Universidad del Bío-Bío
Facultad de Ciencias Empresariales
Departamento de Ciencias de la Computación y Tecnologías de Información

Identificación y uso de las herramientas tecnológicas que permitan obtener Procesos de Negocio desde Sistemas de Información Heredados

Gastón Patricio Márquez Ortega

Memoria para optar al título de
Ingeniero Civil en Informática

Enero 2011

Universidad del Bío-Bío
Facultad de Ciencias Empresariales
Departamento de Ciencias de la Computación y Tecnologías de Información

Identificación y uso de las herramientas tecnológicas que permitan obtener Procesos de Negocio desde Sistemas de Información Heredados

Gastón Patricio Márquez Ortega

PROFESOR GUÍA	:	SR. ALFONSO RODRÍGUEZ RÍOS
PROFESOR INFORMANTE	:	SR. GILBERTO GUTIÉRREZ RETAMAL
NOTA FINAL EXAMEN TÍTULO	:	

Memoria para optar al título de
Ingeniero Civil en Informática

Chillán

Enero 2011

AGRADECIMIENTOS

Deseo mostrar mi más sincero agradecimiento:

Al Sr. Alfonso Rodríguez Ríos, profesor guía. Para mí fue un honor haber trabajado esta memoria bajo su dirección y le estaré siempre muy agradecido porque ha dedicado su valioso tiempo a ello. Por otro lado, agradezco el haber despertado en mí el interés por la investigación.

A mis padres Gastón y Carmen y mi hermano Francisco, ya que ellos han sido un pilar fundamental en mi formación como alumno universitario. Su apoyo constante y fortalezas infinitas hicieron que nunca me rindiera, incluso en tiempos de adversidad. Estaré eternamente agradecido.

A Michelle, por su constante apoyo, alegría y sabias palabras. Gracias a ti todo esto fue posible, ya que tú me enseñaste a ser un hombre mejor. Eres un ángel.

Muchas gracias a todos.

Índice

Índice de figuras

Índice de tablas

RESUMEN

Las organizaciones actualmente están prestando atención a los Procesos de Negocio (BP, Business Process) y éstas reconocen que los BP son un recurso importante para lograr una posición competitiva en el mercado. Al mismo tiempo, dichas organizaciones poseen Sistemas de Información Heredados (LIS, Legacy Information System) que resultan ser importantes para su correcto funcionamiento. Los LIS, además de su composición técnica, abarcan elementos de la estructura organizacional, estrategias, procesos y flujos de trabajo, con los cuales se pueden tomar decisiones sin perder la información almacenada. Pero, en general, esta información no se encuentra explícitamente disponible para los analistas de negocio, lo cual es un problema para la organización. Esta memoria presenta un estudio en que se muestra una forma de obtener modelos de BP a partir de un LIS usando tecnologías disponibles.

ABSTRACT

Organizations today are paying attention to the Business Process (BP) and they recognize that the BP is an important resource to achieve a competitive position in the market. At the same time, these organizations have Legacy Information Systems (LIS) which prove to be important for proper operation. The LIS in addition to its technical composition, comprise elements of organizational structure, strategies, processes and workflows with which decisions can be made without losing the stored information. But overall, this information is not explicitly available to business analysts, which is a problem for the organization. This memory presents a study that shows one way to obtain models of BP from a LIS using available technologies.

Capítulo 1

Introducción

Las organizaciones de hoy participan en mercados cada vez más complejos, cambiantes y competitivos. Esta situación les obliga a estar pendientes de todos los procesos que se llevan a cabo dentro de ellas. En los últimos años han ganado importancia los procesos de negocio pues han permitido que las empresas puedan visualizar la forma en que llevan a cabo sus actividades. Por otro parte, la mayoría de las organizaciones se encuentran fuertemente apalancadas a las Tecnologías de Información y Comunicaciones (TICs), pasando a ser éstas un factor crítico para el éxito de un negocio. No obstante, las TICs tienden a tener un grado de obsolescencia que obliga a su constante actualización. Muchas empresas cuentan con Sistemas de información que están soportados por tecnología obsoleta que los rigidiza. Sin embargo, estos sistemas son muy importantes dentro de una organización puesto que en muchas ocasiones conforman la columna vertebral del flujo de información en una organización y son el principal vehículo para la consolidación de información acerca del negocio. Entonces se entra en un debate ¿Adquirir nuevos Sistemas de Información o modernizar el Software heredado utilizando el enfoque de ADM? [Compute-rs, 2010] [Khusidman & Ulrich, 2007]Muchas empresas en la actualidad se enfrentan a esta interrogante debido a que las aplicaciones de negocios nuevos

están emergiendo, entonces a partir de esto nacen otras pregunta: ¿Cuándo es necesario adquirir una aplicación nueva empresa? ¿Es mejor para modernizar un Sistema heredado o establecer nuevos Sistemas de negocio en su lugar? La adquisición de tecnología nueva para la empresa es una de las opciones importantes hechas por diversos sectores para seguir siendo competitivos. Todas estas interrogantes las deben resolver las personas que están a cargo de la administración de las TIC en las organizaciones. Adquirir un nuevo Sistema puede ser una buena opción, ya que viene con la última tecnología y nuevas aplicaciones. Pero, ¿Se debe desconocer la información que ya está almacenada en los Sistemas anteriores? ¿La integración de la organización con el nuevo Sistema será exitosa? ¿Cuánto tiempo y dinero costará la integración? Según la encuesta relacionada con la modernización de Software hecha por una empresa llamada Software AG [ComputerWorld, 2006] los resultados obtenidos muestran que están "*muy*" o "*extremadamente*" preocupados por la modernización de Software heredado que ya está en la empresa. Por lo anterior, la mejor solución es la modernización de aplicaciones heredadas, porque siguen siendo compatibles con las exigencias de negocio actual, pero pueden aprovechar la tecnología actual. La información financiera y datos de la empresa, independientemente del número de años en el archivo, aún representan la información del negocio principal de la compañía. La modernización de Sistemas heredados no sólo ayudará a mantener procesos de negocio importantes y la información financiera, sino que también pueden contribuir a proporcionar una opción más ágil para una empresa en expansión para satisfacer las demandas que son siempre cambiantes en su industria y mercado.

Considerando lo anterior, parece evidente que uno de los activos más importante de las organizaciones es la información; para algunas incluso puede ser crítico

y esencial para su funcionamiento. En consecuencia es necesario que los datos sean mantenidos a lo largo del tiempo, inclusive si los Sistemas de Información evolucionan. Este problema lo viven muchas organizaciones de distintas áreas, que poseen Sistemas que perduran en el tiempo. La evolución de estos Sistemas se puede realizar mediante estrategias como reingeniería, reemplazo del Sistema o alguna solución híbrida de las anteriores, sin embargo, independiente de la solución adoptada, los datos del Sistema heredado deben ser conservados, aunque no necesariamente con la misma estructura de datos, que puede resultar obsoleta para las nuevas aplicaciones. Entonces, estos Sistemas pueden ser actualizados usando propuestas como la modernización dirigida por la arquitectura (ADM, *Architecture Driven Modernization*) que está orientada a la obtención de nuevas aplicaciones de Software a partir del Software existente [Khusidman & Ulrich, 2007] . En este contexto existen propuestas a través de las cuales se pretende alcanzar una descripción de procesos de negocio tomando como referencia un Sistema de información heredado [Rodríguez & Caro, 2009, Rodríguez & Caro, 2008]. En este proceso se debe contar con una descripción completa de los procesos de negocio de la empresa. Para ello es necesario contar con un lenguaje para obtener los modelos y así lograr el entendimiento de los BP para luego poder adaptarlos y mejorarlos. En la última década nace BPMN (*Business Process Modeling Notation*) [OMG, 2010a] y, por otro lado, el lenguaje UML (*Unified Modeling Language*) [OMG, 2010b] es mejorado para poder representar procesos de negocio. Ambos, la notación y el lenguaje, representan una valiosa herramienta para la descripción de modelos de procesos de negocio. En este trabajo se aborda la implementación, usando la tecnología disponible, de la propuesta presentada en [Rodríguez & Caro, 2009] [Rodríguez & Caro, 2008] en que se define el marco de trabajo para la obtención

de procesos de negocio desde Sistemas de información heredados. En particular, en este trabajo se ha centrado en la creación especificaciones de procesos de negocio descritos usando BPMN a partir de Software heredado, específicamente un conjunto de programas escritos en CoBOL. Aunque existen trabajos relacionados con este planteamiento estos no abordan las tecnologías que llevan a cabo transformaciones [Estes, 2006]o no usan CoBOL como punto de partida [Pérez-Castillo *et al.* , 2010]. Para el trabajo se hizo una revisión sistemática de la literatura, en la que se aplicaron directrices propuestas por [Kitchenham, 2007]. Esto permitirá conectar los trabajos relacionados con las transformaciones de modelos desde LIS hacia BP o de pasos intermedios de transformación y se pondrá especial atención en las herramientas utilizadas. A su vez, se utilizaron casos de estudios [Runeson & Host, 2008], para probar los resultados obtenidos en lel trabajo.

El resto de la memoria donde se muestra el trabajo, comienza definiendo los conceptos relacionados al tema de investigación tales como: Procesos de Negocio, Sistema Heredado, Modelado de Procesos de Negocio, Transformación de modelos y la propuesta LIS2BP. Definido y explicado lo anterior, en el siguiente capítulo de la memoria se detallará la tecnología disponible que sirve para extraer modelos BP a partir de un LIS. Lo anterior implicará que en el siguiente capítulo se hable de los resultados de la investigación. Como tema adicional, se mencionará la propuesta de la herramienta `LIS2BP-TOOL` la cual genera modelos de BP a partir un archivo escrito en CoBOL y, además, se detallará cómo funciona la herramienta. Finalmente, en el último capítulo, se detallarán las conclusiones correspondientes al trabajo.

Capítulo 2

Conceptos relacionados

En este capítulo se presentará en forma resumida una breve descripción de los aspectos más importantes de los conceptos relacionados con el trabajo de investigación.

2.1. Proceso de Negocio

Davenport [Davenport, 1993] define un proceso de negocio como "... *un programa estructurado, un conjunto de actividades diseñadas para un producto específico que está dirigido a un cliente o mercado especifico. Un proceso es, entonces, un orden específico de actividades de trabajo a través del tiempo y el espacio, con un comienzo y un fin, y que tiene claramente definidas las entradas y salidas...*". Por su parte, Rummler & Brache [Rummler & Brache, 1995] aportan que "... *un proceso de negocio es una serie de pasos diseñados para producir un producto o servicio. La mayoría de los procesos (...) son multifuncionales, que abarca el 'espacio blanco' entre las actividades en el organigrama. Algunos procesos muestran como un resultado de un producto o servicio que se recibe por parte del cliente externo de una organización...*" Según ISO-9001 [ISO, 2000] proceso es una actividad que utiliza recursos, y que se gestiona con el fin de permitir que los elementos

de entrada se transformen en resultados [Aguilar-Savén, 2004]. En particular, las primeras definiciones de procesos de negocio permiten identificar un conjunto de actividades o procedimientos que cumplen objetivos específicos o metas de largo plazo, en el contexto de una estructura organizacional, definiendo roles funcionales y relaciones [WfMC, 1999]. Lo esencial de un BP es que está vinculado con las empresas y que en conjunto definen la forma en que ellas alcanzan sus objetivos. Dicho de otra forma, es una parte del negocio donde se describen las funciones del mismo y que involucra recursos que son utilizados, transformados o producidos. Por lo mencionado anteriormente, resulta esencial contar con una forma de representar gráficamente los BP de modo que sean comprensibles para quienes están involucrados en el negocio. En la actualidad, las notaciones más utilizadas para la representación de BP son el Diagrama de Actividad de UML 2.0 (abreviado como UML 2.0-AD) y el Diagrama de Procesos de Negocio de BPMN (abreviado como BPMNBPD). Ello ha permitido mejorar la definición y uso de los BP por parte de analistas de negocio y analistas de sistemas.

2.2. Sistema Heredado

La denominación de Sistema de Información Heredado (LIS, *Legacy Information System*), corresponde a cualquier Sistema de Información que se resiste significativamente a cambios y modificaciones [Caro *et al.* , 2002]. Por otro lado, estos sistemas son muy importantes dentro de una organización, esto significa que si alguno de ellos falla o se detiene, traerá graves consecuencias al funcionamiento de la empresa. Por otro lado, los LIS son considerados complicados por diversos motivos. Dichos sistemas a menudo operan en hadware obsoletos y lentos, cuyo mantenimiento tiene elevados costos y son difíciles de actualizar por falta de componentes adecuados

o de mantenimiento. Las nuevas necesidades de negocios de la empresa por lo general reemplazan equipos y maquinaria por sistemas más modernos, como se aprecia en la Figura 2.1.

