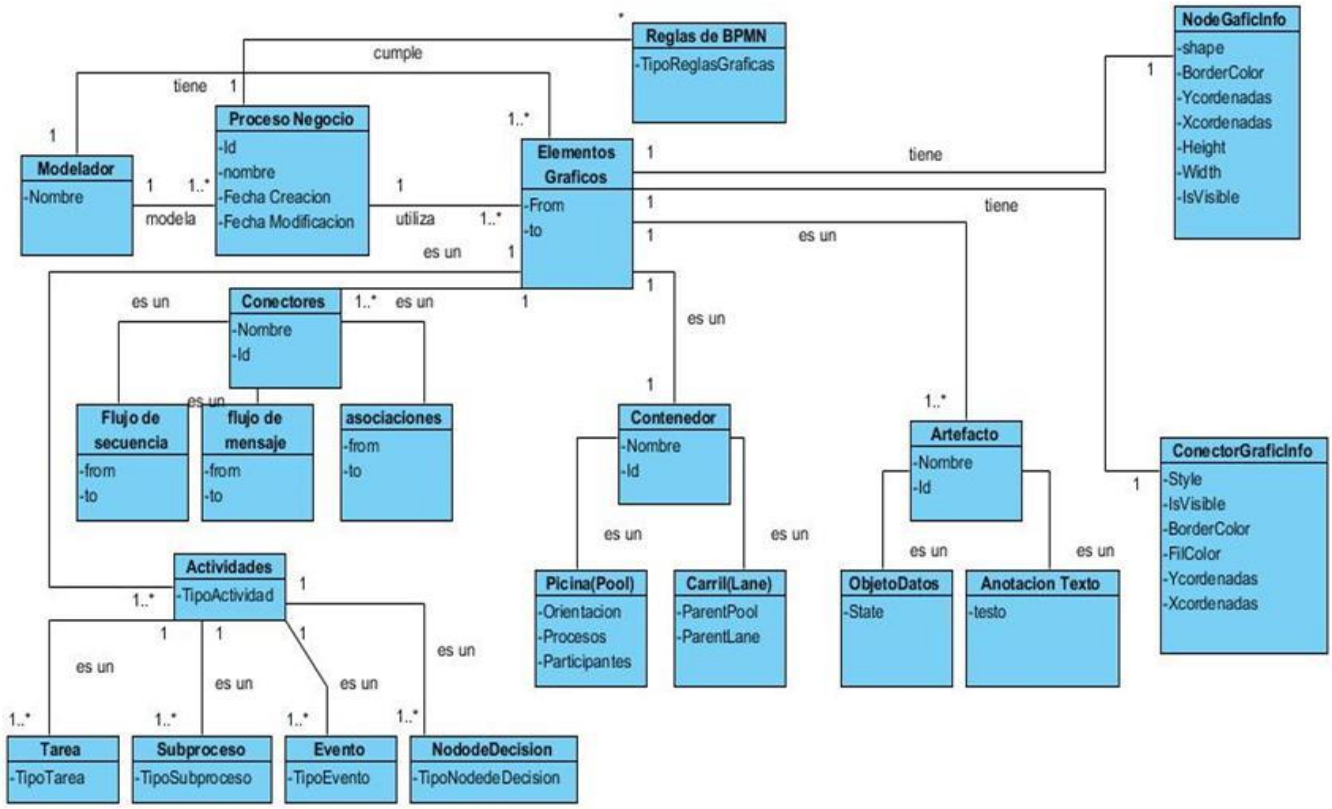


Figura 4: Modelo Conceptual

2.2 Requisitos de software

El objetivo de la definición de requisitos es obtener una clara comprensión del problema a resolver, extraer las necesidades del usuario y derivar de ellas las funciones que debe realizar el sistema. La tarea del análisis de requisitos es un proceso de descubrimiento, refinamiento, modelado y especificación que permite definir la función y el rendimiento del software, indica la interfaz y las restricciones que debe cumplir el software.

Luego de ser precisados los requisitos con que debe contar el sistema, debe lograrse una detallada descripción de cada uno de ellos, de forma que sea entendible para cualquier tipo de destinatario ya sea clientes y/o usuarios. Esta actividad tiene como entrada el listado de requisitos identificados y como salida de esta actividad se incluye un conjunto de artefactos que describen todas las interacciones que tendrán los usuarios con el software y que responden a los requisitos funcionales del sistema (Entidades, 2012).

2.2.1 Técnicas para la captura de los requisitos funcionales

La captura de requisitos permite el desarrollo de mejores sistemas que cumplan con las necesidades y expectativas del cliente. Para llevar a cabo este procedimiento existen diversas técnicas de captura de requisitos, las siguientes fueron usadas en esta investigación:

- **Revisión documental:** se realizó la revisión de todos los documentos de información donde aparecieron los indicadores principales relacionados con la modelación de procesos de negocio en el marco de trabajo Sauxe. Cada uno de los cuales aportó gran parte de la información necesaria.
- **Tormenta de ideas:** se realizó una reunión de varios interesados para expresar sus ideas sobre el problema en cuestión y valorar una posible solución. Cada participante expresó su criterio sin ser interrumpido por ningún otro. Al finalizar la sesión de lluvia de ideas se realizó una recolección de ideas sin duplicidad.

2.2.2 Requisitos funcionales

Los requisitos funcionales comprenden todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un software nuevo o modificado, tomando en cuenta las diversas necesidades de los clientes. Para el sistema propuesto fueron identificados los siguientes requisitos funcionales:

Requisitos de Eventos

RF1: Crear restricciones gráficas de desplazamiento en contenedores Pool para Eventos.

RF2: Crear restricciones gráficas de salida en contenedores Pool para Eventos de inicio.

RF3: Crear restricciones gráficas de entrada en contenedores Pool para Eventos de fin.

Requisitos de Tareas Atómicas

RF4: Crear restricciones gráficas de desplazamiento en contenedores Pool para Tareas atómicas.

RF5: Redimensionar Tareas Atómicas.

Requisitos de Flujo de Secuencia

RF6: Modificar restricciones gráficas de desplazamiento para flujo de secuencia no controlado.

Requisitos de Decisiones

RF7: Crear restricciones gráficas de desplazamiento en contenedores Pool para Decisiones.

Requisitos de Contenedores Pool

RF8: Crear restricciones gráficas de desplazamiento vertical para contenedores Pool.

RF9: Redimensionar contenedores Pool.

Requisitos de Objeto de datos

RF10: Crear Objeto de datos.

RF11: Crear restricciones gráficas de desplazamiento en contenedores Pool para Objeto de datos.

Requisitos de Área de texto

RF12: Crear Área de texto.

RF13: Modificar Área de texto.

RF14: Crear restricciones gráficas de desplazamiento en contenedores Pool para Área de texto.

Requisitos de Flujo de Mensaje

RF15: Crear Flujo de mensaje.

2.2.3 Descripción de Requisitos funcionales

Se describe el requisito funcional con más complejidad de implementación **Redimensionar Contenedores Pool**, se pueden encontrar las demás descripciones en el **Anexo 2**.

Descripción textual del requisito funcional **Redimensionar Contenedores Pool**

Este requisito debe permitir la modificación de tamaño de una pool sobre el lienzo, actualizando a las restricciones de movimiento de todos los elementos que estén contenidos dentro.

Precondiciones	Es creado al menos un tipo de contenedor Pool en el modelador.
Flujo de eventos	
Flujo básico	
1	El modelador muestra los contenedores creados.
2	Se selecciona el contenedor a redimensionar con doble clic.
3	Se redimensiona el contendor.
4	El modelador actualiza las nuevas coordenadas del contenedor cuando se aplica doble clic para deseleccionar el contenedor.
Pos-condiciones	
6 Se obtiene un contenedor Pool con el nuevo tamaño deseado.	
Flujos alternativos	
Flujo alternativo3.a El usuario no modifica el contendor	

7	Va al paso 4.
Pos-condiciones	
8	No se modifica el contenedor.
Validaciones	
	El modelador verifica que las coordenadas no excedan el límite máximo permitido.
	El modelador verifica que las coordenadas no estén por el límite mínimo permitido.
	El modelador verifica que el tamaño no sea menor que los elementos contenidos dentro.
	El modelador actualiza las nuevas coordenadas
Conceptos	Contenedores Pool

2.2.4 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el software debe tener, estas propiedades hacen que el software sea más atractivo, usable, rápido y confiable (Pressman, 2002). Los requisitos no funcionales de hardware y software que debe poseer la biblioteca gráfica son los establecidos por el centro CEIGE al inicio del proceso de desarrollo, a continuación se describen:

Tabla 1: Descripción de los requisitos no funcionales

	Cliente	**Servidor**
Software	Navegador Mozilla Firefox 3.0 o superior Sistema operativo Linux	Sistema operativo Linux en cualquiera de sus distribuciones Apache 2.0 o superior PHP 5.0 configurado con la extensión “pgsql” Ubuntu Server

Hardware	Procesador Pentium IV a 2GHz 1Gb de memoria RAM Tarjeta de red	Procesador Pentium V a 2.5GHz 2Gb de memoria RAM Tarjeta de red 1 Lector de CD 1 UPS Disco duro 160gb

2.2.5 Validación de Requisitos

La validación de requisitos permite definir que los requisitos especificados realmente detallan el sistema que el cliente necesita. Verifica que las especificaciones de requisitos no presentan omisiones, conflictos y ambigüedades, además de que sean correctas las interpretaciones por parte del equipo de desarrollo de software. Entre las técnicas utilizadas para validar los requisitos identificados se utilizó:

- **Revisiones**: Esta técnica se utiliza para corregir cualquier error existente en la documentación o modelado de los requisitos, con el objetivo de encontrar conflictos en el producto, y poder trazar alternativas de solución. El grupo de revisión técnica del Departamento de Tecnología validó los requisitos presentados en dos revisiones realizadas. Ver **Anexo 3**

2.3 Definición de la arquitectura del sistema

Como parte del proceso que define el modelo de desarrollo, se realizó el estudio y análisis a profundidad de los artefactos generados en el negocio y requisitos como paso importante para lograr su comprensión y poder tomar futuras decisiones al diseñar la arquitectura del sistema. En esta actividad se identifican los estilos arquitectónicos y patrones arquitectónicos propicios. (Entidades, 2012)

Teniendo en cuenta que el presente trabajo de diploma está centrado en el desarrollo de un componente para el marco de trabajo Sauxe, no es propósito de los autores crear una nueva arquitectura para dicho desarrollo y por tanto adopta la arquitectura definida por CEIGE.

2.3.1 Patrón arquitectónico Modelo-Vista-Controlador

El patrón conocido como Modelo-Vista-Controlador (MVC) separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres áreas diferentes:

- Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
- Vista: Maneja la visualización de la información.
- Controlador: Controla el flujo entre la vista y el modelo (los datos).

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo, independientemente de la representación visual.

Este patrón es empleado en la capa de control del marco de trabajo Sauxe y determina la estructura de los paquetes internos de los componentes a desarrollar. La solución propuesta se desarrolla sobre la capa de View del marco de trabajo donde implementa el patrón MVC.

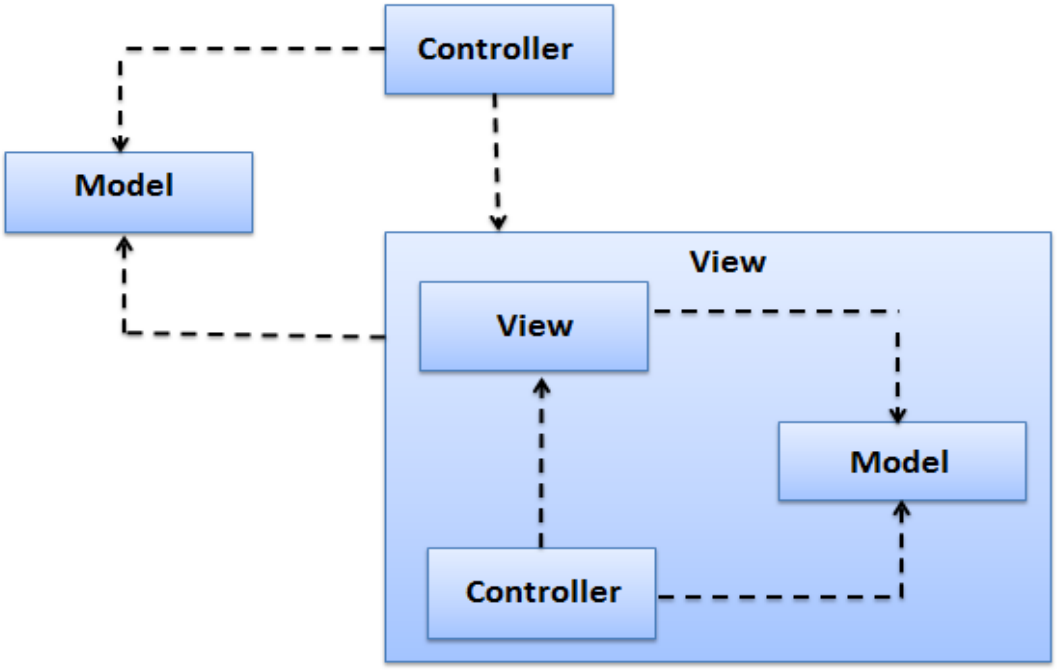


Figura 5: Arquitectura Modelo-Vista-Controlador

Elaboración Propia

2.4 Patrones de diseño

Los patrones de diseño son un conjunto de estrategias, o buenas prácticas, que pueden facilitar el trabajo del programador. La utilización de patrones de software en el desarrollo de una aplicación permite el ahorro de tiempo al programador, quien enfocará sus esfuerzos en el desarrollo de la lógica de la aplicación. Brinda una arquitectura uniforme a la misma, facilitando así su mantenimiento, modificación y expansión (Pressman, 2002).

