



TESINA DE LICENCIATURA

Análisis metodológico de la plataforma IBM WebSphere BPM, y sus equivalentes funcionales en herramientas de licenciamiento de código

TITULO: fuente abierto.

AUTORES: AC. José Nicolás Martínez Garro

DIRECTOR: Mg. Patricia Bazán

CODIRECTOR: Ing. Emilio Lorenzón

CARRERA: Licenciatura en Sistemas

Resumen

BPM (Business Process Management) se encarga de la gestión de procesos de negocio en el marco empresarial. Es un enfoque integral en el cual se modelan las características y restricciones de los procesos en todas las etapas de su ciclo de vida, apuntando fundamentalmente a automatizar la mayor cantidad de tareas de manera de lograr un control integral que permita dar soporte, simular, modelar y monitorear en forma controlada las actividades del proceso.

Este trabajo tiene como finalidad analizar el entorno IBM Websphere BPM haciendo hincapié en las etapas del ciclo de vida de los procesos que este cubre, como así también las capacidades de desarrollo e integración para la interacción de los procesos con aplicaciones reales. Por otro lado, observar las alternativas que ofrece el mercado en herramientas BPM de licenciamiento de código fuente abierto, intentando lograr un grado de integración y cohesión similar al logrado por la herramienta de IBM.

Líneas de Investigación

Las líneas de investigación principales asociadas a este trabajo son Gestión de procesos de negocio (BPM), analizando la cobertura de su ciclo de vida y SOA, considerando la interacción entre los procesos y aplicaciones reales existentes.

BPM es un concepto muy potente desde el punto de vista práctico que ofrece nuevos problemas a resolver, ya sea para el desarrollo como para el estudio. El mayor interés actual en BPM es lograr la mayor interacción del área de tecnología con las arquitecturas orientadas a servicios. De allí nuestro interés en analizar la interacción de BPM con aplicaciones reales existentes, especialmente aquellas basadas en acceso de servicios.

Trabajos Realizados

Se ha analizado un caso de uso concreto relacionado con la solicitud de créditos a una entidad especializada. Utilizando el entorno provisto por IBM Websphere BPM se ha modelado el proceso que implementa dicha solicitud con BPMN e implementado sus componentes utilizando Java y BPEL. Luego logramos invocar el proceso como un *web service* desde una aplicación J2EE.

Luego utilizando Jpbm e Intalio hemos perseguido objetivos similares, aunque con resultados dispares.

Conclusiones

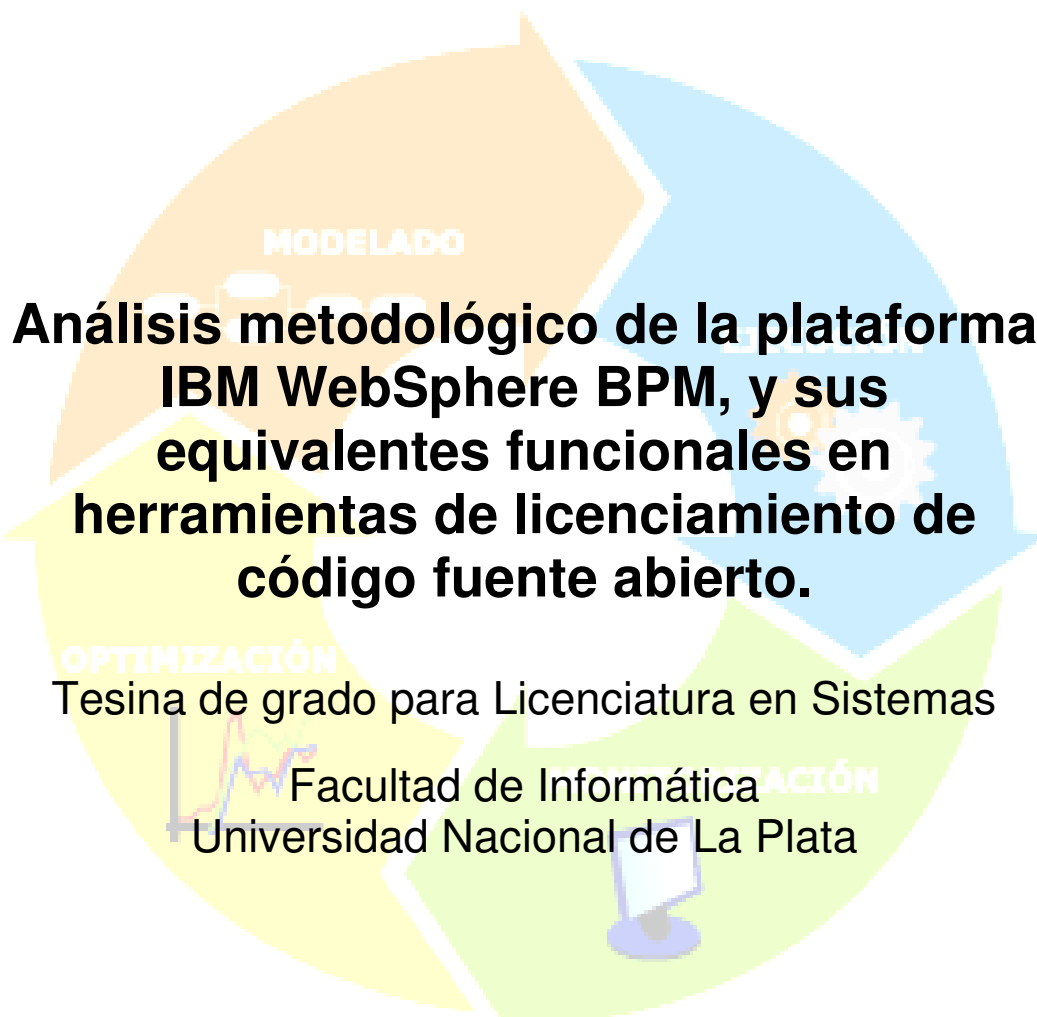
BPM es una metodología integradora para la gestión de procesos de negocio. Por medio de la misma se considera a los procesos en sus distintos estados dentro del ciclo de vida, facilitando la integración de componentes ya existentes en la organización, tal como aplicaciones *legacy*.

Encontramos en el mercado distintos BPMS, tal como lo es IBM Websphere BPM, con una gran potencia para la gestión pero fuertes requerimientos de hardware más una licencia propietaria. Existen equivalentes de tipo *open source* que a pesar de estar desarrollados bajo distintas ópticas aún no logran cubrir la funcionalidad propuesta por el entorno propietario antes citado

Trabajos Futuros

Los trabajos futuros están relacionados con la posibilidad de realizar ingeniería inversa sobre procesos implementados en BPEL para llegar a su equivalente BPMN y no perder consistencia ante la necesidad de modificación de las versiones ejecutables. Por otro lado sería deseable la obtención de mecanismos que permitan conservar la conexión entre las distintas etapas del ciclo de vida dentro de las herramientas open source.

Fecha de la presentación: Junio de 2010



Análisis metodológico de la plataforma IBM WebSphere BPM, y sus equivalentes funcionales en herramientas de licenciamiento de código fuente abierto.

Tesina de grado para Licenciatura en Sistemas

Facultad de Informática
Universidad Nacional de La Plata

Alumno: A.C. José Martínez Garro.

Directora: Mg. Patricia Bazán.

Codirector: Ing. Emilio Lorenzón.

ÍNDICE GENERAL

INTRODUCCIÓN	6
CAPÍTULO 1: MARCO CONCEPTUAL DE BPM.....	8
Motivación y definiciones.....	8
Ciclo de vida de los procesos de negocio.....	10
Diseño y análisis	10
Configuración	10
Representación	11
Evaluación.....	11
Administración.....	12
Clasificación de los procesos de negocio	13
Organizacional vs. Operacional	13
Grado de automatización	13
Grado de repetición.....	13
Grado de estructuración.....	14
Objetivos, estructura y organización	14
Evolución de las arquitecturas de sistemas empresariales.....	15
Desarrollo de aplicaciones tradicionales	15
Las aplicaciones empresariales y su integración	16
Sistemas de planeamiento de recursos empresariales (ERP)	16
Integración de aplicaciones empresariales	16
Integración punto a punto.....	17
Integración basada en un nodo central y repetidores	18
Modelo empresarial y orientación a procesos.....	19
Canales de valor	20
Procesos organizacionales de negocio	21
Procesos B2B.....	22
Manejo de workflow	23
Workflows y aplicaciones	23
Workflows de sistema	24
Workflows de interacción humana	25
Desafíos para el manejo de <i>workflow</i>	25
Servicios de cómputo empresarial	26
Arquitecturas orientadas a servicios	27
Servicios empresariales	28
Aplicaciones basadas en composición de servicios.....	29
Servicios empresariales y arquitecturas orientadas a servicios	29
Bus de servicios empresariales.....	30
Composición de servicios.....	30
Arquitecturas para gestión de procesos de negocio (BPM).....	31
Arquitecturas de gestión de <i>workflow</i>	31
Arquitecturas de sistemas de gestión de <i>workflow</i>	31
Gestión flexible de <i>workflow</i>	32
Web services y su composición en relación a la metodología de workflow	32
Composición de web services en pos de los procesos	33
Composición de servicios en la integración de aplicaciones empresariales	35
Composición avanzada de servicios	35

