Universidad de las Ciencias Informáticas

Herramienta para la transformación de BPMN 1.0 a la tecnología Windows Workflow Foundation 3.5

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

Autores:

Carmen Iliana Rondón Regalado

Michel González Valladares

Tutor:

Ing. Reynier Blanco Zambrano

La Habana

2012


"Sólo los que trabajan tienen
resultados en la vida..."
Anónimo

Declaración de Autoría

Declaración de autoría

Declaramos ser los únicos autores del trabajo titulado “Herramienta para la transformación de *BPMN* a la tecnología *Windows Workflow Foundation* 3.5.” y autorizamos a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

Y para que así conste firmamos la presente a los ___ días del mes de _____ de 2012.

_____ _____

Firma del Autor Firma del Autor

Carmen Iliana Rondón Regalado Michel González Valladares

_____

Firma del Tutor

Reynier Blanco Zambrano

Datos de Contacto

Carmen Iliana Rondón Regalado
Correo: cirondon@estudiantes.uci.cu
Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba
Michel González Valladares
Correo: mvalladares@estudiantes.uci.cu
Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba
Reynier Blanco Zambrano
Correo: rblanco@uci.cu
Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Dedicatoria

Quisiera dedicar el presente trabajo de diploma de manera especial a mamá, papá, hermanito y hermanita, por hacerme sentir orgullosa de la familia que tengo.

Carmen Iliana Rondón Regalado

A la vida por enseñarme a valorarla y a disfrutarla como si fuera hoy el último día…

Michel González Valladares

Agradecimientos

Quiero agradecer:

Primeramente a mi mamá y papá por su amor incondicional y por haberme formado con valores de bien. A mi hermanito por ser un ejemplo de bueno hijo y a Mer reír y llorar conmigo.

A todas mis amistades que estuvieron conmigo en el aula, en el apartamento, en el proyecto.

A mi compañero de tesis por compartir conmigo todo un año de sacrificio.

A mi tutor por soportar nuestros asedios y persecuciones.

A todos los personas que nos revisaron el documento, que nos ayudaron con el diseño, que nos facilitaron códigos, que nos enseñaron y guiaron a lo largo de todo el proceso.

A los que de una manera u otra se interesaron por el desarrollo de la tesis.

Gracias.

Carmen Iliana Rondón Regalado

Primero que todo quiero agradecer a mi madre por ser tan fuerte como es, por resistir tantas cosas que sólo ella sabe y por no rendirse nunca con tal de estar presente en este momento tan especial que sabe que es tan suyo como mío. A mi papá, que siempre está pensando en el bienestar de la familia, a mi hermano por brindarme su apoyo todo este tiempo de estudio y más…, en fin a mi familia en general por haber sacrificado tanto para concebir otro universitario en su seno.

A todos mis compañeros y amigos por brindarme sus conocimientos de forma desinteresada y por estar siempre ahí cuando hacía falta desestresarse cuando el “deber llamaba”… Unos más cercanos que otros, pero todos importantes porque de cada uno he aprendido cosas que me fortalecen para enfrentarme al futuro.

A mi tutor por ser más que tutor compañero y por consentir mis palabras inocuamente.

En especial a mi compañera de tesis que no sólo me apoyo en momentos súper-difíciles, sino que además me levantó el espíritu cuando más tocaba fondo. Soy muy afortunado de haber sido su compañero de tesis y de compartir varios días enteros de trabajo con el objetivo de no rendirnos y de ganarle tiempo al tiempo.

Michel González Valladares

Resumen

Durante todo el ciclo de desarrollo en el Sistema Único de Identificación Nacional (SUIN) los analistas son los encargados de mantener actualizados los modelos de proceso de negocio, generando artefactos que conforman la documentación del proyecto. Por otra parte, los desarrolladores a la hora de interpretar estos modelos en diversas ocasiones los malinterpretan, implementando procesos que se alejan de la definición inicial. Por esta razón, se derivan una serie de problemas que provocan inestabilidad y desacuerdos en el proyecto, prolongándose el tiempo de desarrollo del mismo.

El presente trabajo posee como principal objetivo desarrollar una herramienta que garantice una interpretación automatizada de los modelos de proceso de negocio, generando flujos de trabajo compatibles con la tecnología *Windows Workflow Foundation* 3.5. Para la implementación de la solución se utilizó *Visual Studio* 2010 como herramienta de desarrollo, *C-Sharp* como leguaje de programación y la librería *Telerik* para el diseño de las interfaces.

Con la implementación de la propuesta solución en el Sistema Único de Identificación Nacional se reducen las incompatibilidades entre el diseño del proceso de negocio y su implementación, así como el tiempo de transformación de los diagramas de negocio bajo el estándar *BPMN* 1.0 a la tecnología *Windows Workflow Foundation* 3.5.

Palabras claves: interpretación, incompatibilidades, proceso, transformación.

Índice

Índice

Índice

Índice de Tabla

Índice de Figuras

Introducción

En medio del acelerado desarrollo en el que se encuentra el campo de la informática, cobran creciente auge las aplicaciones orientadas a procesos. Si bien los sistemas informáticos actuales resuelven los problemas de manera eficiente, es necesario satisfacer las transformaciones que demandan los mercados. Los competidores ofrecen nuevos productos, los clientes solicitan modificar sus peticiones y las organizaciones están enfocadas en mejorar la gestión de procesos de negocio, a través de una filosofía que permita diseñar sus arquitecturas automatizando funcionamientos de principio a fin y permitiendo su monitorización y control.

Según Davenport, "*un proceso de negocio representa un conjunto de actividades estructuradas y medidas diseñadas para producir una salida específica para un cliente particular o mercado*" (1). En la medida en que una organización describa, diseñe y monitoree sus procesos de negocio con mayor fluidez podrá responder exitosamente ante la inmensa competencia en la que se encuentra sumergido el mundo informático. La introducción de la Gestión de Procesos de Negocio (*BPM* por sus siglas en inglés), como disciplina de administración y control de procesos por parte de estas empresas, con el objetivo de entrar en un círculo de mejora continua para dar cumplimiento a sus exigencias es la solución más abarcadora hasta el momento.

Estrechamente relacionado al surgimiento de *BPM* se percibe la evolución de los estándares de modelado de procesos de negocio. La industria del modelado de procesos de negocio tiende a centrarse en tres estándares fundamentales (2):

- *BPMN*[1] como notación gráfica para describir los procesos de negocio de un flujo de trabajo y con el propósito de ser comprensible para todos los usuarios participantes.
- *XPDL*[2] como formato para almacenar e intercambiar definiciones de procesos permitiendo a una herramienta modelar procesos en *BPMN* para que otra herramienta pueda leerlos.
- *BPEL*[3] como lenguaje de ejecución concretado en la tecnología de servicios *web*.

[1] Notación de Modelado de Procesos de Negocio (*Business Process Modeling Notation*).

[2] Lenguaje de Definición de Procesos basados en *XML* (*XML Process Definition Language*).

[3] Lenguaje de Ejecución de Procesos de Negocio (*Business Process Execution Lenguage*).

BPMN facilita un lenguaje gráfico común y mejora la comunicación y comprensión de los procesos de negocio para todas las personas que se encuentren involucradas:

- analista de negocio: encargado de realizar los bocetos iniciales de los procesos.
- desarrollador: responsable de garantizar la ejecución de dichos procesos.
- cliente o usuario final: gestiona y controla dichos procesos.

De acuerdo con el ciclo de vida de cualquier *BPMS*[4]*,* es elemental la automatización del proceso de negocio con el objetivo de obtener una aplicación final que gestione las necesidades de cualquier empresa. *Windows Workflow Foundation*[5] (*WWF*) representa uno de los modelos de interpretación para procesos de negocio que existen en el mundo. A pesar de no constituir un *BPMS*, juega un papel fundamental dentro de este ciclo, brindando soporte a las principales tareas que garantizan esta automatización.

WWF y *BPMN* surgen para respaldar toda una arquitectura de desarrollo centrada en la gestión de los procesos de negocio. En la actualidad, el estándar *BPMN* a pesar de ofrecer facilidades para el diseño y la descripción de procesos de negocio, presenta una definición aplicada a sus necesidades que son difíciles de interpretar en paralelo por la tecnología *WWF*, debido a que los modelos de procesos de negocio bajo el estándar *BPMN* que deben ser creados por expertos, deben ser expresivos, legibles y flexibles, cualidades que no pueden ser interpretadas en su totalidad por *WWF,* al no tener la capacidad de cubrir el amplio espectro de diseño que posibilita *BPMN.*

En la práctica, la utilización de estas tecnologías en el desarrollo de *software* requiere de una integración de cada una de las partes involucradas en función del progreso colectivo, por lo que cada uno de los roles que intervienen deben realizar correctamente su trabajo con el objetivo de que no existan trabas en el desarrollo del mismo.

Una buena relación cliente-analista-desarrollador es impulsora del avance en este ámbito. El cliente explica lo que desea obtener del sistema que modelará el analista y que más tarde, con una debida interpretación, el desarrollador llevará a cabo. Por consiguiente, todo el paquete de modelado de los procesos con el que contará la aplicación, será diseñado a través de la notación especificada por las necesidades del proyecto y

[4] Sistema de Gestión de Procesos de Negocio (*Business Process Management System*).

[5] Tecnología que proporciona un modelo de programación, durante el proceso del motor de flujo de trabajo y el para implementar procesos de larga duración como los flujos de trabajo dentro de las aplicaciones *.NET*.

será llevado a la práctica a través de tecnologías que por lo general no pueden ejecutar lo que se definió previamente sin la supervisión debida.

El Sistema Único de Identificación Nacional(SUIN) perteneciente al Centro de Identificación y Seguridad Digital, utiliza *BPMN* 1.0 a través de la herramienta *Altova UModel Enterprise Edition*[6] garantizando elevados niveles de detalle en la definición de los procesos de negocio y la tecnología *Windows Workflow Foundation* 3.5 acompañada del *Bison Framework*[7]*,* permitiendo al proyecto ser más flexible ante cualquier cambio además de mejorar su escalabilidad y velocidad en la trasferencia de tareas de un estado a otro.

En la mayoría de los casos los modelos de procesos de negocio con los que trabaja el analista, llegan a ser mal interpretados o tergiversados por los desarrolladores a la hora de traducirlos en los diagramas de flujos de trabajo, restándole importancia a estos diagramas, los cuales quedan archivados como parte de la documentación del proyecto. Como consecuencia, se termina automatizando un proceso de negocio diferente al concebido por los analistas y los clientes, siendo estos los principales expertos en cuanto a materia de negocio se trata.

Considerando lo anteriormente expuesto se deriva como **problema científico:** la transformación de los modelos de procesos de negocio, representados en *BPMN 1.0* generados por la herramienta *Altova UModel* a *Windows Workflow Foundation 3.5* y *Bison Framework*, genera incompatibilidades entre el diseño del proceso de negocio y su implementación, extendiéndose el tiempo de desarrollo de los procesos en el Sistema Único de Identificación Nacional.

Se plantea como **objeto de estudio** los modelos de procesos de negocio. Para dar respuesta al problema planteado se traza como **objetivo general** desarrollar una herramienta para la transformación de los modelos de procesos de negocio representados en *BPMN 1.0* generados por la herramienta *Altova UModel* a la tecnología *Windows Workflow Foundation* 3.5 y *Bison Framework* y el **campo de acción** se centra en la transformación de los modelos de procesos de negocio representados en *BPMN 1.0* generados por la herramienta *Altova UModel* a la tecnología *Windows Workflow Foundation* 3.5 y *Bison Framework.*

Para orientar la investigación se propone la siguiente **hipótesis:** la creación de una herramienta para la transformación de los modelos de procesos de negocio representados en *BPMN* 1.0 generados por la herramienta *Altova UModel* a *Windows Workflow Foundation* 3.5 y *Bison Framework*, reducirá la

[6]Herramienta de modelado.

[7]*Framework* para la orquestación de procesos de negocio con *Windows Workflow Foundation.*

incompatibilidad entre el diseño del proceso de negocio y su implementación así como el tiempo de desarrollo de los procesos en el SUIN.

Tareas de investigación:

- análisis de los principales conceptos asociados a *BPMN*.
- análisis del estándar de modelación *BPMN*.
- análisis de la tecnología *Windows Workflow Foundation* 3.5.
- análisis de las extensiones de tipo *XMI*.
- análisis de las herramientas que posibiliten la creación de *workflows* a partir de modelos de procesos de negocio representados en *BPMN* 1.0.
- análisis bibliográfico para determinar la metodología y herramientas a utilizar.
- análisis de la integración de *BPMN* con *Windows Workflow Foundation* 3.5.
- definición de los requisitos funcionales y no funcionales de la herramienta.
- extracción de datos de las extensiones de tipo *XMI*.
- definición de la arquitectura de la aplicación a desarrollar.
- aplicación de los patrones de diseño y estándares de codificación en la implementación de la herramienta.
- implementación de la Herramienta para la transformación de *BPMN* 1.0 a *Windows Workflow Foundation* 3.5.
- aplicación de las pruebas unitarias y de sistema a la herramienta.

Para el desarrollo de la investigación se utilizarán los siguientes métodos científicos:

Métodos teóricos

Investigación-acción: método de investigación cualitativa, que permitirá la unión de la teoría y la práctica, orientada siempre a la mejora de los procesos con una participación consciente de los actores involucrados en el mismo, con un alto protagonismo de la práctica, para la construcción de una estrategia participativa y colaborativa. Se utilizará este método ya que para proponer el uso de la herramienta que logre la transformación de *BPMN 1.0* a *WWF* 3.5 es necesario que los usuarios de la misma posean un conocimiento avanzado de la notación de modelado de proceso de negocio y de la tecnología de flujos de trabajo de *WWF* con el objetivo de que la transformación resultante sea lo más equivalente posible.

Analítico-sintético: se analiza toda la información recopilada a través de los diferentes medios bibliográficos que puedan servir para desarrollar mejor el diseño de la herramienta y se realizará un estudio

de los requerimientos de la aplicación que posibilitará la extracción de los elementos más importantes relacionados con el objeto de estudio.

Sistémico: se utilizará este método para estudiar el estándar *BPMN* mediante la determinación de sus componentes, así como la relación entre ellos. Esa relación determina por un lado la estructura y la jerarquía de cada componente en el objeto y por otra parte su dinámica.

Modelación: este método permite la creación de modelos (propuestas, alternativas y estrategias) que visualizan una reproducción simplificada de la realidad y que consisten en descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio.

Métodos empíricos

Observación: es el instrumento universal del investigador, el cual se centra en el estudio del fenómeno objeto de su investigación, observando directamente el desarrollo del proceso. Se realiza de forma consciente y orientada a un objetivo determinado. Este método es utilizado para analizar cada fase de la implementación y se adquiere experiencia de cada tarea, para la mejora de las mismas a la hora de desarrollar la propuesta de solución.

Entrevista: se utiliza como técnica para profundizar en los conocimientos sobre el estándar *BPMN* y los principales aspectos de *Windows Workflow Foundation* 3.5.

Justificación de la investigación:

A partir de la poca integración que existe en el Sistema Único de Identificación Nacional entre los modelos de procesos de negocio realizados por los analistas y su posterior implementación mediante la tecnología *WWF* 3.5, se decide la creación de una herramienta que transforme los modelos de procesos de negocio representados en *BPMN* 1.0 por la herramienta *Altova UModel* a flujos de trabajo de la tecnología *WWF* 3.5, la cual garantice una mayor correspondencia entre los modelos de procesos de negocio iniciales y los procesos finalmente automatizados, además de acortar el tiempo de desarrollo en el proyecto y de fomentar la relación de trabajo existente entre el analista y el desarrollador.

El presente documento constará de 4 capítulos. A continuación se ofrece una pequeña panorámica de lo que se tratará en cada uno de ellos:

Capítulo 1 Fundamentación teórica: se analizan los principales conceptos referentes a las tecnologías *BPMN* y *WWF*. Estudio de las principales herramientas asociadas al modelado de proceso de negocio mediante *BPMN*. Descripción de las tecnologías de desarrollo utilizadas para la propuesta de solución.

Capítulo 2 Propuesta de solución: se realizará una representación de los conceptos más importantes identificados mediante la realización de un modelo de dominio, además de la definición de los requerimientos funcionales y no funcionales de la propuesta de solución, realizando una descripción de los mismos.

Capítulo 3 Análisis y diseño: se define la arquitectura de la propuesta de solución y se realiza el modelo de clases del diseño, con el objetivo de convertir los requerimientos funcionales en una vista interna del *software* a implementar.

Capítulo 4 Implementación y pruebas: se realiza el desarrollo de la herramienta así como la ejecución de las pruebas de caja blanca y negra. Además, se validará la hipótesis y el cumplimiento de los objetivos trazados.

Capítulo 1 Fundamentación teórica

Introducción

La necesidad de crear una herramienta que permita la correcta interpretación de los procesos de negocio diseñados en *BPMN* por la tecnología *WWF* 3.5 queda reflejada en el presente capítulo, en el cual se desarrolla un estudio de los procesos modelados con *BPMN* y las tecnologías asociadas, así como los principales aspectos teóricos, el estudio del arte de la herramienta a desarrollar y las metodologías y herramientas a utilizar.

1.1 Conceptos básicos asociados al dominio del problema.

A continuación se profundizará en algunos de los conceptos más significativos relacionados con el problema planteado como son: proceso de negocio, *BPMN*, tecnologías de flujos de trabajo.

1.1.1 *BPM.*

Engloba un conjunto de herramientas tecnológicas que junto a una nueva filosofía de negocio permite diseñar la arquitectura empresarial modelando los procesos de negocio mediante flujos de trabajo, automatizando su funcionamiento de principio a fin y permitiendo su monitorización y control. A través de *BPM* se aborda el amplio mundo de la empresa a través de sus tres dimensiones: eficaces, transparentes y ágiles. *BPM* presenta tres finalidades que permiten adaptarse a los cambios impulsados por la globalización informática (3):

- mejorar la agilidad de negocio de una organización.
- incrementar la eficacia.
- mejorar los niveles de eficiencia.

1.1.2 Modelado de proceso de negocio.

Las herramientas que facilitan esta actividad visualizan, especifican, construyen y documentan el sistema informático, permitiendo la captura de información de larga vida que forma parte de los artefactos generados por el proyecto en sí.

El modelado de negocio es realizado principalmente por los analistas debido a que tienen la misión de entender cómo es que funciona el negocio, por lo que describen mediante diagramas, la secuencia de pasos que deben seguir las personas beneficiadas con las acciones realizadas que, en ocasiones, crean, modifican o acceden a contenedores de información llamados entidades.

Los objetivos del modelamiento de negocio son (4):

- comprender la estructura y la dinámica de la organización objetivo.

- comprender los problemas actuales de la organización objetivo e identificar el campo de acción incluido en dónde hay un potencial de crecimiento y mejoras.
- evaluar el impacto del cambio en la organización objetivo.
- asegurar que los clientes, usuarios finales, desarrolladores y otros roles tengan un entendimiento común de la organización objetivo.
- obtener, de forma preliminar, los requerimientos del sistema que necesita la organización objetivo.

1.1.3 *BPMN.*

BPMN es un estándar de la Iniciativa de Administración de Procesos de Negocios (*BPMI*), cuyo principal objetivo es "*proporcionar una notación fácilmente comprensible por todos los usuarios del negocio, desde los analistas, los desarrolladores y técnicos, hasta aquellos que monitorizarán y gestionarán los procesos*" (5).

Esta notación ha sido diseñada para coordinar la secuencia de eventos y los mensajes que fluyen entre los diferentes procesos participantes, haciendo de *BPMN* un estándar capaz de facilitar su comprensión por parte del personal no experto; además cubre casi totalmente los patrones de los flujos de trabajo con lo cual se le supone una gran expresividad a la hora de especificar procesos.

Otros objetivos importantes que plantea esta especificación son (6):

- crear puentes entre el diseño de los procesos de negocio y la implementación del proceso.
- que los lenguajes basados en *XML*[8] para describir procesos (como *BPEL4WS*[9]) tengan una notación gráfica.

Dentro de las principales **ventajas de *BPMN*** se encuentran (7):

- considera un único diagrama de proceso de negocio (*BPD*).
- pensado para ser asignado con naturalidad a lenguajes de ejecución como el lenguaje para servicios web de ejecución de procesos de negocio (*BPEL4WS*).
- fácil de entender.
- combina las capacidades del software y la experiencia de negocio para optimizar los procesos y facilitar la innovación del negocio.

[8] Lenguaje de Marcado Extensible (*Extensible Markup Language*).

[9] Lenguaje de Ejecución de Procesos para Servicios Web (*BPEL for Web Servi*ces).

1.1.3.1 Diagramas bajo el estándar *BPMN.*

Uno de los objetivos fundamentales de *BPMN* es habilitar un mecanismo simple para el diseño de los procesos de negocio y a la vez gestionar la complejidad que puedan desencadenar dichos procesos. Para garantizar el cumplimiento de estos requisitos se organizaron los aspectos gráficos en categorías específicas. Un diagrama de proceso de negocio (*BPD*), está diseñado para simbolizar todas las actividades que acontecen en un proceso, incorporando toda la información necesaria para el análisis. Ciertamente un *BPD* no es simplemente un conjunto de cajas y conectores. La semántica de estos diagramas esta implementada en la utilización de elementos gráficos agrupados en cuatro categorías que facilitan su comprensión. Las cuatro categorías son:

- Objetos de flujo
- Objetos conectores
- Artefactos
- Calles

Objetos de flujo: los elementos de flujo son los principales elementos gráficos que definen el comportamiento de los procesos. Un *BPD* posee Objetos de Flujo, de modo que los modeladores no tienen que aprender y reconocer un gran número de formas diferentes. Los tres objetos de flujo son:

- evento
- actividad
- puerta

Objetos conectores: los objetos de flujo se conectan entre ellos en un diagrama para crear el esqueleto básico de la estructura de un proceso de negocio. Hay tres objetos conectores que hacen esta función. Estos conectores son:

- flujo de secuencia
- flujo de mensaje
- asociación

Artefactos: *BPMN* fue diseñado para ofrecer a los modeladores y a las herramientas de modelado flexibilidad a la hora de extender la notación básica y de habilitar un contexto apropiado adicional según una situación específica. Se puede añadir cualquier número de artefactos a un diagrama como sea apropiado para un contexto de proceso de negocio específico. Hasta el momento en *BPMN* sólo se contemplan tres tipos de artefactos *BPD*, los cuales son:

- objetos de datos
- grupo
- mensaje

Calle: muchas metodologías de modelado de procesos usan el concepto de calles como un mecanismo para organizar actividades en categorías separadas visualmente y para ilustrar diferentes capacidades funcionales o responsabilidades. *BPMN* soporta las calles con dos constructores principales. Los dos tipos de objetos calles son:

- piscina
- carril

1.1.4 Modelado de proceso de negocio en *BPMN.*

Con *BPMN,* los procesos de negocio engloban la captura de una secuencia de actividades con información adicional. Modelar un proceso de negocio implica representar cómo una empresa realiza sus objetivos centrales; los objetivos por sí mismos son importantes, pero por el momento no son capturados por la notación (8). En el modelado *BPMN* se hacen notar tres niveles de modelado de proceso:

- mapa de procesos: nivel representado por simples diagramas de flujos de actividades, donde a lo sumo abarcan el nombre de las actividades y tal vez las condiciones de decisión más generales.
- descripción de procesos: en este nivel se incorpora datos más abarcadores del proceso, como son personas responsables de llevarlos a cabo.
- modelos de procesos: procesos bien detallados, con información suficiente para estudiarlos y simularlos. Además esta clase de modelos posibilita la ejecución directa de los procesos o importarlos a herramientas especializadas para esta tarea.

1.1.4.1 Tipos de modelos en *BPMN.*

Al modelar un proceso de negocio enlazamos una variedad amplia de información a una variedad amplia de audiencia. *BPMN* se diseña para cubrir esta gama de uso. Dentro de las posibilidades que brinda, se encuentra la habilitación de varios tipos de modelos, en dependencia del nivel de detalle que se le quiera dar a los mismos. Podemos diseñar el proceso siguiendo tres categorías:

Orquestación: los modelos de orquestación tienden a implicar una perspectiva única de coordinación, por ejemplo, representan una vista específica de negocio u organización del Proceso (9). Si usamos calles para representarlos, este tipo de proceso ocupará una calle aunque pueda interactuar, mediante el flujo de

mensajes, con otros procesos de negocio de la misma clase. En un diagrama bajo el estándar *BPMN* se pueden identificar fácilmente, suelen estar contenidos entre los elementos llamados piscina.

Coreografía: los modelos de coreografía están orientados a describir las interacciones entre un proceso de negocio privado y otro proceso de negocio, o bien un participante del proceso. Los participantes pueden ser roles definidos en el proceso o una entidad específica. En este tipo de procesos únicamente se incluyen aquellas actividades que se usan para comunicar un proceso privado con el exterior, así como las correspondientes estructuras de control de flujo. En *BPMN* las interacciones están asociadas al flujo de mensajes entre participantes. Para distinguir un modelo de coreografía en un diagrama bajo el estándar *BPMN* basta con tener varias piscinas en el diseño.

Colaboración: muestra la interacción entre distintas entidades de negocio. Estas interacciones son definidas como secuencias de actividades que representan el intercambio de mensajes entre las distintas entidades. La colaboración se entiende como la comunicación entre dos o más procesos. Una colaboración en *BPMN* puede contener una coreografía y varias orquestaciones. Específicamente un modelo de colaboración contiene dos o más piscinas representando a los participantes del proceso y a su vez, aunque no es requerido, contendrán una orquestación. Las piscinas tendrán mensajes entre ellas.

1.1.5 Tecnología de flujos de trabajo.

Los diseñadores de procesos cuentan con lenguajes y herramientas de modelado para la definición de los procesos de negocios. El empleo de tecnologías que logren la interpretación y la ejecución de procesos de negocios son de suma importancia para obtener una forma estandarizada de los mismos. La tecnología de flujos de trabajo surge por la necesidad antes explicada.

Un *workflow*[10] (*WF*) crea definiciones basadas en las condiciones del flujo determinado durante el diseño, informa sobre el estado en que se encuentra una actividad, notifica a las personas involucradas en el proceso, etc., pero en general facilita la coordinación del trabajo entorno a las capacidades de compartir información, incrementando las mejoras en el desarrollo de las actividades de una organización.

La *Workflow Management Coalition* define un *WF* como: “La automatización de un proceso de negocio, en su totalidad o en parte, cuando documentos, información o tareas se pasan de un actor a otro, para realizar una acción, de acuerdo con una serie de reglas de procedimiento” (10).

[10] En español: flujo de trabajo.

1.1.5.1 Beneficios del *workflow.*

Entre los beneficios de los *workflow* se pueden encontrar: (11):

- ahorro de tiempo y mejora de la productividad y eficiencia de la empresa, debido a la automatización de muchos procesos de negocio.
- mejora del control de procesos a través de la normalización de los métodos de trabajo.
- mejor atención y servicio al cliente; un incremento en la coherencia de los procesos da lugar a una mayor previsibilidad en los niveles de respuesta a los clientes.
- mejora en los procesos; mayor flexibilidad de acuerdo con las necesidades empresariales.
- optimización de la circulación de información interna con clientes y proveedores.
- integración de procesos empresariales.