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

2.4.1 Patrones GRASP[12]

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

- **Creador:** El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. El cual se evidencia en la clase graphicElement.Pool mediante la función add(), la cual realiza un conjunto de asignaciones de restricciones a los objetos dependiendo de las dimensiones de la misma.
- **Experto:** Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Plantea simplemente que cada objeto realiza la funcionalidad de acuerdo a la información que domina, la cuestión a la hora de diseñar es asignar responsabilidades a la clase que mayor información posee para cumplir con dicha tarea. Un ejemplo de la utilización de este patrón se puede ver en la clase graphicElement.Pool la cual contiene un conjunto de datos de único conocimiento, por lo que ella es la experta en el manejo de ellos. Esta clase implementa un conjunto de funcionalidades para el manejo de estos datos como por ejemplo la función move() la cual mueve a todos los elementos que contiene.

[12] **GRASP:** Patrones generales de software para asignar responsabilidades.

- **Controlador:** El patrón ofrece una guía para tomar decisiones sobre los eventos de entrada, asignando la responsabilidad del manejo de mensajes de los eventos del sistema a una clase controladora, ya que los elementos de interfaz y sus controladores de eventos, no deben ser responsables de controlar los eventos del sistema. El uso de este patrón se evidencia en la clase graphicElement.Base la cual como clase controladora tiene la responsabilidad de escuchar y responder a peticiones realizadas por los elementos Gráficos que heredan de ella.
- **Bajo acoplamiento:** El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Este patrón da soporte a una mínima dependencia y a un aumento de la reutilización; una clase con bajo acoplamiento no depende de "muchas otras" clases para realizar sus tareas, permitiendo que se pueda reutilizar con mayor facilidad y flexibilidad. Este patrón se puede evidenciar en la clase graphicElement.Base, puesto que esta clase implementa un conjunto de funcionalidades que son reutilizables por las clases hijas y que heredan al mismo comportamiento. Dando como resultado el bajo acoplamiento entre las clases y promoviendo la reutilización.
- **Alta cohesión:** La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realizan un trabajo enorme. Fomenta la reutilización, mejorando la claridad y facilidad del diseño. Este patrón se aplica en cada una de las clases, ya que son responsables de dibujarse dado que contienen la información necesaria para realizar estas funcionalidades.

2.4.2 Patrones GoF[13]

- **Singleton (Instancia única):** Dentro de los patrones GOF se clasifica en un patrón de creación a nivel de objetos. Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. La utilización de este patrón de diseño surgió a partir de la aparición de problemas a nivel de clases. Se necesitaba tener una única instancia del objeto Java Script que se iba generando a partir del modelo de proceso creado en la vista, por tanto se decidió utilizar una variante del patrón singleton que se encarga fundamentalmente de que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella. Este patrón se

[13]**GoF:** De su acrónimo en inglés Gang of Four**.**

evidencia en la clase raphaelObj puesto que en todo momento debe existir un Canva, la implementación de este patrón garantiza que solamente una instancia del Canva sea creada.

- **Observer (Observador):** Es un patrón de comportamiento que define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él. Este patrón es usado en nuestro diseño con el objetivo de lograr una coordinación entre las restricciones de los objetos gráficos. Un ejemplo del uso de este patrón lo podemos ver en la relación entre las clases graphicElement.Pool y graphicElement.Base las cuales realizan un una coordinación de las restricciones mediante eventos.

2.5 Modelado del diseño

Esta actividad tiene como objetivo diseñar los componentes más importantes y significativos de la arquitectura en una primera iteración, mientras que en próximas iteraciones dentro de este proceso se diseñan el resto de los componentes de acuerdo a su priorización. Se realiza una identificación y selección de los patrones de diseño a utilizar. El modelo del diseño es el encargado de modelar el sistema teniendo en cuenta las especificaciones de requisitos descriptas previamente. Este facilita la implementación en gran medida ya que en él participan las clases, las interfaces, los conceptos y sistemas necesarios para la solución.

2.5.1 Diagrama de clases del diseño con estereotipos web

Un diagrama de clases del diseño representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Permite visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de convencimiento. Está compuesto por los siguientes elementos: Clase: atributos, métodos y visibilidad. Relaciones: Herencia, Composición, Agregación, Asociación y Uso con sus respectivos estereotipos. Ver **Figura 7 Modelo de clases del diseño**.

Dentro del Domain se genera un diagrama de clases al que se hace referencia en el **Anexo 4.**

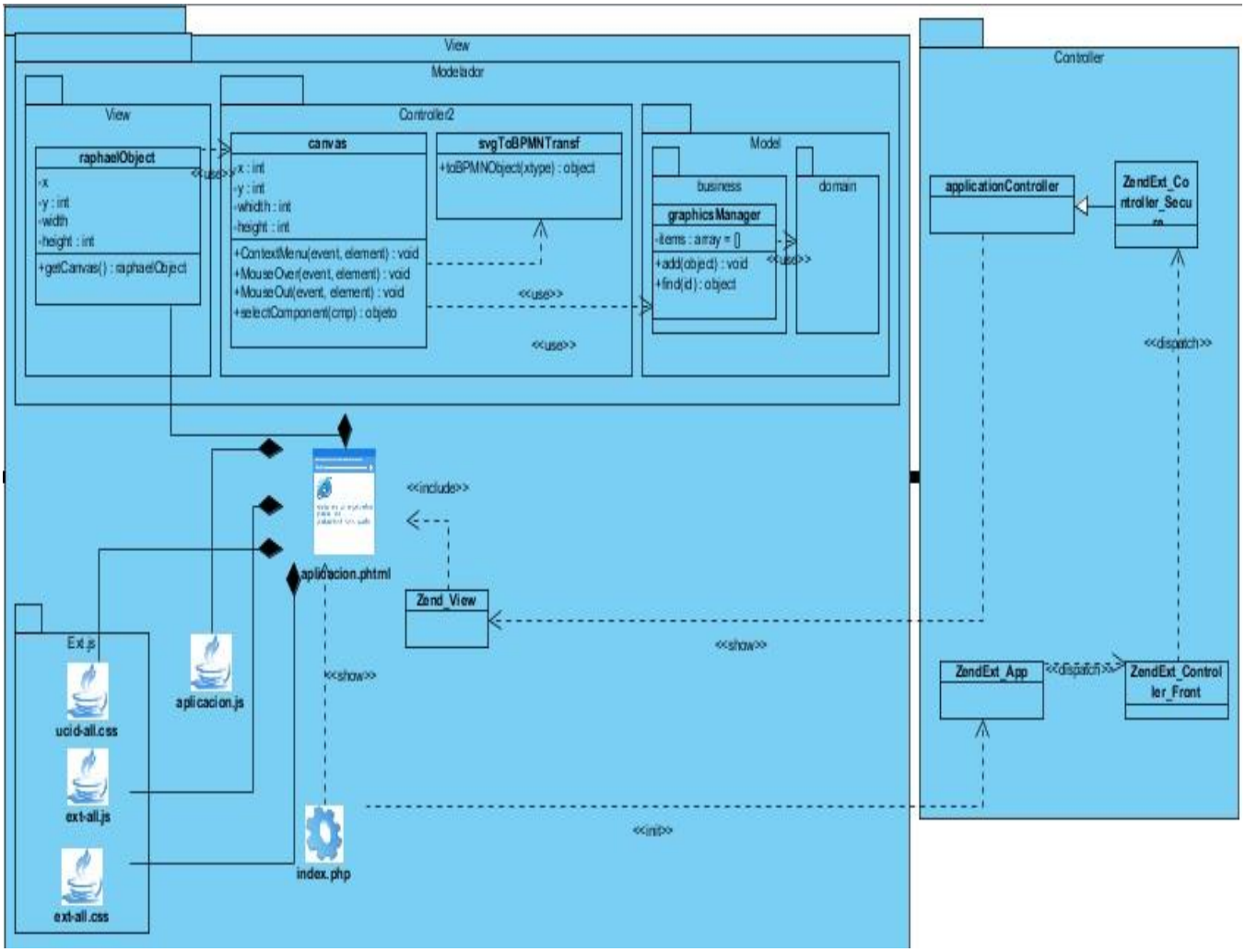


Figura 6: Diagrama de clases del diseño con estereotipos web

Conclusiones parciales

En el desarrollo del capítulo se obtuvo los siguientes resultados:

- Se definieron todos los requisitos, mediante los cuales se identificaron todas las funcionalidades requeridas para la implementación de la biblioteca gráfica.
- Se diseñó el diagrama de clases, lo cual permitió conocer la estructura y las relaciones de las clases que se manejan en la implementación.
- Con el uso de patrones de diseño se logró definir el comportamiento de las clases.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Introducción

En el siguiente capítulo se muestran en primer lugar los estándares de codificación, así como los diferentes artefactos que se desarrollaron en la fase de implementación, como el diagrama de componente y el diagrama de despliegue. Y se especifica el conjunto de validaciones y pruebas que evalúan la calidad del componente desarrollado.

3.1 Implementación

El objetivo principal de esta etapa es tomar como punto de partida el modelo de la fase anterior, se procede a implementar los diseños especificados en el modelo de diseño. La implementación es un proceso de varias fases que empieza cuando se crea una aplicación en el equipo de un desarrollador y termina cuando está instalada y lista para ejecutarse en el equipo de un usuario.

3.1.1 Estándares de Codificación

Uno de los instrumentos que facilitan la calidad del desarrollo del software es la adopción de estándares de estilo y codificación. Las convenciones de código son importantes para los programadores ya que permiten asegurar la legibilidad del código entre distintos programadores, proveer una guía para el encargado de mantenimiento y/o actualización del sistema con código claro y bien documentado; y facilitar la portabilidad entre plataformas y aplicaciones.

Para del desarrollo del Modelador se utilizan las notaciones:

- **Notación Húngara:** esta convención se basa en definir prefijos para cada tipo de datos y según el ámbito de las variables. También es conocida como notación REDDICK (por el nombre de su creador). La idea de esta notación es la de dar mayor información al nombre de la variable, método o función definiendo en ella un prefijo que identifique su tipo de dato y ámbito. (Ejemplo: raphaelObject).
- **CamelCasing:** Para nombrar los métodos y variables que se encuentran en las clases, definiendo que deben comenzar con minúscula y en caso de contener más de una palabra deben comenzar con letra mayúscula. (Ejemplo: raphaelObject).

3.1.2 Convenciones de Nomenclatura

A partir de las notaciones definidas para la codificación de las clases se establece una nomenclatura para las mismas. Los nombres de las clases comenzarán con minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing.

Las clases del modelo están definidas en el Ext.namespace14caxtor.ide. Los elementos que “se dibujan” agregan al nombre de espacio el sufijo graphicsElement, de modo tal que las clases del modelo que se dibujan están definidas en el nombre de espacio caxtor.ide.graphicsElements. A partir de este nombre de espacio se definen las clases, según su nombre.

Ejemplo: caxtor.ide.graphicsElements.eventoInicio define la clase eventoInicio, en el nombre de espacio caxtor.ide.graphicsElements, lo que indica que indica que eventoInicio define una clase cuyas instancias son “dibujables”.

- **Nomenclatura de las funciones:** Los nombres a emplear para las funciones se escribirán con minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing.
- **Nomenclatura de las variables:** Los nombres a emplear para las variables se escribirán con la notación húngara a excepción de los elementos que componen un BPMN definidos en el XSD, que permanecerán invariablemente con el nombre original, en caso de que sea un nombre compuesto se empleará notación CamelCasing. El nombre de las variables estará dado según la notación húngara, es decir se le agregará al nombre original un prefijo que describa el tipo de dato. A continuación se listan los prefijos a utilizar para cada tipo de dato.

Tabla 2: Prefijos para cada tipo de dato

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

- **Nomenclatura de los comentarios:** Los comentarios deben ser claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando. Es de gran importancia que tanto clases como métodos sean correctamente comentados para obtener un código fuente más legible y reutilizable, de manera que se pueda dar mantenimiento de manera más fácil a lo largo del tiempo. Antes de la declaración de una clase o método se escribirá una breve descripción donde se explique la intención del mismo. Este comentario tendrá la siguiente estructura:

Tabla 3: Nomenclatura de los comentarios

[14] **NameSpace:** Nombre de Espacio
Los espacios de nombres son en realidad objetos JavaScript que pueden contener propiedades y métodos (y para ello también las clases).