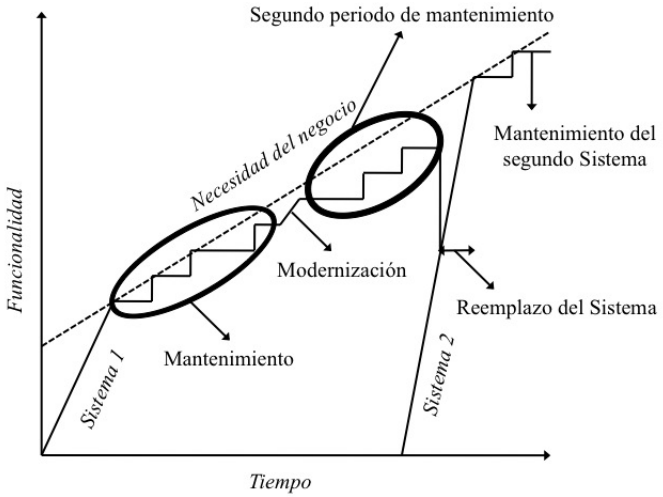


Figura 2.1: Ciclo de vida de un Sistema de Información [Seacord *et al.* , 2003]

En la Figura 2.1 se puede observar que las necesidades de negocio son una recta ascendente, ya que cada vez las exigencias del entorno del negocio aumentan en función del tiempo. Los sistemas (en la Figura se denomina *Sistema 1*) que son parte del negocio se crean desde cero hasta llegar a un punto donde satisfacen (temporalmente o en primera instancia) las necesidades del negocio. En ese punto, el sistema debe ser mantenido periodicamente para cubrir todos los requerimientos que la empresa necesita y llegar a la modernización temporal del Software. Esta modernización temporal es para adaptar el sistema a las nuevas tecnologías que aparecen en el tiempo. Ahora bien, cuando el sistema no es capaz de cubrir todas las necesidades, la empresa tiene la primera opción de crear un nuevo sistema (*Sistema*

2) desde 0 con la tecnología adecuada para cubrir las nuevas necesidades del negocio o la segunda opción que consisten en reemplazar el sistema por otro desde un cierto punto, que en la gráfica se denota como *Reemplazo del Sistema*, con el objetivo de no perder la información ya obtenida del *Sistema 1.* Según [Seacord *et al.* , 2003], la mejor opción para una empresa es la segunda, ya que no se puede perder la información del pasado, debido a que es fundamental para la organización.

Las principales características de los LIS son [Brodie & Stonebraker, 1995] [Bisbal, 1997] [Gold, 1998] :

- Típicamente son grandes, con millones de líneas de código
- Son antiguos, más de 8 años desde su construcción
- Están escritos en un lenguaje heredado (CoBOL, assembler, etc.)
- Se basan en bases de datos heredadas o archivos planos
- Generalmente funcionan en hardware obsoleto que es lento y caro de mantener
- Son autónomos (independientes de otras aplicaciones) generalmente, son difíciles de comprender y no existe documentación suficiente o apropiada acerca de ellos.
- Su mantención implica un alto costo para la organización
- Que generalmente cumplen una "misión-crítica" dentro de la organización

En justificación a lo mostrado en la Figura 2.1, desechar un LIS y reemplazarlo con hardware y software moderno conduce a riesgos de negocio significativos. Reemplazar un sistema heredado es una estrategia de negocios arriesgada por varias razones [Seacord *et al.* , 2003]:

1. Es poco probable que exista una especificación completa de los sistemas heredados. Si existe una especificación, no es probable que tenga los detalles de todos los cambios hechos en el sistema. Por lo tanto, no existe ninguna forma directa de especificar un nuevo sistema que sea funcionalmente idéntico al sistema que se utiliza.

2. Los BP y las formas en que los sistemas heredados operan a menudo están fuertemente entrelazados, como se puede observar en la Figura 2. Estos procesos se diseñaron para aprovechar los servicios del software y evitar sus debilidades. Si el sistema se reemplaza, estos procesos también tendrán que cambiar, con costos y consecuencias impredecibles.

3. Las reglas de negocio importantes están en el software y no suelen estar en ningún documento de la empresa. Una regla de negocio es una restricción que aparece en algunas funciones del negocio y romper esa restricción puede tener consecuencias impredecibles para éste, como se muestra en la Figura 2.2.

En la Figura 2.2 se presenta los componentes que poseen los LIS según [Sommerville, 2005]. Uno de los componentes mostrados son los Procesos de Negocio (punto en el cual se enfoca el trabajo) que son restringidos por las Políticas/Reglas del Negocio.

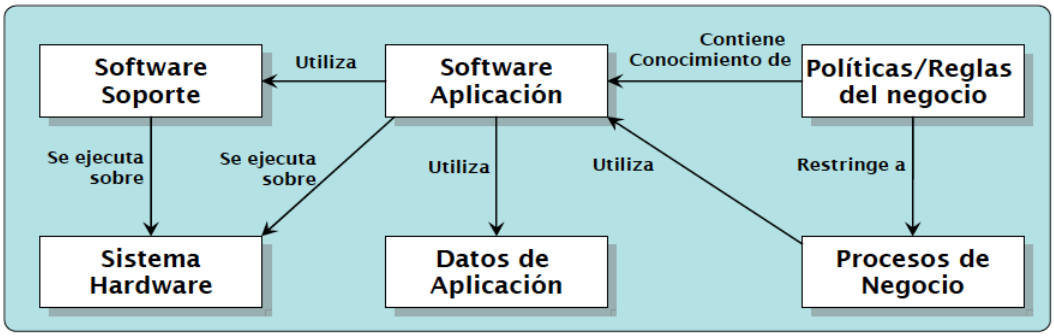


Figura 2.2: Componentes de un LIS [Sommerville, 2005]

2.2.1. Modelado de Procesos de Negocio

La notación Business Process Modeling Notation (BPMN) permite representar modelos de procesos de negocio. Su principal objetivo es proveer de una notación que sea rápida y comprensible por toda la gente de negocio, desde el analista de negocio, pasando por los desarrolladores técnicos responsables de implementar la tecnología que llevará a cabo dichos procesos, hasta las personas de negocio que gestionarán y monitorizarán esos procesos [Rodríguez & Caro, 2009] [Sánchez, 2008]. Si se considera que la gestión se realiza a través de todo el ciclo de vida de los BP, es indudable que el modelamiento tiene un papel significativo. Asimismo, un diagrama en BPMN puede ser transformado en un código ejecutable automáticamente, sin la necesidad de programación. De esta forma, el analista de negocio puede definir, diseñar y generar una solución a sus procesos. Al realizar una secuencia de acciones, BPMN ofrece a los analistas de negocio una forma consistente con su manera de trabajar [Emprendedores, 2008]. Para realizar el diseño, esta notación ofrece un único diagrama con enlaces, los cuales contemplan desde la concepción general hasta la generación del código ejecutable de los procesos, pasando por diagramas de detalle,

tratamiento de excepciones y otros. Este diagrama es llamado Business Process Diagram (BPD).

Las características que posee BPMN son [BizAgi, 2010]:

- BPMN es un estándar internacional de modelado de procesos aceptado por la comunidad.
- BPMN es independiente de cualquier metodología de modelado de procesos.
- BPMN crea un puente estandarizado para disminuir la brecha entre los procesos de negocio y la implementación de estos.
- BPMN permite modelar los procesos de una manera unificada y estandarizada permitiendo un entendimiento a todas las personas de una organización.

A su vez, BPMN se compone de cuatro elementos básicos los cuales son [BizAgi, 2010] [OMG, 2010a] [Sánchez, 2008]:

1. ***Objetos de Flujo:*** son los principales elementos gráficos que definen el comportamiento de los procesos. Dentro de los objetos de Flujo encontramos:

   a) *Eventos*: corresponde a lo que sucede durante el curso de un proceso de negocio, afectan el flujo del proceso y usualmente tienen una causa y un resultado, se encuentran clasificados en 3 tipos: Evento de Inicio, Evento Intermedio y Evento de Fin.

   b) *Actividades:* estas representan el trabajo que es ejecutado dentro de un BP.

   c) *Compuertas:* Son elementos del modelado que se utilizan para controlar la divergencia y la convergencia del flujo. Los 5 tipos de compuertas son:

1) Compuerta Exclusiva
2) Compuerta Basada en eventos
3) Compuerta Paralela
4) Compuerta Inclusiva
5) Compuerta Compleja

2. ***Objetos de conexión:*** Son los elementos usados para conectar dos objetos del flujo dentro de un proceso. Existen 3 tipos de objetos de conexión:

   a) Líneas de Secuencia
   b) Asociaciones
   c) Líneas de Mensaje

3. ***Canales:*** son elementos utilizados para organizar las actividades del flujo en diferentes categorías visuales que representan áreas funcionales, roles o responsabilidades. Existen dos tipos de canales:

   a) Pools: representa a los principales participantes en un proceso. Un *pool* contiene uno o más *lanes*. Un *pool* puede ser abierto (es decir, mostrando los detalles internos) cuando se representa como un rectángulo grande que muestre uno o más *lanes*.
   b) Lanes: se utiliza para organizar y clasificar las actividades en un grupo de acuerdo a la función o rol, y se muestra como un rectángulo que se extiende la anchura o la altura del *pool*. Un *lane* contiene los flujos entre objetos, conexión de objetos y artefactos.

4. ***Artefactos:*** Los artefactos son usados para proveer información adicional sobre el proceso. Existen 3 tipos:

   a) Objetos de Datos: muestran al lector qué datos son necesarios o requeridos para una actividad.

   b) Grupos: los grupos se utilizan para agrupar distintas actividades.

   c) Anotaciones: se usa para explicar mejor el modelo con apuntes.

2.2.2. Transformación de modelos

La arquitectura dirigida por modelos (MDA, *Model Driven Architecture*) es un marco de trabajo que ha sido definido para el desarrollo de software. MDA se encuentra en el ámbito de MDE (*Model Driven Engineering*) y su principal objetivo es permitir la creación de modelos que son totalmente independientes de la implementación tecnológica [Bézivin, 2004]. Uno de los principales objetivos de MDA es separar el diseño de la arquitectura y de las tecnologías de desarrollo, facilitando que el diseño y la arquitectura puedan ser modificados independientemente. El diseño alberga los requerimientos funcionales mientras que la arquitectura proporciona la infraestructura a través de la cual se hacen efectivos los requerimientos no funcionales. Como se muestra en la Figura 2.3, MDA se asegura de que el modelo independiente de la plataforma (PIM), el cual representa un diseño conceptual que concreta los requerimientos funcionales, subsiste a los cambios que se produzcan en las tecnologías de desarrollo y en las arquitecturas software [Harmon, 2004].

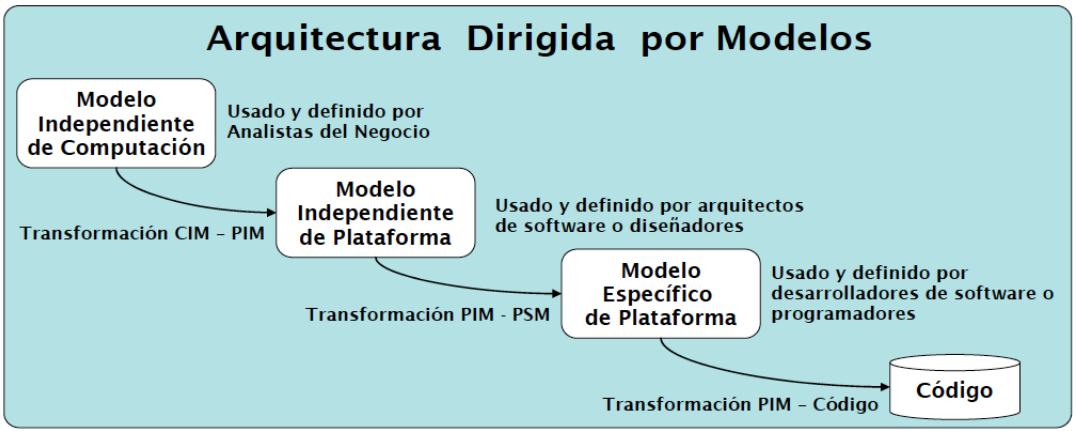


Figura 2.3: Secuencia MDA [Harmon, 2004]

De particular importancia en MDA es la noción de transformación de modelos. Uno de los estándares definidos para la transformación de modelos se denomina QVT (Query/View/Transformation) [QVT, 2005]. La transformación del modelo es también importante para el enfoque de MDA, esto permitirá transformar un modelo Ma a otro modelo Mb independiente de que si sus respectivos meta-modelos MMa y MMb son idénticos o diferentes bajo un meta lenguaje llamado MOF (*Meta Object Facility*) [Bézivin, 2004]. La explicación anterior se resumen en la Figura 2.4.