Ontologías y mapeos de datos	36
Volviendo sobre el concepto de SOA	37
Conceptos relacionados con SOA	40
Metodología para implementar un paradigma de orientación a procesos	42
Estrategia y organización	43
Conclusión	47
CAPITULO 2: ANÁLISIS TÉCNICO Y METODOLÓGICO DE IBM	
WEBSPHERE BPM	48
Descripción de la infraestructura IBM Websphere BPM.	48
Websphere Application Server (WAS)	49
Websphere Enterprise Service Bus (ESB)	50
Websphere Process Server (WPS)	52
Websphere Business Process Modeler (WBPM)	53
Websphere Service Registry and Repository	56
Websphere Integration Developer (WID)	57
Business Process Coreographer (BPC)	59
Gestor de reglas de negocio (<i>Business Rules Monitor</i>)	59
Conclusiones	60
Análisis funcional aplicado	60
Caso de uso: solicitud de crédito	60
Modelado del proceso	61
Participantes del proceso	64
Servicios	64
Interfaces	64
Datos	64
Flujo de la etapa 1 del proceso: Solicitud de crédito	65
Flujo de la etapa 2 del proceso: Estudio de solicitud de crédito	66
Modelo de datos	66
Despliegue del proceso	67
Incorporación de tareas humanas	69
Invocación del proceso como un servicio	71
Conclusiones	77
CAPÍTULO 3: ANÁLISIS TÉCNICO Y METODOLÓGICO DE LOS	
EQUIVALENTES OPEN SOURCE A IBM WEBSPHERE BPM	78
Fundamentos	78
jBPM	79
jPDL (jBPM Process Definition Language)	81
jBPM Modeler	85
Monitoreo de los procesos	86
Modelo de procesos	86
jBPM BPEL	91
Intalio BPMS	92
Conclusiones	96
CAPÍTULO 4: CONCLUSIONES Y POSIBLES LÍNEAS DE INVESTIGACIÓN	
FUTURA	97
Conclusiones	97
Líneas posibles de investigación futura	100
Referencias	101
Anexo 1 – BPMN y los procesos de negocio	103
Introducción a BPMN	103

Objetivos	103
BPMN “habilita” a BPM	103
BPMN modeliza los Servicios Web	104
BPMN y BPD	104
Diagramas BPM	105
Modelización por eventos.....	105
Procesos, sub-proceso y tareas	107
Modelar el flujo de secuencia de un proceso	107
Modelar los puntos de decisión con Gateways	107
Quien hace que – Pool y Lanes	110
Transformación de datos.....	111
Lenguajes de ejecución de negocios.....	111
BPMS orquestan los servicios web.....	111
Integración de BPMN con UML	112
Anexo 2 – jPDL	113
Validación	113
Definición del lenguaje.....	113
Definición del proceso	113
Nodos.....	114
Common node elements (elementos comunes de nodo)	114
End-state (Estado final).....	115
Estado	115
Task-node (Nodo de tarea)	115
Process State (Estado de proceso).....	115
Super-State (Súper estado)	116
Fork (División)	116
Join (Unión).....	116
Decision (Decisión)	116
Transición.....	116
Action (acción)	117
Tarea.....	117

ÍNDICE DE IMÁGENES

CAPITULO 2: ANÁLISIS TÉCNICO Y METODOLÓGICO DE IBM WEBSHERE BPM	48
Figura 1. Descripción de la arquitectura Websphere	48
Figura 2: Componentes de la arquitectura IBM Websphere BPM.	49
Figura 3. Vista de la etapa 1 del proceso: Solicitud de crédito	62
Figura 4. Vista de la etapa 2 del proceso: Estudio de solicitud.....	63
Figura 5: Modelo de datos de la aplicación, construido sobre el motor MSSQL 2000.....	66
Figura 6. Versión BPEL de la etapa 1 del proceso: Solicitud de crédito	68
Figura 7: versión BPEL de la etapa 2 del proceso: Estudio de solicitud de crédito	69
Figura 8: Captura de la interfaz para realizar la solicitud de crédito	73
Figura 9: Captura de la interfaz de comunicación una vez ingresada la solicitud.....	74
Figura 10: Selección de solicitudes para efectuar análisis	74
Figura 11: Visualización de solicitud a analizar	74
Figura 12: Pantalla de información luego del análisis	75
Figura 13: Modelo BPMN del proceso de desarrollo	76
CAPÍTULO 3: ANÁLISIS TÉCNICO Y METODOLÓGICO DE LOS EQUIVALENTES OPEN SOURCE A IBM WEBSHERE BPM.....	78
Figura 14: Arquitectura de jBPM.....	79
Figura 15: Representación en jPDL del proceso “Realizar solicitud de crédito”	87
Figura 16: Formulario de solicitud de crédito	90
Figura 17: Respuesta de solicitud de crédito	90
Figura 18: Invocación de web service con Intalio BPMS	92

ÍNDICE DE TABLAS

CAPITULO 2: ANÁLISIS TÉCNICO Y METODOLÓGICO DE IBM WEBSPHERE BPM	48
Tabla 1: Comparación entre la notación utilizada por Websphere Business Modeler y BPMN.....	55
CAPÍTULO 3: ANÁLISIS TÉCNICO Y METODOLÓGICO DE LOS EQUIVALENTES OPEN SOURCE A IBM WEBSPHERE BPM	78
Tabla 2: Comparación entre BPMN y jPDL.....	84
Tabla 3: Comparación entre BPEL y jPDL.....	84

INTRODUCCIÓN

BPM (*Business Process Management* o Gestión de procesos de negocio) puede definirse como una metodología empresarial cuyo objetivo es mejorar la eficiencia a través de la gestión sistemática de los procesos desarrollados dentro de la organización. Dicha metodología ha adquirido una atención considerable recientemente tanto por las comunidades de administración de negocios como la de ciencia de la computación.

Los miembros de estas comunidades están identificados por diferentes soportes e intereses educacionales; entre ellos encontramos la gente de administración de negocios, que está interesada en mejorar las operaciones de las compañías. Hechos como incrementar la satisfacción de los clientes, reducir los costos de operación, y establecer nuevos productos y servicios a bajo costo son aspectos importantes del manejo de procesos de negocio desde el punto de vista de un administrador.

En la ciencia de la computación encontramos dos comunidades interesadas en BPM: los investigadores con una base de métodos formales, que se encargan de investigar propiedades estructurales de los procesos. Como estas propiedades pueden ser mostradas únicamente usando abstracciones de procesos del mundo real, las actividades de los procesos son reducidas usualmente a letras. Usando esta abstracción se pueden hacer observaciones sobre todo a las propiedades estructurales de los procesos, las cuales son muy útiles para detectar deficiencias estructurales en los mismos.

La comunidad de software está interesada en proveer sistemas robustos y escalables. Como los procesos de negocios son realizados en espacios de información tecnológica compleja, la integración de los sistemas de información existentes se convierte en una base importante para la implementación técnica de los procesos de negocio. Esta última consideración es fundamental para la imposición de la metodología en la industria [1].

A través del presente trabajo daremos un marco teórico sobre el cual sustentar BPM. Analizaremos los motivos existentes para insertar esta metodología en la gestión de procesos de la organización, así como también consideraremos los conceptos fundamentales que permiten realizar implementaciones reales basadas en BPM. Para ello se presenta un caso de estudio sobre el que se aplican los conceptos emergentes del análisis metodológico de las herramientas. Estos casos de estudio serán implementados en una herramienta de licenciamiento propietario, así como en otras opciones disponibles de tipo *open source*.

El aporte principal de este trabajo de tesis es proveer un análisis exhaustivo de la metodología BPM en implementaciones reales, considerando para ello las herramientas disponibles actualmente en el mercado.

Cabe mencionar, que a raíz de este trabajo de investigación se ha realizado la siguiente publicación:

- "Comparación del entorno IBM Websphere BPM y sus equivalentes funcionales en código fuente abierto" [Martinez Garro J., Bazán P., Lorenzón E.] aceptado para su publicación en XII Workshop de Investigadores en Ciencias de la Computación WICC. Universidad Nacional de la Patagonia San

Juan Bosco. Argentina. Mayo 2010. Con referato.
<http://wicc2010.info.unlp.edu.ar/>

La organización del trabajo está dada en 4 capítulos, en los cuales trataremos los siguientes temas

- Capítulo 1: Marco conceptual de BPM. Descripción de los conceptos teóricos y técnicos que conforman a dicha metodología.
- Capítulo 2: Análisis técnico y metodológico de la herramienta IBM Websphere BPM mediante el planteo de un caso de uso concreto. Perseguiremos el alcance de una aplicación orientada a procesos destinada a usuarios finales.
- Capítulo 3: Análisis metodológico y técnico de herramientas *open source* que tengan el perfil de un BPMS. Buscamos de esta manera obtener una funcionalidad equivalente a la alcanzada en el capítulo anterior con la herramienta IBM Websphere BPM.
- Capítulo 4: Conclusiones finales del trabajo. Planteo de posibles líneas futuras de investigación.