Los *workflows* se centran en la habilitación de mecanismos que resuelvan problemas de estandarización en la definición de procesos de una forma no ambigua y que permita su replicación. Sin embargo no es conveniente aferrarse a la tecnología de flujos de trabajo para solucionar cualquier problema. Por ejemplo, un programa informático que nunca vaya a modificarse es preferible implementarlo con un lenguaje de programación estándar que utilizando *WF*, ya que, los *WF* resultan menos potentes que dichos lenguajes, además de ser mucho menos eficientes (12). Siguiendo estas indicaciones, el empleo de sistemas de *WF* está reservado para procesos repetitivos que:

- requieran una legibilidad a un alto nivel, como son los diseñados por expertos sin ningún tipo de conocimiento en programación.
- requieran un control de flujo determinado que garantice la localización del estado actual de un flujo y de sus caminos a seguir.
- contemple la posibilidad de modificar una instancia de un flujo, para que se ejecute de una manera diferente a la que fue concebida inicialmente.
- garanticen un manejo de la tarea en contraposición con una actuación autónoma, en otras palabras: que necesite confirmación de los pasos a seguir.

1.1.5.2 Sistema de Gestión de *Workflow* (*WMS*).

Un *WMS*[11] es "*un sistema que define, crea y gestiona la ejecución de flujos de trabajo mediante el uso de software, siendo capaz de interpretar la definición del proceso, interactuar con los participantes y siempre*

[11] Sistema de gestión de flujo de trabajo (*Workflow Management System*).

que se requiera, invocar el uso de herramientas y aplicaciones" (13). Un *WMS* presenta dos actividades fundamentales, la definición del *WF* que representa el proceso de negocio y su posterior ejecución. Sin duda el proceso de negocio constituye el eslabón principal pero se necesita admitir en el diseño flujos de entrada y de salida, interacción con los usuarios en aquellas actividades que lo requieran y la comunicación con otros sistemas o servicios.

1.1.5.3 Patrones de *workflow.*

La capacidad de expresividad de un lenguaje de representación de *WF* viene dada por la cantidad de patrones que pueda admitir este, por lo que mientras más patrones sea capaz de expresar un lenguaje, más expresivo será. La iniciativa que surge para la creación de patrones de comportamiento de un *WF por* Will Van der Aalst, publica una serie de patrones que deben cumplir un sistema de gestión de *WF*. Estos patrones no solo están orientados a la representación de *WF* sino también al modo de ejecución, donde recursos e información son compartidos entre las acciones que se realizan. Están agrupados en 4 categorías que recogen las principales funcionalidades de estos. Dichas categorías son (12):

- patrones de control de flujo: estos patrones se corresponden con una revisión de los patrones de control de flujo presentados por Will Van der Aalst en los que se definen situaciones de gestión del flujo de procesos de negocio, tales como sincronizaciones o separación paralela de procesos.
- patrones de datos: estos patrones ofrecen una serie de conceptos que pueden aplicarse a la utilización de datos en los sistemas de *WF*. No solo estandarizan los datos y la forma en la que deben utilizarse estos datos en un *WF*, además, definen su interacción con otros *workflows* o sistemas externos.
- patrones de recursos: están centrados en modelos de recursos y su interacción con sistemas de gestión de procesos de negocio, los recursos pueden ser humanos o no, como equipamiento, salas, etc.
- patrones de manejo de excepciones: mediante estos patrones se ofrecen soluciones, dígase un conjunto de acciones a realizar, en caso de producirse una excepción durante la ejecución de un *WF*.

Representación: las operaciones de definición de *WF*, no son más que la transformación de los procesos de negocio concebidos, dotados de artefactos y requisitos que necesita el sistema para su ejecución. Además, la representación contiene toda la información necesaria acerca de los procesos, como la información de comienzo de actividades, condiciones, y reglas de navegación a través de gráficas o mediante un lenguaje semiformal. Algunos sistemas lo hacen mediante una interfaz gráfica, otros, prefieren

definir los procesos a través de extensiones *XML*. Dentro de estos sistemas pueden existir alteraciones dinámicas a la definición del proceso durante el tiempo de ejecución.

Interpretación y ejecución: una vez representado el proceso de negocio como un *WF*, se procede a su interpretación y ejecución, garantizando el cumplimiento de los objetivos finales del proceso. La ejecución de *WF* es un problema de compromiso entre la complejidad del modelo de representación y las necesidades específicas de los usuarios que quieren ejecutar los *WF*. La interpretación de los *WF* requiere que los lenguajes de representación sean formales, lo suficientemente sencillos para ser ejecutados y carezcan de ambigüedad (12). Existe una diferencia notable entre los lenguajes para definir procesos ejecutados por seres humanos y los lenguajes para la representación de flujos de trabajo.

Existen lenguajes bastante legibles orientados a la representación de *WF* pero limitan el control y la ejecución, debido a que resultan ambiguos y menos formales. Otros lenguajes favorecen el control y la ejecución de los flujos de trabajo pero a la vez no complementan la representación. La ejecución de *WF* se basa en la generación de instancias de *WF* para cada ejecución individual. Cada una de estas instancias coordinan la ejecución de los procesos para cada caso particular siguiendo uno de los caminos posibles definidos en el *WF* según de la instanciación de la reglas en cada momento.

1.1.5.4 Motor de *WF.*

Un elemento fundamental dentro del control de la ejecución de los proceso lo constituye el motor de *WF*. Un sistema de interpretación de *WF*, o motor de *WF*, es un componente *software* que toma como entrada un *WF* y mantiene el estado de ejecución de los procesos delegando y distribuyendo las actividades a realizar de los procesos entre actores humanos y aplicaciones *software* (14). En otras palabras, es un sistema que a partir de un *WF* de entrada es capaz de interpretar el modelo, permitir la ejecución de cada una de las tareas y controlar la interacción con los actores involucrados conforme a las reglas definidas en el lenguaje de representación.

1.1.5.5 *Windows Workflow Foundation* (*WWF*).

WWF es una tecnología del *Framework .Net* que permite construir flujos de trabajo que modelen componentes funcionales concretos dentro de un sistema de negocio específico. Mediante un diseñador visual de modelos se puede construir diagramas que representen el diseño y la estructura del programa, facilitando su lectura por otros desarrolladores. El diseñador de flujos de trabajo genera la mayor parte de la infraestructura de comunicación necesaria entre el código y el flujo modelado.

La potencia de *WWF* reside en que, una vez modelada la lógica de negocio mediante un flujo de trabajo, se puede cargar en tiempo de ejecución dicho modelo, hacer modificaciones condicionales en su estructura y ejecutarlo como un subproceso (con su propio contexto de datos y código) dentro de cualquier aplicación cliente.

Una vez que se ha definido un modelo, este se puede reutilizar tantas veces como se quiera dentro de una aplicación o ensamblado. De esta forma, un modelo pasa a ser parte funcional del proyecto y no solo documentación estática. La plataforma *workflow* exporta un marco de desarrollo extensible y adaptable a los distintos requerimientos de cada proyecto.

WWF es utilizado como una herramienta poderosa para el desarrollo de aplicaciones. Ofrece los siguientes beneficios a los desarrolladores (15):

- coordina el trabajo realizado por las personas y el software: las personas juegan un papel importante en el mundo de los sistemas de software relacionados con el flujo de trabajo y procesos. La interacción humana con frecuencia se hace a través de correo electrónico, páginas *Web*, dispositivos móviles, o de otros extremos delanteros. Ofrece además, una serie de características y la infraestructura para manejar con eficacia la interacción humana y todas las cuestiones relacionadas con ella.
- son de larga duración y basado en estados: proporciona un rico marco de los servicios de tiempo de ejecución, que permiten persistir flujos de trabajo con un soporte duradero. Dado que los flujos de trabajo pueden funcionar durante largos períodos de tiempo, el almacenamiento de un flujo de trabajo ejecutado en la memoria no es práctico por muchas razones. Si cada flujo de trabajo ejecutado tiene que ser almacenado en la memoria a la espera de que algo pase, el servidor se quedará sin memoria inmediatamente. Además, si se bloquea el servidor, la memoria volátil se borra y todos los datos se perderán.
- se basan en modelos extensibles: casi todas las partes de la plataforma de *WF* son extensibles. Los flujos de trabajo se componen de distintas acciones, conocidas como las actividades o la creación de nuevas actividades para cubrir sus necesidades. Otras partes de la plataforma, como los servicios (por ejemplo, el seguimiento, la gestión y la persistencia) proporcionada por el motor de ejecución, son también extensibles.
- es transparente y dinámico a lo largo de su ciclo de vida: los flujos de trabajo son transparentes y dinámicos, tanto en tiempo de diseño como en tiempo de ejecución. Está basado en el diseño del modelo declarativo y visual, los flujos de trabajo existentes se pueden modificar sin cambiar el código

fuente. Además, en tiempo de ejecución, se puede consultar el estado del flujo de trabajo, registrar información en un medio permanente para una inspección posterior, e incluso modificar los flujos de trabajo.

1.1.5.5.1 Creación de *workflow* con *WWF.*

WWF ofrece varias maneras para la creación de flujo de trabajo, separando de forma predeterminada la definición de la lógica comercial. Para crear un flujo de trabajo, se cuenta con un diseñador de intermediario que facilita el modelado de la estructura de *WF* y mediante código *C-Sharp* se establece la lógica. Este modo de creación se denomina separación del código.

WWF admite los modos de creación siguientes para la implementación del flujo de trabajo (16):

- **Sólo código**: es el modo de creación establecido para *WWF*. Le permite utilizar *C-Sharp* o código *Visual Basic*[12] para especificar un flujo de trabajo utilizando la *API*[13] establecida por *Windows Workflow Foundation*. Al modelar un flujo de trabajo en solo código se genera automáticamente código en un archivo *Workflow1.designer.cs* y el código que se quiera agregar es alojado en un archivo *Workflow1.cs.* La generación de código trabaja conjuntamente con el diseño del *WF*, adicionando actividades, adicionando hijos a estas actividades y así sucesivamente. Es necesario compilar el flujo de trabajo de solo código.
- **Sin código**: los *WF* sin código están definidos enteramente en un archivo de extensión *.xoml*. Pueden compilar con el compilador de flujo de trabajo de línea de comandos de *Windows Workflow Foundation* o pueden cargar el archivo de marcado de flujo de trabajo en el motor en tiempo de ejecución de flujo de trabajo a través de una aplicación *host*. El uso de una notación *.xoml* brinda mucha más flexibilidad en el manejo y distribución del *WF*, al no estar embebido en un ensamblado.
- **Separación de código**: permite definir los flujos de trabajo utilizando el marcado del flujo de trabajo y combinándolo con *C-Sharp* o *Visual Basic* tras implementaciones. A diferencia del modo de creación sin código, los flujos de trabajo separados por código deben estar compilados y no tener la opción de cargarse directamente en el motor en tiempo de ejecución del flujo de trabajo. Los flujos de trabajo de separación de código almacenan el modelo serializado [14] del *WF* en el archivo de marcado

[12] Lenguaje de programación dirigido por eventos, desarrollado por Alan Cooper para *Microsoft.*

[13] Interfaz de programación de aplicaciones (*Aplication Programming Interface*).

[14] Proceso de convertir el estado de un objeto a un formato que se pueda almacenar o transportar.

Workflow1.xoml y el código en un archivo separado llamado *workflow1.xoml.cs*. El marcado del archivo *.xoml* utiliza la sintaxis conocida como *XAML*[15] para el almacenamiento.

1.2 Estudio de las soluciones asociadas.

1.2.1 Lenguajes existentes para la representación e interpretación de procesos.

XPDL: lenguaje de definición de proceso de tipo *XML* para la descripción de procesos, posee un formato de fichero estándar basado en *XML* a partir del cual se permite el intercambio de información entre herramientas de modelado y herramientas orientas a la ejecución. *XPDL* contiene elementos para llevar a cabo la información gráfica, tal como la posición de X y de Y de los nodos, así como los aspectos ejecutables que serían utilizados para funcionar un proceso. Esto distingue *XPDL* de *BPEL* centrándolo exclusivamente en los aspectos ejecutables del proceso. *BPEL* no contiene elementos para representar los aspectos gráficos de un diagrama de proceso. Existe una estrecha relación entre *BPMN* y *XPDL*, siendo considerada a veces como la notación textual de *BPMN* o a la inversa, *BPMN* la notación gráfica de *XPDL*. La última versión de *XDPL* 2.0 soporta los elementos gráficos de *BPMN* que no contemplaba la versión 1.0 (6).

Por lo tanto *XPDL* y *BPMN* son un binomio a tener muy en cuenta dentro de campo del modelado de procesos de negocio. Para darle efectividad a esta pareja, y siempre que mantengan compatibilidad lo ideal sería encontrar una herramienta que nos permita usar ambas especificaciones de la siguiente manera (6):

- usar *BPMN* para modelar de manera gráfica los modelos de procesos de negocio (lo cual es más amigable tanto para los ingenieros como para los clientes).
- *XPDL* para guardar los modelos e intercambiarlos entre las diferentes aplicaciones.

BPEL: el lenguaje *BPEL* (del inglés, *Business Process Execution Language*) es un lenguaje de orquestación de procesos definido por el comité de estandarización *OASIS*[16]. *BPEL* nació como combinación de los lenguajes *WSFL*[17] de *IBM* y *XLANG*[18] de *Microsoft*. *BPEL* fue un acuerdo entre estas

[15] Lenguaje Extensible de Formato para Aplicaciones (*Extensible Application Markup Language*).

[16] Organización para el Avance de Estándares de Información Estructurada (*Organization for the Advancement of Structured Information Standards).*

[17] Lenguaje de flujo de servicio *web* (*Web Services Flow Language*).

[18] Lenguaje de orquestación implementado por *Microsoft.*

dos empresas para definir un lenguaje común que permitiera realizar conexiones entre las aplicaciones creadas tanto por productos de *Microsoft* como de *IBM* (12).

BPEL es un lenguaje pensado para ser ejecutado, por lo que no tiene una notación gráfica para el diseño de procesos. *BPEL* cuenta con la ventaja de la formalidad en la representación de la orquestación de procesos, ya que es un lenguaje que se ejecuta directamente por lo que no admite ambigüedad (12).

La generación de código *BPEL* a partir de los diagramas *BPMN* lo único que ha logrado es aflorar las diferencias que existen entre este lenguaje de orquestación y la notación de procesos de negocios.

El uso de *XPDL* y de *BPEL* puede ser complementario en el caso de que se utilicen de una manera correcta, pero en realidad, estos dos estándares están pensados para ser utilizados en campos distintos de los procesos de negocio. El primero por ser un modelo representativo, y el segundo por interpretar los flujos de trabajo.

1.2.2 Soluciones asociadas.

BPMN fue, ya se dijo anteriormente, una iniciativa de la *BPMI*. Esta institución fue absorbida por la *OMG*[19] que es una de las organizaciones punteras en la creación de especificaciones para el desarrollo de software orientado a objetos.

BPMN fue adoptada para dar soporte únicamente a los procesos de negocio, por lo que cualquier otro tipo de modelado con fines distintos a los del negocio no estará en su ámbito.

Entre las herramientas que utilizan la notación *BPMN* podemos citar *BizAgi*, *Bonita* e *Intalio*. A continuación se realizará un pequeño análisis de sus comportamientos teniendo como punto de mira la generación de flujos de trabajos a partir de los modelos de negocio diseñados.

Bonita Workflow: *Bonita Workflow* está orientada a definir flujos de procesos orientados al usuario, siendo el motor de *Bonita* el que mantiene la lógica de *workflow* de manera independiente al modelo de negocio de la organización.

La utilización de *Bonita Workflow* trae consigo los siguientes beneficios (17):

- incentiva la eficiencia de equipos de trabajo fomentando la colaboración. Un equipo puede visualizar las tareas concurrentes y cada individuo en tiempo real puede conocer el estado de un proceso,

[19] *Object Management Group.*

permitiendo tener estadísticas a nivel de proceso e instancia de proceso, tiempos de atención de cada tarea y otras métricas en el mismo ámbito.

- reduce los costos y riesgos de automatización de procesos persona - persona y sistema - sistema. Los procesos pueden correr en organizaciones que funcionan en localizaciones geográficas distantes y el *workflow Bonita* permite enlazarlos y aprovecharlos de manera eficiente.
- maneja eficientemente situaciones inesperadas. *Bonita* permite redefinir de manera dinámica y segura un proceso de suerte en el que se pueden incluir eventos que no fueron previamente identificados.
- toma beneficios de algunas características provistas por un servidor de aplicaciones *JEE*[20], como pueden ser el uso de transacciones, autenticación basada en roles y ciclo de vida de aplicaciones, así como también la conexión con sistemas externos.

Esta es una solución *BPM*/*Workflow* que presenta versiones tanto para *Windows* como para *Linux* bajo licenciamiento *LGPL*[21].

Tiene como principal requerimiento la instalación de tres *software* para su posterior uso (*JDK*[22] 1.4 o superior, pero se recomienda emplear *JDK* 1.5*, ANT, JONAS Application Server*). Este *software* está orientado a la creación y modificación de los modelos de proceso de negocio sobre la red.

Exporta las definiciones de los flujos de trabajo en extensión *XPDL* para su posterior modelación en la *web* y ofrece una paleta para la construcción de flujos de trabajos mediante la cual es posible garantizar la mayor cantidad de patrones de diseño de *workflow* para la integración y orquestación de procesos de negocio de manera eficiente y ágil. Posee además un motor *workflow BPM* muy intuitivo y fácil de configurar y usar (17).

BizAgi: fue diseñado para ofrecer resultados inmediatos con herramientas con el objetivo de diagramar procesos en *BPMN*, definir reglas de negocio, orquestar otras aplicaciones, definir interfaz de usuario, optimizar y balancear cargas de trabajo, manejar portales *webs* de trabajo, indicadores de desempeño de procesos, monitor de actividades y mucho más (18).

[20] Edición corporativa de java (*Java Enterprise Edition*).

[21] Licencia Pública General Reducida (*Lesser General Public Licence*).

[22] Paquete de desarrollo de Java (*Java Development Kit)*. *Software* que provee herramientas de desarrollo para la creación de programas en Java.

El termino *BizAgi* proviene de *Business Agility*. Es un sistema que se dedica a la administración de procesos de negocio basado en la metodología *BPM*. Siguiendo un modelo de "no código", permite diseñar, modelar, integrar, automatizar y monitorear los procesos a través de un entorno gráfico y sin la necesidad de programación. Contiene dos componentes fundamentales para la gestión de los procesos, las herramientas *BizAgi Process Modeler y BizAgi BPMSuite*.

Las tareas de definición y descripción de los procesos de negocio son realizadas mediante el *BizAgi Process Modeler*, el cual permite diagramar y documentar dichos procesos bajo el estándar *BPMN*, que además permite exportar los modelos creados a un formato *XPDL* así como *PDF*, *Word*, *Visio*, entre otros, facilitando el intercambio de los diagramas con personas que no tengan el modelador instalado.

La *BizAgi BPM Suite* se encarga de la ejecución, monitoreo y control de los procesos. Una vez creados los modelos de proceso de negocio, dichos modelos pueden ser exportados a *BizAgi BPM Suite*, permitiendo la ejecución de los flujos de trabajo correspondientes. A la vez estos *workflows* pueden definirse en la misma herramienta, a través del módulo de creación representado por el *BizAgi Studio*. *BizAgi Server* representa el módulo de ejecución y monitoreo dentro de *BizAgi BPM Suite* (19).

***Intalio*:** *Intalio* constituye el primer y principal Sistema de Gestión de Negocios *Open Source*. Adopta estándares abiertos y es independiente de cualquier tecnología propietaria. *Intalio* | *BPMS* proporciona todos los componentes necesarios para diseñar, implementar y administrar los procesos de negocio más complejos, incluido el monitoreo de las actividades del negocio, la gestión de reglas de negocio, documentación y gestión de contenidos, integración de sistemas, protocolos de negocio a negocio y herramientas de la web del portal (20).

Entre los componente que lo integran encontramos un diseñador llamado *Intalio Designer*, el cual posibilita la creación de modelos de procesos de negocio con el estándar *BPMN*, que a su vez puede ser convertido a *BPEL* por el propio diseñador lo cual es imprescindible para la ejecución del proceso por el servidor. *Intalio Designer* utiliza componentes de proyectos de la comunidad Eclipse facilitando el trabajo a un usuario familiarizado con el *IDE*[23] de *Eclipse*.

Intalio Server es un componte a parte de la *Intalio Suite*, se integra perfectamente con el *Intalio Designer*, de forma tal que los procesos modelados pueden exportarse directamente al servidor para su posterior ejecución. Como característica principal, esta herramienta no solo ejecuta procesos definidos por el

[23] Entorno de desarrollo integrado (*Integrated Development Environment*).

Designer si no que se acopla con otras herramientas de diseño. Esta modularidad permite una mayor personalización de las herramientas utilizadas y por otro lado, provee más posibilidades para integrar el servidor en un ambiente ya existente por ejemplo como reemplazo o como sistema de resguardo.

Se puede llegar a la conclusión de que hasta el momento no existe en el mundo alguna herramienta que a partir de un estándar *BPMN* permita la creación de flujos de trabajo compatibles con la tecnología *Windows Workflow Foundation* 3.5. Para el análisis del estudio del arte se referenciaron algunas herramientas que en el contexto de *BPM*, posibilitan el modelado con *BPMN* y la definición y ejecución de flujos de trabajo a partir de dichos modelos. Ninguna de estas herramientas permite la transformación de los diagramas de procesos de negocios a la tecnología *WWF* 3.5. Lo más cercano a una posible transformación de *BPMN* a una aplicación orientada a objetos es la generación de un esqueleto para *Java* a través de un generador de código llamado *Axgen*[24], el cual usa ficheros de extensión *XMI*[25] como entrada y plantillas de velocidad para su transformación (*Jakarta Velocity*[26]), en la que se accede al modelo *metadata* por medio de una capa de abstracción *UML* con implementación, la cual provee una vista orientada a objeto del modelo.

La integración de *BPMN* y *WWF* 3.5 en la actualidad no ha sido lograda a pesar de que ambas están encaminadas a resolver problemas en común, por lo que se llega a la conclusión que es necesario la implementación de una herramienta que realice esta transformación en el Sistema Único de Identificación Nacional.

1.3 Ambiente de desarrollo.

1.3.1 Metodología de desarrollo.

En un proyecto de desarrollo de *software* la metodología de desarrollo define quién debe hacer qué, cuándo y cómo debe hacerlo. Las metodologías se basan en la combinación de los modelos de procesos y definen el conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto (21).

[24] *Axgen* es un generador de código que presenta como principal motivación la generación de clases y otros objetos para su uso como una herramienta de mapeo de objetos relacional (O/R *Mapping*) que permite la persistencia transparente de objetos Java con bases de datos relacionales. Viene con un paquete de plantillas listas para usar para la generación de un Puente Relacional de Objetos.

[25] *XML* de Intercambio de Metadatos (*XML Metadata Interchange*).

[26] *Jakarta Velocity* es un proyecto de *software* libre dirigido por la *Apache Software Foundation.* La velocidad es un motor de plantillas basado en Java que proporciona un lenguaje de plantillas sencillo pero potente a los objetos de referencia definido en el código de Java.

***MSF for CMMI*:** el surgimiento de *Visual Studio Team System* trajo aparejado la aparición de una poderosa herramienta para el trabajo con una metodología encaminada a guiar y mejorar el proceso de desarrollo de *software: MSF for CMMI Process Improvement*. Debido a que brinda un entorno flexible y perfectamente ajustable al desarrollo del proyecto, permite que los miembros del equipo puedan asumir uno o varios roles en función de las necesidades del ambiente de desarrollo, realizando actividades que producen resultados de valor observable como son la documentación, el código fuente o el plan del proyecto.

MSF es un proceso alineado con *CMMI* que utiliza el mismo paradigma: iteraciones, roles, y reportes que proporcionan las métricas necesarias para valorar el estado del proyecto. Una de las ventajas de usar el proceso alineado con *CMMI* es la evaluación estándar por la cual se puede comparar la capacidad de desarrollar el *software* en otras organizaciones.

La utilización de *MSF for CMMI* ayuda al equipo a ejecutar procesos de desarrollo de software que satisfagan los requisitos de *CMMI*.

MSF for CMMI proporciona las siguientes herramientas (22):

- una plantilla de proceso para *Team Foundation* que define elementos de trabajo, informes y otras herramientas.
- la guía de procesos.

Las situaciones y los procedimientos de trabajo de los equipos de desarrollo varían ampliamente y la mayoría de las compañías tendrán sus propios procesos correctamente establecidos. Por estos motivos, la guía proporcionada no intenta recomendar un proceso de desarrollo en su totalidad. En su lugar, se describen sólo las actividades más relevantes para mejorar el uso de la plantilla de proceso de *MSF for CMMI*.

Es imprescindible adaptar esta guía a su propia situación, que dependerá del tipo e historial del producto que está desarrollando, la escala del proyecto, los conocimientos de los miembros del equipo y los procedimientos aceptados en su organización. El uso de la guía y la plantilla de *CMMI* ayudan a lograr los objetivos de *CMMI* si lo utiliza como parte de un programa de mejora de procesos.

El modelo *CMMI* se ha diseñado para que se use como base de las iniciativas enfocadas a mejorar los procesos y, en el ámbito de la evaluación, únicamente como ayuda para medir las mejoras. Este enfoque ha dado lugar a resultados mixtos. Resulta demasiado fácil confundir el modelo con una definición de proceso e intentar seguirlo en lugar de considerarlo como un mapa que identifica las lagunas en los procesos existentes que habría que rellenar. El bloque de creación fundamental del modelo *CMMI* es un

área de proceso que define los objetivos y varias de las actividades que se suelen realizar para lograr dichos objetivos (23).

CMMI se traduce en mayor productividad y calidad, resultando en un menor desarrollo y los costes de mantenimiento. El beneficio es el valor económico de la utilización de *CMMI* para crear un proceso de software nuevo y mejorado. Su metodología de beneficio es un proceso de tres partes que consiste en la estimación de costes de las pruebas, los costos totales del ciclo de vida de las pruebas, y los beneficios. Su metodología de beneficio consiste en una variedad de modelos defectuosos utilizados en combinación. Los elementos clave son el modelo del costo de prueba y el modelo de ciclo de vida total de los costos. Algunos de los beneficios se deben a una mayor productividad, resultando en hasta un 50% de disminución en los costos de desarrollo de software. Aumento de la productividad es un factor en el modelo del costo de vida total del ciclo. La prueba y el total de los modelos de costes de ciclo de vida se utilizan para comparar los costos totales del ciclo de vida de las pruebas de software a los de la *CMMI* (24).