2.2.3. La propuesta LIS2BP [Rodríguez & Caro, 2009]

LIS2BP se sustenta en los marcos de trabajo de la arquitectura dirigida por modelos (MDA) y la modernización dirigida por la arquitectura (ADM). El primero, entrega el marco de referencia para la especificación de las transformaciones desde el LIS hacia modelos independientes de computación, en el caso de este trabajo está orientado hacia la representación de modelos de los procesos de negocio embebidos en el LIS. El segundo, provee el marco de referencia para la aplicación de ingeniería

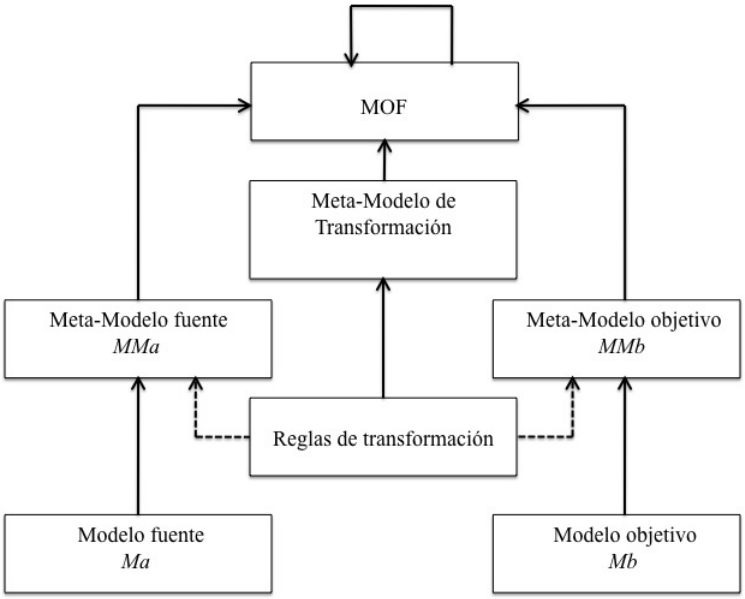


Figura 2.4: Transformación de modelos [Bézivin, 2004]

inversa basada en la arquitectura existente, orientada a lograr definiciones del negocio. La esencia del marco de trabajo LIS2BP la constituye la transformación de la información que puede ser obtenida desde el sistema de información heredado en uno o más modelos de BP. Esta transformación es realizada siguiendo el método M-LIS2BP (ver Figura 2.5), que permite la identificación de todos los componentes de información del LIS y determina su equivalencia en un proceso de negocio. Esto último, se lleva a cabo mediante la aplicación de un conjunto de reglas que transforman los componentes de información en los elementos que componen el proceso de negocio.

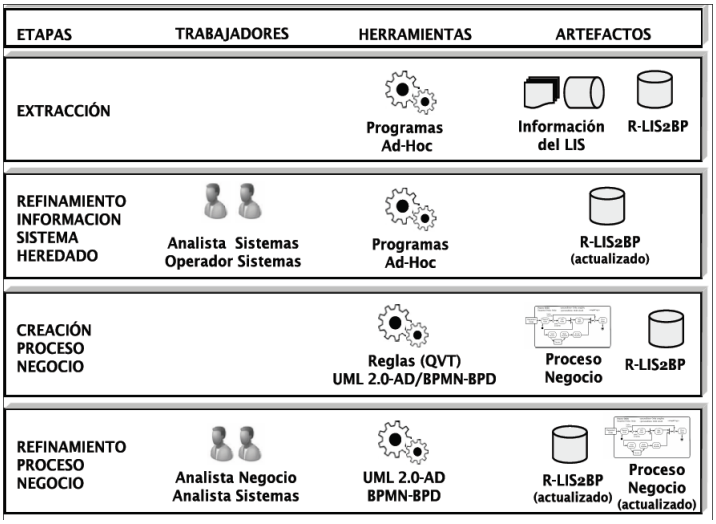


Figura 2.5: Propuesta M-LIS2BP [Rodríguez & Caro, 2009]

El trabajo se centra en las etapas de Extracción y Refinamiento del Sistema Heredado [Rodríguez & Caro, 2009]:

- En la etapa de Extracción, las tareas se realizan en forma automática por lo que no se requiere la intervención de trabajadores. Las herramientas que se utilizan en esta etapa corresponden a un conjunto de programas diseñados ad-hoc (los cuales serán descritos en las siguientes secciones) los que permiten extraer la información disponible en el sistema de información heredado. Estos programas actúan sobre archivos residentes en el computador desde los cuales se obtiene, por ejemplo, los nombres de programas, los nombres y las estructuras de archivos, las secuencias de ejecución de programas (si es que se cuenta con los programas fuentes), los formatos de entradas y salidas, entre otros. Con esta información se hace un llenado inicial del repositorio LIS, el artefacto generado en esta etapa, el cual contendrá la información extraída en forma automática.

- En la etapa de Refinamiento del Sistema Heredado, la principal tarea corresponde a los trabajadores analista de sistemas y operador de sistemas. Ellos, en su calidad de usuarios conocedores del sistema, tienen la responsabilidad de completar la información acerca del LIS. Esta información es relevante, principalmente, en sistemas en los que no se cuenta con los programas fuentes. La información provista por los trabajadores servirá para actualizar el repositorio. También será muy útil la documentación disponible del sistema, ya que se puede obtener en forma directa desde allí alguno de los componentes de información del LIS.

Capítulo 3

Tecnología disponible

El objetivo principal de este trabajo es obtener procesos de negocio que puedan ser visualizados a través de BPMN usando la tecnología disponible. Puesto que se ha verificado que no es posible obtener gráficas de procesos de negocio en BPMN partiendo desde código escrito en CoBOL, el siguiente paso es verificar la existencia de tecnologías disponibles que permitieran obtener modelos BPMN. Dentro de estos archivos se pueden mencionar dos tipos: los archivos XPDL (XML Process Definition Language) [WfMC, 2008] y XML (Extensible Markup Language) [XML, 2010]. El primero de ellos, XPDL es un formato estandarizable para intercambiar definiciones de procesos de negocio, además, es un formato de archivo basado en XML que tiene la particularidad de ser usado para intercambiar modelos de procesos de negocio entre distintas herramientas de modelado. Además es un formato de archivo que muestra la representación gráfica del proceso. XPDL provee paquetes que corresponden a diagramas de procesos de negocio que utilizan XML como mecanismo de intercambio en función de lo cual XPDL puede importar/exportar funciones [WfMC, 2008] (como se ve en la Figura 3.1). A su vez, XPDL con BPMN son muy similares. De hecho, la similitud entre ambos es más fuerte que BPMN con BPEL4WS o BPML [White, 2003]. En la Tabla 3.1 se muestran algunas similitudes. Por otro lado, tenemos los

archivos XML [XML, 2010] que es un metalenguaje extensible de etiquetas el cual propone un estándar para el intercambio de información estructurada entre diferentes plataformas. XML se puede usar en bases de datos, editores de texto, hojas de cálculo, entre otras.

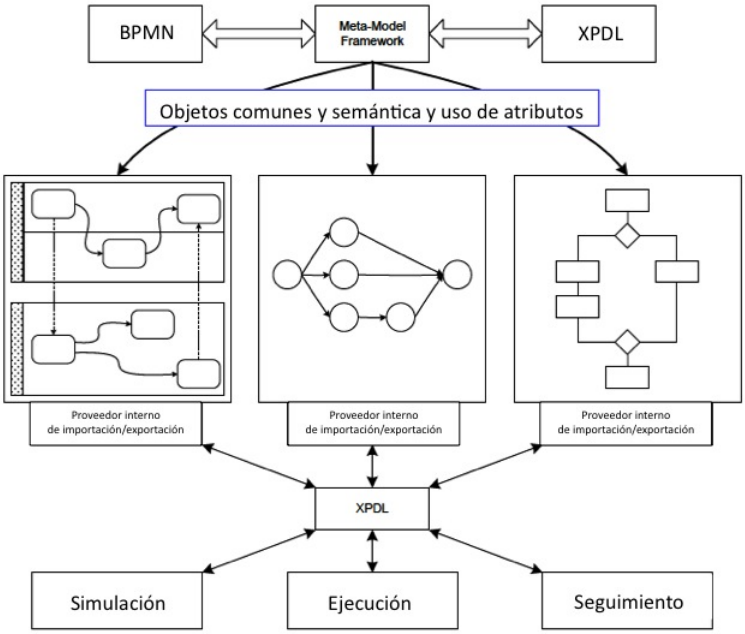


Figura 3.1: Proceso de importar/exportar de XPDL [WfMC, 2008]

A partir de lo anterior se ha dividido el problema a resolver en estudiar las herramientas que nos permitan obtener desde CoBOL archivos `.xpdl` y `.xml` y verificar la existencia de herramientas que nos permitan, partiendo los archivos `.xpdl` y `.xml`, obtener representaciones en BPMN. Ambos puntos son abordados en las secciones 3.1 y 3.2 respectivamente.

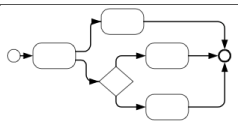
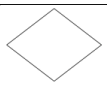
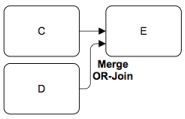
Objeto gráfico de BPMN	Transformación a XPDL
	<WorkflowProcess/>
	<Transition/>
 Tarea	<Activity> <Implementation> <Tool/> <Performer/> </Implementation> </Activities>
 Desición	<Activity> <Route/> <TransitionRestriction> <Split Type="XOR"/> </TransitionRestriction> </Activities> Combined with a: <Transition> <Condition/> <Transition>
	<Activity> <Implementation/> <TransitionRestriction> <Join Type="XOR"/> </TransitionRestriction> </Activities>

Tabla 3.1: Algunos objetos gráficos de BPMN y su transformación en XPDL [White, 2003]

3.1. Archivos `.xpdl`/`.xml`

Como ya se dijera, no existe registro de herramientas capaces de generar directamente modelos BPMN desde de un archivo CoBOL, sólo hay herramientas que hacen el trabajo de forma parcial. Por lo tanto, el trabajo se ha centrado en las herramientas que a partir de CoBOL generan archivos previos a la gráfica BPMN; estos tipo de archivos son los de extensión `.xpdl` o `.xml`. Estas herramientas son:

3.1.1. MoDisco (Model Discovery).

Es una herramienta proporcionada por Eclipse para trabajar con LIS [MoDisco, 2010] [Bruneliere & Madiot, 2010], en la Figura 3.2 se puede observar el proceso de MoDisco. Los LIS abarcan un gran números de técnicas haciendo que el desarrollo de una herramienta que haga frente a los cambios tecnológicos que necesite una organización sea tedioso y que consuma tiempo.

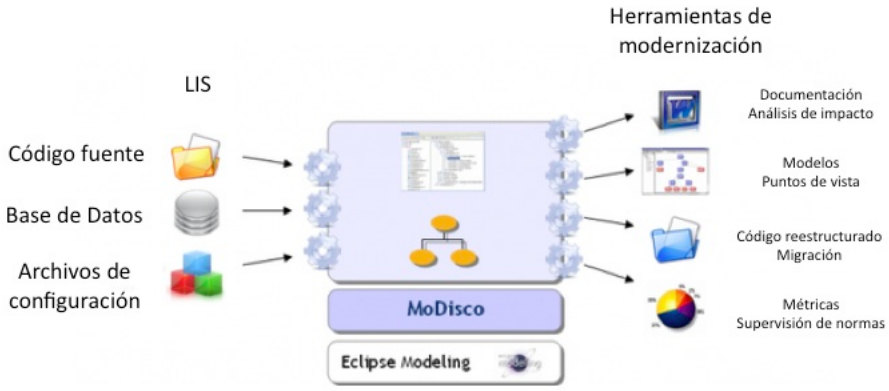


Figura 3.2: Proceso de MoDisco [MoDisco, 2010]

MoDisco proporciona marcos de trabajos extensibles para el desarrollo dirigido por modelos, lo cual implica que utiliza KDM (*Knowledge Discovery Metamodel*)

[KDM, 2010] para apoyar los casos de uso de los programas existentes, dentro de los cuales encontramos los siguientes:

- *Aseguramiento de la calidad:* verifica si un sistema cumple con las cualidades requeridas.
- *Documentación:* extracción de información de un sistema existente para ayudar a comprender el aspecto de éste (estructura, persistencia, flujo de datos, etc).
- *Mejora:* transformación de un sistema existente para integrar mejor las normas de codificación o patrones de diseño.
- *Migración:* transformación de un sistema existente para cambiar un componente, la estructura, el lenguaje, o su arquitectura.