CAPÍTULO 1: MARCO CONCEPTUAL DE BPM

Motivación y definiciones.

BPM se basa en la observación de cada producto que la compañía provee al mercado, lo cual es una salida de un número de actividades ejecutadas. Los procesos de negocio son la clave para organizar estas actividades, y mejorar el entendimiento de sus interrelaciones.

La tecnología de la información en general, y los sistemas de información en particular merecen un rol importante en BPM, porque cada vez más actividades que realizan las compañías son soportadas por sistemas de información. Las actividades de los procesos pueden ser ejecutadas manualmente por empleados de la compañía, o con la asistencia de aplicaciones dedicadas a tal fin. También puede ocurrir que las actividades sean directamente ejecutadas por sistemas sin intervención humana.

Una compañía puede alcanzar sus objetivos de manera eficiente sólo si su gente y los sistemas de información se conducen en la misma dirección, siendo los procesos de negocio quienes facilitan esta colaboración.

En las compañías suele haber una brecha entre los aspectos organizacionales del negocio y la tecnología de información. Es importante hacer mínima esta brecha porque el mercado suele forzar a dar más y mejores productos a los clientes. Los productos que son exitosos hoy, pueden no serlo mañana. El mercado puede inclinarse hacia quien ofrezca el menor producto y que sea más barato.

En un nivel organizacional, los procesos de negocio son esenciales para comprender cómo opera una organización. Aunque también son importantes para el diseño e implementación de sistemas de información flexibles. Estos sistemas proveen la base para la creación rápida de nueva funcionalidad que cree nuevos productos, y también para adaptar ágilmente funcionalidad existente a requerimientos del negocio.

BPM está influenciada por distintos conceptos y tecnologías. Sus raíces nacen en la orientación a procesos de los años '90, cuando se propusieron nuevos modelos de organizar las compañías.

Se puede definir un proceso de negocio como una colección de actividades que pueden tomar distintas clases de entradas para producir una salida, la cual resulta de valor para el cliente.

Esta definición pone énfasis en el comportamiento basado en entradas y salidas, estableciendo sus pre y postcondiciones. Al hablar de colección de actividades no se establece ni un orden de ejecución, ni restricciones de ejecución, con lo cual es una definición bastante libre.

Se puede hablar ya de restricciones entre actividades al decir que un proceso es un conjunto de tareas lógicamente relacionadas ejecutadas para alcanzar una salida para un cliente particular o mercado [1].

El concepto de "lógicamente relacionadas" hace hincapié en las actividades del proceso, al asociar una salida con un cliente. En [1] también se establecen las restricciones de orden en la ejecución de actividades, al mencionar que existe un ordenamiento de actividades en tiempo y lugar, con un comienzo, un final y entradas y salidas claramente identificadas. Además los procesos tienen

clientes externos e internos, y ocurren entre distintas divisiones de la organización.

Así, un proceso de negocio consiste de un conjunto de actividades que son ejecutadas en coordinación dentro de un ambiente organizacional y técnico. Estas actividades alcanzan conjuntamente un objetivo de negocio. Cada proceso es ejecutado por una única organización, pero puede interactuar con procesos de otras organizaciones.

Así, BPM no cubre solamente la representación de procesos de negocio, sino también actividades adicionales. Se incluyen conceptos, métodos y técnicas para soportar el diseño, administración, configuración, representación y análisis de procesos de negocio.

La base de BPM es la representación explícita de procesos de negocio con sus actividades y las restricciones de ejecución entre ellas. Una vez que el proceso está definido se lo somete a análisis, y como resultado de este surge la mejora y ejecución.

Tradicionalmente los procesos pueden ser ejecutados manualmente, aunque con la interacción de sistemas de software para la coordinación de actividades. Estos sistemas son llamados sistemas de BPM, es decir, software genérico que maneja por representaciones de proceso explícitas para coordinar la ejecución de procesos de negocio.

Existen distintas notaciones para el modelado de procesos, aunque su esencia es similar. Las actividades en general son marcadas con rectángulos redondeados, y el orden de ejecución es representado por flechas directas. También se encuentran mecanismos para disparar concurrencias de actividades, y luego sincronizarlas en un punto.

Es importante distinguir que distintas ejecuciones de un mismo proceso pueden producir resultados distintos, con lo cual debemos distinguir entre modelos de procesos e instancias de procesos. Encontramos una relación uno a muchos entre un modelo y sus instancias.

Un modelo de proceso consiste de un conjunto de modelos de actividades y restricciones de ejecución entre ellas. Una instancia representa un caso concreto en el accionar operacional de la organización. Así encontraremos instancias de procesos e instancias de actividades.

Los modelos de procesos son los artefactos principales para implementar procesos de negocios. Su implementación puede ser efectuada con reglas y políticas de la organización, pero también puede ser hecha por sistemas informáticos, utilizando un sistema de manejo de procesos.

Debido a que los procesos son ejecutados en una única organización por definición, el ordenamiento de las actividades puede ser controlado por un sistema de BPM. De esta manera se realizan "orquestaciones" de procesos.

La interacción de un conjunto de procesos de negocio es expresada en una coreografía de procesos. Es importante señalar la ausencia de un agente central que controle la actividad, como en el caso de la orquestación. La coreografía sólo es manejada mediante intercambio de mensajes. De esta manera, para realizar interacciones correctas, los procesos interactuantes necesitan acordar una coreografía común.

Las representaciones gráficas de los procesos se enfocan en la estructura del proceso y en la interacción de los participantes, más que en aspectos técnicos de su realización. Este es un ítem importante en el modelado de procesos,

dado que la definición de los mismos y su comportamiento de interacción no prescriben estrategias de implementación o plataformas.

La intervención de participantes puede cambiar sin que esto afecte el comportamiento externamente visible del proceso [1].

Ciclo de vida de los procesos de negocio

El ciclo de vida de los procesos consiste en fases relacionadas unas con otras. Las mismas estas organizadas en una estructura cíclica mostrando sus dependencias lógicas. Estas no implican un estricto ordenamiento temporal en el cual las fases necesiten ser ejecutadas. Muchas actividades de diseño y desarrollo son conducidas durante cada una de estas fases, y son comunes también aproximaciones incrementales o evolutivas que involucran actividades concurrentes en múltiples fases [5].

Diseño y análisis

El ciclo de vida de los procesos se inicia en la fase de análisis y diseño, en la cual se indaga sobre los procesos de negocio, y sus ambientes técnicos y organizacionales. Basándose en esto, los procesos pueden ser identificados, revisados, validados y representados por modelos de procesos de negocio.

Expresar los modelos mediante notación gráfica facilita la comunicación sobre los mismos, de manera de agilizar el refinamiento y mejora.

En esta fase se usan técnicas tales como validación, simulación y verificación. El modelado de procesos constituye una subfase central dentro del diseño de los mismos. Este se realiza utilizando alguna notación gráfica una vez concluido el proceso de relevamiento y definidas las actividades de mejora.

Al cabo del desarrollo del diseño inicial, el mismo debe ser validado. Un método útil es el de *workshops*, debido a la discusión del diseño por distintos miembros. Básicamente lo que se busca es que todas las instancias del proceso estén reflejadas en el modelo acordado.

También se pueden usar técnicas de simulación para generar ciertas secuencias inesperadas en la ejecución. Además, la idea es poder ejecutar el proceso paso por paso para verificar que tenga el comportamiento esperado en cada etapa.

Configuración

Una vez que el modelo de proceso está diseñado y verificado es necesario implementarlo. Hay distintas maneras de hacerlo: una de ellas sería establecer políticas y procedimientos que los empleados ejecutarán. En este caso no se necesita ningún sistema BPM dedicado.

En caso de que se opte por utilizar un sistema BPM entonces hay que decidir la plataforma en la etapa de configuración. Así, el modelo deberá ser reforzado con información técnica que facilitará el manejo del mismo por parte del sistema BPM.

El sistema deberá ser configurado de acuerdo a la infraestructura de la organización y de los procesos de negocios reinantes. Esto incluye las

interacciones entre los empleados y el sistema como así también la integración de los sistemas existentes con el sistema de gestión de procesos. Esto último en particular es una etapa importante debido que la gran mayoría de las empresas tiene fases de sus procesos implementadas por sistemas que no se quieren desechar, sino que por el contrario, se quieren integrar.

La configuración de un sistema BPM también incluye un aspecto transaccional: se busca que cualquier interacción entre el sistema y la base de datos conserve el protocolo ACID. De esta manera es necesario definir propiedades transaccionales a nivel de los procesos. Un subconjunto de actividades dentro del proceso forma una transacción, de manera de asegurar la ejecución atómica del conjunto.

Por otro lado es necesario señalar que el comportamiento tradicional de transacciones no es una alternativa posible para procesos, ya que no es posible establecer cerramientos para los objetos de datos. Las mismas aún se encuentran en fase de investigación.

Una vez que el sistema ha sido configurado es necesario testarlo para observar que efectivamente presente el comportamiento esperado. Aquí se usan técnicas típicas de *testing* en Ingeniería de software. Los aspectos más importantes son la integración y la performance a nivel de los procesos.