Los cinco niveles de madurez de *CMMI* quedan evidenciados de la siguiente manera:

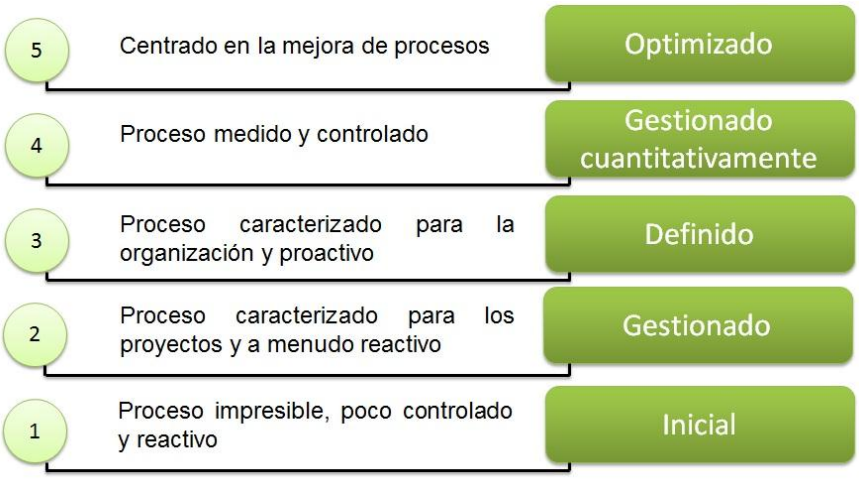


Figura 1 Niveles de madurez de CMMI[27].

1.3.2 Tecnologías a utilizar.

***Framework .Net* 4.0:** *Microsoft .Net Framework* 4.0 es un componente integral de *Windows* que admite la compilación y la ejecución de la nueva generación de aplicaciones y servicios *web*. Sus componentes principales son el *Common Language Runtime* (*CLR*) y la biblioteca de clases *.Net Framework*, que incluye

[27] Todas las fuentes de figuras y tablas fueron elaboración propia de los autores a menos que se indique lo contrario.

ADO.Net[28], *ASP.Net*, *Window Forms* y *Windows Presentation Foundation* (*WPF*). El *Framework .Net* proporciona un entorno de ejecución administrado, un desarrollo e implementación simplificada y la integración con una gran variedad de lenguajes de programación.

Este *Framework* brinda nuevas mejoras y características que lo hacen diferente a sus versiones anteriores, aunque posee la capacidad de funcionar en paralelo con ellas (25):

- mejoras en *CLR*, *ASP.Net* e innovaciones en los lenguajes *Visual Basic* y *C-Sharp*.
- *Entity Framework*[29], donde se permite a los desarrolladores programar con bases de datos relacionales usando objetos *.Net* y *Language Integrated Query*[30] (*LINQ*).
- servicios de datos de *Windows Communication Foundation* (*WCF*), componente que permite crear servicios y aplicaciones que usen protocolo de datos abierto para exponer y usar datos a través de la web.
- compatibilidad con formularios para nuevas mejoras en la librería *AJAX*[31].

***Microsoft Visual Studio* 2010:** *Visual Studio* 2010 es la versión más reciente de esta herramienta que viene acompañada del *Framework .Net* 4.0.

Se ha rediseñado el *IDE* con el fin de mejorar la legibilidad. Se han quitado las líneas y los degradados innecesarios para conseguir una mayor claridad. Las ventanas de documento, como la ventana Editor de código y Vista de diseño, pueden situarse ahora fuera de la ventana del *IDE*. Por ejemplo, puede arrastrar el editor de código fuera del *IDE* para poder verlo en paralelo a la ventana Vista de diseño (26).

Entre sus principales novedades se encuentran (26):

- se describen las nuevas características del lenguaje *C#* y del editor de código. Algunas de estas características son el tipo ***dynamic***, los argumentos opcionales y con nombre, la programación de *Office* mejorada y la varianza.
- Contiene información sobre las principales características y mejoras de *.Net Framework* 4.0.

[28] Conjunto de componentes del *software* que pueden ser usados por los programadores para acceder a datos y a servicios de datos.

[29] Conjunto de tecnologías de *ADO.Net* que permiten el desarrollo de aplicaciones de software orientadas a datos.

[30] Lenguaje Integrado de Consulta.

[31] *JavaScript* asíncrono y *XML* (*Asynchronous JavaScript And XML*).

- Se describen las nuevas funciones de administración del ciclo de vida de las aplicaciones en *Visual Studio* 2010, que reemplazan *Team Foundation Server*.
- Describe las nuevas características del diseñador de informes de *Visual Studio*, el asistente para informes y los controles ***ReportViewer***.

Lenguaje de programación *C#*: leído en inglés "*C-Sharp*", es el primer lenguaje orientado a componentes en la familia de lenguajes *C* y *C++.* Es un lenguaje de programación simple, moderno, orientado a objetos y con un sistema de tipos seguro derivado de *C* y *C++. C#* combina la alta productividad de *Microsoft Visual Basic* y la eficacia bruta de *C++* (27). Se le considera como el lenguaje nativo de *.Net*.

Permite el ahorro de tiempo en la programación ya que tiene una librería de clases muy completa y bien diseñada, posee además características necesarias como la herencia, polimorfismo, encapsulación y los métodos virtuales. Incorpora elementos que facilitan el trabajo de los desarrolladores tales como (27):

- gestión automática de memoria, liberando de ese modo a los desarrolladores de una gestión manual de memoria costosa en tiempo y propensa a errores.
- todo es un objeto. A través del uso innovador de conceptos como empaquetado y desempaquetado, *C#* llena el vacío entre tipos valor y tipos referencia, permitiendo que cualquier dato sea tratado como objeto.
- las propiedades, métodos y eventos son fundamentales. Muchos lenguajes omiten el soporte intrínseco a propiedades y eventos, creando un desajuste innecesario entre el lenguaje y los *frameworks* asociados.
- soporta atributos, que permiten la definición y uso de información declarativa sobre componentes.

LINQ: abre ante los desarrolladores la posibilidad de expresar de una manera uniforme e integrada consultas contra las más diversas fuentes de datos que estos utilizan en sus aplicaciones (colecciones de objetos en memoria, documentos *XML* o bases de datos relacionales, entre otras), mediante las expresiones de consulta que constituyen el principal reflejo en los lenguajes de la tecnología *LINQ*.

LINQ incluye patrones estándar y de fácil aprendizaje para consultar y actualizar datos, y su tecnología se puede extender para utilizar potencialmente cualquier tipo de almacén de datos. *Visual Studio* incluye ensamblados de proveedores para *LINQ* que habilitan el uso de *LINQ* con colecciones de *.Net Framework*, bases de datos *SQL Server*, conjuntos de datos de *ADO.Net* y documentos *XML* (28).

Las ventajas más importantes que aporta *LINQ* al mundo de la programación son (29):

- Sintaxis familiar para escribir consultas.

- Comprobación en tiempo de compilación de errores de sintaxis y seguridad de tipos.
- Compatibilidad mejorada con el depurador.
- Compatibilidad con *IntelliSense*.
- Capacidad para trabajar directamente con elementos *XML* en lugar de crear un documento *XML* contenedor, que es lo que se requiere en *W3C DOM*.
- Modificación de documentos *XML* en memoria de gran eficacia, aún más fácil de usar que *XPath*[32] o *XQuery*[33].
- Funciones de filtrado, ordenación y agrupación eficaces.
- Modelo coherente para trabajar con datos en varios tipos de formatos y orígenes de datos.

XMI: es una especificación de *XML* utilizado para el intercambio de diagramas entre herramientas de modelado. *XMI* surge a partir de la idea de varias industrias de crear un estándar aprovechando los beneficios de *XML* para estandarizar la definición, validación y compartición del formato de los documentos en la *web*.

El principal objetivo de *XMI* es permitir un intercambio de meta-información entre herramientas de modelado, basados en *UML*, y repositorios de meta-información, basados en *MOF*[34], en entornos heterogéneos distribuidos (30). El hecho de incluir tres estándares como *XML*, *UML* y *MOF*, permite a los desarrolladores de sistemas distribuidos compartir modelos de objetos y otra información sobre *Internet*.

Estas son algunas de las ventajas que brinda el uso de *XMI* (30):

- una de las ventajas que proporciona *XMI* es el hecho de trabajar con *Internet* y que está construido en base a estándares industriales como *HTML*, *XML*, *UML*, *MOF*, etc.
- al ser independiente de herramientas, repositorios o aplicaciones, evita que cualquier fabricante pueda crear un formato diferente. Garantizando que distintos productos sean compatibles debido a que permite que compartan información.
- los usuarios pueden decidir que tecnologías utilizar, confiando que su información podrá representarse y transferirse, eliminando la dependencia de una única plataforma.

[32] Lenguaje de ruta *XML* (*XML Path Language*). Es un lenguaje que permite construir expresiones que recorren y procesan un documento *XML*.

[33] Lenguaje de consulta diseñado para colecciones de datos *XML*.

[34] Fondo para Meta-Objeto.

- facilita la forma de organizar la información y meta-información. Siendo más sencillo de usar y comprender que las tradicionales tecnologías de meta-información (relacional repositorio de objetos).

Telerik: es una librería que permite a los desarrolladores crear interfaces de alto rendimiento y visualmente impresionantes. *WinForms Telerik*, inspirada en *WPF* posee componentes que ofrecen una arquitectura única, como *flash* de tipo mezcla de animación, rotación, escalabilidad y transparencias. Además, presenta un diseñador de forma visual y proporcional, que a la vez es interactivo en tiempo de diseño

El conjunto de componentes de *WinForm Telerik* está diseñado para satisfacer las últimas exigencias de negocios a gran escala, rico en características y aplicaciones táctiles. Aprovechando la experiencia de largos años en el desarrollo de la interfaz, *Telerik* proporciona un conjunto completo de componentes de escritorio y herramientas para ayudar al desarrollador a centrarse en la lógica de la base de sus proyectos, y construir proyectos empresariales de alto nivel, con un rendimiento excepcional y con una interfaz de usuario moderna (31).

***UML*:** El lenguaje unificado de modelado (UML) es una notación gráfica para describir programas orientados a objetos. No es un método de diseño, sino una notación que puede ayudar con el diseño de un programa o con su documentación, una vez que el programa esté completo (32).

UML puede ser utilizado para modelar distintos tipos de sistemas: sistemas de software, hardware, y organizaciones del mundo real. Se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Las extensiones de UML para el modelado de negocio aportan elementos muy importantes ya que proporcionan algunas otras vistas de la arquitectura de negocio. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir. Además UML tiene varias ventajas, entre ellas: notación estándar para el análisis y diseño de negocios y sistemas informáticos; es independiente de la arquitectura o el lenguaje que se vayan a seleccionar (o que hayan sido seleccionados) para la realización; incorpora las mejores prácticas a nivel internacional; cuenta con un amplio apoyo entre empresas e instituciones (33).

Altova UModel: esta herramienta *Case*[35] combina una intuitiva interfaz visual con funciones de usabilidad superiores, que favorecen a los usuarios con las más completas ventajas en el modelado de *software.* Las características de *Altova UModel* 2009 para el desarrollo de *software* basado en las capacidades de modelado avanzado son (34):

[35] Ingeniería de *Software* Asistida por Computadora (*Computer Assisted Software Engineer*) destinada a aumentar la productividad en el desarrollo de *software* reduciendo el coste de las mismas en términos de tiempo y dinero.

- soporte para los 14 tipos de diagramas *UML*.
- generación de código fuente en lenguajes *Java*, *C#*, y *VB*.*Net*.
- creación de diagramas de secuencia desde el código fuente de la ingeniería inversa.
- generación de documentación personalizable de proyecto.
- integración con sistemas de control de versiones.
- estrecha integración con *Visual Studio* y *Eclipse.*

1.4 Conclusiones parciales.

El análisis de los principales conceptos asociados al modelado de negocio a través del estándar *BPMN* 1.0 y su posible relación con la tecnología *Windows Workflow Foundation* 3.5 y el estudio de los principales sistemas que, a pesar de garantizar el modelado de procesos de negocio y a la vez gestionar su ejecución, demuestran que no existe hasta el momento una herramienta que garantice una integración directa de *BPMN* 1.0 con la tecnología *WWF* 3.5. Es por ello que en el Sistema Único de Identificación Nacional se llega a la necesidad de la implementación de una herramienta que simplifique y agilice la transformación de los procesos de negocio a los flujos de trabajo.

Capítulo 2 Propuesta de solución.

Introducción

En el presente capítulo se muestra como queda conformado el modelo de dominio y se describe cada uno de los conceptos significativos en el dominio del problema, para un mejor entendimiento de la propuesta de solución. Se realiza el análisis de los requisitos funcionales y no funcionales con los que cuenta la herramienta.

2.1 Descripción del negocio.

El proceso actual es iniciado por el analista a través de la creación del diagrama de proceso de negocio bajo el estándar *BPMN* 1.0 en la herramienta de modelado *Altova Umodel.* Una vez creados los modelos de proceso de negocio, pasan a formar parte de la documentación del proyecto y son utilizados por los desarrolladores para comprender mejor el proceso de negocio y llevar a cabo la implementación de los procesos transformándolos en *workflows* sobre la tecnología *WWF* 3.5. Esta transformación que se lleva a cabo no presenta ninguna automatización que simplifique el trabajo del desarrollador, dependiendo únicamente de la capacidad que posea el mismo para interpretar el diagrama de proceso de negocio bajo el estándar *BPMN* 1.0.

2.2 Descripción de la propuesta de solución.

Se propone el desarrollo de una herramienta que permita la transformación de diagramas de procesos de negocio bajo el estándar *BPMN* 1.0 a *WWF* 3.5. Se tendrán como entradas los ficheros con extensión *XMI* generados por la herramienta *Altova UModel,* los cuales presentan una estructura única para almacenar la información que es gestionada en los modelos de procesos de dominios. De estos ficheros se obtienen las actividades y relaciones que conforman el proceso de negocio, para luego ser transformadas a un flujo de trabajo compatible con la tecnología *WWF* 3.5*,* a través de la aplicación de una serie de patrones de control de flujo que acotarán el amplio espectro de modelación en *BPMN* y que reducirá el tiempo de implementación del desarrollador quien solo tiene que implementar el resto del *workflow* a partir del esqueleto exportado por la herramienta*.* Siguiendo el modo **sólo código** para la creación de un *workflow* en *WWF*, se obtendrá como salida el código fuente de los ficheros ***designer.cs*** y ***.cs*** que conforman el *workflow.*

En el fichero de tipo ***.designer.cs*** se definen todas las actividades que intervienen en la creación del *workflow* bajo la tecnología *WWF*. Presenta una estructura en la que se evidencian bloques de código para

declarar e inicializar las actividades así como para definir sus propiedades, como se puede evidenciar en la siguiente figura.

```
partial class Template.name
{
    #region Designer generated code

    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    [System.Diagnostics.DebuggerNonUserCode]
    [System.CodeDom.Compiler.GeneratedCode("", "")]
    private void InitializeComponent()
    {
        this.CanModifyActivities = true;
        //Initializations
        //Properties
        //
        // Template.name
        //
        //Activities
        this.Name = "Template.name";
        this.CanModifyActivities = false;
    }
    #endregion

    //Declarations
}
```

Figura 2 Estructura del fichero *.designer.cs.*

El fichero con extensión ***.cs*** permite al programador agregar código al diseño del *workflow* enriqueciendo la definición de las tareas automatizadas. Ver figura 3.

```
public sealed partial class Template.name : SequentialWorkflowActivity
{
    public Template.name()
    {
        InitializeComponent();
    }
}
```

Figura 3 Estructura del fichero *.cs*.

Para realizar con éxito esta transformación, el usuario debe conocer que la herramienta posee un conjunto de pautas para su utilización, debido al extenso campo de posibles representaciones de todos los patrones de diseño, que pueden ser simulados bajo el estándar *BPMN* 1.0 y la falta de integración total con la tecnología *WWF* 3.5. Una de estas pautas expone que la herramienta no admite como entrada diagramas de procesos de negocio que contengan modelos de coreografía ni de colaboración, por lo que la representación de múltiples eventos de inicio tampoco será admitida. Para más información ver **Anexo 1**.

Su principal objetivo es crear un puente estandarizado para el vacío existente entre el diseño de procesos de negocio y su implementación, reduciendo el tiempo de comprensión, de transformación y la eliminación de las incompatibilidades que puedan existir a la hora de interpretar estos procesos. La herramienta persigue una arquitectura en capas bien diseñadas para reducir al mínimo el acoplamiento y aumentar la reutilización entre las mismas, siguiendo el principio de separación de responsabilidades.

2.2.1 Patrones de control de flujo.

Para la interpretación de los modelos de procesos de negocio se realizó un estudio de los 20 patrones de control de flujo propuesto por Van der Aalst en su libro *Workflow Pattern*. Se analizaron la representación de los patrones tanto en *BPMN* como en *WWF*. En la siguiente tabla se muestra la correspondencia de cada patrón en ambos modelos, el símbolo “+” significa que se pude representar y el símbolo “-” significa que no pude ser representado.

Patrones de control de flujo	**BPMN**	***WWF***
Secuencia	+	+
Separación paralela	+	+
Sincronización	+	+
Elección exclusiva	+	+
Fusión simple	+	+
Elección múltiple	+	+
Fusión sincronizada estructurada	+	+
Multifusión	+	-

Discriminador estructurado	-	-
Ciclos arbitrarios	+	+
Terminación implícita	+	+
Múltiples instancias sin sincronización	+	+
Múltiples instancias con conocimiento en tiempo de diseño	+	+
Múltiples instancias con conocimiento en tiempo de ejecución	+	+
Múltiples instancias sin conocimiento previo	-	-
Elección aplazada	+	+
Rutina paralela entrelazada	-	+
Hito	-	-
Cancelación de actividad	+	+
Cancelación de instancia	+	+

Tabla 1 Evaluación de los patrones de control de flujo en *BPMN* y *WWF*

De los patrones que se pueden modelar en *BPMN* y *WWF* la herramienta contempla siete de estos patrones. Dentro de estos patrones se puede encontrar (12):

- **secuencia:** describe el proceso más simple dentro de un flujo de control, en el que una actividad es iniciada siempre y cuando se haya completado su antecesora en el mismo proceso.
- **sincronización:** es asociado a procesos que tienen en ejecución múltiples actividades ejecutadas en paralelo. El elemento sincronizador espera a que todas las actividades terminen su ejecución, junta todas las secuencias de actividades en un solo flujo y continúa el proceso con la o las siguientes actividades.
- **separación paralela:** describe un proceso en el que desde una actividad completada, se da lugar a la ejecución de dos o más actividades en paralelo.
- **elección exclusiva:** permite a los *workflows* realizar decisiones a la hora de ejecutar actividades en función de la validez de las ramas escogidas.

- **elección Múltiple:** propone una mejora del patrón de elección exclusiva, debido a que permite la selección y ejecución de tantas ramas sean válidas en el proceso.
- **múltiples instancias sin sincronización:** define la ejecución de una misma actividad en instancias diferentes en un número conocido en tiempo de diseño del *workflow.* Sólo cuando todas las actividades se completan, las siguientes actividades se iniciarán.
- **ciclos arbitrarios:** permite la ejecución de un conjunto de actividades repetidamente.

Una vez desplegada la aplicación, esta posibilita la incorporación de nuevos patrones de control de flujo por parte de usuarios avanzados, lo que responde ante la necesidad de que la misma sea flexible y escalable. Para ello, sólo tendrán que implementar un ensamblado en el que sus clases representen los nuevos patrones a incorporar en la herramienta. Para lograr este objetivo, el conjunto de las clases definidas con los nuevos patrones deberá heredar de la clase *BasePattern* y por consiguiente sobre-escribir los métodos que esta presenta. Como paso final, se deben agregar tantas configuraciones en el ***app.config*** de la herramienta como nuevos patrones se hayan implementado. Estas configuraciones presentan un formato en el cual se deben especificar el nombre del nuevo patrón y del ensamblado en el que se encuentra embebido así como su espacio de nombre.

2.3. Modelo de dominio.

El modelo de dominio representa las entidades importantes (los objetos junto con sus atributos y métodos) del dominio de la aplicación utilizando un modelo de objetos típico de los métodos de análisis orientado a objetos (35).

La representación del modelo de dominio pretende ayudar en la comprensión de los principales conceptos que utiliza la solución para el intercambio de datos entre las distintas capas de la arquitectura.

Como no se identificaron procesos de negocio involucrados en el desarrollo de la propuesta de solución, se realizó un modelo de dominio con el objetivo de presentar una perspectiva general de las entidades o conceptos que intervienen en la creación de este *software*.

Para su elaboración, fue necesario el estudio de las funcionalidades comunes de la herramienta. Con tales fines, se identificaron clases conceptuales y las relaciones que se establecen entre ellas.

A continuación se describen todos los conceptos y sus relaciones presentes en el modelo de dominio:

- **Archivo_XMI**: archivo con extensión *XMI* generado por la herramienta *Altova UModel* que contiene los diagramas de procesos de negocio.
- **DPN**: diagrama de proceso de negocio.

- **Dependencia**: representa las conexiones que vinculan a los objetos que existen en el diagrama de proceso de negocio.
- **Puerta**: representa los elementos de modelado que contemplan cómo el flujo el proceso diverge o converge.
- **ActividadBPMN**: representa las acciones atómicas que son realizadas en el proceso de negocio.
- **Evento**: representa las acciones que afectan el flujo del proceso y por lo general suelen tener un disparador o un resultado.
- **Objeto_flujo**: objetos de flujo que se encuentran contenidos en el diagrama de proceso de negocio los cuales representan las actividades y eventos del diagrama de proceso de negocio.
- **Grafo_actividades**: contempla los elementos fundamentales del archivo *XMI* representados en una estructura de grafo.
- **Identificación_patrones**: concepto que identifica los patrones de control de flujo en el diagrama de proceso de negocio.
- **Árbol_actividades**: representa la secuencia de actividades reflejadas como nodos dentro de un árbol.
- **Clase_controladora**: contiene la lógica de la transformación de los diagramas de proceso de negocio.
- **XML_transformación**: abarca la transformación de cada actividad del diagrama *BPMN* y sus propiedades con sus equivalentes en *WWF* 3.5.
- **Elemento_mapeo**: hace corresponder una actividad dentro del árbol de actividades y sus propiedades con una actividad y sus propiedades en *WWF* 3.5.
- ***Workflow***: representa el *workflow o*btenido por el proceso de trasformación.
- **Actividad**: representa cada una de las actividades comprendidas dentro del *workflow.*

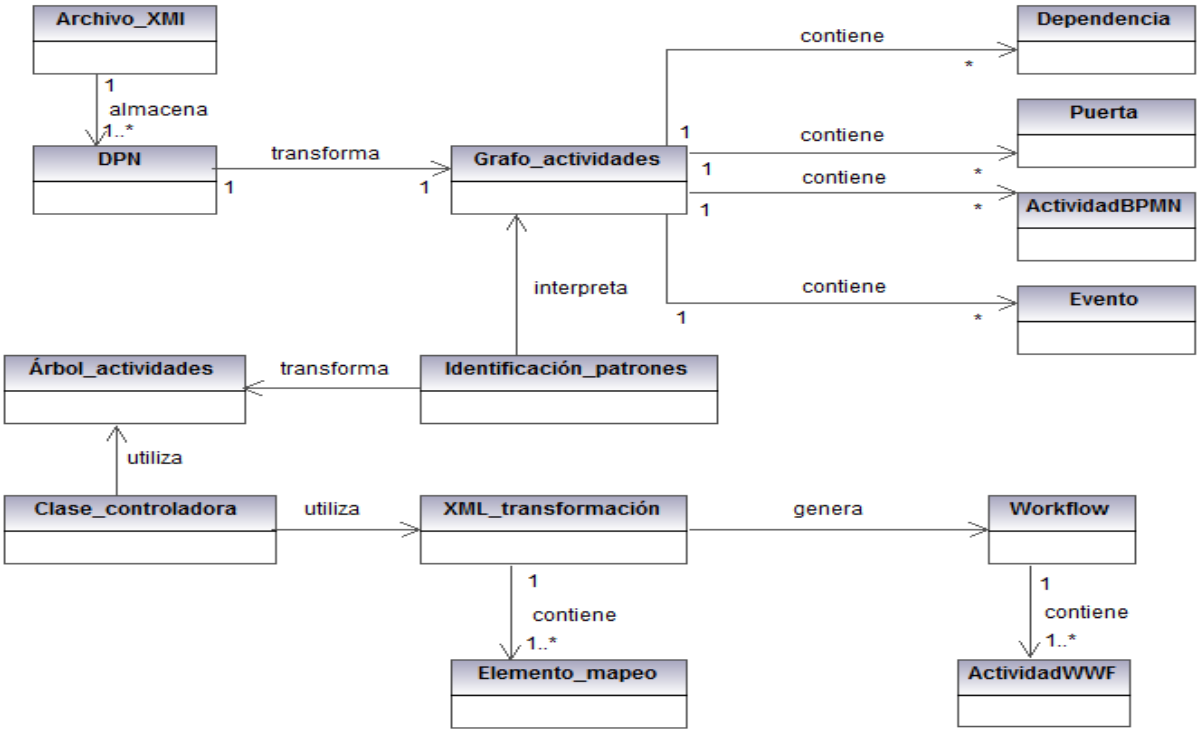


Figura 4 Modelo de Dominio

La herramienta para la transformación de modelos de procesos de negocio bajo el estándar *BPMN* 1.0 generados por la herramienta *Altova UModel* a la tecnología *Windows Workflow Foundation* 3.5 y *Bison Framework* obtiene la lógica de los diagramas de proceso de negocio (DPN) que se encuentra almacenada en un fichero con extensión *XMI* generado por la herramienta *Altova UModel*. De estos ficheros, se filtran los principales componentes (dependencias, puertas, eventos y actividades) con el objetivo de traducirlos en una estructura de dato conexa (grafo de actividades), en la que se identifican una serie de patrones de control de flujo (Identificación_patrones), los cuales son el portal para la posterior conversión en un árbol de actividades. Para llevar esta estructura de datos a *workflow*, la aplicación se apoya en una clase (Clase_controladora) que contiene la lógica de transformación de los diagramas de proceso de negocio la cual, utilizando un *XML* de transformación que contiene los elementos de mapeo, hace corresponder los tipos de actividades en *BPMN* con sus equivalentes en la tecnología *WWF* 3.5.

2.4 Especificación de requisitos funcionales de la herramienta.

La ingeniería de requisitos proporciona el mecanismo apropiado para entender lo que el cliente quiere, analizar las necesidades, valuar la factibilidad, negociar una solución razonable, especificar la solución sin

ambigüedades, validar la especificación, y administrar los requisitos conforme estos en su sistema operacional (35).

2.4.1 Descripción de los roles.

Roles	Objetivo
Desarrollador	Encargado de generar un *workflow* 3.5 a partir de un fichero *XMI*, de configurar la transformación, entre otras funcionalidades que brinda la herramienta.

Tabla 2 Descripción de roles.