Por otro lado, MoDisco tiene otras funcionalidades en la ayuda de obtención de información a partir de un LIS. A continuación se detallarán las funcionalidades:

- **KDM-to-UML2 Converter:** Esta herramienta consiste en convertir modelos de KDM en modelos basados en UML2 en el orden de integración de las herramientas de compilación de KDM con las herramientas de compilación de UML2.
- **Jar2UML Discovery Tool:** Esta herramienta trata sobre la generación de modelos UML desde API's de Java. En palabras simples, esta herramienta selecciona el archivo Jar como entrada y genera un archivo UML como salida.
- **Java Abstract Syntax Discovery Tool:** Esta herramienta trata del descubrimiento de un árbol de sintaxis abstracto (AST, *abstract syntax tree*) del archivo obtenido de la compilación del código java (`.class`)

La razones de haber considerado esta herramienta para el estudio son principalmente a que es la única herramienta que es libre e integra la Ingeniería dirigida por Modelos (MDE) para la extracción de modelos desde LIS. A pesar de que la herramienta aún no integra a CoBOL en su paquete en Eclipse, se esta desarrollando el framework de CoBOL para integrarlo en un futuro próximo.

3.1.2. XML Thunder.

Permite obtener XML/SOAP desde aplicaciones CoBOL o ANSI C [Canam, 2010]. XML Thunder genera correctamente archivos XML desde CoBOL, pero la transformación de CoBOL a documentos XML se considera una tarea difícil cuando se tienen que procesar las relaciones padre-hijo en escenarios donde hay varios programas que están indexados a uno solo. Debido a las limitaciones de CoBOL, si se trata de replicar esta relación como una estructura de datos CoBOL, se tendrá que declarar un tamaño máximo de hijos (tamaño máximo de programas indexados) en el mismo padre (una cláusula `OCCURS`). [Canam, 2004].

Por lo tanto, para solucionar el problema, XML Thunder proporciona dos tipos de controladores de XML, el primero, denominado Document-level XML Handler, se utiliza cuando se tiene un conjunto de datos bien definidos y se conoce el número máximo de elementos que se repiten que serán procesados en cualquier momento. El segundo controlador, llamando Node-level XML Handler se usa cuando no hay control sobre cuántas veces se repetirá la estructura XML en el documento XML de destino. Ésta opción puede ser la ideal cuando se necesita obtener archivos XML desde programas escritos en CoBOL en los casos donde existen relaciones padre-hijo [Canam, 2004].

La razón de la elección de esta herramientas, a pesar de que es un producto de

pago, es la única herramienta que genera archivos de extensión `.xml` enfocado en los procesos de negocio, por lo tanto, lo que se espera es que muestre el archivo `.xml` con toda la información posible sobre el programa escrito en CoBOL.

3.1.3. CB2XML.

Es un convertidor de COBOL a XML escrito en Java y basado en el generador SableCC [Sablecc, 2010]. La conversión XML es estable y se considera lista para cualquier tipo de producción. Uno de los objetivos del proyecto es la creación de utilidades entre XML y CoBOL en tiempo real [cb2xml, 2010]. Esta herramienta fue considerada debido a que al ser una herramienta de libre acceso, es capaz de generar los archivos `.xml`. Por lo tanto, su función principal es entregar la información necesaria del programa CoBOL en `.xml`.

3.1.4. TIBCO.

Es una herramienta que trabaja en la integración de negocio e infraestructura de software [TIBCO, 2010]. Esta herramienta entró al mercado en el año 2004 con la adquisición de Staffware, un pionero entorno de desarrollo. Desde ese momento, la empresa integró BPMN en sus servicios, llamándolo TIBCO iProcess, con plataforma SOA (BusinessWorks y ActiveMatrix), dentro de una iniciativa llamada BPMN+, la cual prometía la integración del usuario central en los procesos principales dentro de un negocio. Durante en pasar de los años, TIBCO logró integrarse con la herramienta Eclipse, con lo cuál se formó TIBCO Business Studio 2.0 [Silver, 2007], entorno de desarrollo creado bajo la plataforma de Eclipse. Esto se considera una gran ventaja debido a que muchas de las propuestas de desarrollo para la automatización del proceso de generación de modelos a partir de LIS, están pensadas en ser desarrolladas

en Eclipse. Además, esta integración permite que TIBCO sea una herramienta de libre acceso.

La decisión de incorporar la herramienta al trabajo es que cumple muchas funcionalidades que son de mucha importancia para extraer modelos desde los LIS, a nivel de desarrollo de un aplicación que haga tal trabajo. Esta herramienta permite exportar/importar archivos del tipo `.xml`/`.xpdl`, por otro lado, permite el modelado de diagramas BPMN y exportar su respectivo archivo `.xpdl`. Así, el aporte de la herramienta es el modelado de los archivos con extensión `.xpdl`.

3.1.5. BizAgi.

El concepto BPM de BizAgi se basa en generar automáticamente una aplicación Web partiendo del diagrama de flujo del proceso sin necesidad de programación [BizAgi, n.d.]. Para lograr esto, el entorno de BPM en BizAgi maneja el ciclo de vida completo de los procesos de negocio: Modelamiento, Ejecución y Mejoramiento Continuo. Cada uno de estas etapas es administrada a través de distintos componentes, los cuales permiten a través de un entorno gráfico y dinámico construir una solución basada en procesos. La siguiente Figura 3.3 explica los pasos para construir una solución BizAgi:

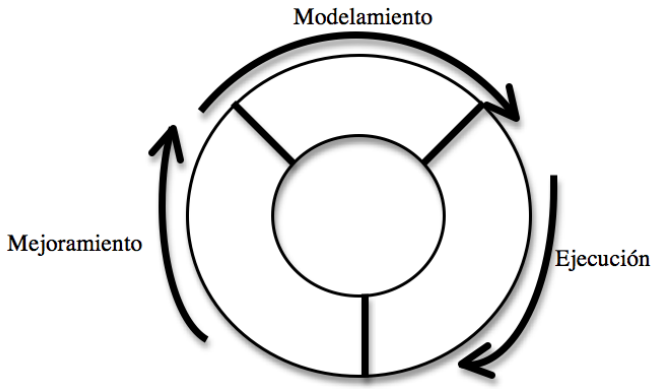


Figura 3.3: Ciclo de vida de un proceso de negocio

A continuación se explicarán los pasos que utiliza BizAgi [XPress, 2009]:

- *Modelamiento:* El primer paso para crear soluciones en BizAgi es definir los procesos. Para esto, el entorno de BPM en BizAgi cuenta con el BizAgi Process Modeler. Este componente es una herramienta que se puede obtener gratuitamente. BizAgi Process Modeler permite obtener diagramas y documentar los procesos en forma fácil y sencilla, y presentar los BP en un formato estándar de BPMN. Luego de diseñar el proceso, el siguiente paso en la construcción de una solución BizAgi es la automatización. BizAgi Studio es el ambiente de construcción con el cual se automatizan los procesos definidos en el BizAgi Process Modeler sin necesidad de programación.

- *Ejecución:* BizAgi BPM Server es el motor que ejecuta y controla los BP construidos con BizAgi Studio. Este se basa en un conjunto de componentes que ofrecen toda la funcionalidad necesaria para administrar los BP en las organizaciones. BizAgi BPM Server basándose en el Modelo construido, se preocupa por la correcta y adecuada ejecución de las diferentes tareas o

actividades que intervienen en el BP controlando y verificando que sean realizadas en el momento adecuado y por la(s) persona(s) o recurso(s) indicado(s), de acuerdo con las directrices, objetivos y otros fundamentos de la empresa.

- *Mejoramiento:* BPM Server de BizAgi cuenta con un completo conjunto de reportes e indicadores de desempeño de los procesos que le permitirán analizar el negocio, identificar cuellos de botella y sus causas, y en general identificar oportunidades de mejoramiento en sus procesos. Con base en estos hallazgos se pueden ajustar los procesos y sus políticas ya sea en tiempo real en la aplicación, o a través de BizAgi Studio para generar una versión mejorada del proceso. Esta nueva versión del proceso puede ser puesta en producción sin necesidad de programación en corto tiempo, solo modificando el modelo de negocio, la aplicación se adapta de forma automática facilitando el mejoramiento continuo y aumentando la productividad de la organización.

Las razones de elección de esta herramienta son similares a las de TIBCO, pero se diferencia en que no se puede programar en esta herramienta, solo trabajar con modelos BPMN.

Capítulo 4

Resultados de la investigación

En el trabajo se encontraron variadas herramientas que hacen de forma parcial todo el trabajo necesario para obtener modelos de negocio a partir de un LIS. Por otro lado, existen varias herramientas que trabajan gráficamente con BPMN y herramientas que trabajan con XML relacionadas con CoBOL, pero ninguna herramienta relaciona BPMN directamente con CoBOL. A su vez, el formato aceptado por las herramientas de modelado para importar modelos de BP es XPDL. Entonces, el trabajo muestra que no existe herramienta tal que a partir de un programa escrito en CoBOL permita generar un archivo con extensión `.xpdl` para luego ser importado por un herramienta de modelado y generar el correspondiente archivo BPMN. Lo único que se puede hacer hasta ahora, es obtener un archivo `.xml` a partir de un archivo CoBOL con dos herramientas: XML Thunder y CB2XML. Ambas herramientas trabajan directamente con programas escritos en CoBOL y generan archivos con extensión `.xml`, pero la diferencia entre ellas es que CB2XML es libre y XML Thunder es de pago.

Siguiendo la propuesta M-LIS2BP [Rodríguez & Caro, 2009] [Rodríguez & Caro, 2008] presentada en la Figura 2.5, en la etapa de Extracción se usan ambas herramientas, XML Thunder y CB2XML para obtener toda la información necesaria

desde un LIS. En esta etapa se tienen algunas restricciones con las herramientas: CB2XML no trabaja con archivos indexados, se necesita transformar los archivos indexados a archivos secuenciales. Por otro lado, XML Thunder si trabaja con archivos indexados y secuenciales.

Una vez obtenido el archivo con extensión `.xml` se debe analizar los datos que fueron extraídos del archivo. Las herramientas mencionadas hacen una transformación automática de conversión de CoBOL a `.xml`, pero no necesariamente se enfoca en los BP o en la reglas de negocio (XML Thunder fue desarrollada bajo reglas de negocio, pero no siempre extrae todas las reglas o procesos que son necesarios). Por lo tanto, la propuesta M-LIS2BP muestra la segunda etapa de Refinación la cual participan los analistas y operadores de sistema quienes intervienen en el código `.xml` y analizan la información que es necesaria. El objetivo de realizar este análisis es generar el archivo de extensión `.xpdl`. Cabe destacar que el archivo mencionado no se puede obtener directamente desde un archivo CoBOL debido a que este archivo es un leguaje de intercambio entre modelos de BPMN y para generar un archivo del tipo `.xpdl` desde CoBOL se deben realizar las etapas mencionadas de la propuesta M-LIS2BP. Entonces, una vez que los analistas y operadores analizan el archivo `.xml` y rescatan la información que es importante, se debe crear un archivo `.xpdl`, con el objetivo de ser exportado por las herramientas de modelado como BizAgi o TIBCO (el esquema de trabajo descrito se resumen en la Figura 9). Ambas herramientas son igual de consistentes y completas, la ventaja que se puede mencionar es que TIBCO esta hecho en Eclipse, por lo tanto, se abren más oportunidades de desarrollo. Por otro lado, ambas herramientas ofrecen asesoría profesional. El resumen del proceso se puede observar en la Figura 4.1.

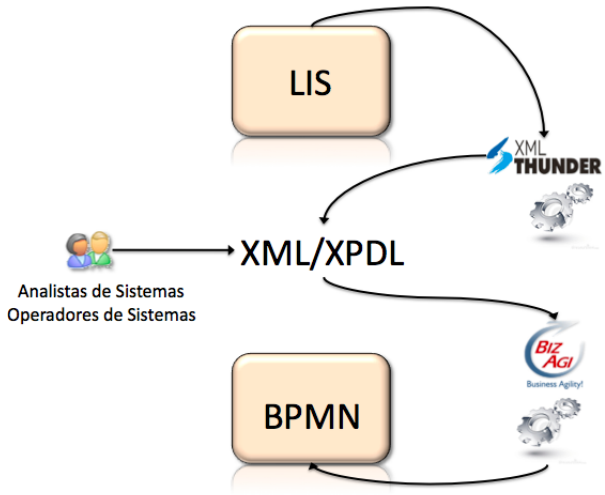


Figura 4.1: Esquema de trabajo

Las dificultades encontradas a la fecha son:

1. La extensión `.xml` es la única que se ha podido generar a partir de códigos en CoBOL y sólo ha sido logrado por la versión de prueba de XML Thunder, esto ha significado en una gran limitante, debido que, al ser una versión de prueba, sólo se ha probado con programas escritos en CoBOL que vienen de ejemplo y, por otro lado, no se sabe cómo se puede conectar con otras herramientas, ya que el acceso a las funcionalidades de la versión de prueba de XML Thunder es restrigido. Entonces, la única manera de obtener un archivo del tipo `.xpdl` es a partir de otro modelo BPMN. Esto significa que automatizar el proceso no es posible ya que el archivo `.xml` debe ser modificado por analistas y operadores de sistema para obtener el archivo con extensión `.xpdl`.