Una vez terminada la fase de *testing*, el paso siguiente es ejecutar el *deploy*. En este último caso puede requerirse además entrenar personal o realizar migración de datos al sistema destino.

Representación

Una vez que se completó la fase de configuración, las instancias de los procesos se pueden representar. Esta etapa abarca el tiempo de ejecución de las instancias. Así, se inician por algún evento instancias de los procesos.

El sistema BPM controla la ejecución de las distintas instancias de los procesos como fue definido en el modelo. El monitoreo es útil para poder conocer la fase en que se encuentran distintas instancias de un mismo proceso.

Se debe poseer información detallada del estado en que se encuentra el proceso. En general hay distintas maneras de efectuar la visualización del estado del proceso, por ejemplo que una instancia que ya se ha completado se ve en color gris, mientras que otra que aun no lo ha hecho se ve azul. La información de monitoreo está basada en los estados de las actividades de los procesos.

La información propia de la ejecución puede volcarse a archivos de *log* (bitácoras), los cuales pueden estar ordenados por lo eventos ocurridos (el inicio o fin de una tarea, etc.). Los resultados son utilizados para la fase de evaluación.

Evaluación

En esta etapa se usa información disponible en los *logs* para lograr evaluar y mejorar los modelos y sus implementaciones. Los *logs* se inspeccionan utilizando monitoreo de actividades y minería de procesos, mediante los cuales

se puede identificar la calidad de los modelos y la adecuación de los mismos al ambiente de ejecución.

El monitoreo de actividades pueden permitir saber por ejemplo si una tarea está consumiendo demasiados recursos, o si está llevando más tiempo del esperado.

Administración

Existen distintos escenarios en el manejo de procesos que deben ser organizados y manejados correctamente. Se necesita entonces almacenamiento estructurado y un retorno eficiente de los artefactos incluidos en el modelo de procesos. En las grandes organizaciones que poseen una alta cantidad de procesos, se necesita un repositorio con mecanismos de consulta eficientes. Los roles que pueden estar interesados en este dominio son:

- **Oficial jefe de procesos:** es el responsable de estandarizar y armonizar los procesos en la empresa. Además, es quien se encarga de la evolución de los procesos ante cambios en los requerimientos.
- **Ingeniero de negocios:** son los expertos en el dominio del negocio, responsables de definir objetivos de los procesos. Suelen no tener formación técnica con lo cual es necesario manejar notaciones simples para lograr una comunicación efectiva.
- **Diseñador de procesos:** son los responsables de modelar los procesos comunicándose con los expertos y otros roles necesarios. Se necesitan capacidades de análisis y comunicación.
- **Participantes de procesos:** son los que conducen el flujo operacional del proceso. Son útiles en la fase de diseño pues son quienes conocen las actividades y las interrelaciones entre las mismas y con otros participantes. El diseñador debe convertir esta información útil en algo coherente.
- **Trabajador con conocimiento:** son quienes usan los sistemas de software en la ejecución de los procesos. Poseen conocimiento detallado del dominio de la aplicación, y pueden ejecutar actividades con autonomía.
- **Responsable del proceso:** cada modelo tiene asignado un responsable, sobre quien recaerá la ejecución correcta y eficiente de todos los procesos de negocios usando este modelo. Debe detectar ineficiencias y mejorarlas, en colaboración con los participantes y diseñadores del proceso.
- **Arquitecto del sistema:** son los responsables de desarrollar y configurar los sistemas BPM, para permitir la representación de los procesos en el contexto de infraestructura tecnológica que se posea.
- **Desarrolladores:** son profesionales de IT que crean artefactos de software necesarios para implementar los procesos. También se encargan de crear interfaces a sistemas existentes.

Todos estos roles deben cooperar para diseñar los procesos y desarrollar soluciones adecuadas que permitan representarlos [1].

Clasificación de los procesos de negocio

Organizacional vs. Operacional

Se pueden encontrar distintos niveles entre las estrategias de alto nivel y los procesos implementados. En el nivel más alto encontramos la especificación de la estrategia de la compañía, la cual debe apuntar de una manera sustentable a obtener una ventaja en el mercado.

En el segundo nivel, la estrategia de negocios ya desciende a objetivos operativos. Al organizar estos podemos encontrar subobjetivos.

En un tercer nivel ya podemos hallar procesos organizacionales. Son procesos de alto nivel, generalmente especificados de manera textual con sus entradas, salidas, resultados esperados y dependencias con otros procesos. Usualmente en estos altos niveles se suelen emplear metodologías informales o semiformales para expresar los procesos y sus componentes.

Dado un proceso organizacional, en general se van a requerir múltiples procesos operacionales al servicio de este. En los procesos operacionales se especifican las actividades y sus relaciones, pero se desechan aspectos de implementación del proceso. Aquí, se usan modelos de procesos para especificar los procesos operacionales. Estos son la base para comenzar a implementar los procesos de negocio, donde los que están ya implementados poseen información sobre la ejecución de las actividades, y del ambiente técnico y organizacional en el que son ejecutados.

Existen así, distintas maneras de implementar procesos, que van desde procedimientos escritos y políticas de la organización hasta el uso de plataformas de representación. En cualquier caso, los procesos implementados refieren a una especificación que permite la representación de los procesos en una plataforma determinada, sea organizacional o técnica.

Grado de automatización

Los procesos de negocio pueden no converger en el nivel de automatización. Existen procesos que se encuentran completamente automatizados, o sea que no requieren intervención humana en ninguna fase. Por ejemplo, el hecho de sacar un ticket para una aerolínea vía web.

También existen múltiples procesos que requieren actividades manuales, pero también pueden incluir actividades automatizadas.

Grado de repetición

Los procesos se pueden clasificar por su grado de repetición: si el grado de repetición es alto son importantes las inversiones en modelamiento y soporte de representación automática, debido a que múltiples instancias del proceso se benefician con esto. En otro extremo encontramos procesos que ocurren solo unas pocas veces. En general puede ser cuestionable esforzarse en el

modelamiento de estos procesos, debido a que el costo por modelar cada instancia es muy alto.

Los procesos son colaborativos cuando requieren mejorar la colaboración entre las personas involucradas. El objetivo del modelamiento y representación no es solo la eficiencia sino también poder efectuar una traza de que es lo que efectivamente se ha hecho al momento, y que relaciones causales entre las actividades realmente se dieron.

En general los procesos con un bajo grado de repetición no están completamente automatizados, y tienen un alto grado de colaboración. Por estas razones suele no requerirse de soluciones automatizadas.

Grado de estructuración

Los procesos son estructurados cuando su modelo prescribe las actividades y sus restricciones de ejecución. Las diferentes opciones de decisión que deberán ser hechas en la representación del proceso deben ser definidas en tiempo de diseño.

Según [1], Leymann y Roller organizaron los procesos de acuerdo a la dimensión de su estructura y repetición. Los *workflows* de producción son bien estructurados y altamente repetitivos.

Los procesos estructurados pueden volverse un obstáculo en caso que los participantes sean trabajadores con conocimiento y aun así ejecuten tareas del proceso. Esto se da pues se pueden saltar ciertas etapas, o etapas que debían ser ejecutadas secuencialmente se ejecutan concurrentemente.

En general para evitar estos problemas suelen usarse actividades ad hoc para dar soporte a sectores desestructurados dentro de los procesos.

Objetivos, estructura y organización

El objetivo máximo de BPM es una mejor comprensión de las operaciones que la compañía ejecuta, y sus relaciones. El núcleo de esto es la representación explícita de los procesos.

La representación explícita permite a los participantes comunicarse acerca de los procesos de una manera eficiente. Esto facilita el análisis, y se pueden efectuar potenciales mejoras.

La clave operacional de BPM es la flexibilidad. Los motivos de cambio pueden ser diversos, y BPM no solo soporta cambios en el ámbito organizacional de la empresa, sino que facilita cambios en el software subyacente, sin que esto provoque un cambio en todos los procesos.

El repositorio de procesos de la organización captura conocimiento de cómo la empresa efectúa sus negocios. Además debemos notar la óptica de continua mejora de los procesos, lo cual permite cambios positivos que indiquen evoluciones en los negocios empresariales.

Se apunta también a reducir la brecha entre los procesos que la empresa realiza y la ejecución de los mismos en software [1].

Evolución de las arquitecturas de sistemas empresariales

Estas arquitecturas están mayormente compuestas por sistemas de información, los cuales pueden ser considerados como sistemas especializados, tales como los de telefonía, autos, etc. La evolución en estos sistemas debe ser analizada con el concepto de separación de patrones dada por Dijkstra.

Es un concepto que tiene diversas aplicaciones, y en el caso del diseño de sistemas de software se aplica a la identificación de conjuntos funcionalmente relacionados. Se los coloca en un paquete (sea un subsistema) con responsabilidades e interfaces claramente establecidas. Este concepto facilita la reusabilidad de componentes. Además agiliza los cambios y reduce el impacto de los mismos.

El segundo principio involucrado es el ocultamiento de información introducido por Parnas [1]. Los cambios que pueden surgir en el mercado, en la legislación, en la tecnología, deben en general ser reflejados por los sistemas. Los cambios deben poder absorberse sin que esto impacte en forma negativa en la forma en que exteriormente se observa al sistema.