2.4.2 Catálogo de requisitos.

A continuación se exponen los requisitos funcionales que conforman este trabajo de diploma:

RF1. Cargar fichero con la extensión *XMI*.

RF2. Gestionar elemento de mapeo.

RF 2.1 Crear elemento de mapeo

RF 2.2 Modificar elemento de mapeo

RF 2.3 Eliminar elemento de mapeo

RF3. Generar *Workflow* 3.5.

RF 3.1 Seleccionar configuración de transformación.

RF 4 Salvar nueva configuración de trasformación.

RF 5 Gestionar propiedad de elemento de *BPMN*.

RF5.1 Adicionar propiedad

RF 5.2 Modificar propiedad

RF 5.3 Eliminar propiedad

RF 6 Agregar componente de *Bison Framework*.

2.4.3 Descripción de requisitos funcionales del sistema.

RF1.Cargar fichero con la extensión *XMI*.

Propósito	Interpretar de un fichero *XMI* generado por la herramienta de modelado *Altova UModel*, los objetos del flujo y dependencias que conforman el proceso modelado.
Roles	Desarrollador.
Precondiciones	1. Se debe haber exportado del *Altova UModel* un fichero

	XMI.	
Conceptos tratados	**Concepto**	**Atributos**
	Node	ID *name* *listClienDependency*
	Dependency	*ID* *dependecyType* *suplier* *client*
	NodeTree	*ID* *name* *properties*
Descripción	1. Se escoge la opción Cargar *XMI*. 2. Se muestra una interfaz de búsqueda 2.1 Seleccionar un fichero *XMI*. 2.2 Seleccionar la opción abrir. 3. Se muestra un árbol de representación con las actividades recogidas del fichero.	
Validaciones	1. Validar que el diagrama de proceso de negocio contenido en el fichero cumpla con las pautas establecidas por la herramienta.	
Postcondiciones	1. Se recogen del fichero los objetos del flujo y relaciones que componen el diagrama.	
Prototipo		

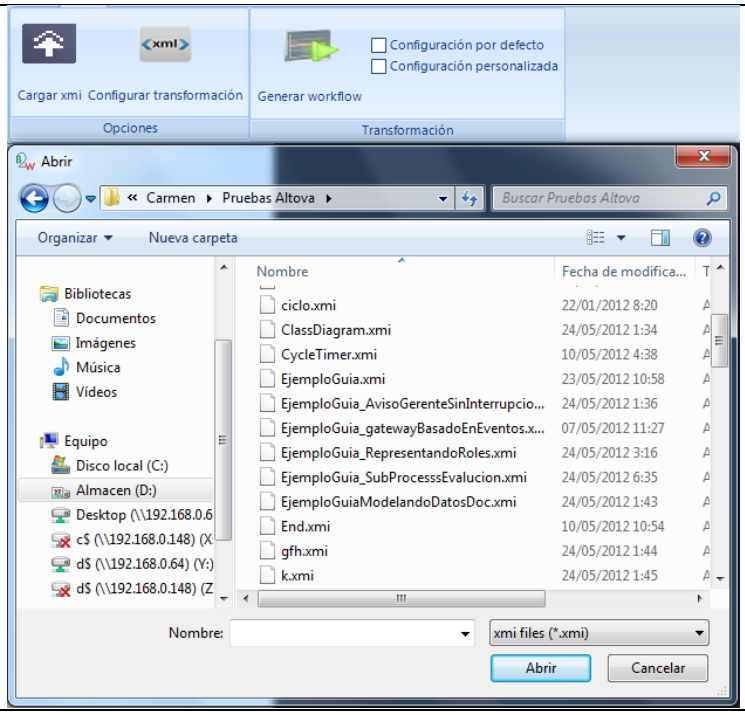
	

Tabla 3 RF1.Cargar fichero con la extensión *XMI*.

El resto de la descripción de requisitos se encuentra en el **Anexo 2** Descripción de requisitos funcionales**.**

2.5 Especificación de requisitos no funcionales.

Las propiedades o cualidades que un *software* debe cumplir son definidas como los requerimientos no funcionales, los cuales deben ser conceptualizados a través de la identificación de una serie de características que representan al *software* no por sus funcionalidades, sino por su usabilidad, confiabilidad, soporte, requisitos de *software* y restricciones de diseño. Con el cumplimiento de estos requisitos se garantiza que el *software* sea de fácil uso y aceptado por los clientes.

A continuación se presentan detalladamente los requisitos no funcionales con los que cuenta la herramienta.

Usabilidad

RnF1. La herramienta podrá ser utilizada por cualquier usuario con las siguientes características:

- conocimientos básicos relativos al uso de una computadora.
- conocimientos básicos del sistema operativo *Windows*.
- conocimientos sólidos relativos a los procesos de negocio que se modelan bajo el estándar *BPMN*.

- Conocimientos sólidos relativos a la definición de flujos de trabajo con la tecnología *Windows Workflow Foundation* 3.5 y *Bison Framework*.

RnF2. La herramienta será distribuida en idioma español.

RnF3. La herramienta poseerá estructura y diseño homogéneos en todas sus pantallas con el objetivo de facilitar el entendimiento de sus funcionalidades.

RnF4. Menú horizontal y desplegable que permita el acceso rápido a las funcionalidades de la herramienta.

Confiabilidad

RnF5. Se garantizará la consistencia de los datos, se realizarán comprobaciones y validaciones en todos los casos posibles.

RnF6. La información manejada por la herramienta será guardada una vez terminada de procesar.

Requisitos de *software*

RnF7. El sistema podrá ser utilizado bajo el sistema operativo *Windows XP* o superior.

RnF8. El sistema requiere de *.Net Framework v*4.0.

Diseño de interfaz

RnF9. Las ventanas del sistema contendrán claro y bien estructurados los datos, y al mismo tiempo permitirán la interpretación correcta e inequívoca de la información.

RnF10. Utilizar una norma que permita la distinción visual entre los elementos de la ventana a través del uso de colores, así como otras técnicas, como tamaños de fuentes.

RnF11. Dirigir la corrección de errores de introducción de datos a una forma clara y fácil de realizar, la entrada de datos incorrecta será detectada claramente por la herramienta.

RnF12. Diseñar el funcionamiento da la herramienta de modo que sea intuitivo, y requiera de información mínima.

2.6 Conclusiones parciales.

Mediante la realización de un estudio completo del negocio relacionado con la transformación de los modelos de procesos de negocio a *workflows* de la tecnología *WWF* 3.5 se demostró la necesidad de desarrollar una herramienta que posibilite la automatización de dicha transformación por la importancia que tiene esta dentro de la implementación de los procesos de negocio. Con la representación del modelo de dominio de la herramienta para la transformación de *BPMN* 1.0 a la tecnología *Windows Workflow Foundation* 3.5 se pudo delinear una propuesta solución donde se reflejan los principales conceptos tratados en el desarrollo de la herramienta, obteniéndose además la descripción de los requisitos

funcionales y no funcionales para darle cumplimiento a los principales objetivos derivados de la presente propuesta solución.

Capítulo 3: Análisis y diseño de la herramienta

Introducción

El desarrollo de soluciones informáticas es un proceso complicado que engloba diversos factores dentro de los cuales, los más importantes son los de comprensión y abstracción. Es por ello que una correcta modelación de cómo estará enfocada la construcción de los mismos, no solo indicará el cómo se construyen las soluciones, sino también cómo se elaboran.

En el presente capítulo se realiza una descripción de la arquitectura de la aplicación para lo cual se detallan sus principales componentes. Se describen además, los principales patrones de diseño de *software* utilizados y los diferentes patrones de *workflow* que son empleados para brindar una solución a la aplicación, así como el diagrama de clases para visualizar el diseño conceptual de la información del sistema y las relaciones entre ellas.

3.1 Arquitectura de la solución.

Una arquitectura que esté bien definida, ayuda a la planificación de recursos y la asignación de tareas, garantizando que la aplicación sea flexible ante la ejecución de futuros cambios.

La herramienta presenta una arquitectura n-capas con el objetivo de que sea escalable, fácilmente analizable y que en un futuro, presente un nivel aceptable de interacción con otras herramientas informáticas. En la vista más abstracta de la aplicación se pueden definir cuatro capas diseñadas para reducir al máximo el acoplamiento y aumentar la reutilización entre cada una de ellas. Estas son la capa de presentación, la capa de negocio, la capa de acceso a datos y la capa de datos, como se muestra en la siguiente ilustración:

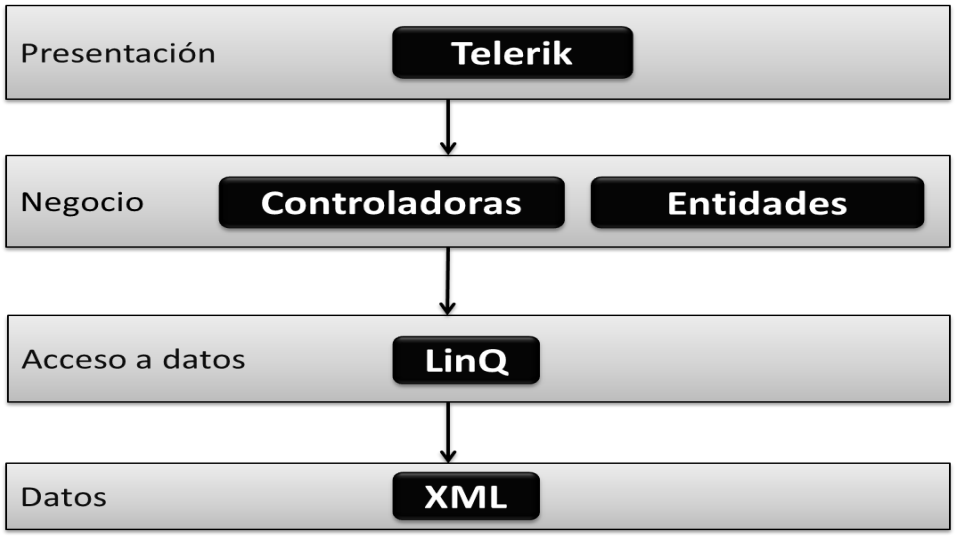


Figura 5 Vista lógica de la arquitectura

La capa de **presentación** está conformada por las interfaces que interactúan con el usuario. Para lograr que los componentes visuales sean de alto rendimiento y agradables a la vista se utiliza la librería ***Telerik***. Esta librería le brinda a la aplicación un estilo visual moderno y reduce el tiempo de desarrollo de las interfaces.

Esta capa presenta una interacción directa con la capa de negocio y permite la visibilidad de los datos de entrada correspondiente a los diagramas de procesos de negocio representadas bajo el estándar *BPMN* 1.0 y a los datos de salida equivalente al *workflow* que se obtiene como resultado del proceso de transformación.

La realización de las funcionalidades con las que cuenta la aplicación tiene lugar en la capa de **negocio**, la cual mantiene separadas las acciones atómicas del negocio adquiriendo la aplicación una gran flexibilidad a la hora de modificar cualquiera de sus funcionalidades. Esta capa se comunica y gestiona la información que se obtiene de la capa de acceso a datos y satisface los requerimientos funcionales del usuario.

Los principales elementos que vienen asociados a esta capa comprenden las entidades y las clases controladoras. Las entidades almacenan la información que es gestionada en el sistema, y las clases controladoras representan la estructura jerárquica con la que cuenta el sistema para lograr su correcto funcionamiento.

La capa de **acceso a datos** se encuentra vinculada con la capa de negocio y la de datos. En esta capa se obtiene la información relacionada con la capa de datos. Esto se logra a través del uso del lenguaje

integrado de consulta ***Linq***, que permite la realización de consultas a las fuentes de datos sin importar de dónde provengan.

Por último, la capa de **datos**, contiene la información de los diagramas de proceso de negocio la cual es almacenada en formato ***XML***.

3.2 Patrones de diseño.

Un patrón de diseño es una solución estándar para un problema común con lo que se puede incrementar o disminuir la capacidad de una implementación, con el objetivo de lograr una finalidad determinada, flexibilizando el código para satisfacer ciertos criterios.

"*Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan"* (36).

Dentro de los principales patrones de diseño existentes se utilizaron los siguientes en la confección de la aplicación:

GRASP[36]: el uso de este conjunto de patrones está totalmente ligado a cada componente desarrollado en el sistema, donde cada uno de ellos posee sólo las funcionalidades acorde a las particularidades que lo caracterizan. Se aplican durante la construcción de diagramas de interacción y colaboración al asignar responsabilidades a los objetos y al diseñar la colaboración entre ellos (37). Dentro de los principales patrones *GRASP* que se evidencian en el desarrollo de la aplicación se pueden identificar:

- alta cohesión: una alta cohesión hace que las clases muy relacionadas no hagan un trabajo enorme. Aumenta la capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico. Ejemplo:

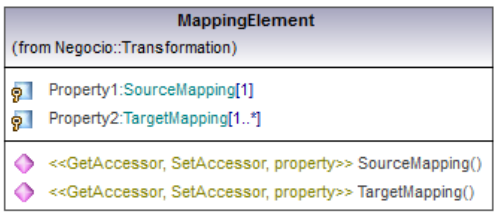


Figura 6 Clase *MappingElement*.

[36] En diseño orientado a objetos representa los patrones generales de *software* para asignación de responsabilidades ("*General Responsibility Assignment Software Patterns*").

- bajo acoplamiento: soporta el diseño de clases más independientes, que reducen el impacto de los cambios y también clases más reutilizables que aumentan la oportunidad de mayor productividad. Ejemplo:

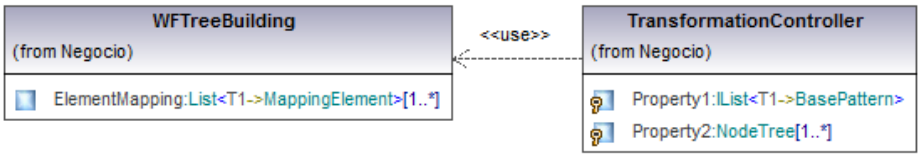


Figura 7 Fragmento del diagrama de clases.

- creador: una instancia de un objeto la tiene que crear el objeto que tiene la información para ello.

GOF: los patrones de la banda de los cuatro (*Gang of four* como sus siglas en inglés lo indican) están considerados como el fundamente de todos los patrones. Ellos los clasifican en tres grupos: creacional, estructural y de comportamiento (39). En la propuesta solución se evidencian dos de estos patrones. Ellos son:

intérprete*:* patrón de diseño perteneciente al grupo de comportamiento de los patrones definidos por la *GOF.* Este patrón propone dado un lenguaje, definir una representación de su gramática junto con un intérprete que utiliza la representación para interpretar frases en el idioma y específicamente en la aplicación fue utilizado a la hora de interpretar los patrones de control de flujo de los diagramas de proceso de negocio como se evidencia a continuación:

```
foreach (BasePattern item in patterns)
{

    var tree = item.FindPattern(nodo, grafoSource, patterns, this);
    if (tree != null)
        arbol.Add_SubArbol(tree);
}
```

Figura 8 Fragmento de código donde se utiliza el patrón intérprete.

compuesto: patrón de diseño perteneciente al grupo de los patrones estructurales definidos por la *GOF*. Este patrón propone preparar objetos en estructuras de árbol para representar jerarquías de parte-todo. Además, permite a los clientes tratar objetos individuales y composiciones de objetos de manera uniforme. Como en la herramienta se trabaja con estructuras de datos de tipo árbol se utiliza este patrón como se evidencia en el siguiente fragmento de código:

```
private static void FillRadWfTree(RadTreeNode treeNode, ArbolG<ActivityWWF> arbol)
{
    if (arbol.EsHoja())
    {
        var rad = new RadTreeNode(arbol.raiz.ActivityName + ":" + arbol.raiz.ActivtyType);
        treeNode.Nodes.Add(rad);
    }
    else
    {
        var rad1 = new RadTreeNode(arbol.raiz.ActivityName + ":" + arbol.raiz.ActivtyType);
        foreach (var subArbol in arbol.subArboles)
            FillRadWfTree(rad1, subArbol);
        treeNode.Nodes.Add(rad1);
    }
}
```

Figura 9 Fragmento de código donde se utiliza el patrón compuesto.

Otros patrones

Encapsulación: propone esconder algunos componentes, permitiendo sólo accesos estilizados al objeto. Se hace uso de este patrón en casi todas las clases que componen al sistema permitiendo que estas solo posean como elementos públicos aquellos que son exclusivamente necesarios. Ejemplo:

```
public string ID { get; set; }
public string name { get; set; }
```

Figura 10 Fragmento de código.

Subclase: propone heredar miembros por defecto de una superclase, seleccionando la implementación correcta a través de resoluciones sobre qué implementación debe ser ejecutada. Se puede encontrar este patrón con más fuerza en las entidades de negocio que por su conceptualización, las funciones y la información que almacenan pueden estar diferenciadas en cierta medida. Ejemplo:

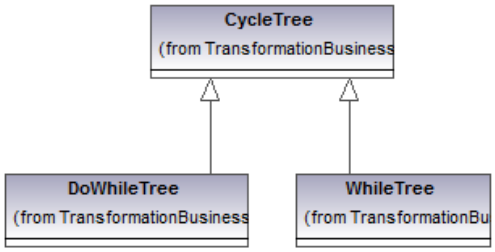


Figura 11 Herencia de clases.

Excepciones: propone introducir estructuras de lenguaje para lanzar e interceptar excepciones. Se identificaron los diferentes tipos de errores a tratar dentro de la herramienta creando clases que permitan identificar cada tipo de error en el momento de ejecución. Ejemplo:

```
public class WrongFormatException:Exception
 {

     public WrongFormatException()
     {
     }
     public WrongFormatException(string message)
         :base(message)
     {

     }
 }
```

Figura 12 Fragmento de código.

3.3 Patrones de *workflow.*

Para la creación de un *workflow* 3.5 se tuvieron en cuenta los diferentes patrones que están presentes en la mayoría de los lenguajes de *workflow*, y sirven para modelar procesos secuenciales, paralelos, o aquellos que incluyan la toma de decisiones. Los patrones de *workflow* pueden ser clasificados en las siguientes categorías:

3.3.1 Patrones de control de flujo.

Secuencia: constituye el pilar fundamental para la construcción de *workflows.* Con el uso de este patrón es posible la ejecución de un conjunto de actividades consecutivas en el mismo orden en el que fueron diseñadas. Dos actividades forman parte de una secuencia si, existe alguna línea de control de flujo que las una sin la presencia de algún tipo de condición asociada.

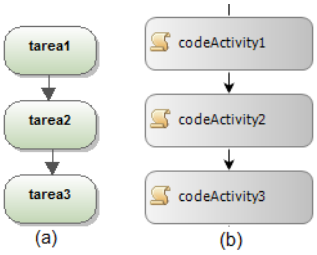


Figura 13 Patrón *workflow* de secuencia en (a) *BPMN* y (b) *WWF.*

Selección: se recorre un camino de los tantos posibles siempre y cuando cumpla con una condición previa. El momento en el cual uno de los posibles caminos es elegido es retrasado hasta el último instante debido a que la selección es basada en factores externos como pueden ser mensajes de llegada, disponibilidad de recursos, temporizadores, entre otros. Hasta el preciso momento en el que la decisión es tomada, cualquiera de los caminos mostrados representa un curso válido de futuras acciones.

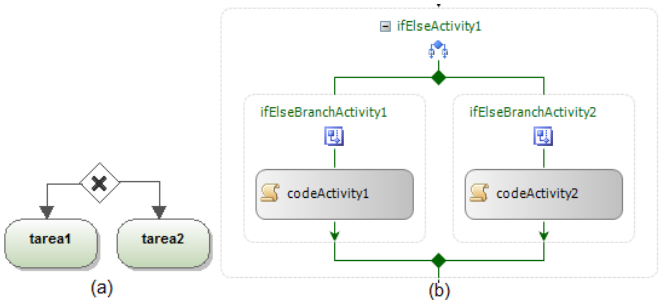


Figura 14 Patrón *workflow* de selección en (a) *BPMN* y (b) *WWF.*

Paralelismo: permite que un solo hilo de ejecución pueda ser dividido en dos o más ramas las cuales pueden ejecutar actividades concurrentemente. Estas ramas pueden o no ser conectadas en la continuación de la construcción del *workflow* deseado.

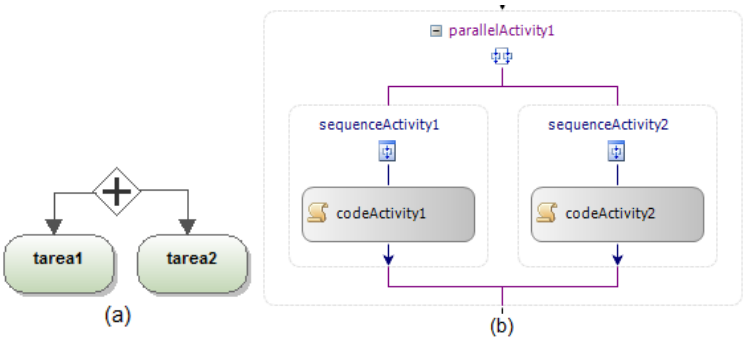


Figura 15 Patrón *workflow* de paralelismo en (a) *BPMN* y (b) *WWF.*

3.3.2 Patrones Estructurales.

Ciclos arbitrarios: garantiza la repetición de un modelo de procesos sin la necesidad de incluir operadores específicos de iteración.

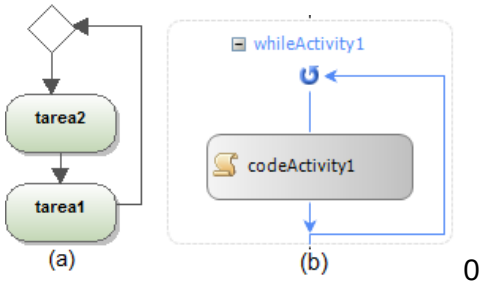


Figura 16 Patrón *workflow* de ciclo en (a) *BPMN* y (b) *WWF*.

3.4 Especificación de clases.

Con el objetivo de brindar una solución a la aplicación a desarrollar se realizó la definición de las clases que son utilizadas para darle cumplimiento a los requisitos funcionales con los que cuenta la aplicación; estas son: las clases entidades, las gestoras y la conectora. En el **Anexo 3** Descripción de las clases del diseño.se encuentran el resto de las descripciones de clase.

Clases entidades: las clases entidades constituyen una representación de los datos que es a menudo persistente a través de los cuales se modela información que posee larga vida y que brindan soporte a las funcionalidades con las que cuenta la aplicación.

Nombre	NodeTree	
Descripción	Entidad que representa un nodo dentro del árbol de actividades	
Nombre del atributo	**Tipo**	**Descripción**
ID	*string*	Identificador del nodo.
name	*string*	Nombre del nodo.
properties	*List<Property>*	Contiene las propiedades pertenecientes al nodo.

Tabla 4 Descripción de la clase *NodeTree*

Clase conectora: maneja directamente el acceso a los ficheros *XMI* generados por *Altova*.

Nombre	***Mapping***
Descripción	Clase que permite la extracción de datos del fichero *XMI.*
Métodos	**Descripción**

GetPackageActivities	Método que permite la captura de las actividades de *BPMN* dentro del fichero.
GetDependency	Método que recoge las relaciones existentes entre las actividades de *BPMN*.
GetGraph	Método que devuelve un grafo equivalente al diagrama compuesto por las actividades y sus relaciones.
GetProperties	Método que captura las propiedades de cada actividad en *BPMN*.

Tabla 5 Descripción de la clase *Mapping*

Clases gestoras: utilizan a las clases conectoras con la finalidad de conformar las clases del negocio lo cual facilita el mantenimiento de las aplicaciones a través de la disminución del esfuerzo de actualización en caso de ocurrir cambios en la fuente de datos. Además, permiten la realización de las pruebas a la aplicación mediante la implementación de conectores de pruebas que no accedan a datos reales.

Nombre	***TransformationController***
Descripción	Clase que garantiza la transformación a través del manejo de la clase conectora y las clases entidades.
Métodos	**Descripción**
FillBPMNTree	Método que a partir de la ruta del fichero *XMI*, crea un árbol con la actividades de *BPMN* identificadas.
AplyPattern	Método que identifica patrones de flujo en el grafo de actividades de *BPMN*.
FillWFTree	Método que crea un árbol con las actividades de *WWF* 3.5.

Tabla 6 Descripción de la clase *TransformationController.*

3.4.1 Diagrama de clases del diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones existentes entre ellos.

En la **Figura 17** se muestra un fragmento del diagrama de clases de la aplicación (Ver versión completa del diagrama de clases en el **Anexo** 4 Diagrama de clases**4**.

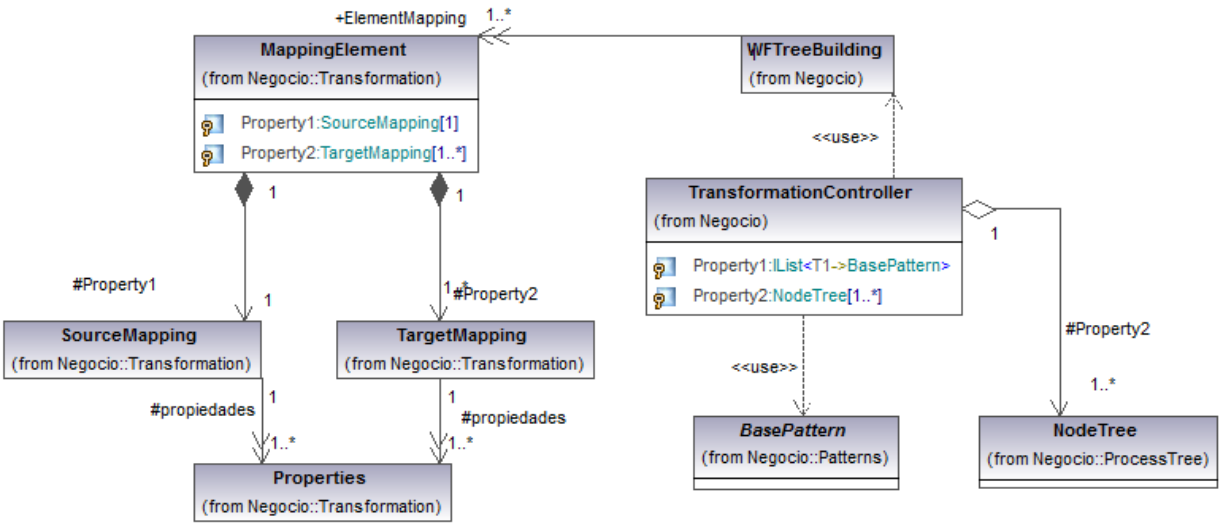


Figura 17 Diagrama de clases del diseño.

3.5 Conclusiones parciales.

Con el diseño de la arquitectura de la aplicación se logrará que la herramienta gane en flexibilidad y escalabilidad a través de una reducción considerable del acoplamiento entre las capas que la componen.