2. Eclipse es por excelencia la herramienta que permite la generación de modelos y documento a partir de un LIS con MoDisco. Entonces, la lógica dice que con

CoBOL debería funcionar, el problema que se presenta es que MoDisco aún está trabajando en la generación de modelos a partir de CoBOL y sólo funciona con LIS basados en Java. Existe una herramienta llamada Cobos [Metrixware, 2010] de la empresa Metrixware que trabaja directamente con Cobol en un entorno de Eclipse y es de libre acceso, pero existen problemas de versiones con MoDisco lo que no permite trabajar con ambas herramientas. TIBCO también es una herramienta basada en Eclipse, pero se presenta el mismo problema que MoDisco relacionado con Cobos. Por último, cuando se trabaja con el framework de KDM (el cual es la base de MoDisco) en Eclipse y se desea generar modelos, no funciona para códigos en Cobol, la herramienta muestra errores.

A continuación se presentará un esquema donde se expone la idea que fue mencionada en los resultados de la investigación. Para probar el resultado del trabajo se usó la versión de prueba de XML Thunder 4 que está disponible para descargar desde la página oficial. Esta versión de prueba trae programas escritos en CoBOL como ejemplo. Por lo tanto, para efectos de prueba, se utilizará uno de estos programas llamado `BANK` el cuál está basado en un pequeño proceso bancario. A continuación, en la Figura 4.2 se muestra un trozo del código.

```
 01  XML-OUTLEN PIC 9(9) VALUE 0.
*Maximum XML Buffer size.
 01  XML-BUFFER-MAX PIC 9(9) VALUE 179500.

*Main data structure.
 01 BANK.
    05 BANK-Id PIC S9(9) COMP.
    05 BANK-Name PIC X(32).
    05 BANK-Incoporation-Date PIC S9(8) COMP.
    05  Account-Counter PIC 99 COMP.
    05 ACCOUNT OCCURS 40 TIMES.
     10 ACCOUNT-Number PIC X(10).
     10 ACCOUNT-Name PIC X(30).
     10 ACCOUNT-Owner PIC X(30).
     10 ACCOUNT-Opening-Date PIC S9(8) COMP VALUE 00010101.
     10 ACCOUNT-Opening-Time PIC S9(6) COMP VALUE 010101.
     10 ACCOUNT-Balance PIC S9(11)V9(2) COMP.
     10 ACCOUNT-Address.
       15 Street-Address PIC X(25).
       15 City PIC X(25).
       15 State-Province PIC X(5).
       15 Country PIC X(25) VALUE 'USA'.
    10 ACCOUNT-Comment PIC X(4000) VALUE '[none]'.


 PROCEDURE DIVISION.
 MAIN.

* Move the XML file name to XML-FILE-NAME
*
      MOVE "bank.xml" TO XML-FILE-NAME.
*
* Read XML file into XML-BUFFER
      OPEN INPUT XML-FILE
      PERFORM VARYING XML-OUTLEN FROM 1 BY XML-FILE-RECSIZE
```

Figura 4.2: Programa escrito en CoBOL usado para la prueba

Al ser un programa ejemplo, este archivo CoBOL es un archivo secuencial. Luego, la herramienta XML Thunder traduce este programa escrito en CoBOL a un archivo con extensión `.xml`, como se muestra en la Figura 4.3.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="BANK">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="BANK-Name" minOccurs="0">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="32"/>
              <xs:whiteSpace value="collapse"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="ACCOUNT" minOccurs="0" maxOccurs="40">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ACCOUNT-Comment" minOccurs="0">
                <xs:simpleType>
                  <xs:restriction base="xs:string">
                    <xs:maxLength value="4000"/>
                    <xs:whiteSpace value="collapse"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
```

Figura 4.3: Trozo de código `.xml`

En esta Figura se puede observar el fin de la etapa de Extracción del método M-LIS2BP. Se obtiene los datos que se pueden extraer desde el programa CoBOL, cabe destacar que la Figura 4.3 sólo muestra un trozo del archivo `.xml`. Es aquí donde actúan los analistas y operadores de sistemas para refinar los datos obtenidos. Entonces, se ha pasado a la etapa de Refinamiento de la Información del LIS, donde una vez que se ha reorganizado el código `.xml` se ha obtenido un archivo `.xpdl`.

```
<?xml version="1.0" encoding="utf-8"?>
<xpdl2:Package
    xmlns:xpdl2="http://www.wfmc.org/2008/XPDL2.1" Id="BANK"
    Name="BANK">
  <xpdl2:PackageHeader>
    <xpdl2:XPDLVersion>2.1</xpdl2:XPDLVersion>
    <xpdl2:Vendor>Savvion</xpdl2:Vendor>
    <xpdl2:Created>Thu Oct 08 16:01:59 IST 2009</xpdl2:Created>
    <xpdl2:Description></xpdl2:Description>
    <xpdl2:Documentation></xpdl2:Documentation>
  </xpdl2:PackageHeader>
  <xpdl2:RedefinableHeader>
    <xpdl2:Author>Gaston</xpdl2:Author>
    <xpdl2:Version></xpdl2:Version>
  </xpdl2:RedefinableHeader>
  <xpdl2:ExternalPackages/>
  <xpdl2:Participants/>
  <xpdl2:Pools>
    <xpdl2:Pool BoundaryVisible="false" Id="BANK" Name="BANK"
        Process="BANK">
      <xpdl2:Lanes>
        <xpdl2:Lane Id="Customer" Name="Customer"
ParentPool="BANK">
          <xpdl2:NodeGraphicsInfos>
            <xpdl2:NodeGraphicsInfo Height="420.0"
ToolId="BizAgi_Process_Modeler"
                Width="1110.0">
              <xpdl2:Coordinates XCoordinate="50.0"
YCoordinate="0.0"/>
            </xpdl2:NodeGraphicsInfo>
```

Figura 4.4: Trozo de código `.xpdl`

Finalmente, se ha obtenido el archivo de extensión `.xpdl` el cual posee las reglas y procesos de negocio que los analistas y operadores de sistema ingresaron al archivo `.xpdl`. Por lo tanto, lo único que queda es importar este archivo con un herramienta para obtener el modelo BPMN. En este caso, se usó la herramienta BizAgi.

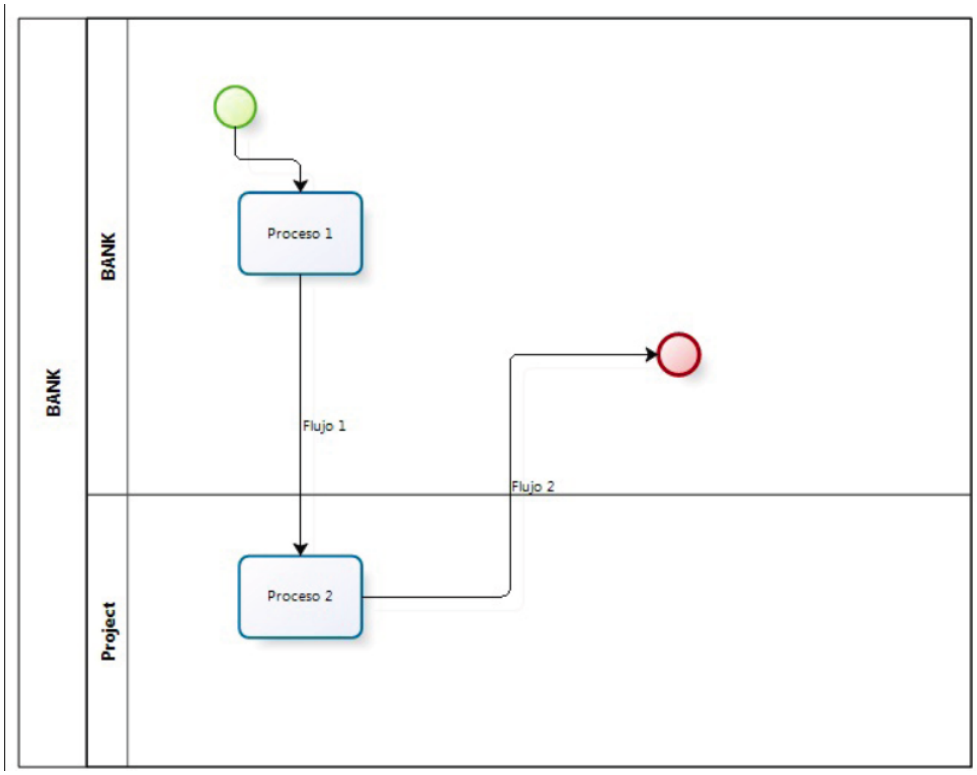


Figura 4.5: BPMN final

En la Figura 4.5 se muestra el resultado. Se puede observar entonces que estos pasos explicados vendrían siendo el objetivo del trabajo. Es interesante en un futuro automatizar este proceso, pero claramente deben intervenir trabajadores que conocen el negocio de las empresas para decidir qué es lo que sirve y obtener un modelo BPMN adecuado.

Capítulo 5

Propuesta LIS2BP-TOOL

Este trabajo considera la confección de una herramienta que permita leer un programa escrito en CoBOL y generar a partir de él un archivo de extensión `.xpdl` con el objetivo de que pueda ser importado con la herramienta BizAgi y genere el modelo BPMN. Esta herramienta se llama `LIS2BP-TOOL`. Está en desarrollo aún, pero actualmente cumple con lo básico para generar modelos BPMN a partir de CoBOL. Esta hecha bajo el entorno de desarrollo JBuilder [Borland, 2010], que es una herramienta derivada de Eclipse. Para la creación de la herramienta, en primer lugar se analizaron las equivalencias entre elementos de LIS y BP (Tabla 5.1) del método M-LIS2BP.

Siguiendo esta conversión, `LIS2BP-TOOL` hace una transformación de las palabras claves dentro de un programa escrito en CoBOL (`CALL`, `GO TO`, `DISPLAY`, `PERFORM`, entre otras) en notación XPDL, cuyo resumen de algunas conversiones de objetos BPMN a XPDL fue presentado en la Tabla 1. Por lo tanto, la herramienta genera un archivo de extensión `.xpdl` el cual cumple con las normas y especificaciones de BPMN y, a su vez, es capaz de ser importado por BizAgi. Todas las transformaciones están basadas en el modelo inicial que provee la herramienta BizAgi en XPDL, debido a que la otra herramienta de desarrollo para XPDL, Together XPDL Workflow Editor,

Información del LIS		Equivalencia	Representación BP	
Nomenclatura	Significado	Proceso de Negocio	UML 2.0-AD	BPMN-BPD
LIS_SC	Programas fuentes			
LIS_SC01	Archivos (claves)	Almacen de Datos	DataStoreNode	Data Object
LIS_SC02	Llamadas a programas	Secuencia Actividades	ObjectFlow	Secuence Flow
LIS_SC03	Entradas	Documentos	DataStoreNode	Data Object
LIS_SC04	Salidas	Documentos	DataStoreNode	Data Object
LIS_XC	Programas ejecutables			
LIS_XC01	Interfaces			
LIS_XC01_1	Roles	Participantes	ActivityPartition	Pool/Lane
LIS_XC01_2	Secuencia de ejecucion	Secuencia Actividades	ObjectFlow	Secuence Flow
LIS_XC01_3	Entradas	Documentos	DataStoreNode	Data Object
LIS_XC01_4	Salidas	Documentos	DataStoreNode	Data Object
LIS_DF	Archivos de datos			
LIS_DF01	Reglas de negocio	Actividades	Actions	Activities
LIS_US	Usuarios			
LIS_US01	Roles	Participantes	ActivityPartition	Pool/Lane
LIS_US02	Distribución Geográfica	Comunicación participantes	ObjectFlow	Message Flow
LIS_TM	Tiempo			
LIS_TM01	Periodicidad de ejecución	Condiciones temporales	-------	StartEventTimer

Tabla 5.1: Equivalencia entre elementos del LIS y BP [Rodríguez & Caro, 2009]

posee un estilo de esquemas XPDL distinto a BizAgi. La diferencia radica en la conexión dinámica de los objetos de BPMN en ambas herramientas. BizAgi da la posibilidad de trabajar con el atributo ‘‘Id’’ de un elemento de BPMN (Activity, flow, entre otros) con el objetivo de conectar el ‘‘Id’’ con otro ‘‘Id’’ y lograr un modelo más consistente. Además, BizAgi tiene más funcionalidades que Together no posee. A continuación, en la Figura 5.1 se mostrará el esquema de trabajo de LIS2BP-TOOL.

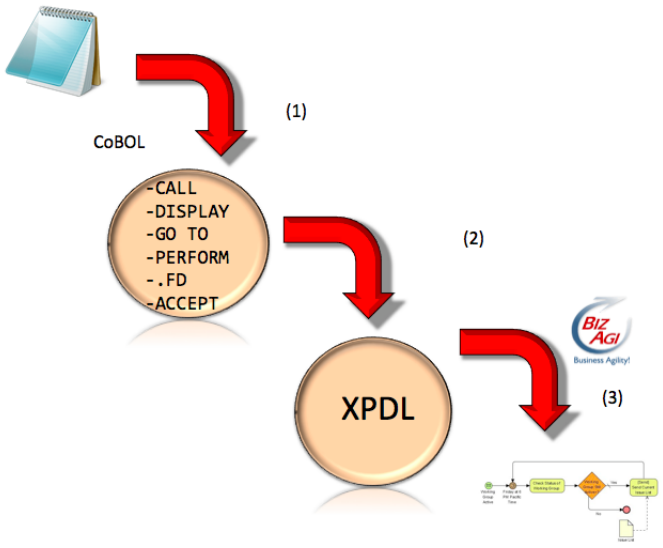


Figura 5.1: Esquema de trabajo de LIS2BPTOOL

En la Figura 5.1 se describen tres pasos:

1. En el paso (1), `LIS2BP-TOOL` lee todo el programa escrito en CoBOL y extrae todas las palabras claves. Una vez hecho lo anterior, estas palabras son almacenadas en una estructura de datos. Es en esta parte donde aún se esta desarrollando para que la extracción de la información sea más completa, ya que sólo por el momento, se extraen palabras que cumplen un proceso de negocio en el archivo. Pero estas palabras, pueden contener más información que es de importancia en el negocio.

2. Una vez hecho lo anterior, en (2) se convierte cada palabra a su equivalente en XPDL y se ingresan a un archivo que tiene el código XPDL que viene por defecto en BizAgi. Cada palabra convertida a XPDL es insertada en una posición especifica dentro de tal archivo.

3. Finalmente, en (3) se obtiene un archivo de extensión `.xpdl` el cual posee todas las palabras claves extraídas en (1) en formato XPDL. Luego, con la ayuda de BizAgi, este archivo es importado y genera el modelo BPMN que representa parcialmente al programa escrito en CoBOL.

Como se mencionó en la Figura 14, existen tres pasos en los cuales la herramienta `LIS2BP-TOOL` extrae palabras que cumplen un rol de negocio y las convierte a un extracto de código en XPDL. Básicamente, la herramienta se basa en el archivo temporal XPDL que ofrece la herramienta de BizAgi. Dentro de este archivo existen distintas sentencias, pero se prestará mayor atención (sólo por ahora, ya que cada sentencia tiene una función en particular, pero la herramienta se enfoca en las conversiones mostradas en la Tabla 5.2. Se pretende agregar más funcionalidades en un futuro) a la siguientes palabras:

XPDL	**CoBOL**
`<Activities/>`	`CALL`
`<Transitions/>`	`GO TO`
`<Artifacts/>`	`DISPLAY, .FD`
`<Associations/>`	conexión de artefacto

Tabla 5.2: Equivalencia entre XPDL y CoBOL

La primera, se relaciona con *Activities* de BPMN y paralelamente corresponde a la palabra `CALL` dentro del archivo CoBOL, luego se tiene a la segunda palabra que se relaciona con *Secuence Flow* y corresponde a `GO TO` del programa escrito en CoBOL. Terminando la idea, se tiene a los artefactos que corresponden a la tercera palabra que va relacionada con la cuarta palabra que corresponde a la conexión entre el artefacto y la actividad; éstos se relacionan con las palabras `DISPLAY`, `.FD`, entre otras dentro del programa CoBOL y que son representados como *Data Object*

en BPMN. Entonces, el programa para obtener las palabras en CoBOL, hace una revisión del archivo y extrae todas las palabras que representen procesos de negocios. Una vez hecho lo anterior, a cada palabra extraída se le asigna un trozo de código XPDL correspondiente a su equivalencia en BPMN. Por ejemplo, si la herramienta selecciona un `CALL`, esta palabra se transforma en:

```
<Activity Id="1" Name="CALL">
   <Description />
   <Implementation>
   <Task />
   </Implementation>
   <Performers />
   <Documentation />
   <ExtendedAttributes />
   <NodeGraphicsInfos>
     <NodeGraphicsInfo ToolId="BizAgi_Process_Modeler" Height="60" Width="90"
BorderColor="-16553830" FillColor="-1249281">
       <Coordinates XCoordinate="270" YCoordinate="74" />
     </NodeGraphicsInfo>
   </NodeGraphicsInfos>
   <IsForCompensationSpecified>false</IsForCompensationSpecified>
</Activity>
```

Figura 5.2: Conversión de `CALL` a XPDL

Y si la herramienta encuentra GO TO, hará la siguiente conversión:

```
<Transition Id="1" From="1 To="1" Name="GO TO">
   <Condition />
   <Description />
   <ExtendedAttributes />
   <ConnectorGraphicsInfos>
     <ConnectorGraphicsInfo ToolId="BizAgi_Process_Modeler" BorderColor="0">
       <Coordinates XCoordinate="137" YCoordinate="193" />
       <Coordinates XCoordinate="203.5" YCoordinate="193" />
       <Coordinates XCoordinate="203.5" YCoordinate="104" />
       <Coordinates XCoordinate="270" YCoordinate="104" />
     </ConnectorGraphicsInfo>
   </ConnectorGraphicsInfos>
</Transition>
```

Figura 5.3: Conversión de `GO TO` a XPDL

Entonces, la herramienta hará todas las conversiones necesarias de todas las

palabras claves que existan dentro de un programa escrito en CoBOL. Luego, una vez hechas todas estas iteraciones, el programa generará un archivo de extensión `.xpdl`, y a partir de éste, la herramienta BizAgi importa el archivo y genera el correspondiente modelo BPMN. En la Figura 5.4 se muestra el diagrama BPMN generado por la herramienta desde un archivo escrito en CoBOL correspondiente a una empresa de la provincia. El archivo cumple funciones básicas de negocio y éstas son representadas en el diagrama BPMN. Cabe destacar que este paso sólo muestra una primera mirada del proceso que hace el programa CoBOL. Por otro lado, es importante señalar que en este modelo no se muestra la intervención al archivo XPDL que deben hacer los analistas y operadores de sistema, sólo muestra la primera mirada general del archivo escrito en CoBOL. La idea en un futuro es que esta herramienta genere un modelo más completo y real del programa CoBOL. Para lograr esto, se deben hacer conexiones a bases de datos que forman un repositorio LIS y obtener todas las palabras de éstas y agregarlas al modelo.

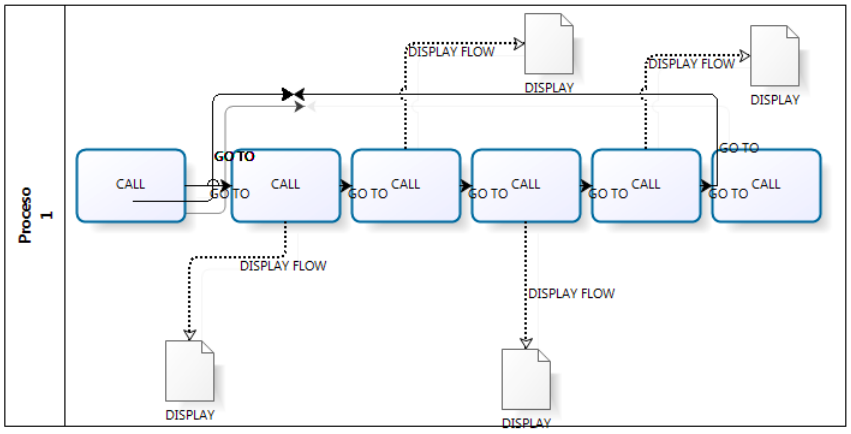


Figura 5.4: Diagrama BPMN obtenido por LIS2BP-TOOL desde un archivo escrito en CoBOL real

5.1. Descripción de la herramienta

El desarrollo de la herramienta está orientado a los pasos básicos que se definieron anteriormente en la Figura 5.1. La herramienta consta de dos elementos: un archivo de extensión `.exe` y un archivo de extensión `.txt` donde se almacena el molde del archivo `.xpdl`. Cuando se menciona molde, se refiere al código XPDL que viene por defecto en BizAgi, así la herramienta lee este molde e ingresa las conversiones necesarias para generar un nuevo archivo XPDL. Para utilizar la herramienta, el usuario debe copiar la carpeta LIS2BPTOOL donde desee. Esta carpeta contiene dos elementos: el programa `LIS2BPTOOL.exe` y el archivo `molde.txt`. Una vez que el usuario eligió donde pegar la carpeta, ejecuta el programa y aparecerá la primera ventana la cual se describe en la Figura 5.5:

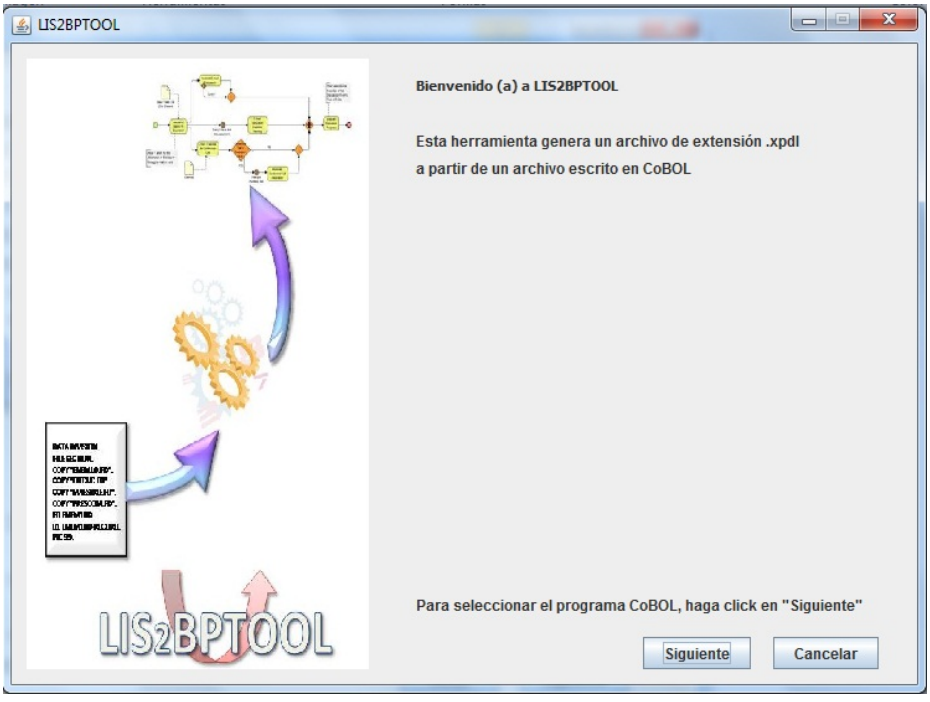


Figura 5.5: Ventana de inicio del programa

Esta pantalla contiene la bienvenida al usuario que utilizará la herramienta. Luego, el usuario debe seleccionar el botón "Siguiente" y aparecerá la segunda ventana que está descrita en la Figura 5.6.

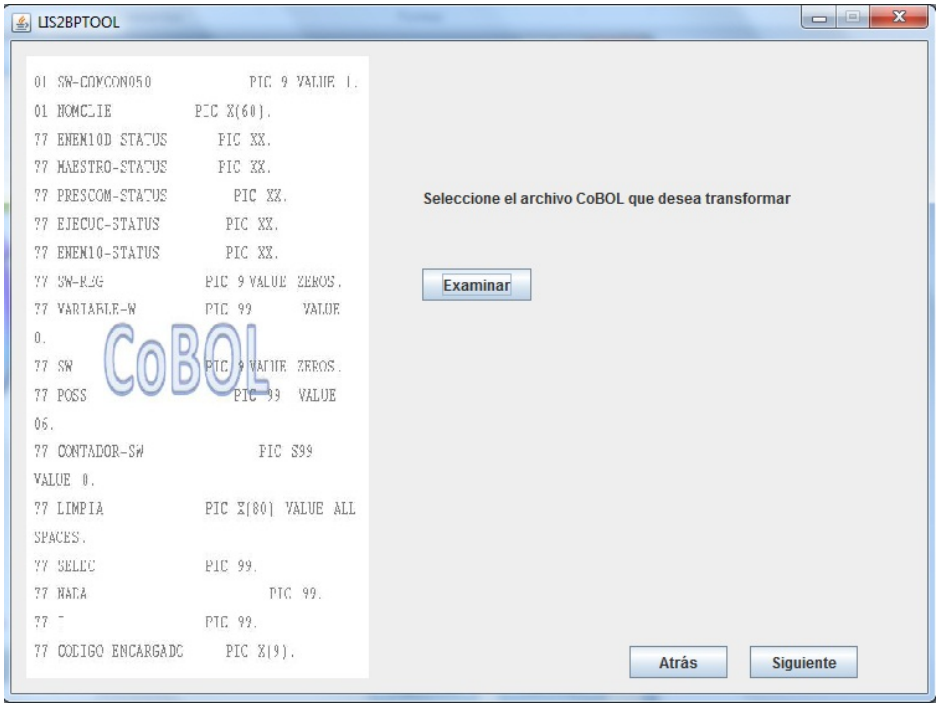


Figura 5.6: Ventana donde al usuario se le muestra la opción de seleccionar el archivo escrito en CoBOL

Hecho eso, el usuario debe seleccionar el archivo escrito en CoBOL como se muestra en la Figura 5.7. La herramienta muestra un mensaje de error si el archivo seleccionado no corresponde a un archivo escrito en CoBOL.

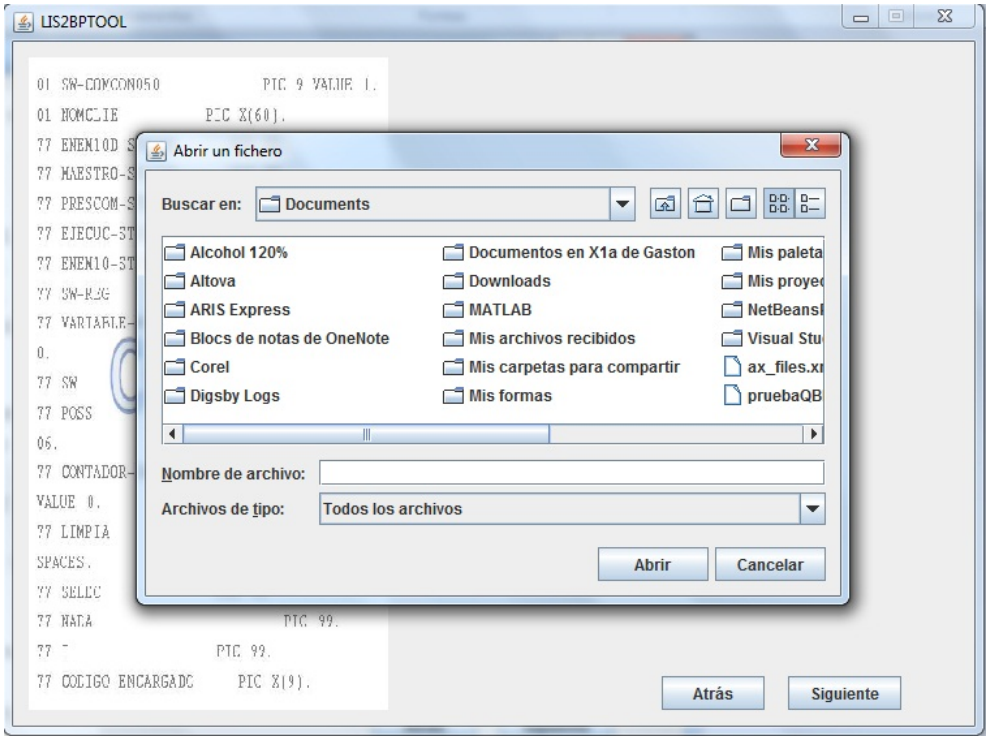


Figura 5.7: Búsqueda del archivo escrito en CoBOL

Una vez que el usuario haya seleccionado el archivo, el programa muestra la ruta del archivo seleccionado más el nombre del archivo seleccionado. La Figura 5.8 detalla lo anteriormente señalado.

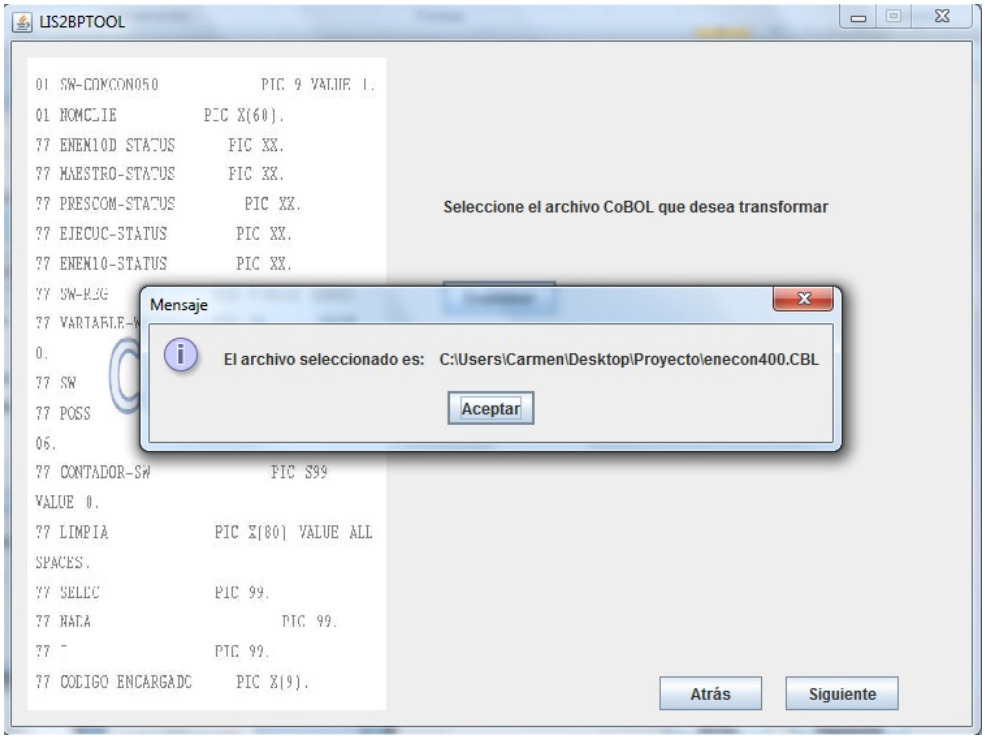


Figura 5.8: Selección del archivo escrito en CoBOL

Luego, el usuario pasa a la siguiente ventana donde aparece la opción de generar el archivo código en XPDL, como se muestra en la Figura 5.9.

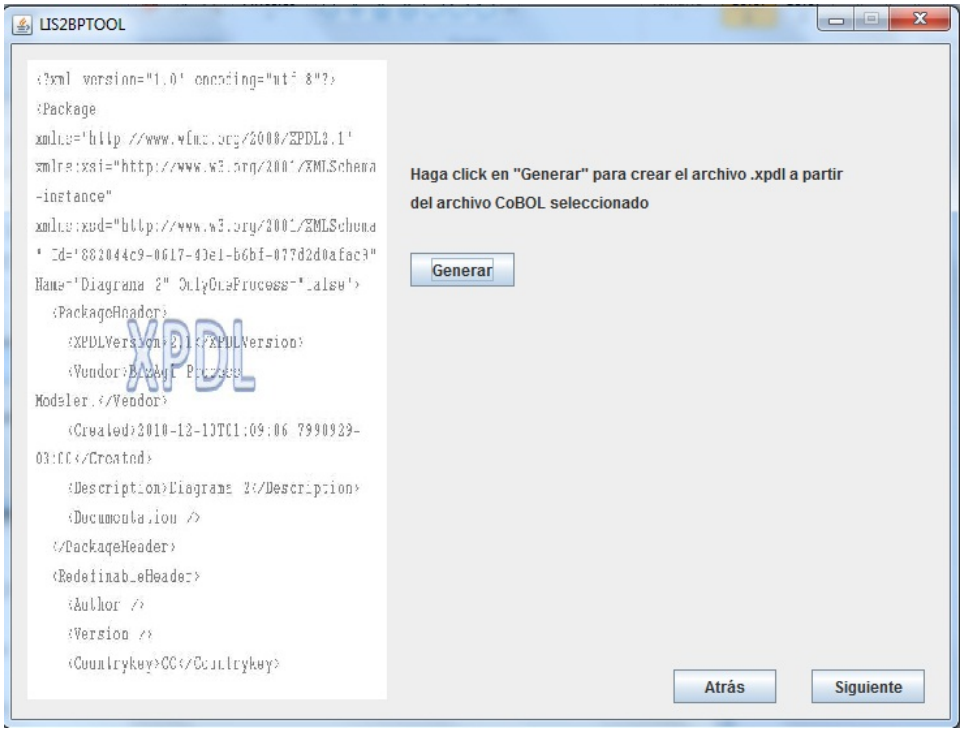


Figura 5.9: Ventana para generar el archivo XPDL

Una vez realizado la transformación, la herramienta muestra un mensaje de que el archivo ha sido creado exitosamente. En caso contrario, la herramienta mostrará un mensaje detallando que no se ha podido crear el archivo XPDL. En la Figura 5.10 se detalla lo dicho.

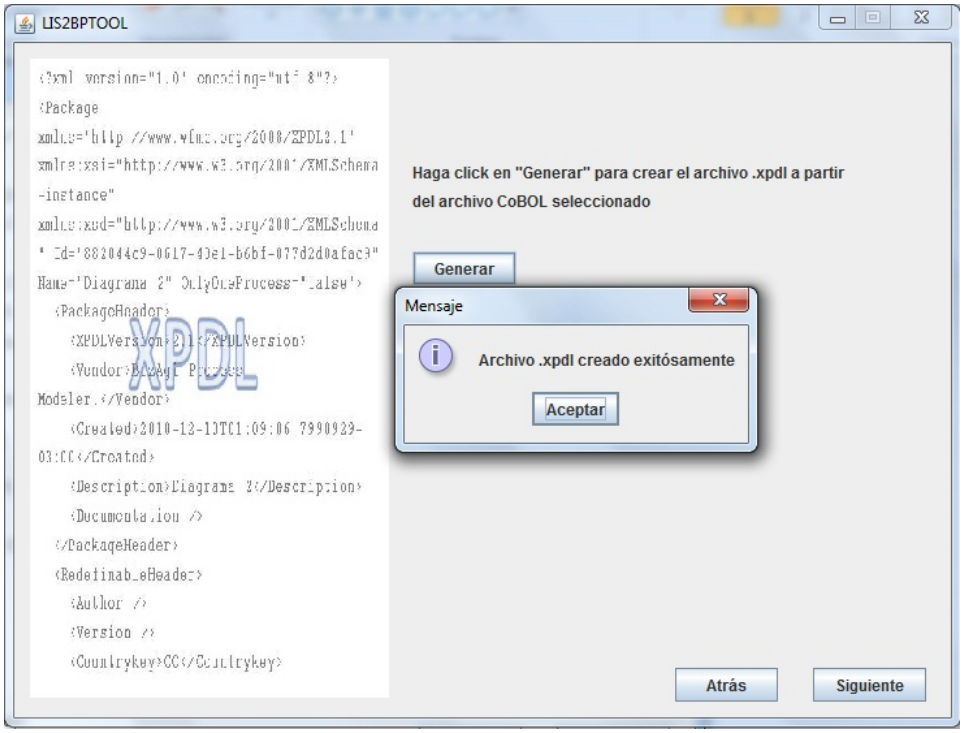


Figura 5.10: Aviso de creación del archivo XPDL