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

3.1.3

Diagrama de Componentes

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

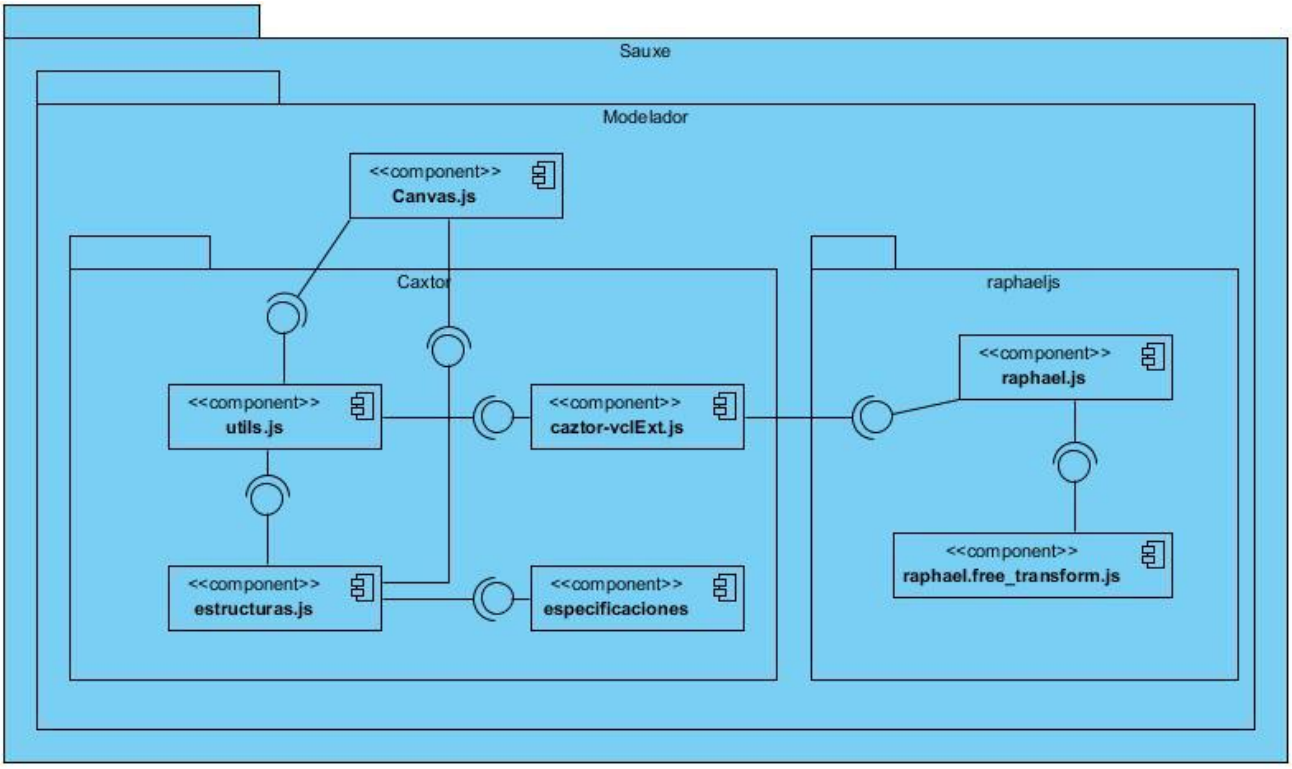


Figura 7: Diagrama de Componentes

Descripción de los Componentes:

- **Especificaciones:** Se encarga de la creación de formularios dinámicos para la especificación de valores de atributos de los objetos BPMN, a través de la operación editar Propiedades, que inspecciona el objeto por parámetro y crea controles para cada atributo editable que contenga el objeto, además de las restricciones de tipo de datos de atributos.
- **Caxtor-vclExt:** Define el conjunto de todos los elementos gráficos. Los elementos gráficos constituyen instancias de objetos que se dibujan sobre el Canvas, a través de la función draw(pintar).
- **Canvas:** Se encarga de representar los elementos gráficos, además de coordinar los eventos que fluyen entre el elemento gráfico y el objeto BPMN asociado a cada elemento. Brindan una funcionalidad cuyo objetivo principal es coordinar la creación de los elementos gráficos, la creación de los elementos BPMN que se asocian a los gráficos, las operaciones de administración de ambos tipos de elementos y de la representación del dibujo una vez que ha concluido con éxito las operaciones antes mencionadas.
- **Estructuras:** Define el conjunto de todos los elementos BPMN, además de la serialización de cada objeto sobre una estructura compleja llamada paquete a través de la invocación a la función

toJSON, y la cual representa el conjunto de todos los procesos de negocio modelados con BPMN. El resultado de esta serialización es la representación en Java Script ObjectNotation (JSON) para el envío de la información para el servidor.

- **Utils:** Define un conjunto de clases responsables, la creación de objetos BPMN a partir de la información recibida desde el componente Canvas, a través de la función toBPMNObject. Además define las clases que administran los paquetes de procesos, las instancias de clases BPMN que se crean, como la serialización de la instancia de paquetes y la petición al servidor, o de los procesos que se definen en Canvas.

3.2 Validación del diseño

Métricas de software

Un elemento clave de cualquier proceso de ingeniería de software es la medición. Se emplean las medidas para valorar la calidad de los productos de ingeniería o de los sistemas que se construyen. Las métricas técnicas para el software de computadora proporcionan una manera sistemática de valorar la calidad basándose en un conjunto de reglas definidas y al ingeniero del software una visión interna en el acto, lo que le permite descubrir y corregir problemas potenciales, así como, una indicación en tiempo real de la eficacia del análisis, del diseño y de la estructura del código, la efectividad de los casos de prueba, y la calidad global del sistema a construir (Digital, 2006).

Para la validación del diseño se utilizarán las métricas:

- **Tamaño operacional de clase (TOC)**: Se centran en el recuento de atributos y operaciones para cada clase individual y los valores promedio para el sistema orientado a objetos. Con la utilización de esta técnica el grado de aceptabilidad y calidad del diseño es directamente proporcional al nivel de reutilización de las clases e inversamente proporcional al grado de responsabilidad y complejidad de las mismas, por tanto entre menor sea el número de métodos en las clases, mayor es el por ciento de aceptabilidad y calidad en el diseño.
- **Relaciones entre clases (RC):** En esta técnica el grado de aceptabilidad y calidad del diseño es directamente proporcional a la reutilización de las clases e inversamente proporcional al acoplamiento, complejidad de mantenimiento y cantidad de pruebas de las mismas, por tanto mientras menor sea las relaciones entre clases, mayor es el porciento de aceptabilidad y calidad del diseño.

Las métricas TOC y RC posibilitan medir los siguientes atributos de calidad:

- **Responsabilidad:** Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.

- **Complejidad del mantenimiento:** Nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto. Puede influir significativamente en los costes y la planificación del proyecto.
- **Complejidad de implementación:** Grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización:** Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.
- **Acoplamiento:** Dependencia o interconexión de una clase o estructura de clase respecto a otras.
- **Cantidad de pruebas:** Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto diseñado.

3.2.1 Tamaño Operacional de Clase (TOC)

Para la evaluación de las clases fueron utilizados umbrales para el tamaño general, la responsabilidad, la complejidad y la reutilización de las clases.

Tabla 4: Tamaño operacional de Clase (TOC)

Atributo de calidad	Aumento del TOC	Categoría	Criterio
Responsabilidad	Implica un aumento de la responsabilic asignada a la clase	Baja	<= PO[15]
		Media	Entre PO y 2*PO
		Alta	> 2*PO
Complejidad de implementación	Implica un aumento de la complejidad implementación de la clase	Baja	<= PO
		Media	Entre PO y 2*PO
		Alta	> 2*PO
Reutilización	Implica una disminución del grado de reutilización de la clase	Baja	> 2*PO
		Media	Entre PO y 2*PO
		Alta	<= PO

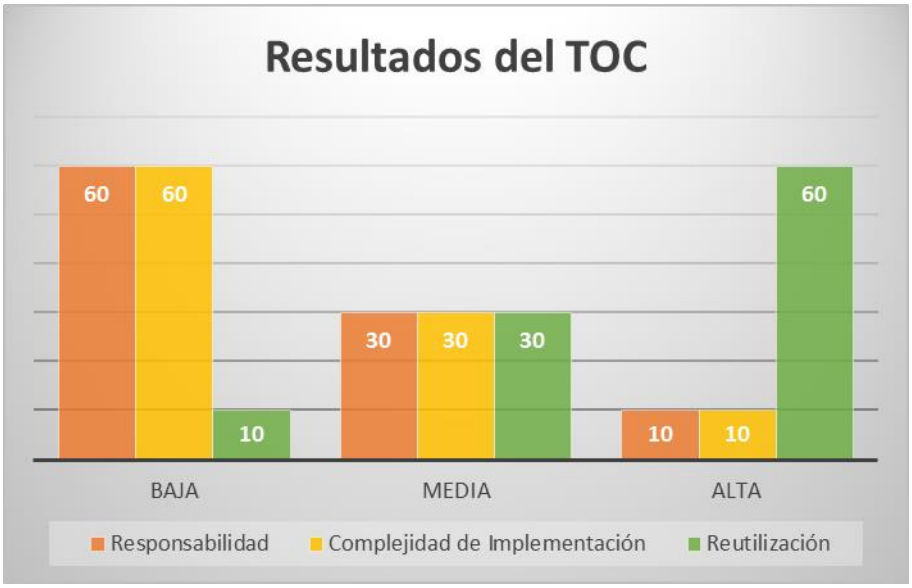
Resultados de la métrica TOC:

Durante la evaluación de la métrica TOC los resultados obtenidos demuestran que los atributos de calidad de las clases se encuentran en un nivel satisfactorio; de manera que se puede confirmar la elevada reutilización con un 60% y cómo se reducen la responsabilidad y la complejidad de implementación en un

[15] **PO:** Promedio Operacional

60%. A continuación el siguiente gráfico (tabla 8) recoge los resultados positivos obtenidos y para ver la representación en % de la incidencia de los resultados en cada atributo de calidad ver **Anexos 4.**

Tabla 5: Resultados de la métrica TOC



3.2.2 Relaciones entre Clases (RC)

Esta métrica está dada por el número de relaciones de uso de una clase con otra. Para la evaluación de las clases fueron utilizados umbrales para el acoplamiento, complejidad de mantenimiento, la reutilización y cantidad de pruebas.

Tabla 6: Atributos de calidad que afecta las RC

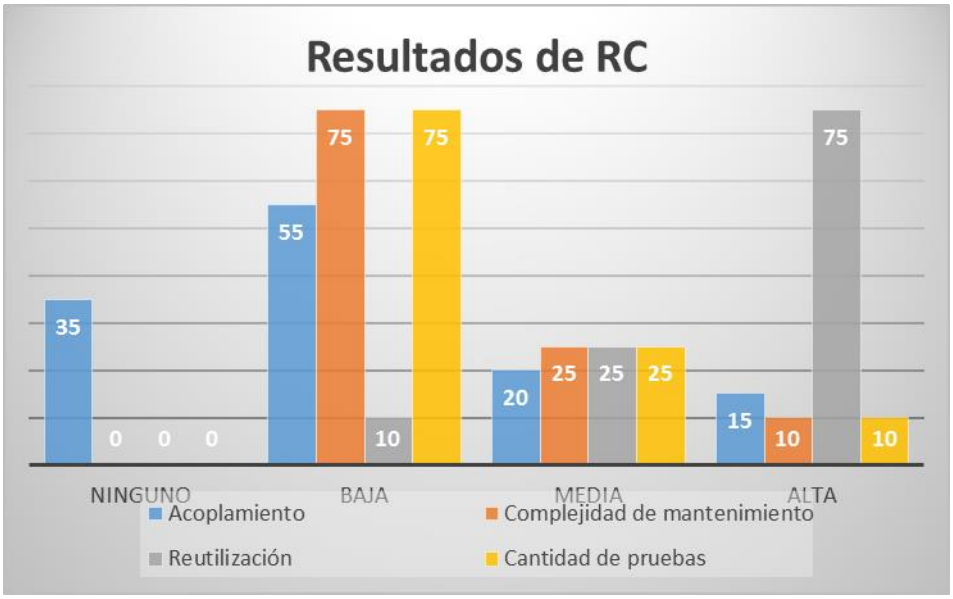
Atributo de calidad	Aumento del RC	Categoría	Criterio
Acoplamiento	Implica un aumento del acoplamiento de la clase	Ninguno	0
		Bajo	1
		Medio	2
		Alto	>2
Complejidad de mantenimie	Implica un aumento de la complejidad de mantenimiento de la clase	Baja	<= PO
		Media	Entre PO y 2*P
		Alta	> 2*PO
Reutilización	Implica una disminución en el grado de reutilización de la clas	Baja	> 2*PO
		Media	Entre PO y 2*P

		Alta	<= PO
Cantidad de pruebas	Implica un aumento de la cantid de pruebas de unidad necesaria para probar una clase	Baja	<= PO
		Media	Entre PO y 2*P
		Alta	> 2*PO

Resultados de la métrica RC:

Luego de realizarse la prueba, los resultados obtenidos demuestran que los atributos de calidad de las clases se encuentran en un nivel satisfactorio; de manera que se puede confirmar la elevada reutilización con un 75%, una baja complejidad de mantenimiento y la cantidad de pruebas en un 75% y un bajo acoplamiento con un 55%. A continuación el siguiente gráfico muestra los resultados obtenidos y para ver la representación en % de la incidencia de los resultados en cada atributo de calidad ver **Anexos 5.**

Tabla 7: Resultados de la métrica RC



3.3 Pruebas

Durante esta fase se desarrollan las pruebas del grupo de calidad del centro verificando el resultado de la implementación. Permite identificar posibles errores en la documentación y el software, es decir requisitos que el producto debería cumplir y que aún no los cumple (Entidades, 2012).

3.3.1 Pruebas funcionales o de Caja Blanca

La prueba de caja blanca, denominada a veces prueba de cristal es un método de diseño de casos de prueba, que usa la estructura de control de diseño procedimental para obtener los casos de prueba. El

mismo permite obtener casos de pruebas que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en vertientes verdadera y falsa; ejecuten todos los bucles en sus límites y con sus límites operacionales; y ejerciten las estructuras internas de datos para asegurar su validez.