La aplicación de los patrones de diseño en el desarrollo de la aplicación ayuda en gran medida en la reducción del tiempo de desarrollo, así como a brindarle a la implementación elegancia, transparencia y simplicidad.

.

Capítulo 4: Implementación y pruebas

Introducción

En el presente capítulo se muestra cómo ha sido implementada la aplicación haciendo uso de los estándares de codificación. Se modelan las partes físicas y la estructura del sistema a través de los diagramas de despliegue y de componentes definidos. Para validar que los requisitos funcionales implementados respondan a las necesidades establecidas por el Sistema Único de Identificación Nacional, se realizan las pruebas unitarias y de sistema. Dichas pruebas garantizan la corrección de errores en la codificación e implementación de la aplicación.

4.1 Estándares de codificación.

La utilización de estándares de codificación y la aplicación de buenas prácticas de programación aseguran la legibilidad del código y proveen una guía para el mantenimiento y actualización del sistema con código claro y bien documentado, además de contribuir con la calidad del *software*.

Con el objetivo de facilitar el mantenimiento del *software* se emplearon los siguientes estándares de codificación:

Pascal: la primera letra en el identificador y la primera letra de cada subsiguiente palabra concatenada se capitalizan. Este estándar se evidencia en los nombres de los métodos y de las clases. Por ejemplo: *AplyPattern*.

Camello: la primera letra en el identificador está en minúscula y la primera letra de cada subsiguiente palabra concatenada es mayúscula. Este estándar se evidencia en los nombres de los atributos. Por ejemplo: *taskType*

Mayúscula: todas las letras en el identificador se capitalizan. Esta convención se utiliza sólo para los identificadores que constan de dos o menos letras. Por ejemplo: *ID*.

Sensibilidad a mayúsculas

- no se deberá utilizar nombres o identificadores que requieran *sensitivity*[37].
- no se deberá crear dos *namespaces* que se diferencien en el uso de las mayúsculas.
- no crear funciones con nombres de parámetros que se diferencien en el uso de la mayúscula.
- no se deberá crear *namespaces* con nombres de clases que se diferencien en el uso de las mayúsculas.

[37] Sensible a las mayúsculas o minúsculas.

- no crear clases con propiedades que se diferencien en el uso de las mayúsculas.
- no crear clases con métodos que se diferencien en el uso de las mayúsculas.

Evitando confusión de nombre y tipo

Distintos lenguajes de programación usan diversos términos para declarar los principales tipos de identificadores. Los diseñadores de librerías de clases deben definir una terminología de lenguaje específica. En la implementación de la aplicación fueron utilizados nombres que describen a sus identificadores en vez de nombres que describen el tipo de identificador.

4.2 Tratamiento de errores.

En un sistema informático se debe velar por la veracidad de los datos, para asegurar la confiabilidad y así producir resultados exactos. Para el diseño de un sistema es imprescindible tener en cuenta no solo la detección de cualquier error, además, se debe realizar un análisis profundo de las diferentes situaciones que se puedan presentar y que constituyen algún tipo de violación o de situación en particular que podría ser motivo de un error dentro del sistema.

Para garantizar una mayor integridad y confiabilidad posible en los datos que utiliza el sistema se adoptan varias estrategias para el tratamiento de errores.

- Se implementaron funcionalidades para validar que la estructura del fichero *XMI* generado por la herramienta *Altova UModel* cumpla con las pautas para los diagramas de procesos de negocio establecidas por la herramienta.
- Los mensajes de error que emitirá el sistema se mostrarán en un lenguaje de fácil comprensión para los usuarios.
- Cada vez que se introduzca información errónea en un formulario o se dejen campos vacíos, se mostrará un mensaje, informando al usuario el error cometido.

4.3 Implementación.

Una vez que se sabe qué funciones debe desempeñar el sistema de información y se ha decidido cómo organizar sus distintos componentes, es el momento de pasar a la etapa de implementación, pero nunca antes. Antes de escribir una sola línea de código es fundamental haber comprendido bien el problema que se pretende resolver y haber aplicado principios básicos de diseño que permitan construir un sistema de información de calidad.

En la implementación se empieza con el resultado del diseño y se implementa el sistema en término de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares

(38). En este flujo de trabajo, se desarrolla la arquitectura y el sistema como un todo, definiendo y aplicando patrones y estándares para la codificación con el fin de alcanzar una óptima organización del código.

Los diagramas de componentes y despliegue conforman el modelo de implementación; el diagrama de componentes describe la organización y dependencia de cada uno de los componentes implementados, representándose la distribución de los mismos en los distintos nodos físicos en los que funcionará la aplicación.

4.3.1 Diagrama de componentes.

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Representan todos los tipos de elementos *software* que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente.

El diagrama de componentes describe la descomposición física del sistema de software (y, eventualmente, de su entorno organizativo) en componentes, a efectos de construcción y funcionamiento.

La descomposición del diagrama de componentes se realiza en términos de componentes y de relaciones entre los mismos (39).

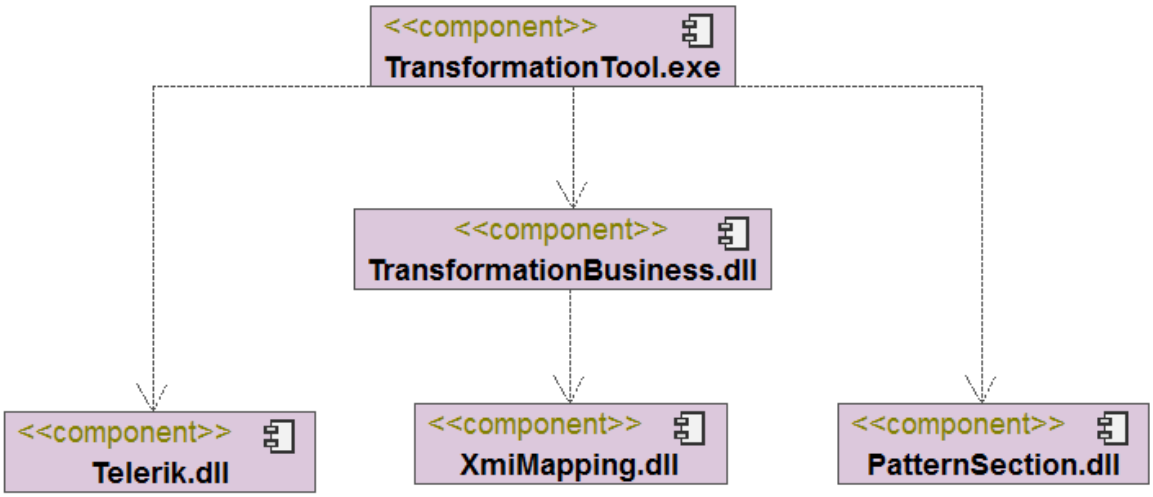


Figura 18 Diagrama de Componentes.

Los distintos componentes de la aplicación fueron implementados siguiendo la arquitectura definida. En el ejecutable *TransformationTool* se encuentran las clases de la capa de presentación que contiene las interfaces de usuarios las cuales son enriquecidas con la librería *Telerik* de manera cómoda. El componente

TransformationBusiness recoge la lógica del negocio, implementando las clases entidades y controladoras que representan la capa de negocio. Este componente se relaciona de manera directa con el *TransformationTool*, garantizando el procesamiento de los datos recogidos. El componente *XmiMapping* implementa las clases encargadas de la captura de información en los ficheros cargados por la aplicación, representando la capa de acceso a dato y la librería *PatternSection* brinda a través de su implementación la posibilidad de la gestión de los patrones de interpretación de los diseños de los diagramas de proceso de negocio.

4.4 Prototipos de la aplicación.

Para el diseño de los diferentes prototipos de interfaz de usuario, se siguieron una serie de pautas que conjuntamente con las facilidades que brinda la librería *Telerik*, permiten que la aplicación gane en facilidad de uso.

4.5.1 Interfaces.

El resto de las imágenes se encuentran en el **Anexo** 5 Interfaz de gestionar elemento de mapeo**5.**

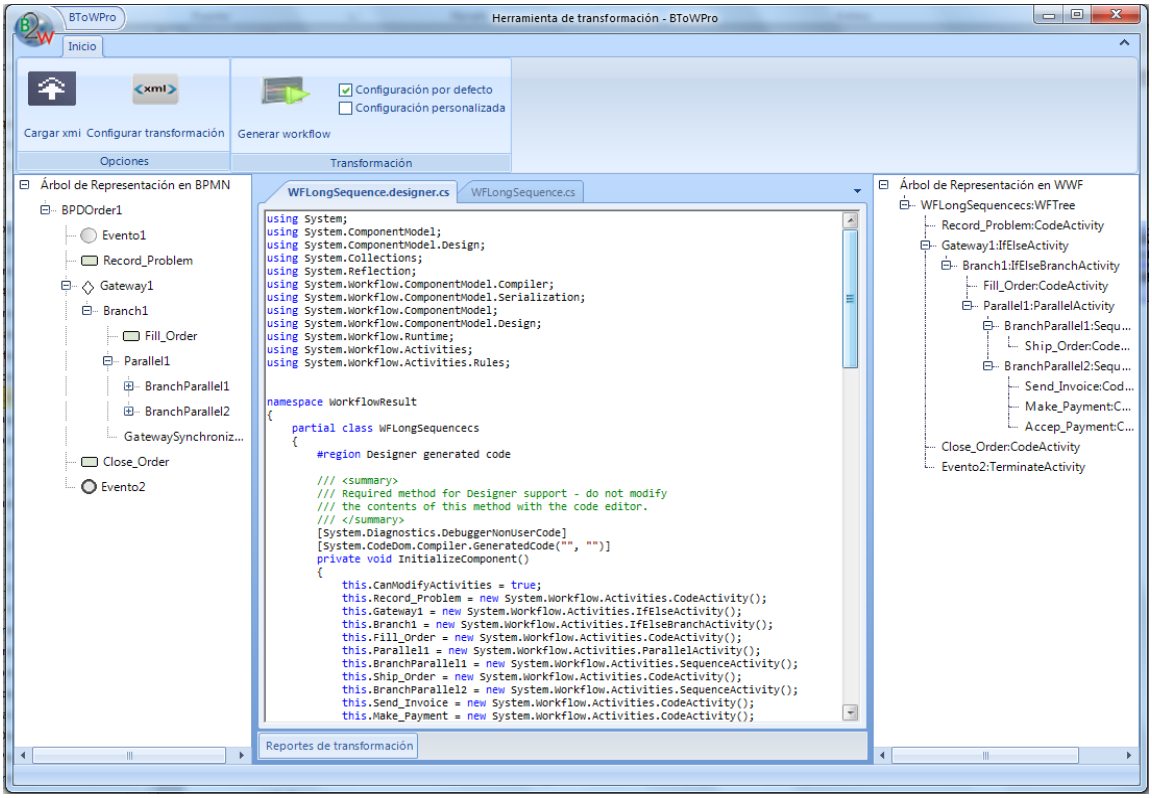


Figura 19 Interfaz principal.

4.4.2 Pautas del diseño.

Página principal: está compuesta por un menú horizontal en la parte superior y el área de trabajo que ocupa el resto del espacio en la ventana.

Tipografía: el tipo de letra que se utilizara es la *Segoe UI* y como tipografía alternativa la *Segoe UI Symbol*.

Subtítulos primarios: se agruparon algunas secciones en dependencia de las funcionalidades como *labels*, botones, tablas, etc. garantizando una estructura cómoda.

Botones en general: los botones en la página principal varían en el tamaño de acuerdo a la descripción y la imagen de fondo, con el texto siempre por debajo de la imagen. Para los botones que contienen simplemente la descripción se definieron dos tamaños de botones, uno mínimo para las palabras pequeñas, otro para las palabras medianas. Para la tipografía de los botones se usará *Segoe UI* normal 8.

Titulares o *labels* de los cuadros de textos: se utilizará la tipografía Segoe UI 8 en su versión normal, color negro.

4.5 Diagrama de despliegue.

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria (40).

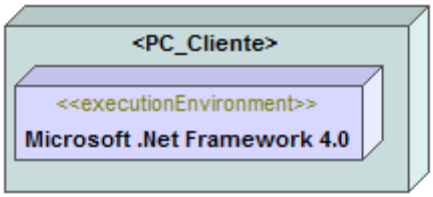


Figura 20 Diagrama de Despliegue.

La herramienta de transformación de diagramas de procesos de negocio representados en *BPMN* 1.0 a *Windows Workflow Foundation* 3.5 estará disponible en cada una de las estaciones de trabajo de los desarrolladores del Sistema Único de Identificación Nacional que presenten como ambiente de ejecución *Microsoft .Net Framework* 4.0 y como sistema operativo *Windows XP Sp3* o superior.

4.6 Pruebas.

Las pruebas del *software* son un elemento crítico para la garantía de calidad del *software* y representa una revisión final de las especificaciones, del diseño y de la codificación.

Las pruebas de *software*, son los procesos que permiten verificar y revelar la calidad de un producto *software*. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa (41).

En la medida en que culminan las iteraciones del ciclo de vida de los sistemas es necesaria la verificación del cumplimiento de los objetivos propuestos para la etapa en cuestión. En cada iteración, el equipo de desarrollo obtiene como resultado de su trabajo una serie de componentes que se traducen en el cumplimiento de los requerimientos funcionales del sistema a desarrollar, es por ello que la realización de las pruebas del sistema más allá de chequear el cumplimiento o no de los objetivos de la iteración, verifican y revelan la calidad del *software*.

4.6.1 Pruebas unitarias.

Las pruebas unitarias se realizan sobre el código de la aplicación con el objetivo de demostrar que estos no contienen errores a la hora de generar las salidas pertinentes, asegurando que la solución no presenta errores de lógica de programación y que las respuestas son las correctas ante un conjunto de datos de entrada determinados por el probador.

Mediante los métodos de prueba de caja blanca, el ingeniero del *software* puede obtener casos de prueba que (41):

- garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
- ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas.
- ejecuten todos los bucles en sus límites y con sus límites operacionales.
- ejerciten las estructuras internas de datos para asegurar su validez.

La prueba de caja blanca del *software* se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del *software* proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles.

A continuación se muestra la realización de la prueba unitaria al método *IsCycleEnd.* El resto de las pruebas unitarias a los principales métodos de la herramienta se pueden apreciar en el **Anexo 6** Pruebas de caja blancaAnexo .

```
[TestMethod()]
public void IsCycleEndTest()
{
    var node2 = new TaskActivity();
    node2.taskType = TaskType.Task;
    node2.identificador = "Udbd6c3e2-e2e9-406e-8b5a-c22c1fd898bb";
    node2.nombre = "Task";
    var prop2 = new List<Property>();
    prop2.Add(new Property { PropertyName = "ActivityType", Instancevalue = "Task" });
    prop2.Add(new Property { PropertyName = "LoopType", Instancevalue = "None" });
    var clientDependecy2 = new List<Dependency>();
    clientDependecy2.Add(new Dependency
    {
        Identificador = "Uaa1697bb-f66e-4fc0-82f4-b41bb3b12642",
        Client = "Udbd6c3e2-e2e9-406e-8b5a-c22c1fd898bb",
        Suplier = "U5be5d420-9199-4d83-9225-31bbca9d2115",
        DependecyType = DependecyType.SequenceFlow,
        properties = new List<Property>()
    });
    node2.clientDependecy = clientDependecy2;
    node2.propiedades = prop2;
    node2.BorderEvents = new List<string>();
    var node3 = new EventNodo { identificador = "U5be5d420-9199-4d83-9225-31bbca9d2115" };
    var prop3 = new List<Property>();
    prop3.Add(new Property { PropertyName = "EventType", Instancevalue = "End" });
    prop3.Add(new Property { PropertyName = "Result", Instancevalue = "None" });
    node3.clientDependecy = new List<Dependency>();

    node3.propiedades = prop3;
    var vertices = new List<Node> { node2, node3 };

    GrafoMA grafo = new GrafoMA(); // TODO: Initialize to an appropriate value
    for (int i = 0; i < vertices.Count; i++)
        grafo.InsertarVertice(vertices[i]);
    foreach (Node nodo in vertices)
        foreach (Dependency t in nodo.clientDependecy)
            grafo.InsertarArco(t);
    TransformationController target = new TransformationController(); // TODO: Initialize to an appropriate value
    Node n = node3; // TODO: Initialize to an appropriate value
    //GrafoMA grafo = null; // TODO: Initialize to an appropriate value
    bool expected = false; // TODO: Initialize to an appropriate value
    bool actual;
    actual = target.IsCycleEnd(n, grafo);
    Assert.AreEqual(expected, actual);
}
```

Figura 21 Prueba unitaria al método *IsCycle.*

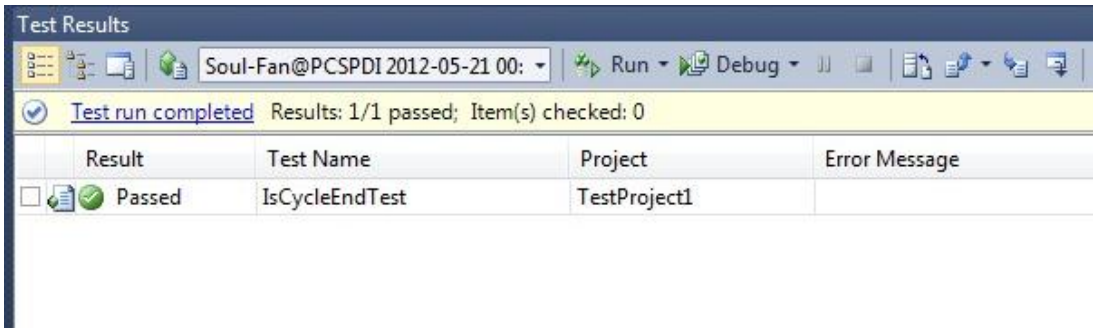


Figura 22 Resultado de la prueba unitaria al método *IsCycleEnd.*

4.6.2 Pruebas de sistema.

Las pruebas de caja negra, también denominadas prueba de comportamiento, se centran en los requisitos funcionales del *software*. O sea, la prueba de caja negra permite al probador obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa.

Los casos de prueba de caja negra pretenden demostrar que (41):

- las funciones del software son operativas.
- la entrada se acepta de forma correcta.
- se produce una salida correcta.
- la integridad de la información externa se mantiene.

4.6.2.1 Diseño de casos de pruebas.

El diseño de casos de pruebas consiste en la confección de los distintos casos de pruebas según la identificada previamente. Definen un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Cada técnica de prueba proporciona unos criterios distintos para generar estos casos o datos de prueba. Con la aplicación de los casos de pruebas se obtuvieron resultados, identificando diferentes errores en la herramienta. A continuación se muestra el diseño de caso de prueba de la funcionalidad Agregar componente de *Bison Framework*, el resto de los casos de pruebas se encuentran en el **Anexo 7** Diseños de caso de prueba

Anexo .

Escenario	Descripción	Nombre	Tratamiento de errores	Respuesta del sistema	Flujo central
EC 1.1 Agregar un componente	El sistema debe mostrar los campos para realizar la inserción del componente	V *NewActivity*	N/A N/A	El sistema limpia los campos para agregar otro componente si se desea.	1. Mostrar el campo para agregar el nuevo componente. a) Nombre del componente. b) Tratamiento de errores. 2. Insertar los datos.
EC 1.2 Agregar un componente que ya existe.	El usuario agrega un componente que ya existe.	I *WebActivity*	N/A	El sistema muestra un mensaje informando que ya existe el componente.	
EC 1.3 Cancelar Opción	El sistema limpia los campos y cierra la ventana.	NA	NA	Se cierra la ventana.	1. Seleccionar la opción Cancelar.

Tabla 7 Diseño de caso de prueba del RF 6 Agregar componente del *Bison*

En cada iteración se realizaron rigurosamente las pruebas a la herramienta con el objetivo de detectar cualquier error o fallo no previsto y registrar las no conformidades para ser corregidas posteriormente. A continuación se muestran los resultados obtenidos para la segunda iteración de pruebas. En el **Anexo 8** se muestran las no conformidades pertenecientes a la primera iteración de pruebas.

Iteración 1					
Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección de errores	Importancia
RF 1	1	Se muestra el árbol de representación de ficheros *XMI* con diagramas erróneos	Cargar fichero *XMI*	Al cargar el fichero	Significativa
RF 2	2	Al arrastrar y soltar una actividad del *Bison* no se muestra en la interfaz	Gestionar elemento de mapeo	Al arrastrar un componente del *Bison*	Significativa
RF 2	3	Al no seleccionar ningún elemento de la lista de mapeo elimina el primero de la lista	Gestionar elemento de mapeo	Al seleccionar la opción Eliminar seleccionado	No significativa
RF 3	4	No se valida que se halla cargado un fichero *XMI*	Generar *workflow*	Al seleccionar la opción generar *workflow*	No significativa
RF 3	5	No se valida la selección de alguna configuración	Generar *workflow*	Al seleccionar la opción generar *workflow*	No significativa
RF 3	6	El árbol de representación con las actividades de	Generar *workflow*	Al seleccionar la opción generar *workflow*	No significativa

		WWF y Bison Framework muestra elementos con nombre repetidos			
RF 5	7	Se permite la introducción de números en el campo del nombre de la propiedad	Adicionar propiedades del elementos de *BPMN 1.0*	Al insertar los datos	No significativa
RF 6	8	Al insertar un nuevo componente del *Bison* se coloca en el listado de componentes de *WWF* 3.5	Agregar componente de *Bison Framework*	Al adicionar el nuevo componente	No significativa

Tabla 8 No Conformidades de la segunda iteración.

4.6.3 Resultados de las pruebas.

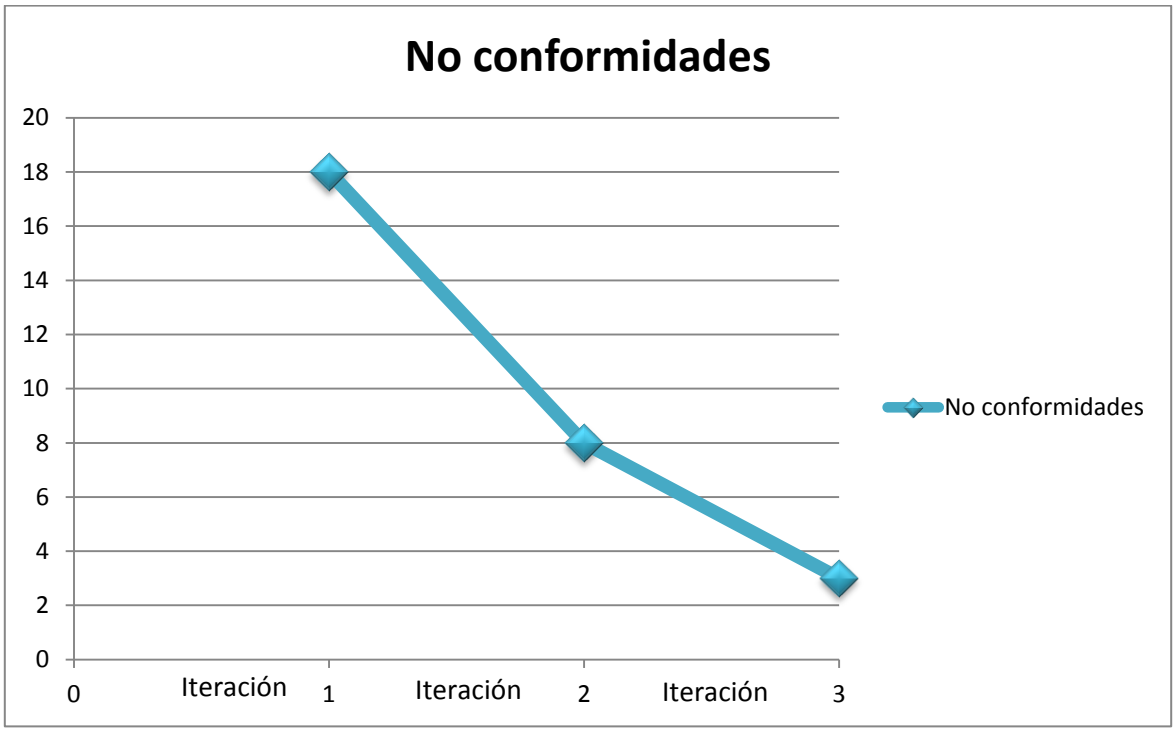


Figura 23 No conformidades por iteraciones.

La herramienta fue sometida a tres iteraciones de prueba que arrojaron en su mayoría fallos de validaciones y errores ortográficos. En la primera iteración se detectaron 18 no conformidades de las cuales 5 representaban fallos significativos de la herramienta, siendo erradicados durante la etapa de pruebas. Para la segunda iteración con 8 no conformidades se mantuvieron los errores de validaciones, algunos más significativo que otros pero sin grandes repercusiones para la herramienta. En la tercera iteración se detectaron 3 no conformidades reflejando una reducción de los errores.

Las no conformidades derivadas durante la etapa de pruebas fueron corregidas en la medida que eran detectadas. Actualmente quedan no conformidades que serán eliminadas en futuras iteraciones de pruebas. También se realizaron pruebas unitarias a los diferentes métodos implementados arrojando resultados satisfactorios como se muestra en la **Figura 22**.

4.6.4 Validación de la hipótesis.

Dentro de las pruebas realizadas se transformaron algunos diagramas de proceso de negocio existentes en el SUIN. De una total de 17 procesos en el SUIN se tomaron 3 procesos para realizar el análisis de tiempo

que les lleva a los desarrolladores implementar estos procesos, que representa el tiempo que tarda en diseñar los *workflow* y el porcentaje aproximado de incompatibilidades que se aprecian en los *workflow* implementados con respecto a los diagramas de proceso de negocio creados por los analistas.

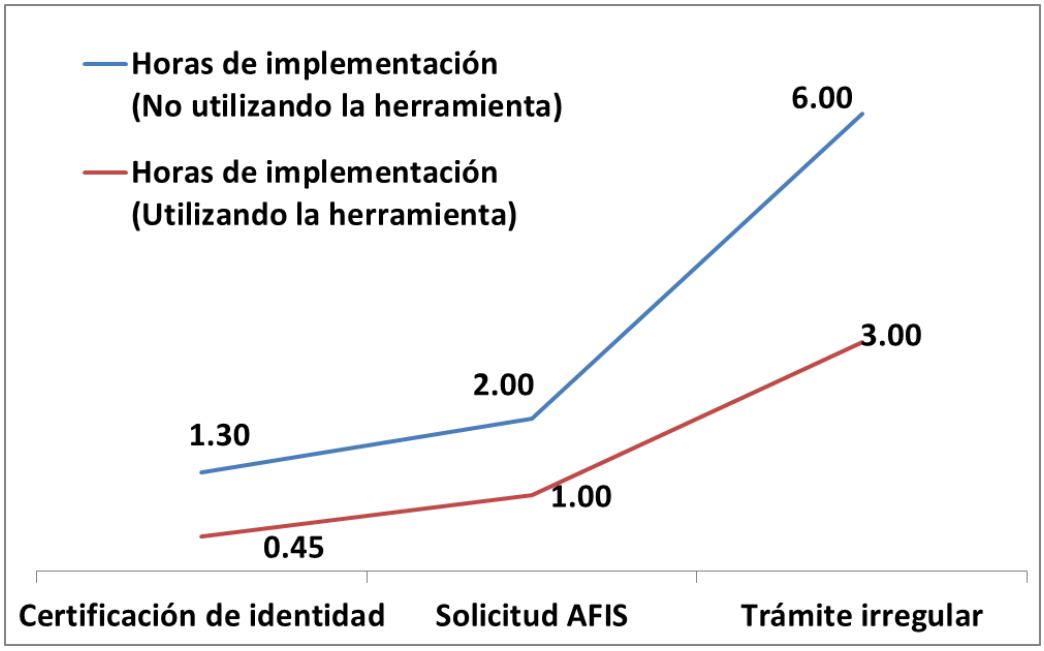
Procesos del SUIN	Tiempo de implementación(horas)	Incompatibilidades aproximadas entre el *workflow* y el modelo de proceso de negocio (%)
Certificación de identidad	1.30	25
Solicitud *AFIS*[38]	2.00	30
Trámite irregular	6	50

Tabla 9 Tiempo de implementación e incompatibilidades entre los *workflow* y el diagrama de proceso de negocio en los procesos del SUIN.