Finalmente, aparece la última pantalla donde el programa detalla que el archivo `codigo.xpdl` ha sido creado en la carpeta LIS2BPTOOL para que el usuario lo importe a BizAgi y pueda trabajar en él, como se muestra en la Figura 5.11.

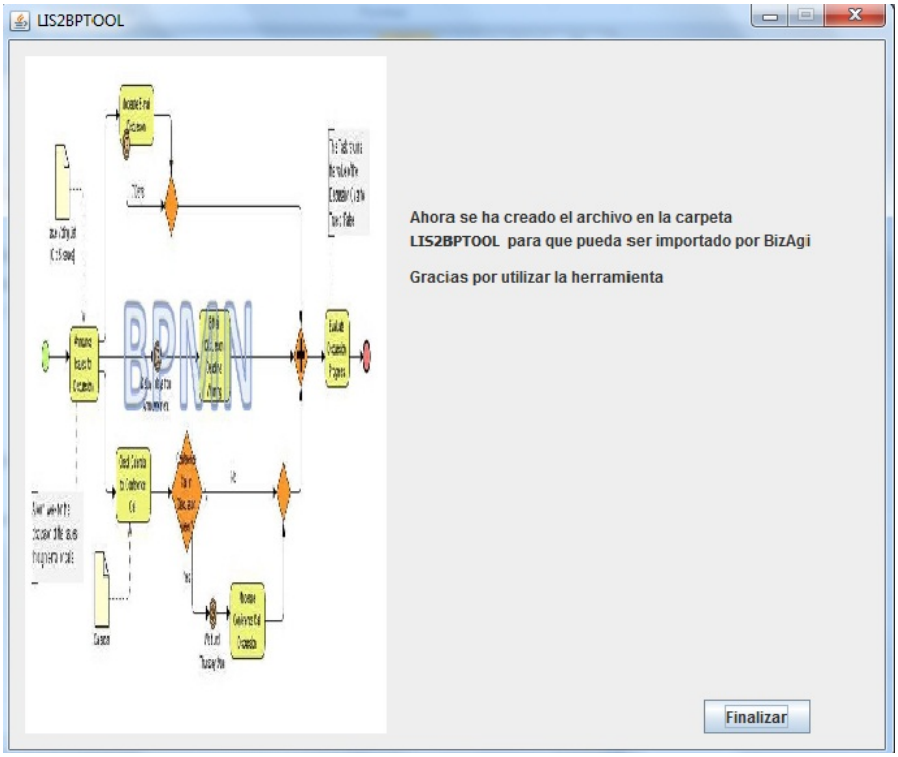


Figura 5.11: Ventana final del programa

La herramienta fue desarrollada en Window 7, por lo que puede existir problemas de permisos de usuario, ya que en este Sistema Operativo, para trabajar o modificar ciertas carpetas (como por ejemplo `C:/`) se tiene trabajar como administrador. En caso de que la herramienta tenga problemas para crear el archivo XPDL, basta simplemente con darle privilegios de administrador al programa `LIS2BPTOOL.exe`.

Capítulo 6

Conclusiones

La modernización de los LIS es un tema que ha sido fundamental para las organizaciones actuales. Obtener los BP a partir de los LIS sin perder la información inherente al negocio que estos contienen es importante para los analistas de negocio con el objetivo de tomar decisiones. En el presente trabajo se ha generado modelos de BPMN a partir de los programas fuente escritos en CoBOL que forman parte de un LIS. Básicamente, este proceso consiste en generar archivos de tipo `.xml` y `.xpdl` para que puedan ser importados por herramientas de modelado y generar los BPMN que se necesitan. Como se mencionó, no existen herramientas que automaticen el proceso descrito, es por eso que se deben buscar y analizar todas las herramientas disponibles con el fin de lograr los modelos BPMN. Dentro de este trabajo se han encontrado herramientas que hacen los procesos parcialmente pero ninguna que haga el proceso total, por lo tanto, se deben realizar procesos por separado. En la memoria se menciona el método M-LIS2BP con el objetivo de posicionar el tabajo en las dos primeras etapas de dicho método. La primera etapa, que se refiere a la Extracción de la información de un LIS, consiste en generar un archivo de extensión `.xml` para obtener todos los datos que se encuentran en un LIS. Hay herramientas como XML Thunder que aparte de extraer los datos de un LIS basado en CoBOL, extrae además,

reglas y procesos de negocio, pero la única herramienta de libre acceso CB2XML sólo trabaja con programas escrito en CoBOL secuenciales. Siguiendo las etapas del método M-LIS2BP, se pasa a la etapa de Refinamiento de Información del LIS. Aquí, los archivos con extensión `.xpdl` cumplen un rol importante, debido a que estos archivos son creados a partir de la información del archivo `.xml` más la intervención de analistas y operadores de sistema que conocen el negocio y que saben que datos son los relevantes, así, generan un archivo con extensión `.xpdl` para que pueda ser importado por una herramienta de modelado como BizAgi y TIBCO y generar el modelo BPMN correspondiente.

Todas las herramientas encontradas van cambiando con el tiempo, como es el caso de MoDisco. Ésta es una herramienta poderosa para trabajar con LIS, pero como se mencionó en este trabajo, aún está en desarrollo para ser utilizada con más códigos heredados, como es el caso de Cobol. En consecuencia, esta herramienta en un futuro cercano se esperan nuevas versiones que podrán integrar a Cobol usando como entorno de modelado TIBCO o el mismo Eclipse. Pero, por otro lado, se tiene a XML Thunder con BizAgi, ambas herramientas van de la mano en la generación de código del tipo `.xml` para la generación de BPMN. XML Thunder tiene todas las herramientas y pasos para generar un esquema XML a partir de un archivo Cobol, pero existen limitantes como es el acceso al programa, debido a que como es pago, sólo se puede tener acceso a la versión de prueba, lo cual limita las transformaciones desde CoBOL a `.xml`.

Como trabajo futuro se pretende analizar las posibilidades de mejorar la herramienta que hace el proceso de obtención de modelo de BP (bajo todas las restricciones mencionadas) `LIS2BPTOOL`. Esta herramienta es una primera versión de lo que se quiere realizar, pero XPDL ofrece más propiedades que se pueden ocupar

para mejorar el modelo. Lo modelos BPMN tiene propiedades específicas que se detallan en el archivo XPDL, pero estas propiedades no se encuentran detalladas en CoBOL. Lo interesante sería integrar estas propiedades al archivo XPDL en función de lo que el archivo CoBOL puede entregar.

Bibliografía

[Aguilar-Savén, 2004] Aguilar-Savén, R. 2004. Business process modelling: Review and framework. *International Journal of Production Economics*, **90**, 129–149.

[Bézivin, 2004] Bézivin, Jean. 2004. In Search of a Basic Principle for Model Driven Engineering. *The European Journal for the Informatics Professional, Vol. V, No. 2.*

[Bisbal, 1997] Bisbal, J. 1997. An Overview of Legacy Information Systems Migration. *Proceedings of the APSEC'97/ICSC'97: Joint 1997 Asia Pacific Software Engineering Conference and International Computer Science Conference*, 529–530.

[BizAgi, n.d.] BizAgi. *Solución de Gestión de Procesos de Negocio, BPM y Automatización de flujos de trabajo.*