Entre las técnicas de prueba de caja blanca se selecciona aplicar a la solución propuesta la prueba de camino básico. La cual permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental, y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Para la realización de la misma, primero se procede a enumerar las sentencias del código de la funcionalidad **isWithinBoundaries()** la cual se puede ver en la **Figura 8**, y a partir del mismo se construye el grafo de flujo asociado.

```
function isWithinBoundaries() {
    if(callback.displayName =='Pool'){           1
    var containedObjects=callback.containedObjects; 2
    var a=getBBox()[0];  2   var b=getBBox()[1];  2
    var c=getBBox()[2];  2   var width=0;         2
    var height=0; 2
    var desplazamientoX =0;  2
    var desplazamientoY =0;  2
    var xd=this.ft.items[0].el.getX(); 2
    var yd=this.ft.items[0].el.getY(); 2
    var whidthd=this.ft.items[0].el.getWidth(); 2    var heightd=this.ft.items[0].el.getHeight(); 2
    callback.fireEvent('movementRestriction',{x:xd,y:yd, width:whidthd, height:heightd}); 2
     if(containedObjects[0]){ 3
         desplazamientoX=containedObjects[0].getBBoxWT().getBBox().x; 4
         desplazamientoY=containedObjects[0].getBBoxWT().getBBox().y;  4
                 for (i=0;i<containedObjects.length;i++){ 5
                     if(containedObjects[i].getBBoxWT().getBBox().x<desplazamientoX) 6
                         desplazamientoX=containedObjects[i].getBBoxWT().getBBox().x; 7
                     if(containedObjects[i].getBBoxWT().getBBox().y<desplazamientoY) 8
                         desplazamientoY=containedObjects[i].getBBoxWT().getBBox().y; 9
                     if(containedObjects[i].getBBoxWT().getBBox().x+containedObjects[i].getBBoxWT().getBBox().width > width) 10
                         width=containedObjects[i].getBBoxWT().getBBox().x+containedObjects[i].getBBoxWT().getBBox().width;   11
                     if(containedObjects[i].getBBoxWT().getBBox().y+containedObjects[i].getBBoxWT().getBBox().height>height) 12
                         height=containedObjects[i].getBBoxWT().getBBox().y+containedObjects[i].getBBoxWT().getBBox().height;  13
                 } 14
                 return {
             x:  a.x+5 <= desplazamientoX && b.x > width && b.x <= raphaelObject.a-8 && a.x >= 2.5,  15
             y:  a.y+5 <= desplazamientoY && c.y > height && a.y >= 5 && c.y<raphaelObject.b-10      15 }  ;
                     }
     return {
             x:  b.x <= raphaelObject.a-8 && a.x >= 2.5,  16
             y:  a.y >= 5 && c.y<385                      16
            } ;
            }
    else{
        var xpool=callback.movementRestriction.x; 17
        var ypool=callback.movementRestriction.y; 17
        var xobj=getBBox()[0].x; 17
        var yobj=getBBox()[0].y;            17
        return {
             x:  xpool+25<=xobj ,           17
            y:  ypool+5<=yobj               17
              };
        } 18
}
```

Figura 8: Código fuente de la funcionalidad isWithinBoundaries()

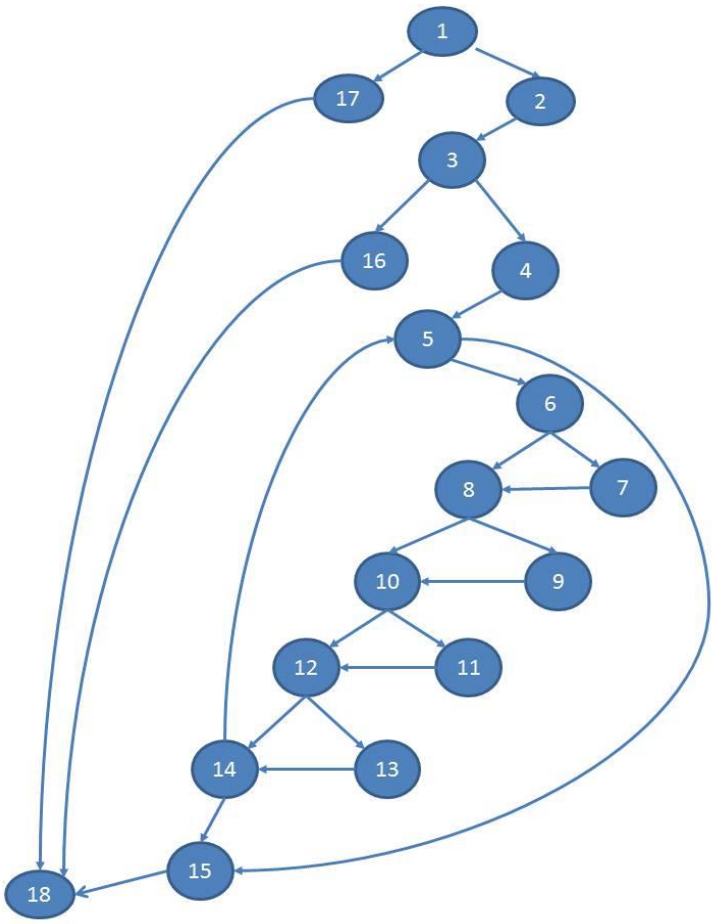


Figura 9: Grafo de flujo asociado a la funcionalidad isWithinBoundaries()

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

1. **V(G)= (A-N) + 2** // Siendo "A" la cantidad total de aristas y "N" la cantidad total de nodos.

   V(G)= (25-18) + 2
   V(G)= 9
2. **V(G) = P+1** // Siendo "P" la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

   V(G)=8+1
   V(G)= 9
3. **V(G)= R** // Siendo "R" la cantidad total de regiones, para cada formula "V(G)" representa el valor del cálculo.

V(G)= 9

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, dando como resultado 9, lo que indica que existen 9 posibles caminos donde el flujo básico puede circular, y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Seguidamente se representan los caminos básicos por los que puede recorrer el flujo:

Camino # 1: 1, 2, 3, 16, 18

Camino #2: 1, 17, 18

Camino # 3: 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 18

Camino#4: 1, 2, 3, 4, 5, 15, 18

Camino#5: 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 15, 18

Camino#6: 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 14, 15, 18

Camino #7: 1, 2, 3, 4, 5, 6, 8, 10, 11, 12, 14, 15, 18

Camino #8: 1, 2, 3, 4, 5, 6, 8, 10, 12, 13, 14, 15, 18

Camino #9: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 5, 15, 18

Para cada camino se realiza un caso de prueba, se pondrá un ejemplo de uno de ellos:

Caso de prueba para el Camino básico #2:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El valor de un atributo que tiene ese objeto **callback.displayName** será igual de **"Pool".**

Condición de ejecución: El valor de un atributo que tiene ese objeto **callback.displayName se llama "Pool"**.

Entrada: callback.displayName== "Pool"

Resultados esperados:

```
var xpool=callback.movementRestriction.x;
var ypool=callback.movementRestriction.y;
var xobj=getBBox()[0].x;
var yobj=getBBox()[0].y;
            return {
                x:  xpool+25<=xobj ,
 y:  ypool+5<=yobj
}
```

3.3.2 Pruebas funcionales o de Caja Negra

Las pruebas funcionales o de caja negra se centran en los requisitos funcionales del software. Permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los

requisitos funcionales de un programa. La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a base de datos externas, errores de rendimiento y errores de inicialización y terminación. A diferencia de la prueba de caja blanca, que se lleva a cabo previamente en el proceso de prueba, la prueba de caja negra tiende a aplicarse durante las fases posteriores de la prueba, ya que esta ignora intencionalmente la estructura de control y centra su atención en el campo de la información (Pressman, 2002).

Aplicación de la Prueba de Caja Negra:

Particiones de Equivalencia es la técnica empleada para confeccionar los casos de prueba de Caja Negra, la misma es una de las más efectivas al examinar los valores válidos e inválidos de las entradas existentes en el subsistema. Este método define un conjunto de clases de equivalencia o escenarios de pruebas donde serán analizados un conjunto de datos de entrada como a continuación se muestra en el requisito funcional **Crear restricciones gráficas de desplazamiento en contenedores Pool para Eventos**, tomado caso de estudio.

Condiciones de Ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.
- Se debe crear al menos un tipo de Evento**.**

Requisitos a probar

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Crear restricciones gráficas de desplazamiento en contenedores Pool para Eventos.	Al ser creado un Evento se crean las restricciones de desplazamiento dentro del contenedor que le permitirá al evento moverse dentro del contendor.	EP 1: Crear evento y moverlo dentro del contenedor.	– Se crea un contenedor de tipo Pool. – Se crea un Evento arrastrándolo hacia el contenedor Pool**.** – Al ser creado el elemento se crean las restricciones de desplazamiento. – Se desplaza el elemento por todo el contenedor sin

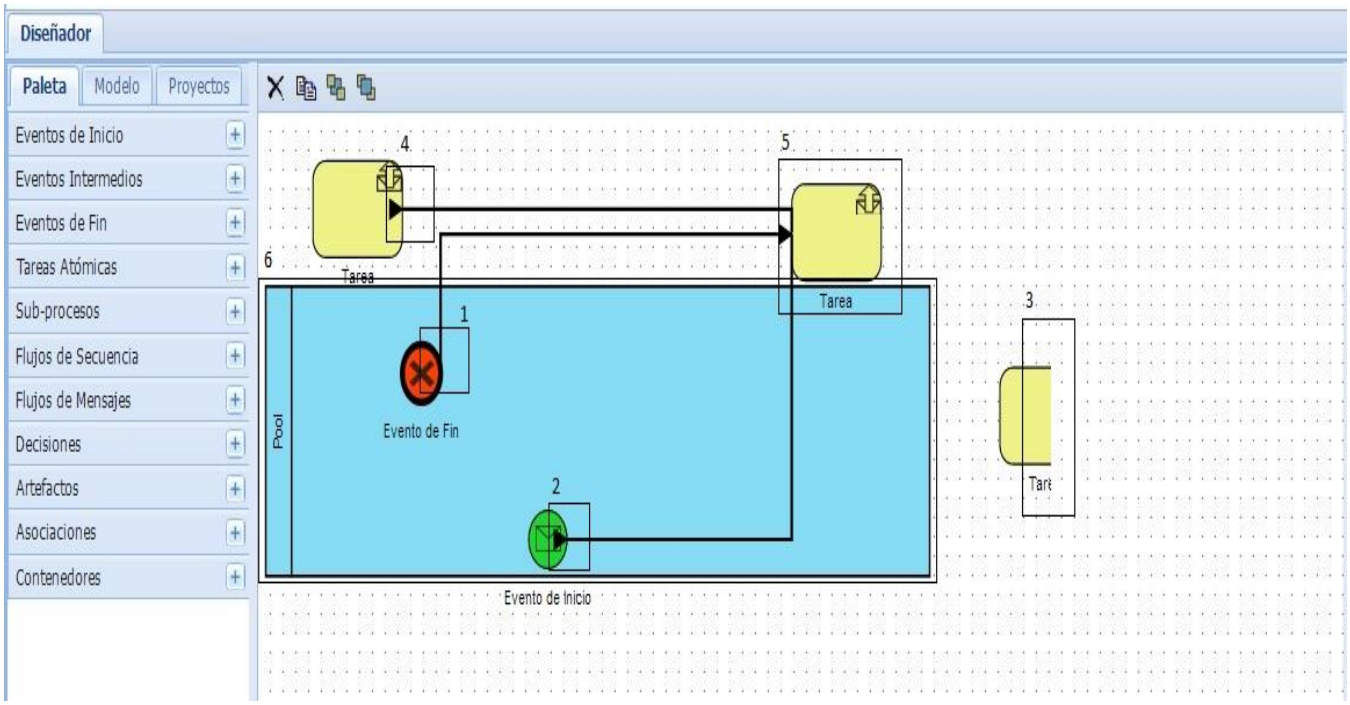
		que este pueda salir de las coordenadas del contenedor.
	EP1.2: Crear eventos fuera del contenedor.	– Se crea un Evento arrastrándolo hacia el lienzo**.** – El modelador no permite crear el elemento.

Como resultado se obtuvieron valores satisfactorios en el conjunto de casos de pruebas desarrollados. Para consultar el resto de los casos de pruebas realizados ver **Anexo 5.**

3.3.3 Validación de las variables de investigación

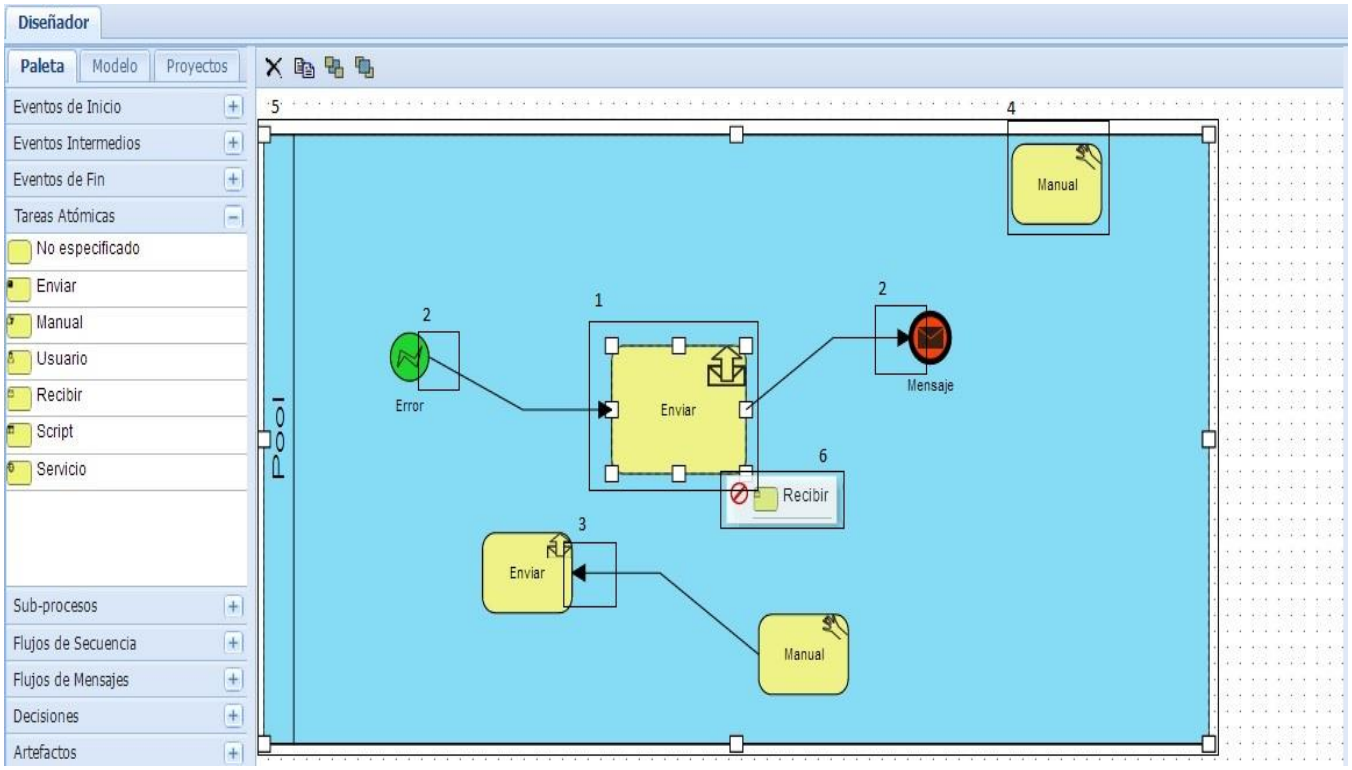
El mejoramiento del modelador con el desarrollo de la biblioteca gráfica, lo determina el comportamiento de los elementos gráficos en el proceso de creación de un diseño, como se plantea en la idea a defender al inicio de la investigación. A continuación la solución se somete a un análisis para verificar que en efecto se mejora el comportamiento de los elementos gráficos. Las restricciones sobre los elementos gráficos son las que definen el comportamiento de estos. En función de estas restricciones los elementos gráficos logran obtener características de comportamiento propias, dando como resultado un correcto modelado siguiendo las especificaciones de BPMN. A continuación se realiza una comparación en cuanto a las restricciones antes y después de implementar esta biblioteca gráfica, con el objetivo de demostrar el mejoramiento del comportamiento de los elementos gráficos.