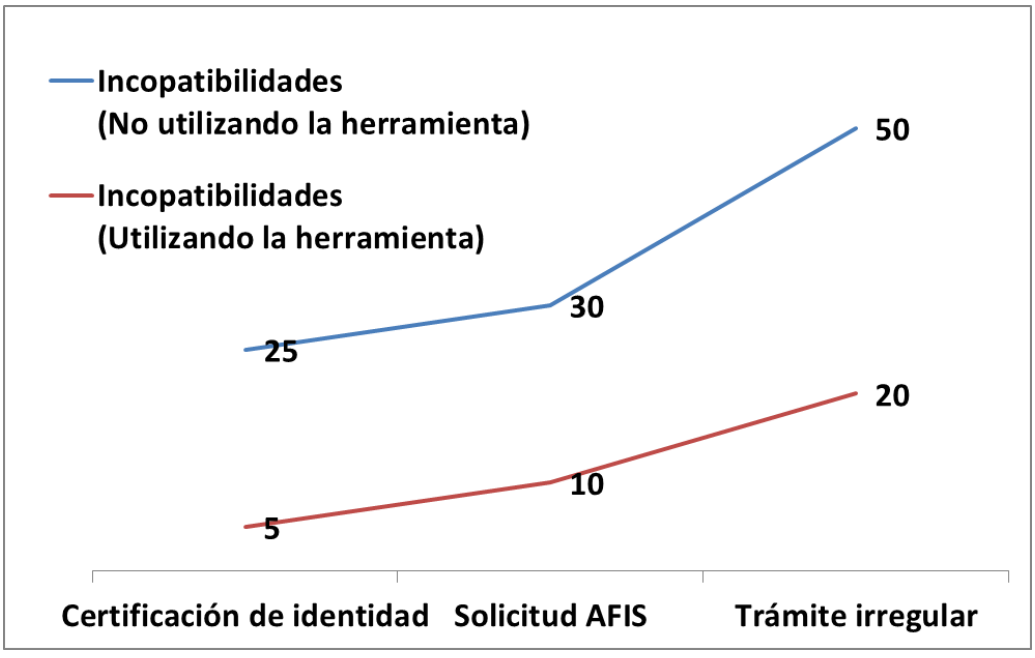
Procesos del SUIN	Tiempo de implementación(horas)	Incompatibilidades entre el *workflow* y el modelo de proceso de negocio (%)
Certificación de identidad	0.45	5
Solicitud AFIS	1	10
Trámite irregular	3	20

Tabla 10 Tiempo de implementación e incompatibilidades entre los *workflow* y el diagrama de proceso de negocio en los procesos del SUIN utilizando la herramienta.

[38] Sistema de Autentificación de Impresión de Dedo (*Authentication Finger Print System*).



Gráfica 1 Horas de implementación.



Gráfica 2 Incompatibilidades en la transformación.

Analizando los resultados de las pruebas realizadas se comprueba la reducción, tanto en el tiempo de implementación de los procesos como en las incompatibilidades entre los *workflows* y los diagramas de procesos de negocio iniciales por parte de la herramienta. En el **Anexo 9** Ejemplo de transformación de *BPMN* 1.0 a *WWF* 3.5.se pude apreciar un ejemplo de la transformación de un diagrama de proceso de negocio a un *workflow* sobre la tecnología *WWF* 3.5**.**

4.7 Beneficios de la herramienta.

El desarrollo de una herramienta para la transformar los modelos de procesos de negocio representados en *BPMN* 1.0 mediante herramienta de modelado *Altova UModel* a flujos de trabajo de la tecnología *Windows Workflow Foundation* 3.5 permite al Sistema Único de Identificación Nacional:

- reducir las incompatibilidades que existen a la hora de los desarrolladores interpretar los diagramas de procesos de negocio.
- agilizar las funciones de implementación en el proyecto, garantizando un diseño preliminar de los *workflows*, lo cual acorta el tiempo de representación de estos por parte de los desarrolladores, quienes junto a los analistas, simplifican el tiempo de interpretación de los procesos de negocio y su posterior transformación a la tecnología *WWF* 3.5 y *Bison Framework*.
- garantiza en mayor nivel la automatización final de los procesos inicialmente definidos por los analistas del proyecto.

4.8 Conclusiones parciales.

El uso de los diferentes estándares de codificación aportó legibilidad y limpieza al código generado en la implementación de la herramienta. En la etapa de implementación se llega a una solución final describiéndose cada uno de los componentes que la conforman siguiendo la arquitectura definida, lo que posibilitó la obtención de una herramienta que garantiza la transformación de *BPMN* 1.0 a *Windows Workflow* Foundation 3.5 y *Bison Framework*. Durante la fase de pruebas se detectaron deficiencias en el funcionamiento de la herramienta las cuales garantizaron una mejor validación de los objetivos que se persiguen con la misma.

Conclusiones generales

El proceso de investigación e implementación de la “Herramienta para la transformación de procesos de *BPMN* 1.0 a la tecnología *Windows Workflow Foundation* 3.5” permitió llegar a las siguientes conclusiones:

- con el análisis de los principales conceptos asociados a los modelos de negocio a través del estándar *BPMN* 1.0 y su posible relación con la tecnología *Windows Workflow Foundation* 3.5 se logró un mejor entendimiento de la problemática planteada, mostrando la necesidad de obtener una herramienta que simplifique y agilice la transformación de los procesos de negocio a los flujos de trabajo, que tienen lugar en el Sistema Único de Identificación Nacional.
- con la representación del modelo de dominio, además de trazar un bosquejo en el que se evidencian, en su forma más abstracta, los principales conceptos con los que cuenta la aplicación, se esclarecen las principales funcionalidades con las que cuenta la misma.
- la definición de una arquitectura de n-capas junto a la correcta utilización de los patrones de diseño contribuye en la elaboración de una aplicación flexible y escalable.
- a partir de las pruebas realizadas a la herramienta, donde fueron detectados diferentes errores en el funcionamiento de la misma, se logró perfeccionar el trabajo realizado garantizando un mayor nivel de calidad de software y así darle cumplimiento a los objetivos planteados.
- con la entrega de una aplicación funcional que garantiza la transformación de los diagramas de negocio bajo el estándar *BPMN* 1.0 a la tecnología *Windows Workflow Foundation* 3.5, se reducen las incompatibilidades entre el diseño del proceso de negocio y su implementación, así como el tiempo de desarrollo de los procesos en el Sistema Único de Identificación Nacional, con lo cual se le da cumplimiento a la hipótesis planteada.

Recomendaciones

Con el objetivo de incorporar mejoras a la herramienta se recomienda:

- implementar otros patrones de control de flujo con el objetivo de garantizar la interpretación de modelos de proceso de negocio más expresivos.
- adaptar la herramienta para otras versiones de *BPMN.*

Referencia bibliográfica

1. **Davenport, Thomas.** *Process Innovation: Reengineering work through information technology.* Cambridge, Massachusetts : s.n., 1993.

2. ebizq.net. [Online] [Cited: 3 23, 2012.] http://www.ebizq.net/topics/human_centric_bpm/features/7852.html.

3. **Heyl, Bernhard Hitpass.** [Online] junio 2011. http://www.noticias.utfsm.cl/web/uploads/pdf/de6cb32b1d59df4686b39873df9f91032e146cbb.pdf.

4. Modelo del Negocio con RUP y UML Parte 1. *Modelo del Negocio con RUP y UML Parte 1.* [Online] 2009. [Cited: 12 11, 2011.] http://www.slideshare.net/david.motta/modelo-del-negocio-con-rup-y-uml-parte-1.

5. **BPMI.** *Business Process Notation Specification.* 2006.

6. **Pérez, Juan Diego.** Notaciones y lenguajes de procesos. Una visión global. [En línea] [Citado el: 26 de 11 de 2012.] http://www.lsi.us.es/docs/doctorado/memorias/Perez,%20Juan%20D.pdf.

7. **Alfonso Rodríguez, Eduardo Fernández Medina, Mario Piattini.** [Online] 11 21, 2005. [Cited: 12 9, 201.] http://www.criptored.upm.es/cibsi/cibsi2005/presentaciones/sesion11/Hacia_una_definicion_de_procesos_de_negocios_seguros.pdf.

8. **Stephen A. White, PHD Derek Miers.** *Guía de Referencia y Modelado BPMN.* EEUU : s.n., 2009.

9. —. *Guía de Referencia y Modelado BPMN.* EEUU : s.n., 2009.

10. **Workflow Management Coalition.** Workflow Management Coalition. *Workflow Management Coalition.* [Online] Febrero 1999. [Cited: 12 10, 2011.] http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf.

11. **Rodriguez, Sandra Melyna Lopez.** Scribd. [Online] [Cited: 12 9, 2011.] http://es.scribd.com/doc/42813565/Flujo-de-trabajo#outer_page_7.

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

13. **Workflow Management Coalition .** Workflow Management Coalition . [Online] 7 1998. [Cited: 12 9, 201.] http://www.wfmc.org/ .

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

15. **Bravo, Lorenzo.** Windows Workflow Foundation Mucho más que documentación. [Online] [Cited: 12 2, 2011.]

16. **Bukovics, Bruce.** *Pro WF Windows Workflow Foundation in .Net 3.5.* New York : s.n., 2008.

17. **Basurto, Cristhian Kirs Herrera.** Scribd. [Online] http://es.scribd.com/doc/2068969/Instalacion-de-Bonita-Workflow#outer_page_1.

18. Productos Representado por Datamatic. [Online] [Cited: 12 13, 2011.] http://facilplan.com/datamatic/bizagi.html.

19. Los beneficios del Modelador Bizagi-Aplicaciones Empresariales.com. [Online] 8 12, 2005. http://www.aplicacionesempresariales.com/los-beneficios-del-modelador-bizagi.html.

20. Intalio. [Online] [Cited: 12 14, 2011.] http://bpms.intalio.com/product.

21. **P.Letelier.** [Online] [Cited: 12 9, 2011.] http://www.google.com.cu/url?sa=t&rct=j&q=la+metodolog%C3%ADa+de+desarrollo+define+qui%C3%A9n%2C+debe+hacer+qu%C3%A9%2C+cu%C3%A1ndo+y+c%C3%B3mo+debe+hacerlo&source=web&cd=3&ved=0CDIQFjAC&url=http%3A%2F%2Fwww.dsic.upv.es%2Fasignaturas%2Ffacultad%2Flsi%2Fdo.

22. **Microsoft.** MSDN MSF for CMMI Process Improvement v5.0. [Online] 11 25, 2011. http://msdn.microsoft.com/es-es/library/dd997574.aspx.

23. MSDN. [Online] 2012. [Cited: 12 11, 2011.] http://msdn.microsoft.com/es-es/library/ee461556.aspx.

24. **Rico, David F.** *ROI of software Process Improvement. Metric for Project Managers and Software Engineers.* 2004.

25. MSDN. *MSDN.* [Online] [Cited: 12 9, 2011.]

26. MSDN. [Online] 2012. [Cited: 3 8, 2012.] http://msdn.microsoft.com/es-es/library/dd547188.

27. **Archer, Tom.** *C# a Fondo.* España : s.n., 2001. 84-481-3246-7.

28. **Microsoft.** LINQ (Language-Integrated Query). *LINQ (Language-Integrated Query).* [Online] [Cited: 12 9, 2011.] http://msdn.microsoft.com/es-es/library/bb397926.aspx.

29. —. MSDN. [Online] [Cited: 12 9, 2011.] http://msdn.microsoft.com/es-es/library/bb655883%28v=vs.90%29.aspx.

30. **M., Pérez J. García.** *XMI: XML Metadata Interchange.* Valencia : s.n.

31. WinForms Controls. [Online] [Cited: 3 1, 2012.] http://www.telerik.com/products/winforms.aspx.

32. **Douglas Bell, Mike Parr, Alfonso Vidal, Romero Elizondo, Guillermo Levine Gutierrez.** *Java para estudiantes.* Naucalpan de Juárez, Edo. de México : Pearson Education, 2003.

33. **Booch, Grady , Rumbaugh, James y Jacobson, Ivar.** *El lenguaje unificado de modelado.* 2006.

34. [Online] enero 17, 2011. [Cited: diciembre 9, 2011.] http://www.altova.com/umodel.html.

35. **Pressman, Roger S.** *Ingeniería de Software un enfoque practico.*

36. **Joaquín.** Patrones de diseño.Análisis y diseño.Ingeniería de software. [Online] 2003. [Cited: 3 10, 2012.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

37. **Outes, Ignacio Calvo.** Scribd. [Online] 2008. http://es.scribd.com/doc/53520152/14/Patrones-GRASP.

38. **Turossi, AUS Gustavo.** Scribd. [Online] 2008. http://es.scribd.com/doc/51305810/50/Implementacion.

39. **Falgueras, Benet Campderrich.** *Ingeniería del software.* s.l. : UOC, 2003.

40. **Marca Huallpara, Hugo Michael,Quisbert Limachi,Nancy Susana.** *ANALISIS Y DISEÑO DE SISTEMAS II Diagrama de Despliegue.*

41. Scribd. [Online] 1 2011. [Cited: 4 25, 2012.] http://es.scribd.com/doc/50741997/Pruebas-de-Software.

42. BPMDaySeminarioBPMN3Dic09. *BPMDaySeminarioBPMN3Dic09.* [Online] [Cited: 12 12, 2011.] http://www.club-bpm.com/BPMday/BPMDaySeminarioBPMN3Dic09.pdf.

43. Instalación y primeros pasos con Bonita Workflow. [Online] karmacracy, 2003. [Cited: 12 13, 2011.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=bonitaWorkflow.

44. Modelos de gestión de la calidad de software: CMMI que es? niveles ? beneficicios? *Modelos de gestión de la calidad de software: CMMI que es? niveles ? beneficicios?* [Online] 1 3, 2008. [Cited: 12 11, 2011.] http://juanmarcosteoria2.blogspot.com/2008/01/para-el-enriquecimiento-de-los-lectores.html.

45. **Provencio, Francisco Recio y David.** .Net Framework. *.Net Framework.* [Online] 12 10, 2003. [Cited: 12 9, 2011.] http://www.desarrolloweb.com/articulos/1328.php.

46. **Archer, Tom.** *C# a Fondo.* España : s.n., 2001. 84-481-3246-7 .

47. BPEL: estandarización de la gestión de procesos. [Online] [Cited: 12 20, 2012.] http://www.iworld.com.mx/iw_SpecialReport_read.asp?iwid=4135&back=2&HistoryParam=.

48. *BPEL y Orquestación de BPEL y Orquestación de.* **Díaz, Jonathan Saúl Torres.**

49. **Quispe-Otazu, Rodolfo.** ¿Que es la Ingenieria de Requisitos?| Computación e Informática. [Online] 8 2007. [Cited: 2 10, 2012.] http://www.rodolfoquispe.org/blog/que-es-la-ingenieria-de-requisitos.php.

50. Especificaciones De Requerimientos. [Online] [Cited: 2 10, 2012.]

51. **Shaw, David Garlan y Mary.** *An introduction to Software Architecture.*

52. **Zambrán, Reynier Blanco.** *Documento de arquitectura del Proyecto Identidad Cuba.*

53. [Online] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

54. [Online] 2007. [Cited: 3 1, 2012.] http://es.downv.com/download-Telerik-RadControls-de-WinForms-10226620.htm.

55. [Online]

56. **Bravo, Lorenzo.** Windows Workflow Foundation Mucho más que documentación. [Online] [Cited: 12 2, 2011.] http://www.dnmplus.net/articulos/windows-workflow-foundation-mucho-mas-que-documentacion.aspx.

57. **Emprendimientos Corporativos S.A.** iProfesional.com. *iProfesional.com.* [Online] 2010. [Cited: diciembre 12, 2011.] http://www.iprofesional.com/notas/51865-Los-beneficios-de-la-inteligencia-de-negocios.html.

58. **Identidad Cuba.UCI.** *Proyecto Técnico.DIR.* 2009.

59. **Proyecto Identidad Cuba.UCI.** *Arquitectura de Software.* [Documento] 2010.

60. **The Data Warehousing Institute.** tdwi. [Online] [Cited: Diciembre 10, 2011.]

61. **Inteligencia de Negocios S.A.** in. *in.* [Online] 2011. [Cited: 12 10, 2011.] http://www.idensa.com/.

62. **Gravitar.** Gravitar Información sin Límites. *Gravitar.* [Online] 2011. [Cited: diciembre 12, 2011.] http://www.gravitar.biz/index.php/herramientas-bi/pentaho/.

63. **Casales Cabrera, María Evilia.** Faultad de Ciencias. Universidad Nacional Autónoma de México. *Faultad de Ciencias. Universidad Nacional Autónoma de México.* [Online] 2009. [Cited: noviembre 21, 2011.] http://hp.fciencias.unam.mx/~alg/bd/dwh.pdf.

64. **Clemente García, Gerardo.** Riunet. *Riunet.* [Online] 2008. [Cited: 12 11, 2011.] http://riunet.upv.es/manakin/bitstream/handle/10251/2505/tesisUPV2842.pdf.

65. **Berzal, Fernando.** [Online] http://elvex.ugr.es/idbis/db/docs/intro/F%20Modelo%20multidimensional.pdf.

66. **Valencia Arcos, Janeth del Carmen and Guevara Lenis, Jorge Eduardo.** [Online] 6 2007. [Cited: 12 12, 2011.] http://bibdigital.epn.edu.ec/bitstream/15000/445/1/CD-0827.pdf.

67. **Sinnexus.** Sinnexus. *Sinnexus.* [Online] 2011. [Cited: 12 13, 2011.] http://www.sinnexus.com/business_intelligence/olap_avanzado.aspx.

68. **Microsoft.** MSDN. *MSDN.* [Online] 2012. [Cited: 4 5, 2012.] http://msdn.microsoft.com/es-es/library/ms190457.aspx.

69. **Pressman, Roger.** *Ingenieria del Software: Un Enfoque Practico.* s.l. : McGraw-Hill, 2006.

70. **Escobar Domínguez, René Raydel.** NETRONICS. *NETRONICS.* [Online] 2011. [Cited: 11 12, 2011.] http://www.netronycs.com/clasificacion_de_base_datos.html.

71. **Universidad de Holguín.** Plataforma de Aprendizaje de la Universidad de Holguín. *Plataforma de Aprendizaje de la Universidad de Holguín.* [Online] 2011. [Cited: 11 12, 2011.] http://www.google.com.cu/url?sa=t&rct=j&q=base+de+datos&source=web&cd=17&ved=0CHoQFjAGOAo&url=http%3A%2F%2Fmoodle.uho.edu.cu%2Fmod%2Fresource%2Fview.php%3Finpopup%3Dtrue%26id%3D3528&ei=e_K6T5TRGoPqgAfa0qXnCg&usg=AFQjCNGRBL--sywqToaJjf426okhnuX3-w&cad=rja.

72. **Identidad Cuba.** Modelo de datos Identidad. [Online] 2011. [Cited: 12 12, 2011.] http://intranet.uci.dev/bdsite/Shared%20Documents/Release/Identidad/ID-BD1011-2012-Modelo%20de%20Datos-IDENTIDAD.dm1.

73. **W3 Schools.** W3 Schools. [Online] 2010. http://www.w3schools.com/soap/default.asp.

74. **Sparxsystems.** Enterprise Architect. [Online] 2011. [Cited: 1 14, 2012.] http://www.sparxsystems.com.ar/resources/tutorial/physical_models.html.

75. SGuía. [Online] 2009. [Cited: 3 1, 2012.] http://sg.com.mx/guia/node/702.

76. Axegen Anything from XMI Generator . [Online] http://axgen.sourceforge.net/.

77. **Ladreda, Fernández.** *Información General de CMMI.*

78. **W.M.P. van der Aalst, A.H.M ter Hofstede, B. Kiepuszewski, and A.P. Barros.** *Workflow Patterns.*

79. **Nick Rusell, Arthur H.M. ter Hofstede, Wil M.P. van der Aalst, Nataliya Mulyar.** *Workflow Control-Flow Patterns A Review View.*

80. **Marco Zapletal, Wil M.P. van der Aaslt, Nick Rusell, Philip Liegl, Hannes Werthner.** *Patterns-based Analysis of Windows Workflow.*

Anexo 1 Pautas paras los diagramas de proceso de negocio.

1. **Eventos iniciales**

- Los diagramas deben tener un único evento inicial, para representar el punto inicial del proceso de negocio.
- Los eventos iniciales de tipo *Rule*, *Link*, no contienen equivalente en *WWF* 3.5

2. **Eventos finales**

- Los diagramas deben tener como mínimo un evento final para representar la terminación de un proceso de negocio.
- Los eventos finales de tipo *Terminate* hasta ahora no presentan equivalentes en *WWF* 3.5.

3. **Eventos Intermedios**

- Los eventos intermedios en *BPMN* 1.0 de tipo *Link*, no tiene equivalente en *WWF* 3.5.
- Los eventos en la frontera de tipo *Timer*, *Conditional*, *Multiple*, *Message* no son transformados a la tecnología *WWF* 3.5.
- La dirección del flujo de salida de los eventos de frontera no debe ser hacia atrás.

4. **Tipos de modelo de *BPMN* 1.0**

- La herramienta soporta hasta el momento diagramas de orquestación. En caso de presentar interacción con otros participantes del proceso, son representados si estos no implican otro proceso.

5. **Ciclos arbitrarios**

- La herramienta solo reconoce ciclos post-condición y pre-condición, de manera que el principio o fin de un ciclo debe ser un *Gateway* o *Gateway Exclusive*.

6. **Subprocesos**

- Lo subprocesos de tipos *Adhoc* no tienen equivalente en *WWF* 3.5.
- Los eventos iniciales y finales no deben estar en la frontera.

Anexo 2 Descripción de requisitos funcionales.

RF2. Gestionar elemento de mapeo.

Propósito	Crear, modificar o eliminar un elemento de mapeo	
Roles	Desarrollador	
Precondiciones		
Conceptos tratados	*sourceMapping*	*activityTree* *properties*
	targetMapping	*activityWWF* *properties*
Descripción	1. Se muestra un listado con los elementos de *BPMN* 1.0 mostrando: • Tareas • Subprocesos • Eventos 2. Se muestra un listado con los componentes de WWF 3.5 y Bison Framework mostrando: • Nombre del Componente 3. Crear elemento de mapeo 3.1 Seleccionar un elemento de *BPMN* 3.2 Se muestran los campos para gestionar las propiedad del elemento de *BPMN* ver RF 5. 3.3 Se arrastra el o los componentes equivalentes dentro de *WWF* o *Bison Framework*. 3.4 Seleccionar la opción aceptar. 4. Modificar elemento de mapeo 4.1 Se muestra un listado con los elemento de mapeo creados 4.2 Seleccionar un elemento de mapeo de la lista 4.3 Seleccionar la opción modificar seleccionado para modificar el elemento de mapeo 4.4 Mostrar los valores de las propiedades del elemento de *BPMN* ver RF 5 4.5 Se muestran los componentes equivalentes dentro de *WWF*	

	y *Bison Framework.* 4.6 Eliminar o arrastrar otras componentes de *WWF* o *Bison Framework* si se desea modificar las actividades equivalentes. 4.7 Seleccionar otro elemento de *BPMN* si se desea modificar el elemento de *BPMN* 1.0. 4.8 Seleccionar la opción aceptar. 7. Eliminar elemento de mapeo. 7.1 Seleccionar un elemento de mapeo de la lista. 7.2 Seleccionar la opción eliminar seleccionado si desea eliminar un elemento de mapeo.
Validaciones	
Postcondiciones	Se muestra el árbol de representación con a las actividades de *WWF* 3.5 Se muestras el código fuente de los ficheros *.designer.cs* y *.cs* que conforman el *workflow*.
Prototipo	

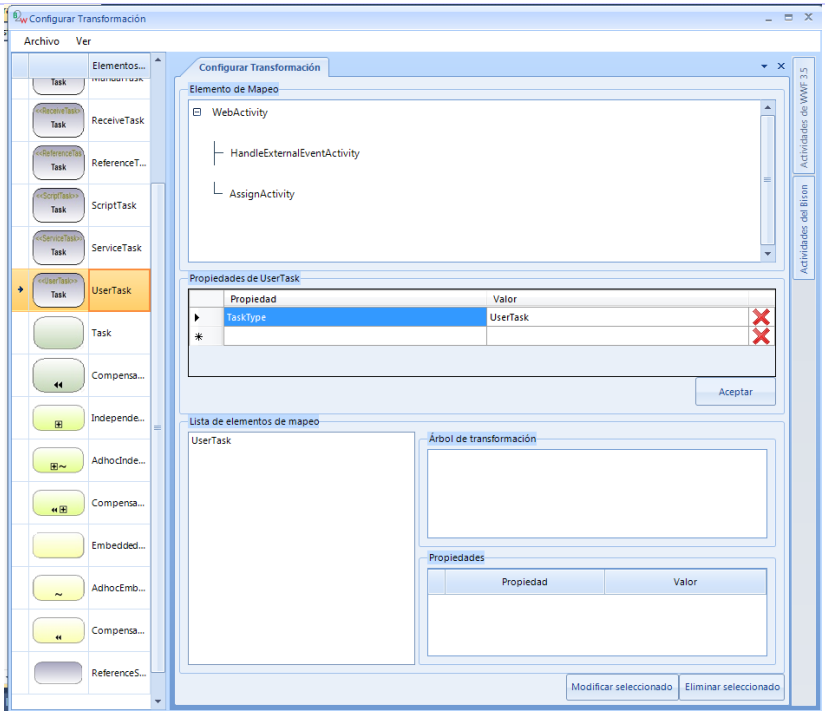


Tabla 11 RF 2. Gestionar elemento de mapeo.

RF 3. Generar *workflow* 3.5.