Una arquitectura de software define una estructura que organiza los elementos de un sistema de software y sus recursos. Ambos están representados por subsistemas. En una arquitectura, estos subsistemas pueden tener responsabilidades y relaciones con otros subsistemas.

La arquitectura de software no hace referencia a la estructura interna de los subsistemas, es decir que a su vez estos pueden tener su propia arquitectura interna [1] [5].

Desarrollo de aplicaciones tradicionales

En los principios de la programación las aplicaciones eran desarrolladas de cero, sin reutilizar componentes o granularidad. De esta manera se desarrollaban módulos idénticos múltiples veces, resultando así un proceso costoso e ineficiente. También resultaba muy complicado mover una aplicación de una máquina a otra.

Luego surgieron los sistemas operativos como una centralización en la administración de los recursos de sistemas, y como una interfaz entre las distintas aplicaciones de usuario y los recursos del sistema.

Más tarde comenzó a utilizarse el almacenamiento de datos, pasando desde el manejo individual de los datos por cada aplicación hasta la creación de subsistemas dedicados para el intercambio de datos. En este último ítem encontramos a las bases relacionales, las cuales proveen los principios de independencia física y lógica de los datos, limitando así la cantidad de cambios necesarios en las aplicaciones ante cambios surgidos en las estructuras de datos.

Posteriormente a esto surgen las aplicaciones con claro interés puesto en la interfaz gráfica, lo cual facilita la interacción con el usuario. Surgen paralelamente a esto empleados con gran conocimiento y familiaridad con las aplicaciones y el dominio, siendo estos los destinatarios de interfaces cada vez más amigables.

Las aplicaciones empresariales y su integración

Basándose en los sistemas de bases de datos relacionales y en las interfaces orientadas a usuarios se han desarrollado gran cantidad de aplicaciones. Muchos de estos sistemas son aplicaciones empresariales. Cuando convivían distintas aplicaciones de este tipo se buscó crear un nuevo *middleware* que provea la integración de las mismas. Estos fueron los sistemas integrados de aplicaciones empresariales. Surgieron con esto numerosos problemas con aplicaciones distintas que debían manejar la misma estructura de datos, debido a que como muchas veces quedaban replicadas estructuras idénticas, se producían muchas inconsistencias. Como contraposición a esto surgen los ERP.

Sistemas de planeamiento de recursos empresariales (ERP)

El gran objetivo de los ERP es proveer una base de datos integrada que se extiende por distintas partes de la organización, a la cual se accede desde varios puntos de la misma, generalmente a través de una estructura que responde a la arquitectura cliente servidor.

Con el correr del tiempo surgieron sistemas de manejo de cadena de suministros (SCM) y sistemas de manejo de relaciones entre clientes (CRM). El máximo objetivo de los sistemas SCM es soportar el planeamiento, operación y control de canales de suministro, incluyendo manejo de inventario, de proveedores y distribuidores, y planeamiento de demanda.

Un problema frecuente es que en las organizaciones se poseen distintos sistemas que no poseen la misma base de datos que el sistema corporativo, con lo cual se produce redundancia de información, y así se vuelve a repetir el problema de años atrás: al presentarse información redundante es necesario actualizar en forma distribuida cada una de las bases de datos.

También podemos encontrarnos con el escenario de un *call center*, por ejemplo, en que los clientes llaman para consultar su información. Si el mismo no puede acceder a un entorno en que su estado esté completamente integrado, puede originarse algún desfasaje. O sea, todo proviene de manejar transacciones en las cuales se decide almacenar información redundante en sistemas que no guardan relación entre sí, aun teniendo por ejemplo una conexión a la misma red. La primera forma de plantear una integración de estos datos sería en forma manual por parte del cliente, lo cual obviamente es propenso a errores.

En las grandes organizaciones se optó por desarrollar una aplicación integrada, lo cual no es aplicable al nuevo contexto debido al tamaño. Ahora el problema es desarrollar un sistema de integración de aplicaciones empresariales.

Integración de aplicaciones empresariales

Las empresas enfrentan el desafío de integrar sistemas que se han desarrollado en distintas plataformas por años. Muchos sistemas fueron desarrollados en forma independiente, cada uno con sus bases de datos respectivas. Los problemas de heterogeneidad de datos se dan cuando unidades de datos se almacenan múltiples veces en aplicaciones diferentes.

Los datos de cliente son almacenados en un sistema ERP, y en un sistema de manejo de relación de clientes. Las estructuras de ambos sistemas no tienen porqué ser iguales, ni inmediatamente comparables.

Un primer nivel de heterogeneidad se da en cuanto a diferencias en los nombres de los campos entre ambos sistemas. El segundo nivel se refiere en cuanto a la semántica de los atributos. Las tecnologías de integración de datos apuntan a resolver estos temas sintácticos y semánticos.

Integración punto a punto

Se suelen observar múltiples aplicaciones que necesitan ser integradas, frecuentemente varias instancias de un tipo específico de aplicación, como un ERP, que en muchos casos ejecutan distintas versiones del software.

La tecnología de integración de aplicaciones está basada en *middleware* ya conocido. La idea es tomar ventaja de estas tecnologías para integrar los datos presentes en plataformas heterogéneas, además de los procesos que estos sistemas realizan.

Hay que tener en cuenta que la integración de estas aplicaciones requiere esfuerzos de diseño e implementación considerables. Así, unir cada par de aplicaciones requiere un esfuerzo de $N \times N$, o sea que se requieren un total de N^2 interfaces para integrar las aplicaciones.

En la computación empresarial los cambios son constantes, y se espera que la arquitectura de un sistema responda a los mismos en una forma eficiente y efectiva. La integración punto a punto no responde bien a los cambios, y la razón es la intensa conexión de las interfaces. Cualquier cambio puede requerir reconfigurar las interfaces, y esto conlleva mucho esfuerzo.

Una plataforma de implementación de la integración es el intercambio de mensajes. Vemos que no es necesario un control centralizado debido a la interconexión entre todos los componentes, aunque sí es necesario garantizar la distribución de los mensajes. De cualquier manera este problema no está resuelto debido a que cualquier cambio en la configuración de aplicaciones requiere efectuar una modificación en la estructura de comunicación.

Se suelen usar colas de mensajes para garantizar la distribución de los mismos. El cliente usa una aplicación que integra una determinada cantidad de sistemas llamada aplicación de integración. Esta aplicación manda un mensaje a otra aplicación que puede por ejemplo ser un ERP.

De esta manera, el mensaje es insertado en la cola de mensajes del sistema ERP, este lo recibe e invoca la funcionalidad requerida. Esto produce resultados que son enviados a la aplicación de integración, por medio de un comunicado colocado en la cola de mensajes respectiva. Una vez recibido, la aplicación de integración prepara un mensaje y lo envía al sistema de manejo del canal de suministro. Efectivamente se efectúa una conexión punto a punto debido a que quien envía el mensaje debe codificar el receptor en el mismo. Igualmente, el problema de la arquitectura punto a punto en cuanto a la adaptación frente a los cambios no disminuye ante el intercambio de mensajes. La cooperación de aplicaciones puede darse en el sistema de integración. Una aplicación se encarga de implementar el proceso que indica como debe realizarse la cooperación, lo cual implica la no existencia de un modelo de proceso que permita facilidad a la hora de la comunicación y de los cambios.

Integración basada en un nodo central y repetidores

Este paradigma está basado en un nodo central y un conjunto de repetidores directamente asociados a este, aunque los repetidores no se encuentran conectados entre sí. De esta manera, el *middleware* está representado por la integración centralizada de aplicaciones, y las aplicaciones que deben ser integradas son reflejadas por los repetidores. Las mismas interactúan entre ellas por medio de la integración centralizada de aplicaciones.

Una característica importante es el hecho de que quien envía el mensaje no necesita codificar el receptor del mismo y cada mensaje se envía al nodo de integración de aplicaciones. Así, el *hub* está configurado de manera tal que la estructura del mensaje puede ser usada para detectar el/los receptor/es del mensaje.

La principal ventaja se da en el hecho de que la cantidad de conexiones se puede disminuir. Si queremos conectar N aplicaciones, no necesitamos más de $N \times N$ conexiones. Debido a que cada aplicación está ligada al *hub* central, N interfaces van a ser suficientes. Por estas últimas, las relaciones específicas entre las aplicaciones se pueden reflejar en la configuración del *middleware*.

Se busca en general que el nodo central provea adaptadores que oculten la heterogeneidad de las distintas aplicaciones. Así, cada aplicación requerirá el desarrollo de un adaptador dedicado para unirse al *hub*. Esto último puede requerir muchos recursos, aunque es necesario para que las aplicaciones puedan interactuar entre sí.

Desde un punto de vista técnico, se pueden usar “agentes” como mensajeros para implementar esta arquitectura. Son sistemas de software que permiten a los usuarios definir reglas de comunicación entre aplicaciones, posibilitando desacoplar de las aplicaciones la implementación y el cambio en los canales de comunicación. La forma de definir como se efectuará la comunicación es declarativa, por lo cual la implementación se efectúa mediante la declaración de estructuras de comunicación. El nivel de respuesta ante los cambios se mejora debido a que el emisor del mensaje no debe implementar los cambios en forma local, pudiendo estos ser especificados de una forma declarativa en el nodo central.