[BizAgi, 2010] BizAgi. 2010. BPMN: Business Process Modeling Notation. *Corporate Headquarters.*

[Borland, 2010] Borland. 2010. Borland Solutions. *http://www.borland.com/.*

[Brodie & Stonebraker, 1995] Brodie, M, & Stonebraker, M. 1995. Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach. *Morgan Kaufman Publishers.*

[Bruneliere & Madiot, 2010] Bruneliere, Hugo, & Madiot, Frederic. 2010. How to deal with your IT legacy? Reverse Engineering with MoDisco. *INRIA AtlanMod and Mia-Software.*

[Canam, 2004] Canam. 2004. Transforming COBOL Sequential files to XML Documents.

[Canam, 2010] Canam. 2010. XML Thunder. *http://www.canamsoftware.com.*

[Caro *et al.* , 2002] Caro, Angélica, Bocca, Jorge, & Campos, Daniel. 2002. Migración de Sistemas Heredados: Una metodología basada en el uso de herramientas de kdd (knowledge discovery in databases). *Revista Ingeniería de Sistemas*, 49–73.

[cb2xml, 2010] cb2xml. 2010. CB2XML. *http://cb2xml.sourceforge.net/.*

[Compute-rs, 2010] Compute-rs. 2010. *Adquisición de nuevas aplicaciones de negocio o modernización de sistemas heredados.*

[ComputerWorld, 2006] ComputerWorld. 2006. Los sistemas heredados preocupan más a las empresas usuarias de TI.

[Davenport, 1993] Davenport, Thomas. 1993. Process Innovation: Reengineering work through information technology. *Harvard Business School Press, Boston.*

[Emprendedores, 2008] Emprendedores. 2008. BPMN y su influencia en la Gestión de procesos. *http://www.blog-emprendedor.info/la-bpmn-y-su-influencia-en-la-gestion-de-procesos/.*

[Estes, 2006] Estes, Don. 2006. From Legacy to BPM. *Cutter Consortium Enterprise Architecture Service Executive Update, Volume 8*, 22–22.

[Gold, 1998] Gold, N. 1998. The Meaning of "Legacy Systems". *SABA Project Report: PRSABA-01.*

[Harmon, 2004] Harmon. 2004. *The OMG's Model Driven Architecture and BPM.*

[ISO, 2000] ISO. 2000. Norma Internacional ISO 9001 - Sistemas de Gestión de la Calidad. *Secretaría Central de ISO, Suiza.*

[KDM, 2010] KDM. 2010. Knowledge Discovery Metamodel. *http://www.kdmanalytics.com/kdm/.*

[Khusidman & Ulrich, 2007] Khusidman, V, & Ulrich, W. 2007. *Architecture-Driven Modernization: Transforming the Enterprise.*

[Kitchenham, 2007] Kitchenham, S. 2007. Guidelines for performing systematic literature reviews in software engineering. *Software Engineering Group School of Computer Science and Mathematics Keele, Keele University.*

[Metrixware, 2010] Metrixware. 2010. *Cobos.*

[MoDisco, 2010] MoDisco. 2010. MoDisco. *http://www.eclipse.org/MoDisco/.*

[OMG, 2010a] OMG. 2010a. Business Process Management Notation. *www.bpmn.org.*

[OMG, 2010b] OMG. 2010b. Unified Modeling Language. *www.omg.org.*

[Pérez-Castillo *et al.* , 2010] Pérez-Castillo, Ricardo, Weber, Barbara, García-Rodríguez, Ignacio, & Piattini, Mario. 2010. Modernizing Legacy Systems through runtime Models. *Alarcos Research Group.*

[QVT, 2005] QVT. 2005. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. *OMG Adopted Specification ptc/05-11-01*, 204.

[Rodríguez & Caro, 2008] Rodríguez, Alfonso, & Caro, Angélica. 2008. Hacia la Obtención de Procesos de Negocio desde Sistemas de Información Heredados. *Encuentro de Informática y Gestión (EIG), Temuco, Chile.*

[Rodríguez & Caro, 2009] Rodríguez, Alfonso, & Caro, Angélica. 2009. LI2BP : Una propuesta para obtener Procesos de Negocio a partir de los Sistemas Herdados. *XII Conferencia de Ingeniería de Requisitos y Ambientes de Software (IDEAS). Medellín, Colombia*, 29–42.

[Rummler & Brache, 1995] Rummler, & Brache. 1995. Improving Performance: How to manage the white space on the organizational chart. *Jossey-Bass, San Francisco.*

[Runeson & Host, 2008] Runeson, P., & Host, M. 2008. Guidelines for conducting and reporting case study research in software engineering. *Springer*, 131–164.

[Sablecc, 2010] Sablecc. 2010. Sablecc. *http://sablecc.org/.*

[Sánchez, 2008] Sánchez, Manuel. 2008. Introducción a BPMN: Una breve introducción a la estandarización del modelado de procesos de negocio.

[Seacord *et al.* , 2003] Seacord, Robert, Plakosh, Daniel, & Lewis, Grace. 2003. *Modernizing Legacy Systems: Software Technologies, Engineering Processes and Business Practices.* SEI Series in Software Engineering.

[Silver, 2007] Silver, Bruce. 2007. The BPMS Report: TIBCO iProcess Suite 10.6. *BPMS Watch*.

[Sommerville, 2005] Sommerville, Ian. 2005. *Ingeniería de Software*. Addison Wesley.

[TIBCO, 2010] TIBCO. 2010. TIBCO Developer Network. *http://developer.tibco.com*.

[WfMC, 1999] WfMC. 1999. Workflow Management Coalition: Terminology Glossary. 65.

[WfMC, 2008] WfMC. 2008. Process Definition Interface: XML Process Definition Language. *The Workflow Management Coalition Specification*.

[White, 2003] White, Stephen. 2003. XPDL and BPMN. *SeeBeyond, United States*.

[XML, 2010] XML. 2010. XML. *World Wide Web Consortium, www.w3.org*.

[XPress, 2009] XPress, BizAgi. 2009. BizAgi, Descripción Funcional. *BizAgi*.