Propósito	Obtener un *workflow* 3.5 que contenga la transformación de *BPMN 1.0 a WWF* 3.5.	
Roles	Desarrollador	
Precondiciones	1. Se debe haber cargado un fichero *XMI*. Se debe haber seleccionado una configuración de transformación	
Conceptos tratados	**Concepto**	**Atributos**
Descripción	1. Seleccionar la configuración de transformación 2.. Se selecciona la opción de generar a *workflow* 3.5 3. Se muestra una interfaz para salvar el fichero 3.1 Se selecciona la ruta donde se quiere guardar el fichero 3.2 Se introduce el nombre del *workflow* 3.3 Se selecciona la opción guardar.	

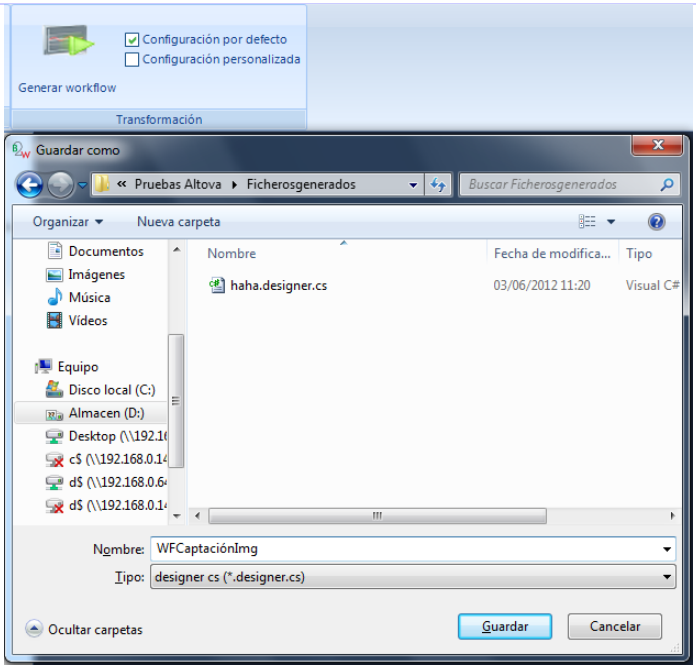
Validaciones	
Postcondiciones	1. Se muestra el árbol de representación con a las actividades de *WWF* 3.5 2. Se muestras el código fuente de los ficheros *.designer.cs* y *.cs* que conforman el *workflow*.
Prototipo	

Tabla 12 RF 3 Generar *workflow* 3.5.

RF 4 Salvar nueva configuración de transformación.

Propósito	Salvar una configuración de transformación con los elementos de mapeo entre *BPMN y WWF 3.5*	
Roles	Desarrollador.	
Precondiciones		
Conceptos tratados	**Concepto**	**Atributos**
	sourceMapping	*activityTree* *properties*
	targetMapping	*activityWWF*

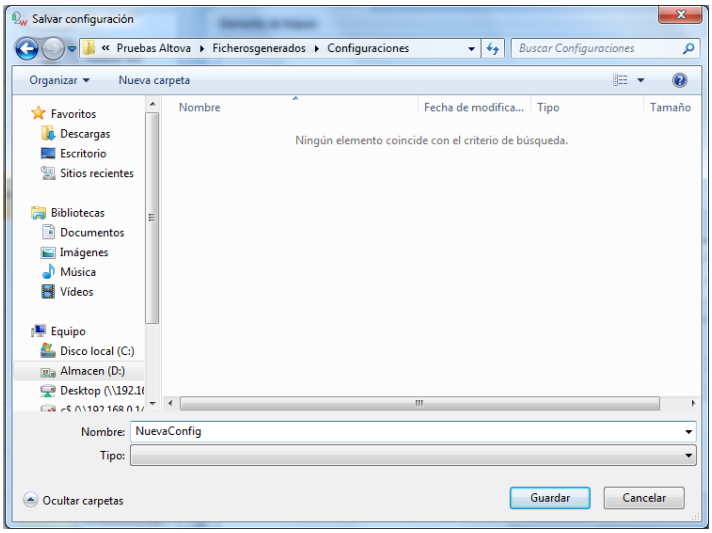
		properties
Descripción	1. Se selecciona la opción salvar configuración 2. Se muestra una interfaz de búsqueda 3. Seleccionar la dirección donde se desea guardar el archivo 4. Introducir el nombre del archivo. 5. Seleccionar la opción guardar.	
Validaciones		
Postcondiciones	1. Se guarda la nueva configuración de transformación	
Prototipo		

Tabla 13 RF 4 Salvar nueva configuración de transformación.

RF 5 Gestionar propiedad de elementos de *BPMN*

Propósito	Gestionar las propiedades de un elemento de *BPMN* 1.0	
Roles	Desarrollador.	
Precondiciones		
Conceptos tratados	**Concepto**	**Atributos**

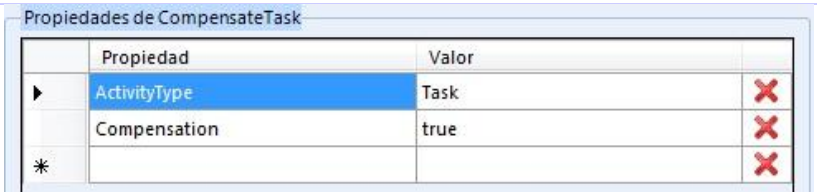
	Properties	*name* *value*
Descripción	1. Se muestran los campos para adicionar las propiedades a.) Propiedad b.) Valor 2. Adicionar propiedad. 2.1 Insertan los datos si se desea adicionar una propiedad. 3. Modificar propiedad. 3.1 Se muestran los campos con los valores a.) Propiedad b.) Valor 3.2 Modificar el valor de los campos si se desea modificar la propiedad. 4. Eliminar propiedad. 4.1 Seleccionar una propiedad si se desea eliminar. 4.2 Seleccionar la opción eliminar	
Validaciones		
Postcondiciones	1. Se gestionan las propiedades de un elemento de *BPMN* 1.0	
Prototipo		

Tabla 14 RF 5 Gestionar propiedad de elemento de *BPMN* 1.0.

RF 6 Agregar componente de *Bison Framework.*

Propósito	Agregar un nuevo componente del *Bison Framework*
Roles	Desarrollador.
Precondiciones	

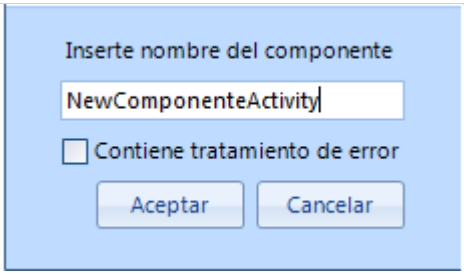
Conceptos tratados	**Concepto**	**Atributos**
	TargetMapping	*activityWWF* *properties*
Descripción	1. Mostrar el campo para agregar el nuevo componente. a) Nombre del componente b) Tratamiento de errores 2. Insertar los datos. 3. Si está registrado el componente se muestra un mensaje informativo de que existe el componente. 4. Seleccionar la opción cancelar si desea terminar.	
Validaciones	1.Verificar que no esté vacío el campo del nombre del componente	
Postcondiciones	1. Se registra en la aplicación la nueva actividad del *Bison* Framework.	
Prototipo		

Tabla 15 RF 6 Agregar componente de *Bison Framework*.

Anexo 3 Descripción de las clases del diseño.

Nombre	***TaskTree***	
Descripción	Entidad que representa un elemento de tipo tarea dentro del árbol de actividades de *BPMN.*	
Nombre del atributo	**Tipo**	**Descripción**
taskType	*TaskType*	Tipo de tarea.
ID	*string*	Identificador del nodo.
name	*string*	Nombre del nodo.
properties	*List<Property>*	Contiene las propiedades pertenecientes al nodo.

Tabla 16 Descripción de la clase *TaskTree.*

Nombre	***SubProcessTree***	
Descripción	Entidad que representa un elemento de tipo subproceso dentro del árbol de actividades de *BPMN.*	
Nombre del atributo	**Tipo**	**Descripción**
subprocessType	*SubProcessType*	Tipo de subproceso.
ID	*string*	Identificador del nodo.
name	*string*	Nombre del nodo.
properties	*List<Property>*	Contiene las propiedades pertenecientes al nodo.

Tabla 17 Descripción de la clase *SubProcessTree*.

Nombre	***GatewayTree***
Descripción	Entidad que representa un elemento de tipo *gateway* dentro del árbol de actividades de *BPMN.*

Nombre del atributo	**Tipo**	**Descripción**
gatewayType	*GatewayType*	Tipo de gateway.
ID	*string*	Identificador del nodo.
name	*string*	Nombre del nodo.
properties	*List<Property>*	Contiene las propiedades pertenecientes al nodo.

Tabla 18 Descripción de la clase *GatewayTree*.

Nombre	***EventTree***	
Descripción	Entidad que representa un elemento de tipo *evento* dentro del árbol de actividades de *BPMN.*	
Nombre del atributo	**Tipo**	**Descripción**
eventType	*EventType*	Tipo de evento.
ID	*string*	Identificador del nodo.
name	*string*	Nombre del nodo.
properties	*List<Property>*	Contiene las propiedades pertenecientes al nodo.

Tabla 19 Descripción de la clase *EventTree.*

Nombre	***CycleTree***	
Descripción	Entidad que representa un cicle dentro del árbol de actividades de *BPMN.*	
Nombre del atributo	**Tipo**	**Descripción**
eventType	*EventType*	Tipo de evento.
ID	*string*	Identificador del nodo.
name	*string*	Nombre del nodo.

Tabla 20 Descripción de la clase *CycleTree*.

Nombre	***ActivityWWF***	
Descripción	Entidad que representa una actividad dentro de *WWF*.	
Nombre del atributo	**Tipo**	**Descripción**
activityType	*string*	Tipo de actividad.
activityName	*string*	Nombre de la actividad.
properties	*List<Property>*	Propiedades de la actividad.

Tabla 21 Descripción de la clase *ActivityWWF*.

Nombre	***MappingElement***	
Descripción	Clase que contiene un elemento de BPMN con su equivalente en *WWF*.	
Nombre del atributo	**Tipo**	**Descripción**
sourceMapping	*SourceMapping*	Elemento de *BPMN*.
targetMapping	*Árbol<TargetMapping>*	Equivalente en *WWF*.

Tabla 22 Descripción de la clase *MappingElement*.

Nombre	***WFTreeBuilding***
Descripción	Clase que permite la construcción de un árbol con las actividades de *WWF.*
Métodos	Descripción
GetTargetBySource	Método que permite obtener un *TargetMapping* por un *SourceMapping.*
BuildSourceMapping	Método que construye un *SourceMapping* con los datos de una entidad *NodeTree.*

FillWFTree	Permite la construcción del árbol de actividades de *WWF.*

Tabla 23 Descripción de la clase *WFTreeBuilding*.

Nombre	***BasePattern***
Descripción	Clase que contempla la interpretación de un patrón de control de flujo dentro del diagrama de proceso de negocio.
Métodos	**Descripción**
CreateID	Genera un identificador para un objeto de tipo *NodeTree*.
FindPattern	Método que detecta un patrón de control de flujo.
GetPriority	Método que permite obtener la prioridad del patrón.

Tabla 24 Descripción de la clase *BasePattern*.

Anexo 4 Diagrama de clases.

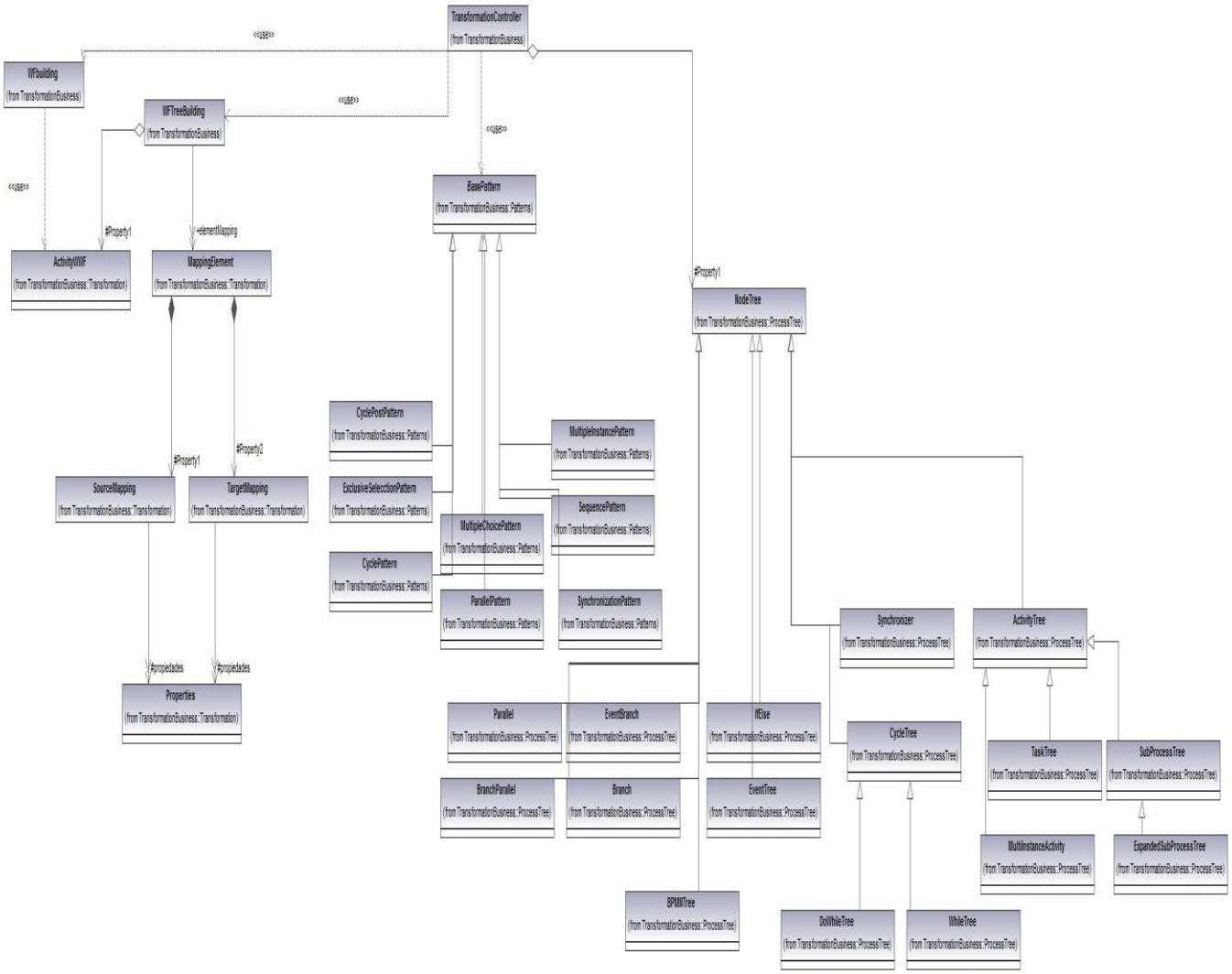


Figura 24 Diagrama de clases de diseño.

Anexo 5 Interfaz de gestionar elemento de mapeo.

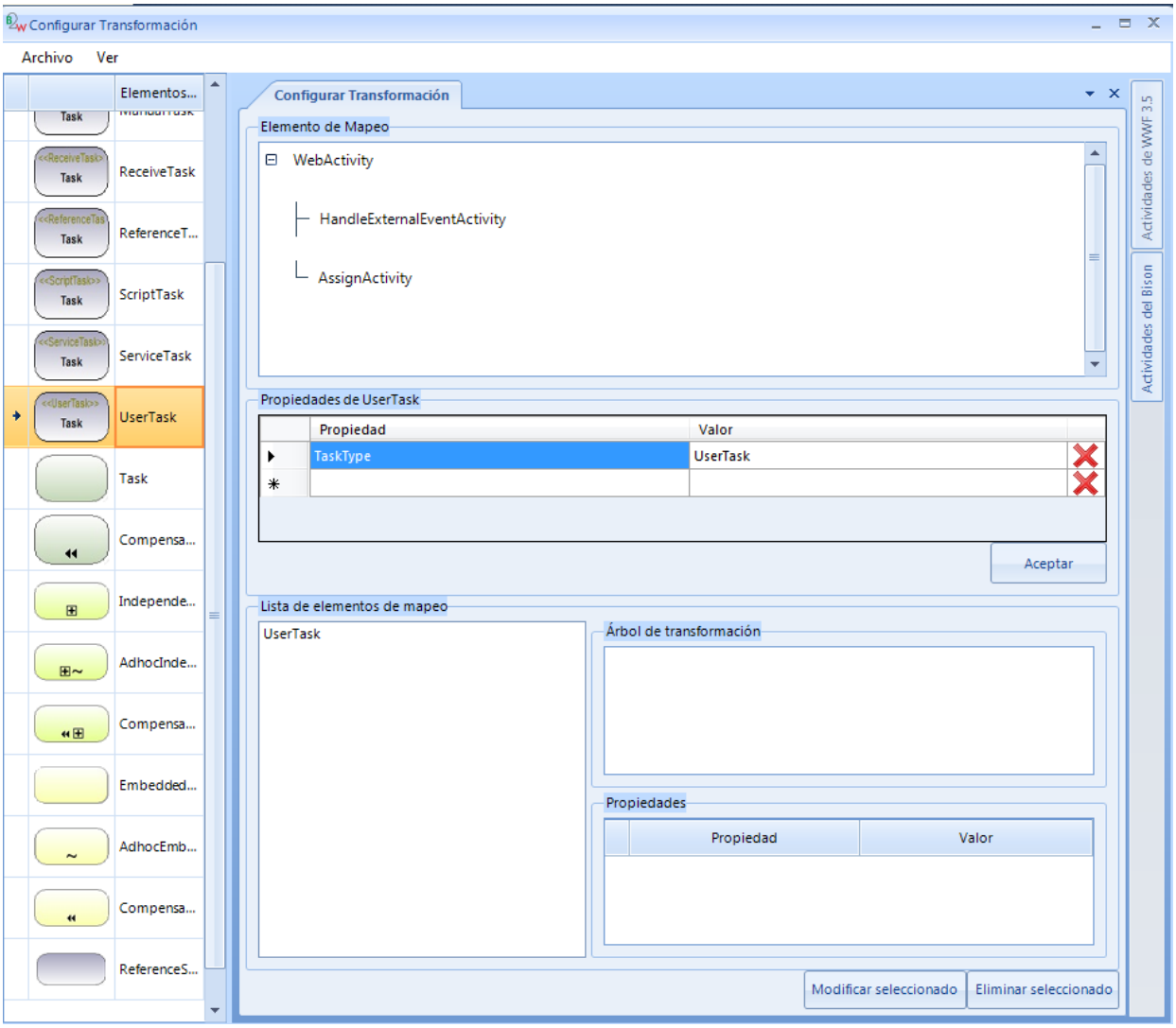


Figura 25 Interfaz de gestionar elementos de mapeo.

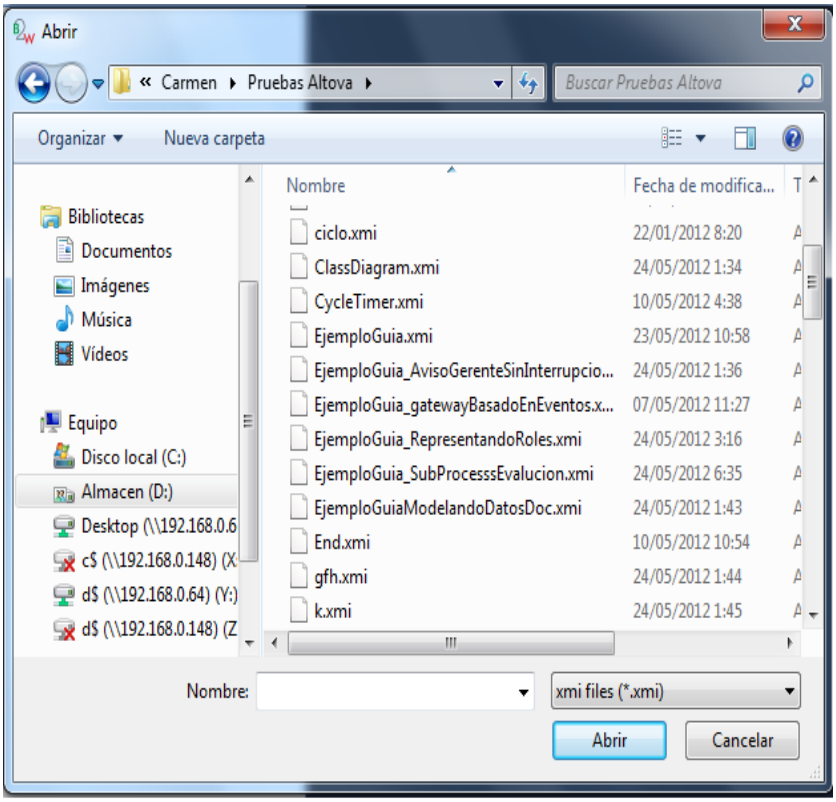


Figura 26 Interfaz de búsqueda del fichero *XMI*.

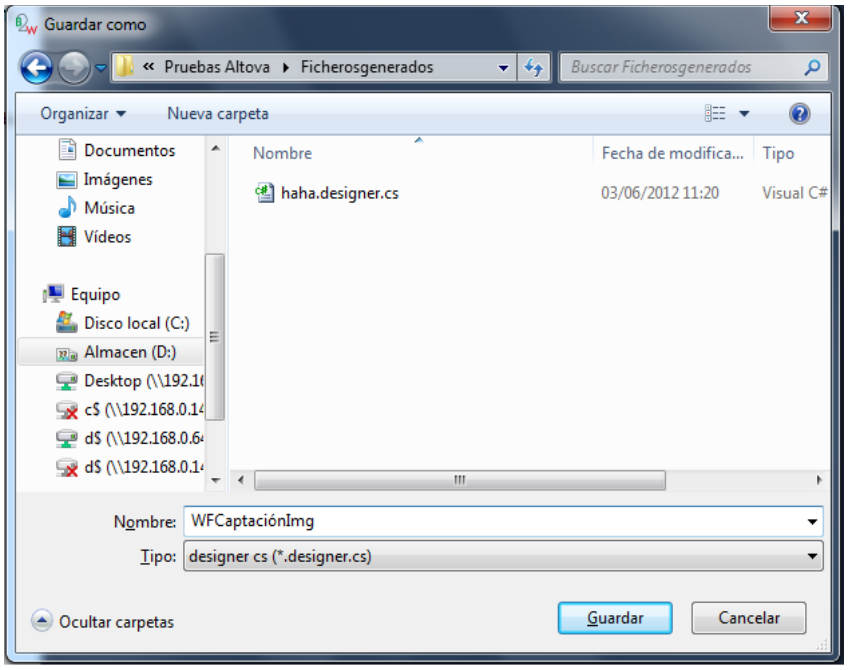


Figura 27 Interfaz de salvar *workflow*.

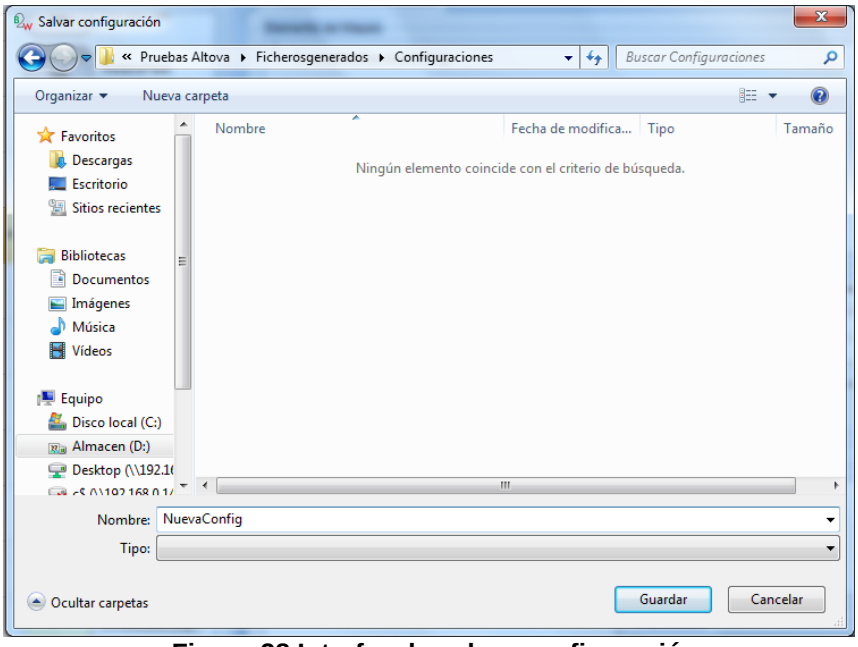


Figura 28 Interfaz de salvar configuración.