Antes:



1. Los eventos de fin no deben dejar comenzar un flujo.
2. Los eventos de inicio no deben dejar terminar un flujo.
3. El área de modelado crea sus dimensiones de manera estática por lo que siempre tendrá las mismas dimensiones.
4. Los conectores se dibujan de una sola manera lo que provoca un mal dibujado en el diseño.
5. Los elementos no tienen restricciones de creado, por lo que se puede crear fuera de la Pool.
6. La Pool no tiene la propiedad de redimensionarse, ni ningún elemento. Esto trae como desventaja que no se pueda realizar un diseño correctamente, ya que la Pool no tiene suficiente espacio.

Después:



1. Los elementos de tipo tarea ya contienen la propiedad de redimensionarse.
2. Los elementos contienen un conjunto de restricciones para no violar las especificaciones de BPMN, por ejemplo: el elemento “evento de inicio”, no permite terminar un flujo en él y el elemento “evento de fin”, no permite comenzar un flujo.
3. Los conectores se dibujan de diferentes maneras dependiendo del posicionamiento de los elementos que conecta.
4. Existen un conjunto de restricciones que no permiten que los elementos se desplacen fuera del elemento contenedor Pool.
5. Los elementos de tipo Contenedores Pool ya contienen la propiedad de redimensionarse. También existen un conjunto de restricciones para los elementos contenedores, con el objetivo de mantener a sus elementos dentro una vez que se redimensione la Pool.
6. Los elementos contienen restricciones para poderse crear, el modelador no permite crear un elemento encima de otro, ni fuera de los elementos contenedores Pool.

La comparación anterior muestra que se ha mejorado el comportamiento de los elementos gráficos.

Conclusiones parciales

Después de concluida la etapa de implementación y prueba se concluye que:

- La aplicación de las métricas de validación de diseño, sentó de manera eficiente las bases para la implementación de las funcionalidades.
- Al aplicar las pruebas de caja negra, se obtuvieron importantes resultados sobre el comportamiento del sistema. Dichas pruebas permitieron comprobar los requisitos funcionales definidos para el componente a partir de los diseños de casos de pruebas, evaluándose la calidad de la biblioteca gráfica.

CONCLUSIONES

Una vez finalizado el presente trabajo se puede concluir que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos, resaltando que:

- Con el estudio de sistemas que modelan procesos de negocio se obtuvieron características visuales, las cuales fueron utilizadas para la implementación.
- La generación de los artefactos durante el desarrollo de las etapas de análisis, diseño e implementación proporcionó la obtención de una biblioteca gráfica para el modelador del marco de trabajo Sauxe.
- La validación del diseño mediante la aplicación de las métricas, las pruebas realizadas, demostró que la solución cumple con los requerimientos, la eficiencia y la estabilidad necesaria para ser utilizado en el modelador.
- Las pruebas de caja negra permitieron detectar y corregir los errores no detectados durante la implementación, posibilitando cumplir con las especificaciones requeridas y la validación de la biblioteca implementada.

De manera general se solucionó el problema existente a través de la realización de todas las tareas definidas, obteniéndose como resultado una Biblioteca Gráfica, que permitirá, a través de sus funcionalidades, contribuir a la definición del comportamiento de los elementos gráficos en el modelador del marco de trabajo Sauxe.

RECOMENDACIONES

- Aprovechar el estudio realizado en esta investigación para futuras versiones que se realicen del modelador.
- Realizar un estudio de las nuevas funcionalidades que brinda la nueva versión de la librería Raphael.js versión 2.0.
- Realizar un estudio de los siguientes plugin para la librería Raphael.js versión 2.0:
  - Raphael.InlineTextEditing
  - Raphael.FreeTransform

TRABAJOS CITADOS

ABC. 2013. http://www.definicionabc.com. [En línea] 2013. http://www.definicionabc.com/general/proceso.php#ixzz2IoFMZYik.

Apache2., La biblia Servidor. **Kabir, Mohammed J.**

Baranovskiy, Dmitry. 2012. http://raphaeljs.com/. *Raphaël—JavaScript Library. Raphaël—JavaScript Library.* [En línea] 13 de 01 de 2012. http://raphaeljs.com/..

Barros, Oscar. 1994. *"Reingeniería de Procesos de negocio".* Chile : Editorial Dolmen, 1994.

Beecle, Ken Schwaber y Mike. 2002. Agile Software Development with Scrum. *Prentice Hall.* 2002.

Benghazi, Jose Luis Garrido Bullejos Kawtar. 2009. http://www.ugr.es. [En línea] 2009. [Citado el: 14 de 02 de 2009.] http://www.ugr.es/~mnoguera/collaborative_systems-business_processes_10-11.pdf.

Bizagi. 2013. Bizagi Process Modeler. [En línea] 2013. http://www.bizagi.com/docs/BPMNbyExampleSPA.pdf.

bonitasof. 2013. www.bonitasoft.com. [En línea] 2013. www.bonitasoft.com.

Cummunity, Jbos. 2013. jBPM-Jbos Cummunity. [En línea] 2013. http://www.jboss.org/jbpm.

Digital, Dirección General de Gobierno. 2006. Estándares de codificaciones de sistemas. s.l. : Gobierno del Chubut, 2006.

Entidades, Centro de Informatización de Gestión de. 2012. *Modelo de desarrollo de software v1.1.* 2012.

Esser. 2009. Secure Programming with the Zend-Framework. 2009.

Hitpass, Freund Ruecker. 2011. Manual de Referencia y Guía BPMN 2.0. Chile : s.n., 2011.

ISO-9001. 2008. *ISO-9001.* 2008.

—. 2005. *ISO-9001.* 2005.

Kawtar Benghazi, Jose Luis Garrido Bullejos. 2009. http://www.ugr.es. [En línea] 2009. [Citado el: 14 de 02 de 2009.] http://www.ugr.es/~mnoguera/collaborative_systems-business_processes_10-11.pdf.

Learning Ext JS. . **Frederick, Shea, Ramsay, Colin y Blades, Steve 'Cutter'. 2008.** s.l. : Packt Publishing Ltd, 2008.

León, Carlos Ponce de. 2013. Modelado De Procesos De Negocio (Bpm). [En línea] 2013. http://www.buenastareas.com/ensayos/Modelado-De-Procesos-De-Negocio-Bpm/1581374.html..

Mayora, Alex. 2010. *MANUAL PARTICIPANTE DE LA HERRAMIENTA INTALIO.* 2010.

NetBeans, Community. 2005. http://netbeans.org/. [En línea] NetBeans Community, 10 de 5 de 2005. http://netbeans.org/.

OoCities. 2009. http://www.oocities.org. [En línea] 2009. http://www.oocities.org/es/avrrinf/tabd/Foro/Foro_UML.htm.

Oroya, Víctor Paú Avila. 2010. *INFORMÁTICA Y SOCIEDAD .* Perú : Universidad Nacional Mayor de San Marcos, 2010.

Orozco, Jesús Graterol – Francisco Hernández – Yamil. 2010. Herramientas BPMS. [En línea] 2010. http://kuainasi.ciens.ucv.ve/adsi2010-2/HTML_Herramientas_BPMS/BPM.htm.

Pantiagozo, Manue lGarcía. 2002. Revista Industrial Data. s.l. : Instituto de Investigación FII, 2002, Vols. FII - UNMSM Nº 9.

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

RAE. *Definición de biblioteca.*

Revista Industrial Data . **García Pantigozo, Manuel. 2002.** s.l. : Instituto de Investigación FII, 2002, Vols. FII - UNMSM Nº 9.

Ruiz, Francisco. 2006. Proceso Software y Gestión del Conocimiento 4c Procesos de Negocio. España : Grupo Alarcos Dep., 2006.

Stapleton, Jennifer. 2003. DSDM Business Focused Development. s.l. : DSDM Consortium, 2003.

White, Stephen A. 2009. *Guía de Referencia y modelado BPMN.* Lighthouse Point, Florida, USA : Future Strategies Inc, 2009.

Zaratiegui, J. R. 1999. *La gestión por procesos: Su papel e importancia en la empresa.* 1999.

ANEXOS

Anexo 1

Tabla 8: Descripción de las Clases del Modelo Conceptual.

Nombre de la Clase	Descripción
Modelador	Herramienta en las que se definen los procesos de negocio.
Procesos de Negocio	Definición de todas las actividades, así como el orden en que son ejecutadas
Elementos Gráficos	Definición de todos los elementos de modelado.
Reglas BPMN	Reglas de modelado para los procesos de negocio
Conectores	Elemento gráfico para crear asociaciones entre objetos
Flujo de secuencia	Conector que indica el orden en se realizan las tareas
Flujo de mensajes	Conector que representa el envío de mensajes entre procesos
Asociaciones	Asociación de un artefacto a un elemento gráfico
Actividad	Representa algún tipo de acción dentro del proceso
Contendor	forma de agrupar los elementos primarios de modelado
Artefactos	Es un elemento gráfico para proporcionar información adicional sobre los procesos
Tareas	Es una actividad atómica que se incluye dentro del proceso, se usa cuando el trabajo que representa dentro del proceso no puede ser dividido a nivel mayor de detalles
Subprocesos	Es una actividad compuesta que se incluye dentro del proceso. Es compuesta en el sentido que se puede dividir para proveer un mayor nivel de detalles
Eventos	Es algo que ocurre durante transcurso de un proceso de negocio
Nodos de decisiones	Se utiliza para controlar la convergencia y divergencia de múltiples flujos de secuencia.
Piscina (Pool)	Representa un participante en el proceso
Carril (Lane)	Se usa para particionar las piscinas
Objeto de datos:	Proveen información sobre lo que necesitan las actividades para ser ejecutadas y lo que producen. Es considerado un artefacto pues no tienen efecto directo sobre el flujo de mensaje o el flujo de secuencia del proceso

Conector Grafic Info	Contiene la información acerca de los objetos de conexión
Node Grafic Info	Contiene información acerca de los objetos de flujo relativo el origen, las coordenadas, el tamaño, el color, etc.

Tabla 9: Diccionario de datos de Procesos de negocio y NodeGraficinfo.

Nombre de la entidad	**Procesos de negocio.**					
Descripción de la entidad	Representan la definición de todas las actividades, así como el orden en que son ejecutadas.					
Nombre del atributo	**Descripción**	**Tipo**	**¿Puede ser nulo?**	**¿Es único?**	**Restricciones**	
					Clase válidas	**Clases no válidas**
id	Atributo que identifica un proceso de negocio.	Entero	No	Sí	Cualquier combinación de números.	No admite letras o caracteres especiales.
nombre	Nombre o descripción del proceso de negocio	Letras	No	Sí	Cualquier combinación de letras	No admite números o caracteres especiales.
Fecha de creación	Fecha de elaboración del proceso de negocio.	Alfanumérico	No	No	Combinación de números y letras.	No admite caracteres especiales.
Fecha Modificación	Fecha cuando se ha realizado una variación al	Alfanumérico	No	No	Combinación de números y letras.	No admite letras o caracteres

	proceso de negocio.					especiales.
Nombre de la entidad	NodeGraficinfo					
Descripción de la entidad	Contiene información acerca de los objetos de flujo relativo el origen, las coordenadas, el tamaño, el color, etc.					
Nombre del atributo	**Descripción**	**Tipo**	**¿Puede ser nulo?**	**¿Es único?**	**Restricciones**	
					Clase válidas	**Clases no válidas**
shape	Arreglo de objetos para dibujar un elemento BPMN	Objeto	No	No	Admite solo objetos de tipo Raphael.js	No admite otro tipo de objeto
Border Color	Representa el color que contendrá el borde de la figura.	Alfanumérico	No	No	Cualquier combinación de letras y números.	No admite caracteres especiales.
Ycordenadas	Representa la coordenada que ocupará la figura en el eje vertical.	Numérico	No	No	Cualquier combinación números.	No admite números, ni caracteres especiales.
Xcordenadas	Representa la coordenada que ocupará la figura en el eje horizontal.	Numérico	No	No	Cualquier combinación números.	No admite letras, ni caracteres especiales.
Height	Representa la altura del objeto	Numérico	No	No	Cualquier combinación números.	No admite letras, ni caracteres especiales.