El nodo central usa reglas para manejar las dependencias entre aplicaciones, y basándose en estas puede usar información sobre la identidad del emisor, el tipo del mensaje y el contenido del mismo para decidir en qué cola colocarlo. Además de efectuar la distribución de los mensajes, los agentes también realizan tareas de transformación de los mismos con el propósito de efectuar mapeo de datos entre las aplicaciones, para manejar adecuadamente los problemas de heterogeneidad. Estas transformaciones son realizadas por medio de los adaptadores de las aplicaciones.

Cada aplicación está ligada a un agente, y en particular al componente de evaluación de reglas del mismo. Ante la recepción de un mensaje, el agente evalúa las reglas y decide en que cola colocar el mensaje.

Las colas se usan para garantizar la recepción de los mensajes, y además cualquier cambio en la comunicación se maneja a través del agente. No se requiere mucho esfuerzo para reflejar cambios, solo basta redefinir las reglas.

La publicación y suscripción es un mecanismo para unir las aplicaciones a los agentes, buscando que las aplicaciones puedan adherirse a determinados mensajes o tipos de mensajes, pudiendo las mismas publicar mensajes

también. Esta información enviada por la publicación y la suscripción es utilizada por el nodo central para la retransmisión de mensajes.

Usando esta arquitectura de agente se puede desarrollar un sistema de integración. Este se comunica con el agente y con las aplicaciones de *back end* vía el agente y los adaptadores de las propias aplicaciones.

Las aplicaciones pueden interactuar de formas muy diversas, desde invocaciones simples hasta interacciones complejas entre múltiples aplicaciones. Estas últimas consisten de una serie de actividades representadas por una invocación a una aplicación, además de existir restricciones de ejecución entre las mismas. Las restricciones pueden estar determinadas por dependencias de datos, por lo que una función en el ERP puede ser iniciada solamente cuando se extrae determinada información del cliente del sistema de manejo de relaciones del cliente.

Este esquema de agentes y mensajes tiene ciertos inconvenientes. El primero de ellos es que el agente contiene cierta lógica, oculta en las reglas. La programación de estas puede volverse una labor compleja por las dependencias que pueden darse entre las mismas, y el hecho de cambiar una regla puede tener implicancias en el comportamiento global del sistema.

La razón principal de estos problemas es la “pérdida” conceptual que se da en la integración de aplicaciones, ya que tanto la integración de datos y procesos requieren gran actividad de programación y configuración de bajo nivel, tanto de adaptadores como de los agentes de mensajes.

La integración de datos suele darse mediante actividades de mapeo, lo cual requiere un modelo de datos acordado entre todas las aplicaciones y que reside en los agentes. Este modelo global suele no ser explícitamente desarrollado, pero es común encontrarlo oculto en las reglas de mapeo de datos efectuadas por los adaptadores.

En escenarios típicos de integración de aplicaciones, la funcionalidad de los sistemas de integración está organizada en forma de un proceso. Este proceso consiste en la ejecución de un conjunto de actividades con restricciones de ejecución apuntando a la obtención de un objetivo. En estos escenarios la representación de los procesos está embebida en las reglas que los agentes manejan, y parece ser más apropiado alcanzar una representación explícita del mismo.

Ya en el próximo nivel de los sistemas de información encontramos el manejo de *workflow*, lo cual identifica especificaciones de procesos que contribuyen a resolver el proceso de integración de aplicaciones. Igualmente, antes de entrar en el manejo de *workflow* encontramos las influencias dadas por la administración del negocio, lo cual impacta en el modelo empresarial y la orientación a procesos [1] [5] [25].

Modelo empresarial y orientación a procesos

La administración del negocio influye también a la arquitectura BPM, encontrando en relación a esto dos conceptos importantes: los canales de valor, que permiten considerar funcionalmente las actividades de la empresa y verificar su contribución al éxito económico de la organización; y por otro lado

la orientación a procesos como la forma de organizar las actividades empresariales.

Canales de valor

Son un mecanismo para organizar el trabajo de una compañía de manera de conducirlo a alcanzar sus objetivos de negocio.

Su creador, Porter, habla de que la configuración de cada actividad encarna la manera en que la misma es ejecutada, incluyendo los recursos humanos e intelectuales utilizados, junto con los arreglos organizacionales.

La cooperación que se da entre las distintas organizaciones para alcanzar sus objetivos hace que interactúen sus canales de valor, formando un ecosistema llamado sistema de valor. Así, el canal de valor de una organización es una estructura rica internamente, representada por funciones de grano grueso, las cuales pueden ser descompuestas en funciones de grano fino por medio de mecanismos de descomposición funcional.

Todas las funciones que una compañía ejecuta deben contribuir al éxito de la organización. Las funciones primarias de un canal de valor son:

- Logística de entrada: garantizan la recepción de material, información y otros recursos requeridos para la ejecución del negocio. Las logísticas de entrada interactúan significativamente con pares del negocio para coleccionar y seleccionar ofertas, negociar contratos, organizar el transporte y manejar bienes e información entrantes.
- Operaciones: agregan funcionalidades responsables de darle valor agregado a los productos generados por la compañía. En caso de compañías de manufactura, los productos son generados por la función de operación de negocios.
- Logística de salida: una vez que los productos se encuentran realizados se requiere distribuirlos a almacenes u otros centros para que puedan finalmente llegar a los clientes.
- Marketing y ventas: se organizan funciones para colocar los productos en los mercados y venderlos. La función más básica es organizar y conducir una compañía a colocar un nuevo producto en el mercado.
- Servicios: una vez que el producto es vendido, la empresa necesita mantener el contacto con los clientes, tanto por cualquier falla que pueda haber con el producto como para recabar información para futuras realizaciones comerciales.

Las relaciones entre las funciones de negocio que una empresa ejecuta en el contexto de un sistema de valor son identificadas y capturadas en un proceso de negocio. De esta manera, debido a la complejidad de las grandes organizaciones, la granularidad de los procesos de negocio debe estar alineada con los objetivos de la función de negocio particular representada por el proceso [1].

Procesos organizacionales de negocio

En los 90's se consideró la orientación a procesos como un mecanismo potente, no solo para la representación de las actividades desarrolladas en la empresa sino también para estudiar y mejorar las relaciones entre las actividades.

La aproximación general de la reingeniería de procesos de negocio es una visión holística de una empresa, donde los procesos son el instrumento principal para ordenar las operaciones de la misma. Esta reingeniería se basa en el entendimiento que los productos y servicios que una empresa ofrece al mercado, son provistos a través de procesos de negocio, y rediseñar los mismos bajo un enfoque radical puede ser el camino al éxito.

La orientación a procesos surge básicamente como un concepto para introducir eficiencia. Este enfoque divide grandes actividades de trabajo complejo a tareas de baja granularidad, las cuales pueden ser ejecutadas por sectores especializados.

Utilizar directamente la adaptación en las organizaciones modernas resulta ineficiente debido a que los pasos dentro de un proceso están relacionados unos con otros, por lo que se requiere de información contextual durante todo el ciclo. Además se agrega la complejidad de los distintos actores intervinientes en cada una de las etapas del proceso.

La perspectiva de procesos busca combinar múltiples unidades de trabajo de baja granularidad para formar áreas de mayor granularidad, y así lograr reducir la carga de trabajo. La desventaja de este enfoque es la necesidad de poseer empleados con conocimientos suficientes del dominio.

Desde un punto de vista organizacional, la orientación a procesos caracteriza las operaciones de una empresa usando procesos de negocio. Como en realidad estos son enfoques diferentes, el punto en común de ambos es que los procesos de negocio de alto nivel son expresados de manera informal, buscando así que toda organización no posea más de una docena de procesos organizativos. Se suele usar para los mismos la misma codificación que para los canales de valor, aunque los niveles de abstracción son diferentes.

Los sistemas de información son recursos que los empleados con cierto bagaje pueden utilizar y sacar ventaja de los mismos. Uno de los aspectos principales de la reingeniería de procesos es combinar funciones de baja granularidad conducidas por varias personas, en unidades de mayor granularidad, y apuntar a dar soporte a los trabajadores con conocimiento para ejecutar estas tareas con sistemas de información dedicados.

BPM se basa en la gestión de los recursos de la organización, sobre todo de los sistemas de información, debido a la cualidad de los mismos de permitir a los actores del proceso ejecutar las actividades de manera eficiente. Estos sistemas tienen implicancias a su vez en los procesos de negocio, ya que estos últimos no serían posibles sin el soporte de los sistemas.

Los distintos interesados en los procesos son muy importantes, sobre todo en los procesos organizativos. Pero la influencia también se da en sentido inverso. Cuando hablamos de los distintos interesados nos estamos refiriendo a socios, clientes, y sobre todo el personal de la organización.

Los procesos organizacionales son influidos por ciertas actividades que la compañía ejecuta, como por ejemplo la administración, organización, control y optimización de los procesos de negocio.