Anexo 6 Pruebas de caja blanca.

```
[TestMethod()]
public void GetDistanceTest()
{
    var node2 = new TaskActivity();
    node2.taskType = TaskType.Task;
    node2.identificador = "Udbd6c3e2-e2e9-406e-8b5a-c22c1fd898bb";
    node2.nombre = "Task";
    var prop2 = new List<Property>();
    prop2.Add(new Property { PropertyName = "ActivityType", Instancevalue = "Task" });
    prop2.Add(new Property { PropertyName = "LoopType", Instancevalue = "None" });
    var clientDependecy2 = new List<Dependency>();
    clientDependecy2.Add(new Dependency
    {
        Identificador = "Uaa1697bb-f66e-4fc0-82f4-b41bb3b12642",
        Client = "Udbd6c3e2-e2e9-406e-8b5a-c22c1fd898bb",
        Suplier = "U5be5d420-9199-4d83-9225-31bbca9d2115",
        DependecyType = DependecyType.SequenceFlow,
        properties = new List<Property>()
    });
    node2.clientDependecy = clientDependecy2;
    node2.propiedades = prop2;
    node2.BorderEvents = new List<string>();
    var node3 = new EventNodo { identificador = "U5be5d420-9199-4d83-9225-31bbca9d2115" };
    var prop3 = new List<Property>();
    prop3.Add(new Property { PropertyName = "EventType", Instancevalue = "End" });
    prop3.Add(new Property { PropertyName = "Result", Instancevalue = "None" });
    node3.clientDependecy = new List<Dependency>();

    node3.propiedades = prop3;
    var vertices = new List<Node> { node2, node3 };

    GrafoMA target = new GrafoMA(); // TODO: Initialize to an appropriate value
    for (int i = 0; i < vertices.Count; i++)
        target.InsertarVertice(vertices[i]);
    foreach (Node nodo in vertices)
        foreach (Dependency t in nodo.clientDependecy)
            target.InsertarArco(t);
    Node vert1 = node2; // TODO: Initialize to an appropriate value
    Node vert2 = node3; // TODO: Initialize to an appropriate value
    double expected = 1; // TODO: Initialize to an appropriate value
    double actual;
    actual = target.GetDistance(vert1, vert2);
    Assert.AreEqual(expected, actual);
}
```

Figura 29 Prueba de caja blanca al método *GetDistance.*

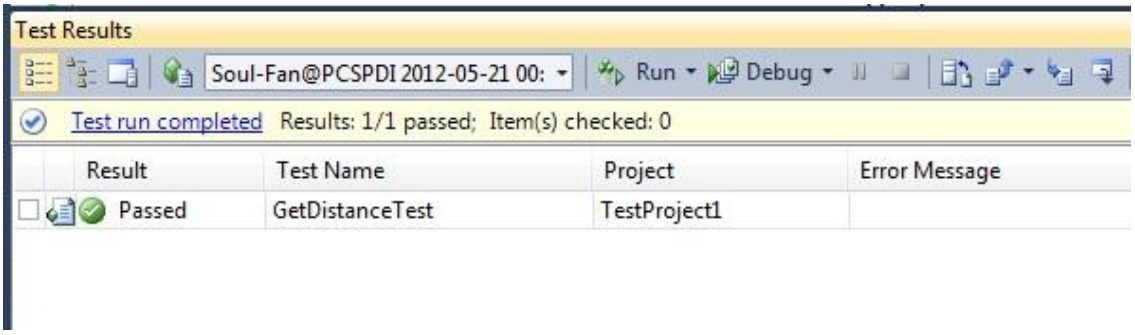


Figura 30 Resultado de la prueba al método *GetDistance.*

```
[TestMethod()]
[DeploymentItem("TransformationBusiness.dll")]
public void IsCommponenTModelTest()
{
    WFbuilding_Accessor target = new WFbuilding_Accessor(); // TODO: Initialize to an appropriate value
    ActivityWWF nodeTree = new ActivityWWF(); // TODO: Initialize to an appropriate value
    nodeTree.ActivtyType = "CodeActivity";
    bool expected = false; // TODO: Initialize to an appropriate value
    bool actual;
    actual = target.IsCommponenTModel(nodeTree);
    Assert.AreEqual(expected, actual);
}
```

Figura 31 Prueba de caja blanca al método *IsComponentModel.*

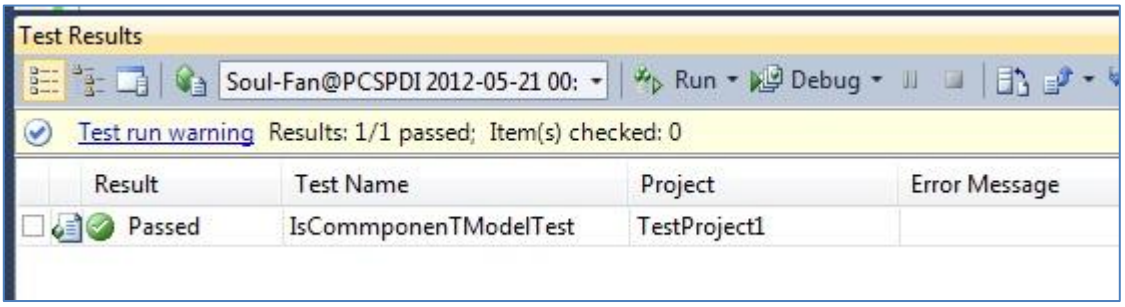


Figura 32 Resultado de la prueba al método *IsComponentModel.*

Anexo 7 Diseños de caso de prueba.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Cargar fichero	El sistema brinca la opción de buscar el fichero *XMI*.	El sistema muestra un árbol de representación con elementos cargados del fichero	1. Se escoge la opción Cargar XMI. 2. Se muestra una interfaz de búsqueda 2.3 Seleccionar un fichero XMI. 2.4 Seleccionar la opción abrir. 3. Se muestra un árbol de representación con las actividades recogidas del fichero.

Tabla 25 Caso de prueba al RF 1 Cargar fichero con extensión *XMI.*

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 3.1 Seleccionar configuración de transformación	El sistema muestra la opción para seleccionar una configuración de transformación	El sistema carga la configuración seleccionada	1. Se selecciona una configuración de transformación
EC 3.2. Generar *workflow* 3.5	El sistema muestra la opción de generar *workflow*	Se muestra un árbol de representación con las actividades de *WWF* y los ficheros que conforman el *workflow*	1. Se selecciona la opción generar *workflow* 2. Se muestra una interfaz para salvar el fichero 2.1 Se selecciona la ruta donde se quiere guardar el fichero 2.2 Se introduce el nombre del *workflow* 2.3 Se selecciona la opción guardar.

Tabla 26 Caso de prueba al RF 3 Generar w*orkflow* 3.5.

Escenario	Descripción	Respuesta del sistema	Flujo central

EC 4.1 Salvar configuración	El sistema brinda la opción de salvar la nueva configuración	El sistema limpia la lista de elementos de mapeo para crear una nueva configuración	1.Se selecciona la opción salvar configuración 2. Se muestra una interfaz de búsqueda 3. Seleccionar la dirección donde se desea guardar el archivo 4. Introducir el nombre del archivo. 5. Seleccionar la opción guardar.

Tabla 27 Caso de prueba al RF 4 Salvar nueva configuración de trasformación.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 2.1 Gestionar elemento de mapeo	Se muestra la interfaz de gestionar elemento de mapeo		
EC 2.2 Crear elemento de mapeo	Se muestra un listado con elementos de *BPMN*, un listado con las actividades de *WWF* y otro listado con las actividades de del Bison *framework*	Se adiciona un elemento de mapeo a la lista de elementos de mapeo	1. Seleccionar un elemento de *BPMN* 2. Se muestran los campos para gestionar las propiedad del elemento de *BPMN* ver RF 3. Se arrastra el o los componentes equivalentes dentro de *WWF* o *Bison Framework*. 4. Seleccionar la opción aceptar

EC 2.3 Crear elemento de mapeo sin actividad de *WWF* o *Bison Framework* equivalente	Se muestra un listado con elementos de *BPMN*, un listado con las actividades de *WWF* y otro listado con las actividades de del Bison framework	El sistema muestra un mensaje de error indicando q no se mapeado el elemento de *BPMN*	1. Seleccionar un elemento de *BPMN* si se desea crear un elemento de mapeo 2. Se muestran los campos para gestionar las propiedad del elemento de *BPMN* ver RF 5 3. Se arrastra el o los componentes equivalentes dentro de *WWF* 3.5 o *Bison Framework*. 4. Seleccionar la opción Aceptar.
EC 2.3 Modificar elemento de mapeo	Se muestra un listado con los elementos de mapeo creados hasta el momento	El modifica el elemento de mapeo	1. Se muestra un listado con los elemento de mapeo creados 2. Seleccionar un elemento de mapeo de la lista 3. Seleccionar la opción modificar seleccionado para modificar el elemento de mapeo 4. Mostrar los valores de las propiedades del elemento de *BPMN* ver RF 5 5. Se muestran los componentes equivalentes dentro de *WWF* y *Bison Framework*. 5.1 Eliminar o arrastrar otras componentes de *WWF* o *Bison Framework* si se desea modificar las actividades equivalentes. 6. Seleccionar otro elemento de *BPMN* si se desea modificar el elemento de *BPMN* 1.0. 7. Seleccionar la opción aceptar.
EC 2.5 Eliminar elemento de mapeo	Se muestra un listado con los elementos de mapeo creados hasta el momento	Se elimina el elemento de mapeo de la lista de elementos de mapeo	1. Seleccionar un elemento de mapeo de la lista. 2. Seleccionar la opción eliminar seleccionado si desea eliminar un elemento de mapeo.

Tabla 28 Caso de prueba la RF 2 Gestionar elemento de mapeo.

Escenario	Descripción	Propiedad	Valor	Respuesta del sistema	Flujo central

EC 5.1 Adicionar propiedad	Se muestran los campos para adicionar una propiedad	V TaskType	V Manual	El sistema adiciona una propiedad al elemento de *BPMN*	1. Se muestran los campos para adicionar las propiedad a.) Propiedad b.) Valor 2. Insertan los datos si se desea adicionar una propiedad
EC 5.2 Adicionar propiedad incorrectamente	Se muestran los campos para adicionar una propiedad	I TaskType789 V *Trigger*	V Manual I *Link*,,?\	El sistema muestra un mensaje de error indicando que existen campos incorrectos	1. Se muestran los campos para adicionar las propiedad a.) Propiedad b.) Valor 2. Insertan los datos si se desea adicionar una propiedad
EC 5.2 Adicionar dejando campos vacíos	Se muestran los campos para adicionar una propiedad	V *Trigger*	I vacío	El sistema muestra un mensaje de error indicando que existen campos vacíos	1. Se muestran los campos para gestionar las propiedades a.) Propiedad b.) Valor 2. Modificar los valores si se desea modificar la propiedad
EC 5.2 Modificar propiedad	Se muestran los campos para modificar una propiedad	V *Trigger*	V Message	El sistema modifica los valores de la propiedad	1. Se muestran los campos para gestionar las propiedades a.) Propiedad b.) Valor 2. Modificar los valores si se desea modificar la propiedad
EC 5.2 Modificar propiedad incorrectame	Se muestran los campos para modificar una propiedad	V *Trigger*	I ?>Message	El sistema muestra un mensaje de error indicando	1. Se muestran los campos para gestionar las propiedades

nte				que existen campos incorrectos	a.) Propiedad b.) Valor 2. Modificar los valores si se desea modificar la propiedad
EC 5.2 Modificar propiedad dejando campos vacíos	Se muestran los campos para modificar una propiedad	V Trigger	I vacío	El sistema muestra un mensaje de error indicando que existen campos vacíos incorrectos	1. Se muestran los campos para gestionar las propiedades a.) Propiedad b.) Valor 2. Modificar los valores si se desea modificar la propiedad
EC 5.2 Eliminar propiedad	Se muestran las propiedades adicionadas	NA	NA	El sistema elimina la propiedad	3. Seleccionar una propiedad si se desea eliminar 3.1 Seleccionar la opción Eliminar

Tabla 29 Caso de prueba al RF 5 Gestionar propiedad de elemento de *BPMN* 1.0.

Anexo 8 No conformidades de la primera iteración.

Iteración 1					
Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección de errores	Importancia
RF 1	1	En el árbol de representación no se muestran todos los elementos del diagrama	Cargar fichero *XMI*	Al cargar el fichero	Significativa
RF1	2	En el árbol de representación existen elementos que no tienen nombre	Cargar fichero *XMI*	Al cargar el fichero	Significativa
RF1	3	Al cargar varios ficheros *XMI* no se limpia el árbol de representación	Cargar fichero *XMI*	Al cargar el fichero	No significativa
RF 2	4	Al seleccionar un elemento de *BPMN* 1.0 se borra el nombre del elemento	Gestionar elemento de mapeo	Al seleccionar un elemento de *BPMN* 1.0	No significativa
RF 2	5	Al arrastrar y soltar una actividad de *WWF* se muestra otra actividad	Gestionar elemento de mapeo	Al arrastrar y soltar una actividad de *WWF* 3.5	Significativa
RF 2	6	Al seleccionar la opción modificar seleccionado y no	Gestionar elemento de mapeo	Al seleccionar la opción modificar seleccionado	No significativa

		seleccionar un elemento de mapeo se muestra elimina de la lista el primero			
RF 2	7	Al seleccionar la opción eliminar seleccionado se elimina el último de la lista	Gestionar elemento de mapeo	Al seleccionar la opción eliminar seleccionado	No significativa
RF 2	8	No se valida que se adicione un elemento de mapeo si actividad de *WWF* o *Bison Framework* equivalente	Gestionar elemento de mapeo	Al adicionar un elemento de mapeo	Significativa
RF 3	9	No se valida que no estén seleccionado las dos opciones del configuración	Generar *workflow*	Al seleccionar la opción generar *workflow*	No significativa
RF 3	10	El árbol de representación de *WWF* muestra elementos sin nombre	Generar *workflow*	Al seleccionar la opción generar *workflow*	No significativa
RF 5	11	No se validan que existan campos vacíos	Gestionar propiedad de elemento de *BPMN* 1.0	Al adicionar o modificar una propiedad	No significativa

RF 5	12	No se permite eliminar una propiedad	Gestionar propiedad de elemento de *BPMN* 1.0	Al eliminar una propiedad	Significativa
RF 5	13	Al introducir números en el campo de propiedad se deshabilitan todos los campos	Gestionar propiedad de elemento de *BPMN* 1.0	Al adicionar o modificar una propiedad	No significativa
RF 6	14	Palabras con errores ortográficos (componente)	Adicionar componente del *Bison Framework*.	Al mostrarse la interfaz	No significativa
RF 6	15	Al seleccionar la opción cancelar se limpian los campos pero no se cierra la interfaz	Adicionar componente del *Bison Framework*.	Al seleccionar la opción Cancelar	No significativa
RF 6	16	Al seleccionar la opción Aceptar no se limpian los campos	Adicionar componente del *Bison framework*.	Al seleccionar la opción Aceptar	No significativa
RF 6	17	Al adicionar un nuevo componente del *Bison Framework* no se muestra en el listado del Bison	Adicionar componente del *Bison Framework*.	Al cerrar la interfaz	No significativa
RF 6	18	No se valida que se queden	Adicionar componente del	Al adicionar un nuevo	No significativa

		campos vacíos	*Bison Framework*.	componente	

Tabla 30 No conformidades de la primera iteración.

Anexo 9 Ejemplo de transformación de *BPMN* 1.0 a *WWF* 3.5.

En el ejemplo que se muestra a continuación, se observa un diagrama de proceso de negocio perteneciente al SUIN.

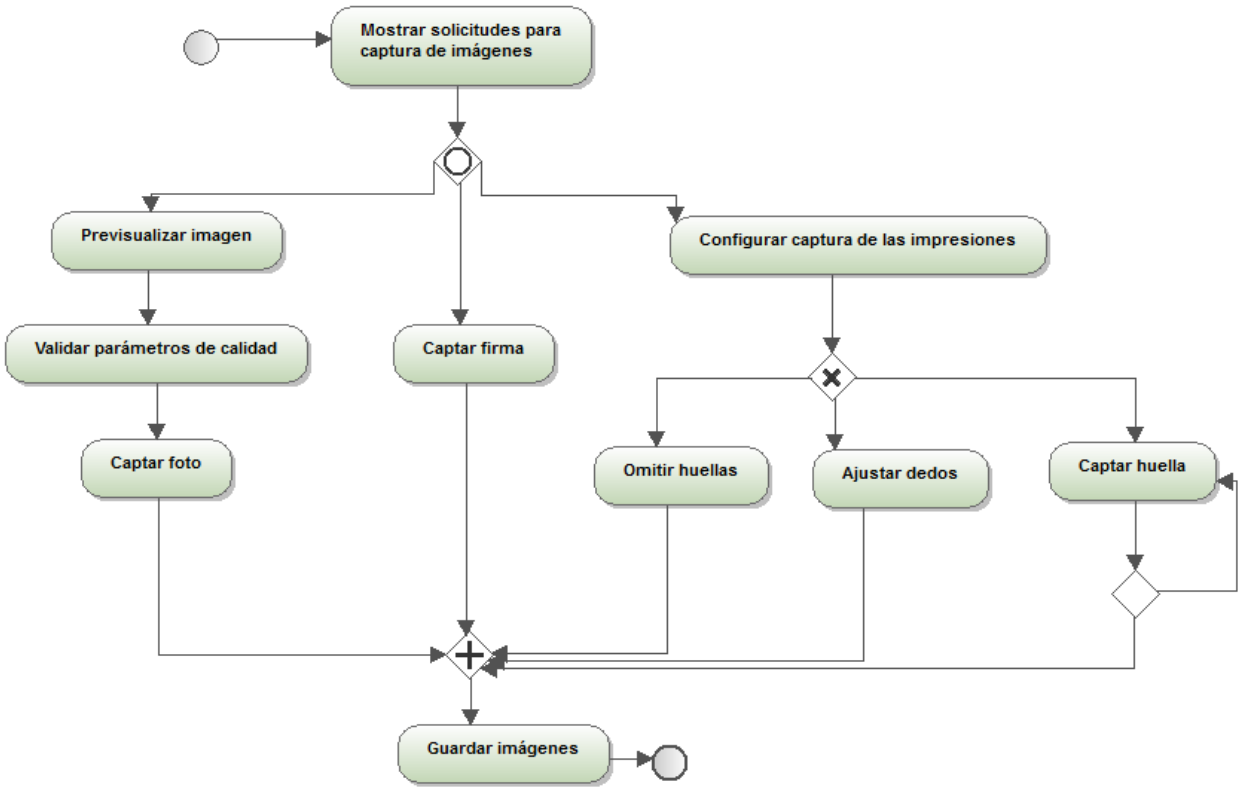


Figura 33 Diagrama de proceso de negocio captación de imagen.

En la tabla que se muestra a continuación se muestran los patrones de control de flujo identificados en el diagrama.

Patrón	Representación del patrón
Secuencia	

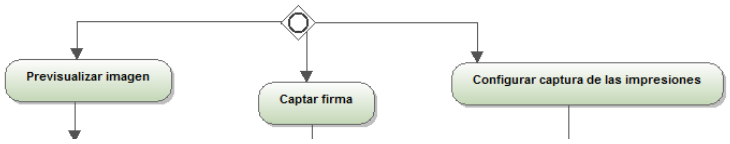
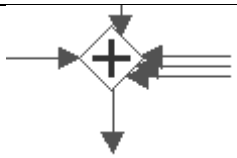
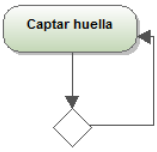
Múltiple elección	
Elección exclusiva	
Sincronización	
Ciclo arbitrario	

Tabla 31 Representación de patrones de control de flujo de captación de imagen.

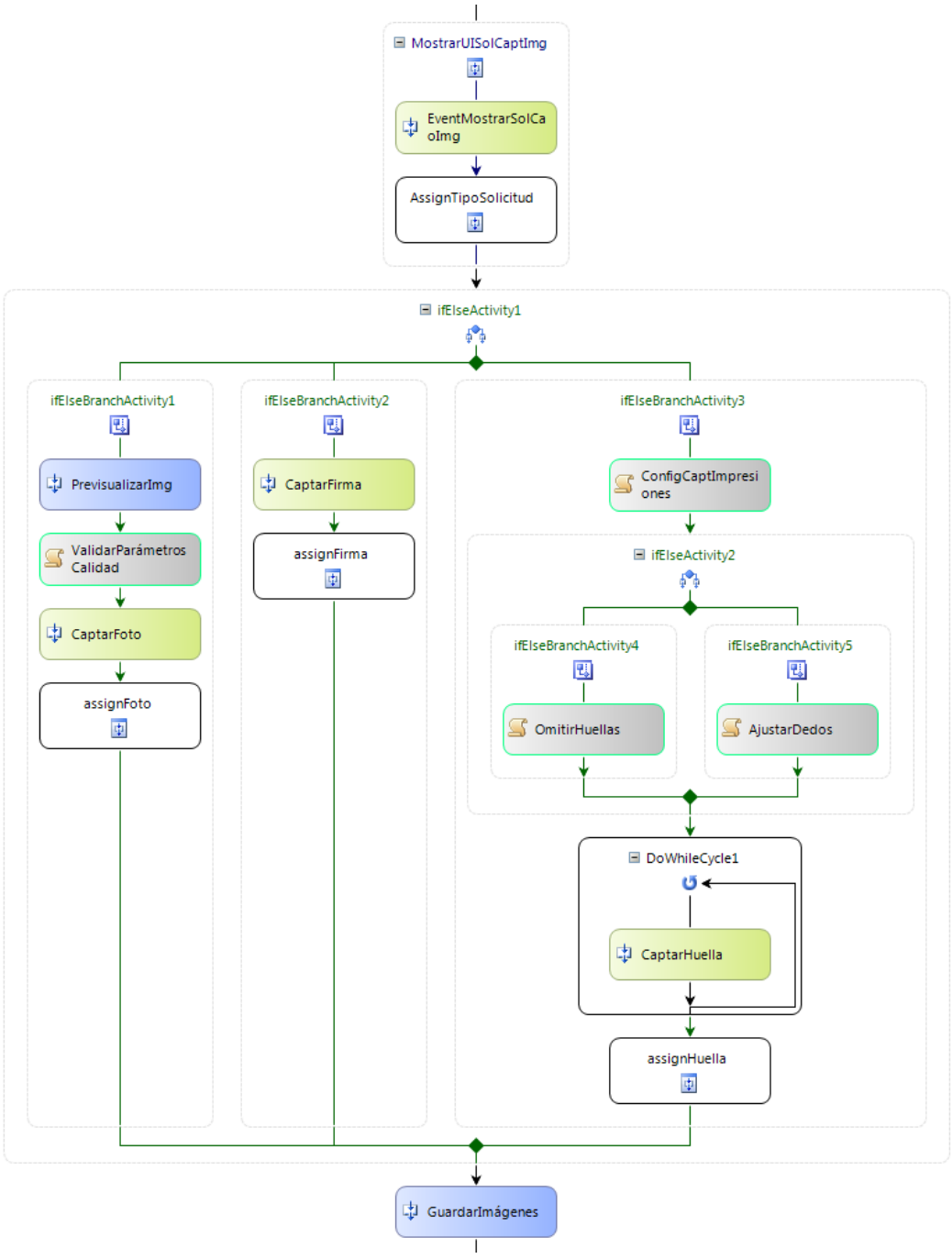


Figura 34 *Workflow* de captación de imagen implementado por un desarrollador.

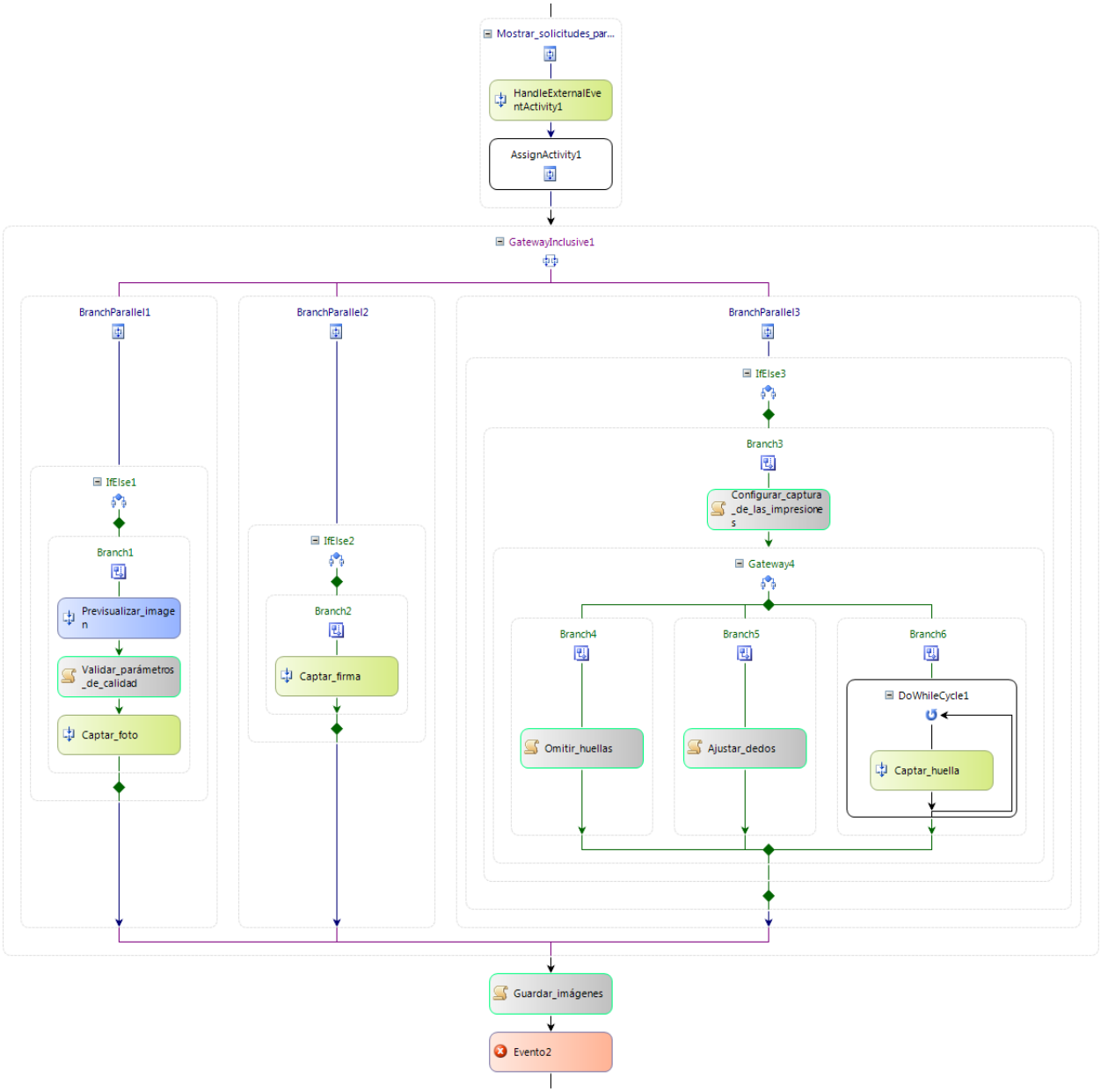


Figura 35 *Workflow* de captación de imagen generado por la herramienta.

Analizando el diagrama de proceso de negocio y el *workflow* implementado por el desarrollador, se observan a simple vista una serie de divergencias con respecto a los patrones representados entre ambos

modelos, infiriéndose un 30% aproximado de incompatibilidades entre el diagrama de proceso de negocio y su implementación. Dentro de las incompatibilidades detectadas se encuentran:

- no se representa el patrón de control de flujo múltiple elección.
- el patrón elección exclusiva no está representado correctamente, puesto que el ciclo queda omitido.

Sin embargo, con el uso de la herramienta se reducen estas incompatibilidades en un 0%. Es importante aclarar que para que exista una correcta transformación del modelo de proceso de negocio a su equivalente flujo de trabajo de la tecnología *WWF* 3.5, es necesario que a la hora de modelar los procesos de negocio se respeten las pautas de diseño requeridas por la herramienta. Por lo que mientras más simple y legible sin importar que tan expresivo sea la descripción de los procesos de negocio, menor será el porciento de incompatibilidades que se detectará con el uso de la herramienta.

El cálculo de porciento de incompatibilidades es realizado a través de la regla de tres que se muestra en la figura a continuación:

$$\%I = \frac{P * 100}{T}$$

Figura 36 Ecuación para realizar el cálculo de incompatibilidades.

%I: por ciento de incompatibilidades a calcular.

P: cantidad de patrones que son omitidos en la representación del *workflow.*

T: total de patrones representados en el diagrama de proceso de negocio.

Analizando el tiempo de implementación del *workflow*, limitándose a las tareas de diseño de forma manual, al desarrollador le tomó 45 minutos diseñar el esqueleto del *workflow*. Sin embargo, mediante el uso de la herramienta se redujo en un 80% el tiempo de implementación, llevándole tan solo 9 minutos al desarrollador en obtener el *workflow* deseado de acorde a sus especificaciones.