Width	Representa el ancho del objeto	Numérico	No	No	Cualquier combinación números.	No admite letras, ni caracteres especiales.

Anexo 2

Tabla 10: Descripción textual del requisito Crear restricciones gráficas de desplazamiento en contenedores pool para Eventos

Precondiciones	Debe estar creado un tipo de contenedor Pool.
Flujo de eventos	
Flujo básico	
1	Se selecciona un tipo de evento de la paleta de opciones.
2	Se crea el evento arrastrándolo hacia el contendor de tipo Pool.
3	Se crean las restricciones de desplazamiento.
Pos-condiciones	
4	Quedan creadas las restricciones de desplazamiento en el contenedor Pool para el evento.
Relaciones	**Requisitos Incluidos** **Paso 1 del flujo básico:** Crear evento.
	Extensiones **N/A.**
Conceptos	**Evento**

Tabla 11: Descripción textual del requisito Crear restricciones gráficas de salida en contenedores pool para Eventos de inicio

Precondiciones	Debe estar creado un contenedor de tipo Pool.
Flujo de eventos	
Flujo básico	
1	Se selecciona evento de inicio de la paleta de opciones.
2	Se crea el evento de inicio arrastrándolo hacia el contendor de tipo Pool.
3	Se crean las restricciones de salida.
Pos-condiciones	
4	Quedan creadas las restricciones gráficas de entrada para el evento de inicio seleccionado.
Relaciones	**Requisitos Incluidos** **Paso 1 del flujo básico:** Crear evento de inicio
	Extensiones **N/A.**
Conceptos	**Eventos de fin**

Tabla 12: Descripción textual del requisito Crear restricciones gráficas de entrada en contenedores pool para Eventos de fin

Precondiciones	Debe estar creado un contenedor de tipo Pool.
Flujo de eventos	
Flujo básico	
1	Se selecciona evento de fin de la paleta de opciones.
2	Se crea el evento de fin arrastrándola hacia el contendor de tipo Pool.
3	Se crean las restricciones de entrada.
Pos-condiciones	
4	Quedan creadas las restricciones gráficas de entrada para el evento de fin seleccionado.
Relaciones	**Requisitos Incluidos** **Paso 1 del flujo básico:** Crear evento de fin
	Extensiones **N/A.**
Conceptos	**Eventos de fin**

Tabla 13: Descripción textual del requisito Crear restricciones gráficas de desplazamiento en contenedores pool para Tareas atómicas

Precondiciones	Debe estar creado un tipo de contenedor Pool.
Flujo de eventos	
Flujo básico	
1	Se selecciona la tarea atómica de la paleta de opciones.
2	Se crea la tarea atómica arrastrándola hacia el contendor de tipo Pool.
3	Se crean las restricciones de desplazamiento.
Pos-condiciones	
4	Quedan creadas las restricciones de desplazamiento en el contenedor Pool para el objeto de datos.
Relaciones	**Requisitos Incluidos** **Paso 1 del flujo básico:** Crear tareas atómicas
	Extensiones **N/A.**

Conceptos	**Tareas atómicas**

Tabla 14: Descripción textual del requisito Redimensionar Tareas atómicas

Precondiciones	Se creado al menos un tipo tarea atómica
Flujo de eventos	
Flujo básico	
1	El modelador muestra las tareas creadas.
2	Se selecciona la tarea a modificar con doble clic.
3	Se modifica la tarea a las dimensiones deseadas.
4	Se actualizan las nuevas restricciones gráficas de la tarea al dar doble clic para quitar la selección.
Pos-condiciones	
6 Quedan modificadas las dimensiones y las restricciones gráficas del contenedor Pool	
Flujos alternativos	
Flujo alternativo 3.a Se cancela la opción de modificar con doble clic	
	Paso 4.
Pos-condiciones	
	No se modifica el contenedor.
Validaciones	
	Las dimensiones de la tarea no pueden ser mayores que la Pool.
Relaciones	**Requisitos Incluidos** **Paso 1 del flujo básico:** Crear Tareas atómicas
	Extensiones **N/A.**
Conceptos	**Tareas atómicas**

Tabla 15: Descripción textual del requisito Modificar restricciones gráficas de desplazamiento para Flujo de secuencia no controlado

Precondiciones	Se creado al menos un tipo un flujo de secuencia no controlado
Flujo de eventos	

Flujo básico		
1	El modelador muestra el flujo creado	
2	Se selecciona uno de los elementos al que está conectado	
3	Se modifica las restricciones de desplazamiento al mover el elemento	
Pos-condiciones		
6	Quedan modificadas las restricciones para flujo de secuencia no controlado	
Relaciones	**Requisitos Incluidos**	**Paso 1 del flujo básico:** Crear flujo de secuencia no controlado
	Extensiones	**N/A.**
Conceptos	**Flujo de secuencia no controlado**	

Tabla 16: Descripción textual del requisito Crear restricciones gráficas de desplazamiento en contenedores pool para Decisiones.

Precondiciones	Debe estar creado un tipo de contenedor Pool.	
Flujo de eventos		
Flujo básico		
1	Se selecciona una decisión de la paleta de opciones.	
2	Se crea decisión arrastrándola hacia el contendor de tipo Pool.	
3	Se crean las restricciones de desplazamiento.	
Pos-condiciones		
4	Quedan creadas las restricciones de desplazamiento en el contenedor Pool para decisiones.	
Relaciones	**Requisitos Incluidos**	**Paso 1 del flujo básico:** Crear Decisiones
	Extensiones	**N/A.**
Conceptos	**Decisiones**	

Tabla 17: Descripción textual del requisito Crear restricciones gráficas de desplazamiento vertical para contenedores Pool

Precondiciones	

Flujo de eventos		
Flujo básico		
1	Se selecciona contenedor de tipo Pool de la paleta de opciones.	
2	Se crea el contendor Pool arrastrándola hacia el lienzo.	
3	Se crean las restricciones de desplazamiento vertical.	
Pos-condiciones		
4	Quedan creadas las restricciones de desplazamiento vertical para el contenedor Pool.	
Relaciones	**Requisitos Incluidos**	**Paso 1 del flujo básico: Crear contenedor Pool**
	Extensiones	**N/A.**
Conceptos	**Pool**	

Tabla 18: Descripción textual del requisito Crear Objeto de datos.

Precondiciones	Debe estar creado un contenedor de tipo Pool
Flujo de eventos	
Flujo básico	
1	Se selecciona el Objeto de datos de la paleta de opciones.
2	Se crea un objeto de datos arrastrándola hacia el contendor de tipo Pool
Pos-condiciones	
3	Se crea un Objeto de datos.
Conceptos	**Objeto de datos**

Tabla 19: Descripción textual del requisito Crear restricciones gráficas de desplazamiento en contenedores pool para Objeto de datos

Precondiciones	Debe estar creado un tipo de contenedor Pool.
Flujo de eventos	
Flujo básico	
1	Se selecciona el objeto de datos de la paleta de opciones.
2	Se crea el objeto de datos arrastrándolo hacia el contendor de tipo Pool.
3	Se crean las restricciones de desplazamiento.

Pos-condiciones		
4 Quedan creadas las restricciones de desplazamiento en el contenedor Pool para el objeto de datos.		
Relaciones	**Requisitos Incluidos**	**Paso 1 del flujo básico:** Crear Objeto de datos
	Extensiones	**N/A.**
Conceptos	**Objeto de datos**	

Tabla 20: Descripción textual del requisito Crear Área de Texto.

Precondiciones	Debe estar creado un contenedor de tipo Pool
Flujo de eventos	
Flujo básico	
1 Se selecciona área de texto de la paleta de opciones.	
2 Se crea el área de texto arrastrándola hacia el contendor de tipo Pool.	
Pos-condiciones	
3 Se crean un área de texto.	
Conceptos	**Área de texto**

Tabla 21: Descripción textual del requisito Modificar Área de texto.

Precondiciones	Debe estar creada un área de texto.	
Flujo de eventos		
Flujo básico		
1 El modelador muestra el área de texto.		
2 Se da clic derecho abrir especificaciones y se cambia el nombre por el que se desee.		
Pos-condiciones		
3 Se modifica el contenido del área de texto.		
Relaciones	**Requisitos Incluidos**	**N/A**

	Extensiones	**N/A.**
Conceptos	**Área de texto**	

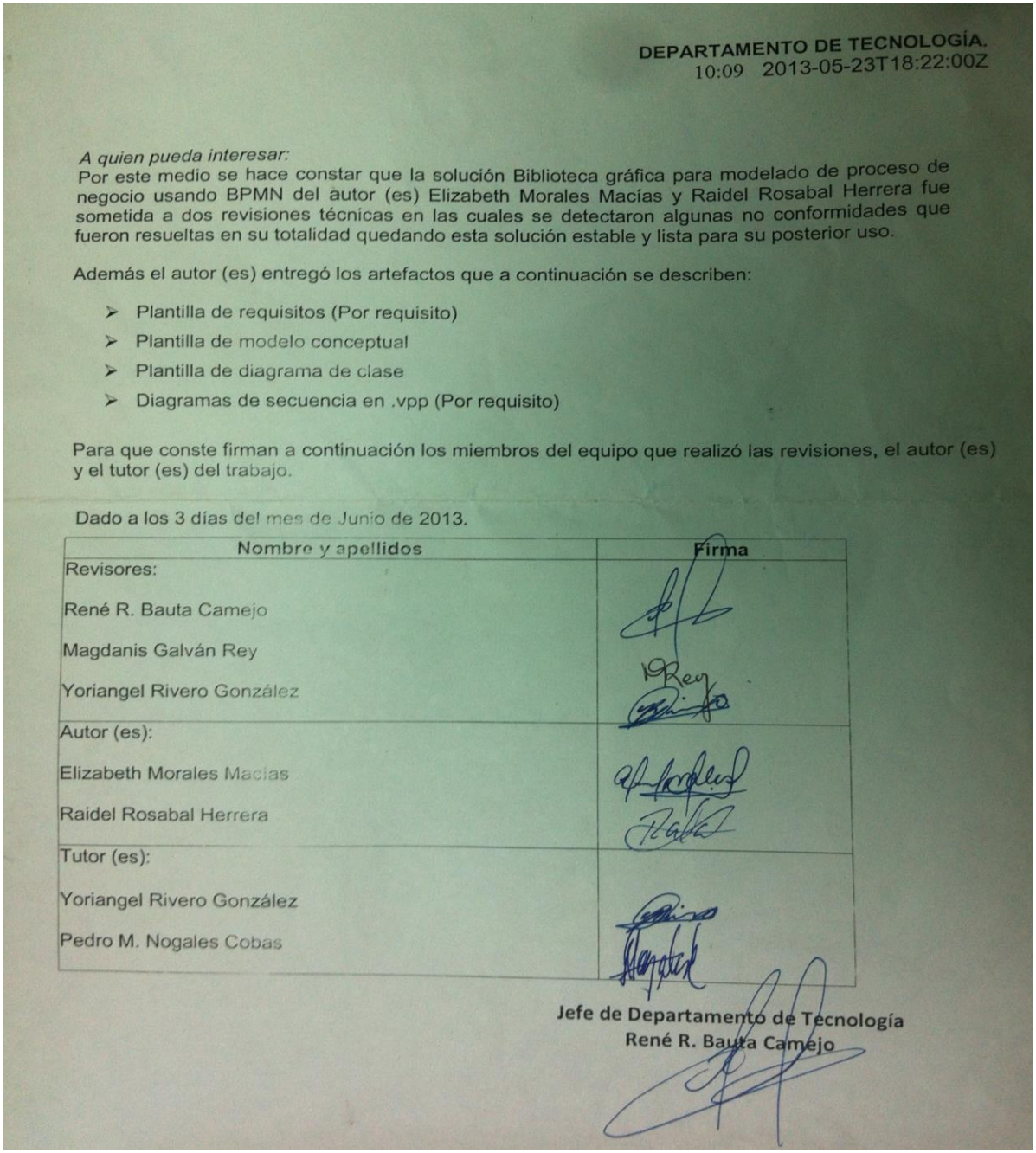
Tabla 22: Descripción textual del requisito crear restricciones gráficas de desplazamiento en contenedores pool para área de texto

Precondiciones		Debe estar creado un tipo de contenedor Pool.
Flujo de eventos		
Flujo básico		
1	Se selecciona área de texto de la paleta de opciones.	
2	Se crea el área de texto arrastrándola hacia el contendor de tipo Pool.	
3	Se crean las restricciones de desplazamiento.	
Pos-condiciones		
4	Quedan creadas las restricciones de desplazamiento en el contenedor Pool para el área de texto**.**	
Relaciones	**Requisitos Incluidos**	**Paso 1 del flujo básico: Crear Área de Texto**
	Extensiones	**N/A.**
Conceptos	**Área de texto**	

Tabla 23: Descripción textual del requisito Crear Flujo de Mensaje.

Precondiciones		**Debe estar creado un contenedor de tipo Pool** **Deben estar creadas dos tareas atómicas.**
Flujo de eventos		
Flujo básico		
1	**Se selecciona el flujo de mensaje de la paleta de opciones.**	
2	**Se crea el flujo de mensaje dando clic en una tarea primero y luego en la otra.**	
Pos-condiciones		
6	**Se crean un flujo de mensaje entre las tareas.**	
Conceptos	**Flujo de mensaje**	

Anexo 3



DEPARTAMENTO DE TECNOLOGÍA.
10:09 2013-05-23T18:22:00Z

A quien pueda interesar:
Por este medio se hace constar que la solución Biblioteca gráfica para modelado de proceso de negocio usando BPMN del autor (es) Elizabeth Morales Macías y Raidel Rosabal Herrera fue sometida a dos revisiones técnicas en las cuales se detectaron algunas no conformidades que fueron resueltas en su totalidad quedando esta solución estable y lista para su posterior uso.