Dentro de la administración y organización encontramos actividades que permiten la identificación de los procesos, como la selección de roles y personas responsables, y el despliegue de los procesos dentro de la organización. También se incluirían en esta etapa la designación del equipo de manejo del proceso y del “jefe” del proceso.

Debido a que cada proceso contribuye a alcanzar uno o más objetivos de la organización, es necesario monitorear las actividades para ver la eficiencia del proceso y que metas se han alcanzado. Aquí se determinan entonces indicadores clave de performance, ya sean indicadores técnicos como tiempo de respuesta promedio, como así también indicadores de aspectos específicos de dominio, tales como la reducción de la tasa de error o de costos.

El área de control desarrolla métodos para medir indicadores de performance e instalarlos en los procesos operacionales, de manera tal de identificar deficiencias y lograr corregir y mejorar el proceso.

Los procesos organizacionales son de alta granularidad, e involucran gran cantidad de personas y actividades; suelen ser descriptos mediante mecanismos textuales, incluso formas orientadas a formularios. Los elementos de estos formularios incluyen el nombre del proceso, una persona responsable, los objetos direccionados, las entradas y resultados del proceso, proveedores y clientes, ambos de los cuales son procesos organizacionales también.

Igualmente, es necesario aclarar que a este nivel los procesos son vistos como cajas negras, es decir que no se dan detalles sobre su estructura interna. Tampoco se consideran en este punto las restricciones de ejecución sobre las actividades del mismo, lo cual se aplica en un nivel de procesos operacionales.

Para poder representar las relaciones entre los procesos organizativos, se los suele representar a través de un mapa, donde cada proceso es graficado como un bloque, y sus dependencias con flechas. Existen distintos tipos de dependencias, como transferencia de información y de bienes físicos.

Un aspecto de cuidado son las interfaces de los procesos organizativos: se corre el riesgo de que sean poco claras o ineficientes. Estas interfaces serán descompuestas en interfaces de los procesos organizativos, que son los que efectivamente ejecutan el proceso organizativo [1].

Procesos B2B

Hemos visto ya que los sistemas de valor representan colaboraciones entre canales de valor de múltiples compañías. Estas colaboraciones de alto nivel son realizadas por interacción de procesos, cada uno de los cuales corre en una compañía distinta, en un escenario B2B (empresa a empresa). En cada canal de valor interviniente suele haber un participante en la colaboración B2B, lo cual forma una estructura interna y denota su contribución a la colaboración. Cabe analizar por ejemplo, si los procesos B2B creados mediante la unión de un conjunto de procesos existentes realmente cubren los requerimientos. Cuestiones a chequear son la ausencia de *deadlock*, criterios estructurales, etc. El problema puede estar agravado por el hecho de que los procesos internos son un recurso importante dentro de las empresas. Debido a que muchas empresas no desean exponer sus procesos internos al exterior, la colaboración B2B no se puede basar en el detalle local de los procesos sino en el comportamiento visible externamente de los mismos, y los modelos asociados que los representan.

Manejo de workflow

Los desarrollos en arquitectura de software empresarial y en BPM están relacionados con el manejo de *workflow*. El logro principal que se busca alcanzar aquí es la representación explícita de las estructuras de los procesos a través de modelos, y la representación controlada de los procesos basándonos en los modelos creados con anterioridad.

La aproximación a través del manejo de modelos facilita un mayor grado de flexibilidad, debido a que los modelos pueden ser adaptados a nuevos requerimientos, y las modificaciones resultantes se pueden usar inmediatamente para representar los procesos de negocio.

El término *workflow* consiste en la automatización de un proceso de negocio, en su totalidad o en parte, y en el cual se intercambian documentos, información o tareas de un participante a otro, para provocar la acción de acuerdo a un conjunto de reglas procedimentales.

Un sistema de manejo de *workflow* es un sistema que permite definir, crear y manejar la ejecución de flujos de trabajo a través del uso de software, que corre en uno o más motores, y que es capaz de interpretar la definición del proceso, interactuar con los participantes del *workflow*, y donde sea requerido invocar el uso de herramientas y aplicaciones IT.

La tecnología de *workflow* es capaz de soportar procesos de negocio dentro de un sistema dado o dentro de un conjunto de aplicaciones, lo que permite efectivamente integrar estos sistemas. Sin embargo esta tecnología posibilita también representar procesos en los que hay seres humanos activamente involucrados, y así mejorar la colaboración entre los trabajadores con conocimiento.

Workflows y aplicaciones

Tradicionalmente los sistemas se diseñan e implementan no solamente mediante la codificación de funciones sino también mediante un establecimiento del orden de estas funciones, es decir, la lógica del proceso realizado por la aplicación.

Con una complejidad creciente y una demanda mayor de adaptación del sistema a nuevos requerimientos, la codificación de la lógica del proceso tiene un severo inconveniente: cualquier modificación realizada por la aplicación requiere una modificación del código fuente. El código no solo necesita ser modificado sino testeado y mantenido, con lo cual se requieren muchos recursos.

La tecnología de manejo de *workflow* puede ser utilizada para facilitar la modificación de la lógica del proceso realizado por aplicaciones. Las funciones de una aplicación son pasos en el *workflow*, y cada componente usa un modelo de *workflow* para representar las funciones. Por la modificación de la lógica del proceso especificada en los modelos de *workflow*, se puede modificar el comportamiento de las aplicaciones sin codificar.

Hoy en día, la mayor cantidad de aplicaciones empresariales, como las aplicaciones de planeamiento, poseen un componente *workflow* que facilita la

adaptación flexible de los procesos de negocio dentro de estos sistemas. Observar que usamos el término “componente de *workflow*” además de “sistema de manejo de *workflow*”, debido a que un componente no es un sistema aislado sino que está embebido en la aplicación.

Podemos concluir que una aplicación de *workflow* única consiste de actividades y su correspondiente ordenamiento causal y temporal, donde las mismas son realizadas por un sistema común. Los *workflow* de aplicación múltiple contienen actividades que son realizadas por sistemas de múltiples aplicaciones, proveyendo así una integración de las mismas.

En el caso de los *workflow* de aplicación única, hay un componente dedicado que es alimentado con modelos de *workflow* que capturan la lógica del proceso así como información de ejecución técnica. Este componente usa funciones realizadas por la aplicación y provee procesos a un nivel más alto, que es la interfaz gráfica de usuario.

En el caso de *workflows* de múltiples aplicaciones, un sistema de manejo de *workflow* dedicado se asegura que las aplicaciones sean invocadas como fue especificado en el modelo de proceso. También se encarga de la transferencia de datos entre aplicaciones.

La integración de aplicaciones es efectuada por el sistema de manejo de *workflow*, usando adaptadores similares a los que se usan en un ambiente tradicional de aplicaciones empresariales.

Workflows de sistema

En los *workflows* de sistema las actividades son ejecutadas automáticamente por software. De esta manera los trabajadores con conocimiento no interactúan con las aplicaciones, y suelen no requerirse interfaces de usuario. Las restricciones de ejecución se especifican en un modelo de proceso, y el sistema de manejo de *workflow* se asegura que el orden de invocaciones se haga de acuerdo a dicho modelo.

Un *workflow* de sistema consiste de actividades que son implementadas vía software sin usuarios involucrados.

Los escenarios de integración de aplicaciones empresariales son candidatos típicos para *workflows* de sistema. El diseño e implementación de *workflows* de los mismos puede ser clasificada como programación de alto nivel, donde la funcionalidad provista por aplicaciones caracteriza la construcción de bloques que son organizados dentro de un *workflow* de sistema.

En plataformas de integración de aplicaciones empresariales sin un componente de proceso dedicado, la interacción entre las aplicaciones se representa mediante reglas, las cuales son utilizadas para enviar mensajes basándose en su tipo o contenido. De estas reglas no se puede deducir la totalidad del proceso fácilmente, además de que efectuar cambios puede resultar sumamente engorroso, debido a que una regla puede desencadenar otras reglas, y de esta manera obtener efectos no deseados.

Se pueden usar técnicas para el modelado de procesos, a fin de proveer una representación explícita de las relaciones entre las aplicaciones. Los modelos de proceso proveen la base conceptual para definir cuando y bajo qué

condiciones se invocan actualmente las aplicaciones en el contexto del escenario de integración. Por todo esto, podemos concluir que lo más adecuado es un componente de procesos dedicado para el modelado y representación de los procesos.

Workflows de interacción humana

Para introducir este concepto es útil discutir su desarrollo. Un precedente importante son los sistemas de automatización de oficina, desarrollados por los años '80. El objetivo de este tipo de sistemas era dar soporte a la organización, y lograr colaboración en el trabajo que incluía varias personas. Esto surge debido a que no es suficiente dar soporte a actividades individuales sino que es necesario considerar las relaciones entre las distintas actividades que se desarrollan, además de la colaboración entre las personas que intervienen.

Para esta colaboración se manejan repositorios de datos consolidados, y junto con estos y la mejora de entregas de resultados entre los intervinientes, se obtienen grandes mejoras.