Además el autor (es) entregó los artefactos que a continuación se describen:

- Plantilla de requisitos (Por requisito)
- Plantilla de modelo conceptual
- Plantilla de diagrama de clase
- Diagramas de secuencia en .vpp (Por requisito)

Para que conste firman a continuación los miembros del equipo que realizó las revisiones, el autor (es) y el tutor (es) del trabajo.

Dado a los 3 días del mes de Junio de 2013.

Nombre y apellidos	Firma
Revisores: René R. Bauta Camejo Magdanis Galván Rey Yoriangel Rivero González	
Autor (es): Elizabeth Morales Macías Raidel Rosabal Herrera	
Tutor (es): Yoriangel Rivero González Pedro M. Nogales Cobas	

Jefe de Departamento de Tecnología
René R. Bauta Camejo

Anexo 4

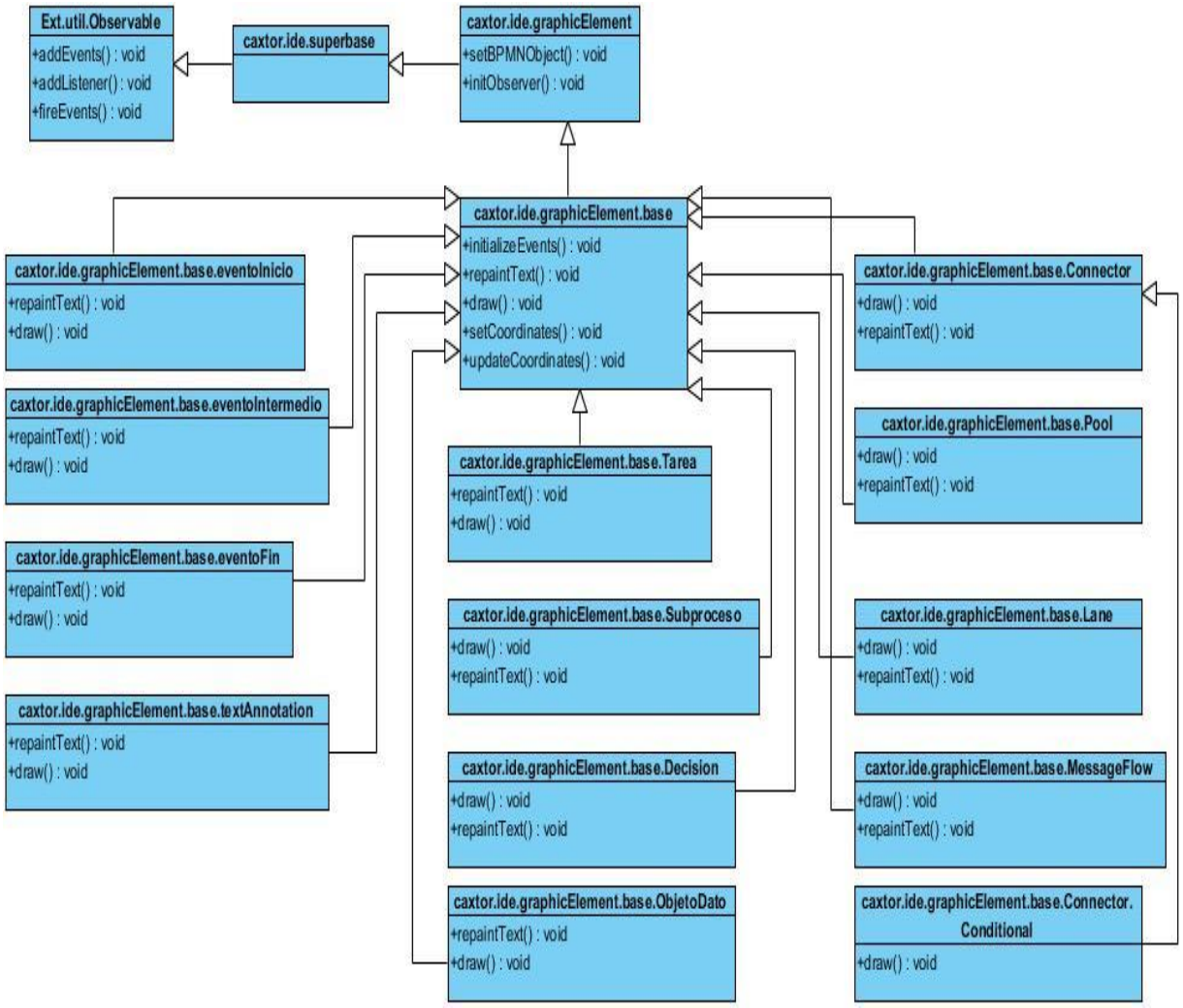


Figura 10: Diagrama de clases

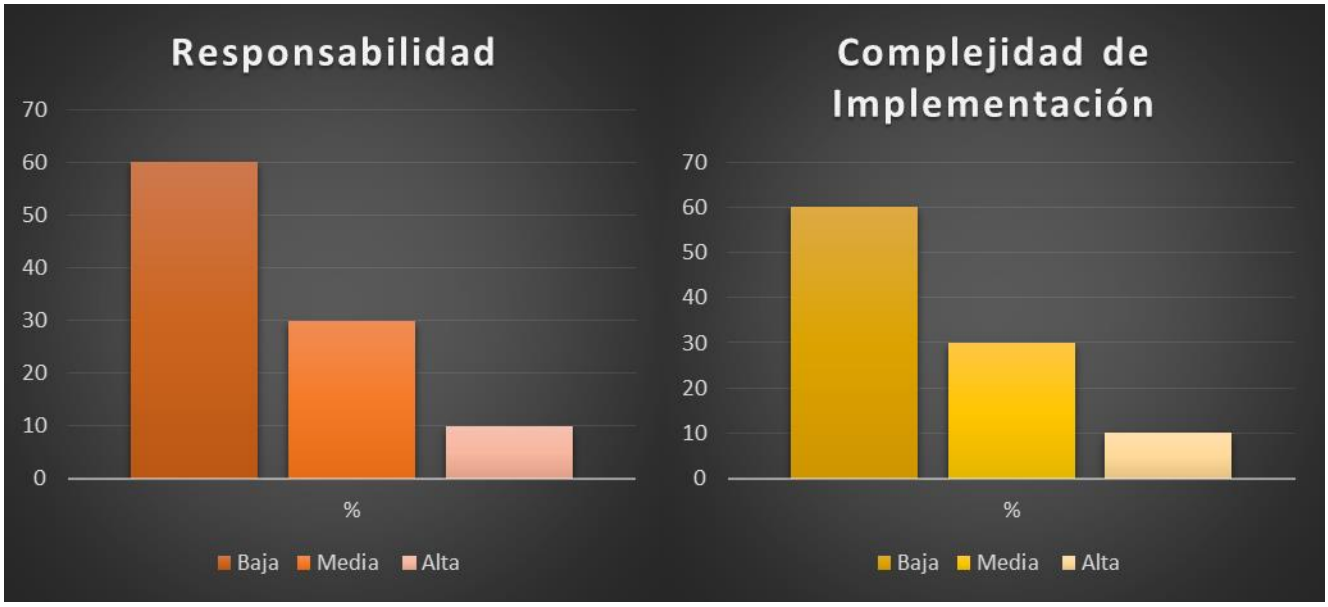
Anexo 5

Validación del diseño

Resultados de la métrica TOC

Representación en % de la incidencia de los resultados en los atributos de calidad:

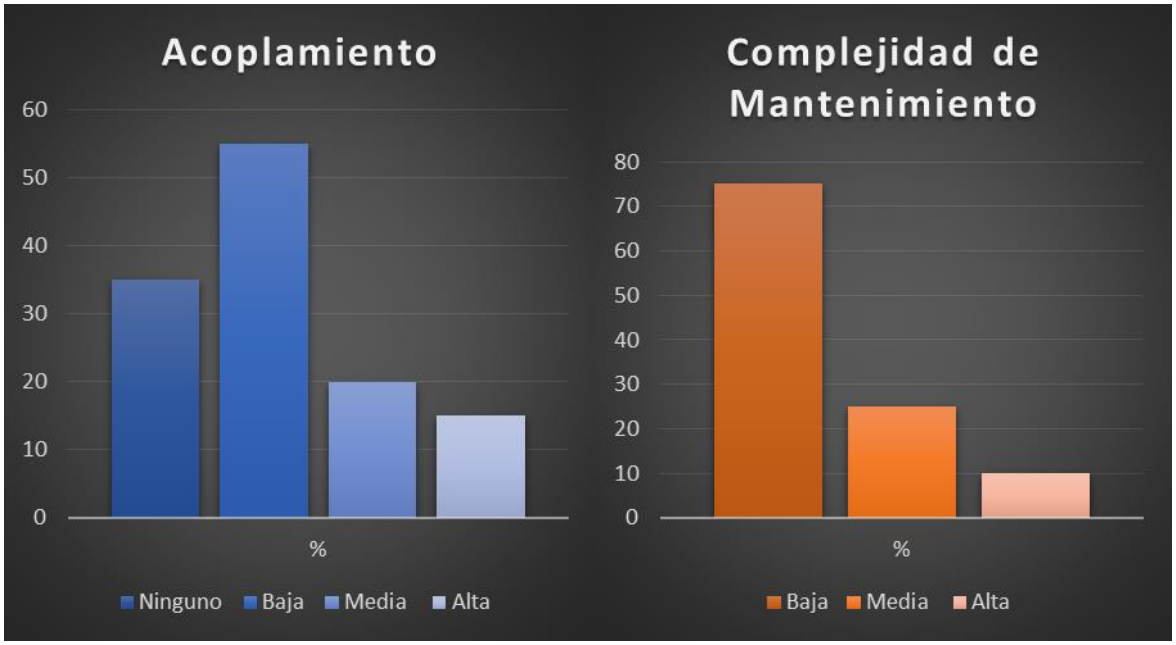
Tabla 24: Resultados de la métrica TOC

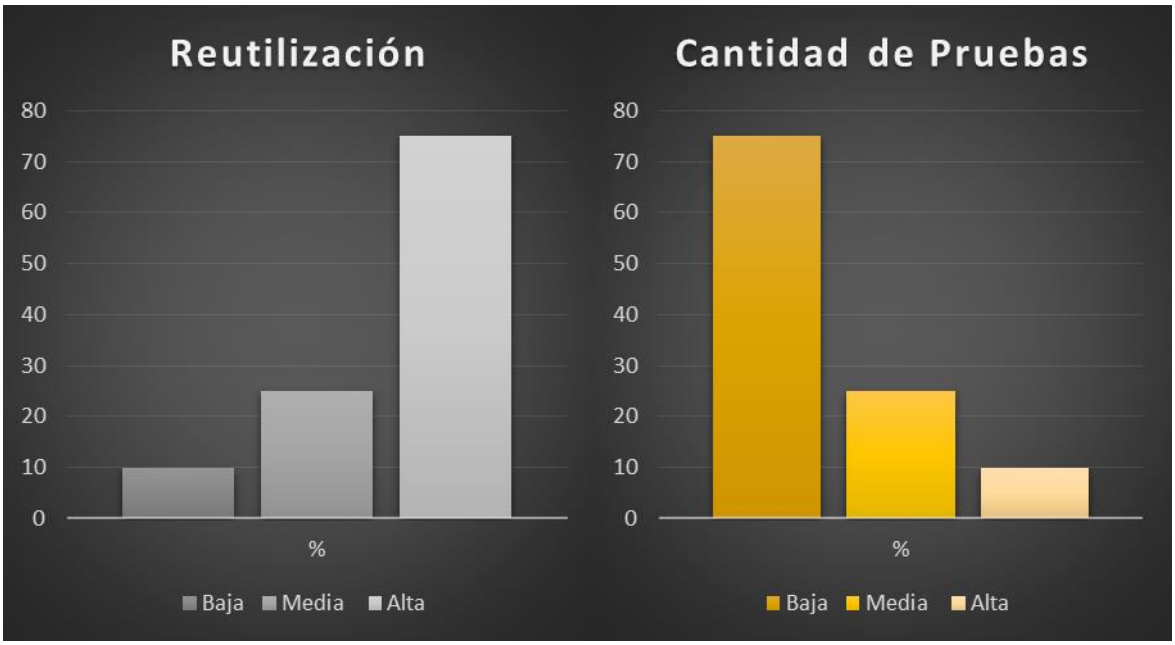


Resultados de la métrica RC

Representación en % de la incidencia de los resultados en los atributos de calidad:

Tabla 25: Resultados de la métrica RC





Anexo 6

Aplicación de la Prueba de Caja Negra al requisito funcional **Crear restricciones gráficas de desplazamiento en contenedores Pool para decisiones.**

Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.
- Se debe crear al menos un tipo de Evento**.**

Tabla 26: Caso de Prueba Crear restricciones gráficas de desplazamiento en contenedores Pool para Decisiones

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Crear restricciones gráficas de desplazamiento en contenedores Pool para decisiones.	Al ser creado una decisión se crean las restricciones de desplazamiento dentro del contenedor que le permitirá a la decisión moverse dentro del contendor.	EP 1: Crear Decisión y moverla dentro del contenedor.	– Se crea un contenedor de tipo Pool. – Se crea una decisión arrastrándolo hacia el contenedor Pool**.** – Al ser creado el elemento se crean las restricciones de desplazamiento. – Se desplaza el elemento por todo el contenedor sin que este pueda salir de las coordenadas del contenedor.
		EP1.2: Crear decisión fuera del contenedor.	– Se crea una decisión arrastrándolo hacia lienzo**.** – El modelador no permite crear el elemento.

Aplicación de la Prueba de Caja Negra al requisito funcional **Crear restricciones gráficas de desplazamiento en contenedores Pool para Área de texto.**

Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.
- Se debe crear un área de texto.

Tabla 27: Caso de Prueba Crear restricciones gráficas de desplazamiento en contenedores Pool para Área de texto

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Crear restricciones gráficas de desplazamiento en contenedores Pool para Área de texto	Al ser creado un área de texto se crean las restricciones de desplazamiento dentro del contenedor que le permitirá al área de texto moverse dentro del contendor.	EP 1: Crear Área de texto y moverla dentro del contenedor.	– Se crea un contenedor de tipo Pool. – Se crea un área de texto arrastrándolo hacia el contenedor Pool**.** – Al ser creado el elemento se crean las restricciones de desplazamiento. – Se desplaza el elemento por todo el contenedor sin que este pueda salir de las coordenadas del contenedor.
		EP1.2: Crear Área de texto fuera del contenedor.	– Se crea un área de texto arrastrándolo hacia lienzo**.** – El modelador no permite crear el elemento.

Aplicación de la Prueba de Caja Negra al requisito funcional **Crear restricciones gráficas de desplazamiento en contenedores Pool para Objeto de datos.**

Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador
- Se debe crear un Objeto de dato**.**

Tabla 28: Caso de Prueba Crear restricciones gráficas de desplazamiento en contenedores Pool para Objeto de datos.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Crear restricciones gráficas de desplazamiento en contenedores Pool para Objetos de dato.	Al ser creado un objeto de dato se crean las restricciones de desplazamiento dentro del contenedor que le permitirá al elemento creado moverse dentro del contendor.	EP 1: Crear Objeto de dato y moverlo dentro del contenedor.	– Se crea un contenedor de tipo Pool. – Se crea una decisión arrastrándolo hacia el contenedor Pool**.** – Al ser creado el elemento se crean las restricciones de desplazamiento. – Se desplaza el elemento por todo el contenedor sin que este pueda salir de las coordenadas del contenedor.
		EP1.2: Crear Objeto de dato fuera del contenedor.	– Se crea un objeto de dato arrastrándolo hacia lienzo**.** – El modelador no permite crear el elemento.

Aplicación de la Prueba de Caja Negra al requisito funcional **Crear Área de texto**

Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.
- Se debe crear un área de texto.

Tabla 29: Caso de Prueba Crear Área de texto

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Crear Área de Texto.	Se crea un área de texto	EP 1: Crear un área de texto en un	– Se crea un contenedor de tipo Pool.

	contenedor de tipo Pool	– Se crea un Área de texto
	EP1.2: Crear un área de texto en lienzo.	– Se crea un área de texto fuera en el lienzo**.** – El modelador muestra el puntero el error al intentar crear el elemento.

Aplicación de la Prueba de Caja Negra al requisito funcional **Crear Flujo de mensaje**

Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.
- Se debe crear dos tareas atómicas.
- Se crea un flujo de mensaje.

Tabla 30: Caso de Prueba Crear Flujo de mensaje

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Crear Flujo de mensaje.	Se crea un flujo de mensaje que indica el orden en que se intercambian mensajes entre dos entidades.	EP 1: Crear un flujo de mensaje entre dos entidades	– Se crea un contenedor de tipo Pool. – Se crean las dos tareas atómicas. – Se selecciona el flujo de mensaje, se crean con un clic en cada tarea un flujo de mensaje.
		EP1.2: Crear un flujo de mensaje entre una tarea y un contenedor de tipo Pool.	– Se crea un contenedor de tipo Pool. – Se crea una tarea atómica. – Se selecciona el flujo de mensaje, se crean con un clic en

			la tarea y otro en el contenedor de tipo Pool un flujo de mensaje**.** – El modelador muestra el puntero el error al intentar crear el elemento.

Aplicación de la Prueba de Caja Negra al requisito funcional **Crear Objeto de datos**
Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.
- Se debe crear un objeto de datos

Tabla 31: Caso de Prueba Crear Objeto de datos

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Crear Objeto de datos.	Se crea un Objeto de datos	EP 1: Crear un objeto de datos en un contenedor de tipo Pool.	– Se crea un contenedor de tipo Pool. – Se crea un Objeto de datos.
		EP1.2: Crear un área de texto fuera del contenedor de tipo Pool.	– Se crea un Objeto de datos en el lienzo**.** – El modelador muestra el puntero el error al intentar crear el elemento.

Aplicación de la Prueba de Caja Negra al requisito funcional **Crear restricciones gráficas de desplazamiento en contenedores Pool para Tareas Atómicas**
Condiciones de ejecución.

- Se debe crear un contenedor de tipo Pool en el modelador.
- Se debe crear al menos un tipo de tarea atómica

Tabla 32: Caso de Prueba Crear restricciones gráficas de desplazamiento en contenedores Pool para Tareas Atómicas

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Crear restricciones gráficas de desplazamiento en contenedores Pool para Tareas Atómicas.	Al ser creada una tarea atómica se crean las restricciones de desplazamiento dentro del contenedor que le permitirá a la tarea moverse dentro del contendor.	EP 1: Crear una tarea atómica moverla dentro del contenedor.	– Se crea un contenedor de tipo Pool. – Se crea una tarea atómica arrastrándola hacia el contenedor Pool**.** – Al ser creada la tarea atómica se crean las restricciones de desplazamiento. – Se desplaza la tarea atómica por todo el contenedor sin que esta pueda salir de las coordenadas del contenedor.
		EP1.2: Crear una tarea atómica fuera del contenedor.	– Se crea una tarea atómica arrastrándola hacia lienzo**.** – El modelador muestra el puntero el error al intentar crear el elemento.

Aplicación de la Prueba de Caja Negra al requisito funcional **Crear restricciones gráficas de salida en contenedores Pool para Eventos de inicio**

Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.
- Se debe crear Evento de inicio**.**
- Se deben crear una Tareas Atómicas

Tabla 33: Caso de Prueba Crear restricciones gráficas de salida en contenedores Pool para Eventos de inicio

Nombre del	Descripción	Escenarios de	Flujo del escenario

requisito	general	pruebas	
1: Crear restricciones gráficas de salida en contenedores Pool para Eventos de inicio.	El evento de inicio solo permite salida de flujo secuencia no controlado hacia cualquier otro elemento de modelado.	EP 1.1: Crear flujo de secuencia no controlado de salida en contenedores de tipo pool para eventos de inicio.	– Se crea un contenedor de tipo Pool en el modelador. – Se crea un evento de inicio arrastrándolo hacia el contenedor. – Se crea una tarea atómica arrastrándola hacia el contenedor. – Se selecciona un flujo de secuencia no controlado con un clic, se da clic primero sobre el evento de inicio creado y después clic sobre la tarea atómica. – Se crea un flujo de secuencia no controlado de salida entre el evento de inicio y la tarea.
		EP1.2: Crear flujo de secuencia no controlado de entrada en contenedores de tipo pool para eventos de inicio.	– Se crea un contenedor de tipo Pool en el modelador. – Se crea un evento de inicio arrastrándolo hacia el contenedor. – Se crea una tarea atómica arrastrándola hacia el contenedor. – Se selecciona un flujo de secuencia no controlado con un clic, se da clic primero sobre la

			tarea atómica creada y después clic el evento de inicio. – El modelador no permite crear un flujo de secuencia no controlado entre el evento de inicio y la tarea.

Aplicación de la Prueba de Caja Negra al requisito funcional **Crear restricciones gráficas de entrada en contenedores Pool para Eventos de fin.**

Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.
- Se debe crear Evento de fin
- Se deben crear una Tareas Atómicas

Tabla 34: Caso de Prueba Crear restricciones gráficas de entrada en contenedores Pool para Eventos de fin.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Crear restricciones gráficas de entrada en contenedores Pool para Eventos de fin.	El evento de fin solo permite entrada de flujo secuencia no controlado hacia cualquier otro elemento de modelado.	EP 1.1: Crear flujo de secuencia no controlado de entrada en contenedores de tipo pool para eventos de fin.	– Se crea un contenedor de tipo Pool en el modelador. – Se crea un evento de fin arrastrándolo hacia el contenedor. – Se crea una tarea atómica arrastrándola hacia el contenedor – Se selecciona un flujo de secuencia no controlado con un clic, se da clic primero sobre la tarea atómica creada y

		después clic el evento de fin. – Se crea un flujo de secuencia no controlado de entrada entre el evento de fin y la tarea.
	EP1.2: Crear flujo de secuencia no controlado de entrada en contenedores de tipo pool para eventos de inicio.	– Se crea un contenedor de tipo Pool en el modelador. – Se crea un evento de fin arrastrándolo hacia el contenedor. – Se crea una tarea atómica arrastrándola hacia el contenedor. – Se selecciona un flujo de secuencia no controlado con un clic, se da clic primero sobre el evento de fin creado y después clic sobre la tarea atómica. – El modelador no permite crear un flujo de secuencia no controlado entre el evento de fin y la tarea.

Aplicación de la Prueba de Caja Negra al requisito funcional **Redimensionar tareas atómicas**

Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.
- Se deben crear una Tareas Atómicas

Tabla 35: Caso de Prueba Redimensionar tareas atómicas

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario

1: Redimensionar Tareas Atómicas.	Se modifican las dimensiones de la tarea atómica.	EP 1: Redimensionar una tarea atómica contenida dentro de una pool.	– Se crea un contenedor de tipo Pool. – Se crea una tarea atómica arrastrándola hacia el contenedor Pool**.** – Se selecciona la tarea atómica dando doble clic sobre esta. – Se redimensiona hasta el tamaño deseado.

Aplicación de la Prueba de Caja Negra al requisito funcional **Modificar restricciones gráficas de desplazamiento para flujo de secuencia no controlado.**

Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.
- Se debe crear una tarea atómica.
- Se debe crear un evento.
- Se debe crear una relación mediante el flujo de secuencia no controlado.

Tabla 36: Caso de Prueba Modificar restricciones gráficas de desplazamiento para flujo de secuencia no controlado.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Modificar restricciones gráficas de desplazamiento para flujo de secuencia no controlado.	Al ser creada una relación mediante un flujo de secuencia no controlado, la tarea o el evento pueden ser movidos por el contenedor, modificando en cada movimiento las restricciones del flujo	EP 1: Crear flujo de secuencia no controlado y moverlo dentro del contenedor.	– Se crea un contenedor de tipo Pool. – Se crea una tarea arrastrándola hacia el contenedor Pool**.** – Se crea un evento de inicio arrastrándolo hacia el contendor. – Se selecciona el flujo de

	a la hora de moverse.		secuencia no controlado de la paleta de opciones. – Se crea la relación dando clic sobre un elemento primero y luego el otro. – Se mueve cualquiera de los dos elementos por el contenedor, modificándose al mismo tiempo las restricciones del flujo.

Aplicación de la Prueba de Caja Negra al requisito funcional **Crear restricciones gráficas de desplazamiento vertical para contenedores Pool**

Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.

Tabla 37: Caso de Prueba Crear restricciones gráficas de desplazamiento vertical para contenedores Pool

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Crear restricciones gráficas de desplazamiento vertical para contenedores Pool	Al ser creado un contenedor Pool se crean las restricciones de desplazamiento vertical que permitirá moverse en el lienzo.	EP 1: Crear contendor Pool y moverlo en el lienzo	– Se crea un contenedor de tipo Pool. – Al ser creado el elemento se crean las restricciones de desplazamiento. – Se desplaza el elemento el lienzo de manera vertical.

Aplicación de la Prueba de Caja Negra al requisito funcional **Crear Objeto de datos.**

Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.

- Se debe crear un objeto de datos.

Tabla 38: Caso de Prueba Crear Objeto de datos

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Crear Objeto de datos.	Se crea un Objeto de datos	EP 1: Crear un objeto de datos en un contenedor de tipo Pool.	– Se crea un contenedor de tipo Pool. – Se crea un Objeto de datos.
		EP1.2: Crear un área de texto fuera del contenedor de tipo Pool.	– Se crea un Objeto de datos en el lienzo**.** – El modelador muestra el puntero el error al intentar crear el elemento.

Aplicación de la Prueba de Caja Negra al requisito funcional **Modificar área de texto**

Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.
- Se debe crear un área de texto.

Tabla 39: Caso de Prueba Modificar área de texto

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Modificar Área de texto	Se modifica el contenido del texto del área	EP 1: Modificar dimensiones de un área de texto contenidas dentro de una pool.	– Se crea un contenedor de tipo Pool. – Se crea un área de texto arrastrándola hacia el contenedor Pool**.** – Se selecciona clic derecho abrir

	especificaciones y se pone el texto deseado.

Aplicación de la Prueba de Caja Negra al requisito funcional **Redimensionar contendores Pool**

Condiciones de ejecución

- Se debe crear un contenedor de tipo Pool en el modelador.

Tabla 40: Caso de Prueba Redimensionar contenedores Pool

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Redimensionar contendores Pool.	Se modifican las dimensiones del contenedor.	EP 1: Redimensionar un contenedor	– Se crea un contenedor de tipo Pool. – Se selecciona el contenedor dando doble clic sobre este. – Se redimensiona hasta el tamaño deseado.

GLOSARIO

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

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

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

PHP: (PHP HypertextPreprocessor) lenguaje de programación.

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

SVG: (Scalable Vector Graphics) gráficos vectoriales escalables.

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

Workflow: gestión electrónica de procesos de negocio.

.js: extensión para ficheros de tipo JavaScript.