En la actualidad, los *workflows* de interacción humana realizan partes de extensos procesos que poseen partes automatizadas y otras no. El objetivo es dar soporte a partes automatizadas mediante el control activo de las actividades ejecutadas de acuerdo a los modelos de procesos.

Por todo esto, podemos deducir que los *workflows* de interacción humana son aquellos en los que hay humanos activamente involucrados, y los mismos interactúan con sistemas de información.

Los sistemas de manejo de *workflow* tienen en cuenta la organización en la cual el proceso se inserta. En particular, se busca definir el rol interviniente en cada etapa del proceso de *workflow*. Obviamente hablamos de roles como grupos de personas que pueden llegar a realizar una tarea. Esto introduce cierta flexibilidad, ya que al momento en que el *workflow* entra en ejecución se puede asociar a una tarea alguien que esté disponible en ese momento.

Algunos de los objetivos que se atribuyen a los *workflows* de interacción humana son la reducción de los tiempos de espera, evitar trabajo redundante, y mejorar la interacción entre los trabajadores y los sistemas de información subyacentes.

Este tipo de *workflows* suele requerir interfaces gráficas y amigables. El concepto principal es la lista de ítems de trabajo. Mediante las mismas, los usuarios saben si disponen o no de la posibilidad de ejecutar una determinada tarea. En caso de estar a su alcance dicha tarea, proveen un conjunto de datos de entrada, y cuando la actividad se completa los trabajadores informan al *workflow*, el cual computa cual es el estado actual y determina las actividades a realizar consecuentemente.

Desafíos para el manejo de *workflow*

Existen distintos aspectos en este tipo de sistemas que aún es necesario mejorar. Por ejemplo encontramos:

- Ausencia de soporte adecuado para los trabajadores con conocimiento: este tipo de arquitecturas tienen un efecto bastante potente sobre el trabajo diario de las personas. Los métodos de almacenamiento de datos o las metodologías de programación utilizadas para desarrollar los sistemas no son de interés para los usuarios, con lo cual es sumamente relevante tener en consideración las necesidades de estos a la hora del diseño.

Este tipo de sistemas no representa solo los procesos sino también el ambiente organizacional en que estos se insertan, con lo cual las personas son representadas mediante sus habilidades, competencias y posicionamiento en la organización. Esta información es útil para seleccionar a las personas destinadas a ejecutar actividades.

También debemos considerar el rol de los trabajadores con conocimiento. Los modelos del *workflow* prescriben el flujo de los procesos, y el *workflow* se encarga de observar que las condiciones se ajusten a este modelo. Por lo cual es fundamental la participación de estos empleados en la conformación del modelo.

- Integración técnica: en este tipo de sistemas se presentan los mismos problemas de integración que en los sistemas tradicionales de integración de aplicaciones empresariales. Los problemas que se encuentran son: la falta de interfaces bien documentadas, problemas a nivel de código como por ejemplo errores en invocaciones directas, diferencias entre la granularidad de las aplicaciones integradas y la granularidad de las actividades del *workflow*. Para solucionar estos problemas se pueden usar arquitecturas como SOA (*Service Oriented Architecture*).
- Soporte de procesos sin sistemas de *workflow*: no todos los ambientes son adecuados para un *workflow*: por ejemplo si no se prevén cambios frecuentes, entonces simplemente con hacer una codificación del proceso estaríamos frente a una solución adecuada. En el caso de la administración de bases de datos hay procedimientos que son representados siguiendo un modelo de procesos, o también en ambientes de publicación donde por ejemplo un *workflow* de impresión puede ser visto como una herramienta para describir y ejecutar los pasos para llegar a resultados publicables.
Los sistemas de aplicaciones empresariales realizan literalmente miles de procesos. Estos pueden ser adaptados para adecuarse a necesidades de la compañía, y en muchos casos los procesos son realizados dentro del sistema, sin necesidades de integración. Si los procesos predefinidos no se pueden adaptar, entonces se puede usar funcionalidad de modelado para procesos integrados, y así crear nuevos procesos [1] [25].

Servicios de cómputo empresarial

Actualmente, la orientación a servicios es una de las mayores tendencias en ingeniería de negocios y tecnología de software. La idea principal de la orientación a servicios es capturar la funcionalidad relevante relativa al negocio

como un servicio, y proveer a los clientes la información lo suficientemente detallada como para que lo usen. Esta definición va más allá de los servicios que son realizados por sistemas de software.

La computación orientada a servicios usa interfaces bien especificadas, las cuales usan lenguajes para su definición, y son empleadas para combinar varios servicios en una aplicación nueva con orientación a servicios. Su principio de organización básica son las arquitecturas orientadas a servicios [1].

Arquitecturas orientadas a servicios

Los servicios son tareas computacionales débilmente acopladas que se comunican vía una red (en el caso de los *web services* por Internet), y que juegan una relación cada vez más creciente en las interacciones B2B. Las arquitecturas orientadas a servicios son aquellas que se enfocan en como se describen los servicios, y se organizan para dar soporte a la búsqueda automática y dinámica de servicios para lograr su uso. Los sistemas que tienen interacciones manualmente fijadas no son orientados a servicios, sino tipo EDI.

De esta manera podemos definir que un servicio es aquel que captura funcionalidad con un valor de negocio, y que está listo para ser usado. Estos son provistos por servidores, para lo cual requieren una descripción que pueda ser accedida y entendida por potenciales clientes. Los servicios de software son servicios provistos por sistemas de software.

Así, las arquitecturas orientadas a servicios son arquitecturas de software que proveen un ambiente para describir y encontrar servicios de software, y para ligar servicios a clientes. Las descripciones de servicios de software proveen un nivel de detalle que facilita a los clientes ligarse a los servicios e invocarlos.

Este tipo de arquitecturas son esencialmente importantes en ambientes donde muchos servicios se encuentran disponibles, y donde este conjunto de servicios cambia constantemente.

En una arquitectura orientada a servicios, las organizaciones pueden usar servicios ofrecidos por otras compañías, y las compañías pueden también proveer servicios al mercado. La visión es que los sistemas usen funcionalidad de negocio provista por servidores, de manera que se da una reutilización de funcionalidad de granularidad gruesa. Esto permite desarrollar nuevas aplicaciones con mayor facilidad, y menos costos de desarrollo y mantenimiento.

Los tres roles primarios que se dan en una SOA son el proveedor, el cliente y el registro de servicios. A su vez, entre estos, se dan las relaciones de publicación, requerimiento/respuesta y asociación (ligadura). El proveedor del servicio publica especificaciones del servicio en un registro de servicios, y el cliente busca en este registro interfaces de su interés. Así de esta manera obtiene información que puede permitirle invocar un determinado servicio que satisfaga sus necesidades [2].

Servicios empresariales

En ambientes de computación empresarial, la funcionalidad de los sistemas se puede describir y proveer vía servicios. Los servicios deben ser descriptos de tal manera que su especificación esté desacoplada de su implementación. Una especificación detallada de los servicios facilita una configuración flexible de los mismos mediante composición, y así lograr brindar funcionalidad compleja.

Debemos notar que la realización de los servicios puede cambiar, mientras que su especificación puede no hacerlo. Como resultado de esto, las partes individuales de una aplicación compleja basada en servicios se pueden intercambiar sin que esto requiera rediseñar la aplicación.

La orientación a servicios es también uno de los factores que influyen la integración de aplicaciones empresariales. Así, se pueden desarrollar aplicaciones que toman provecho de funcionalidad provista a través de interfaces estándar.

La arquitectura de servicios empresariales está basada en el entendimiento de que las aplicaciones complejas serán incrementalmente construidas sobre funcionalidad ya existente. Esta funcionalidad es brindada por sistemas de recursos que constituyen un gran activo de las empresas. El mayor objetivo es hacer que esta funcionalidad sea reusable.

Existen un conjunto de tendencias actuales que motivan a la creación de servicios empresariales:

- Un alza en el poder del cliente: los servicios de valor agregado son esenciales, porque los clientes pueden cambiar de proveedor sin mucho esfuerzo. Las experiencias positivas por parte de cliente son fundamentales.
- Transparencia de los sistemas: Internet se caracterizó por atraer a los clientes y proveedores a la infraestructura IT de la empresa. Si las aplicaciones están pobremente integradas esto quedará rápidamente expuesto.
- Un aumento en la interacción mediada por computadora con clientes y proveedores: las empresas buscan diferenciarse por los servicios que ofrecen a los clientes. Muchas veces el cliente puede resultar mal atendido si las aplicaciones solo reflejan un lado de su información personal, quizás debido a una pobre integración en los sistemas empresariales.
- Productos como servicios: la percepción de las empresas crece de acuerdo a los servicios que provee. Estos pueden ser brindados a través de servicios empresariales provistos por el back end de aplicaciones de la empresa. Sin embargo, los servicios brindados por terceros también pueden ser integrados, y de esta manera brindar al cliente mejores servicios y aplicaciones.
- Aplicaciones multinivel: hay una tendencia a las aplicaciones multinivel, en donde cada uno de estos es provisto por empresas distintas. Es decir que una organización puede brindar servicios de valor agregado mediante colaboración con otras empresas, además de facilitar la

creación de nuevas aplicaciones basándose en recursos ya existentes [1] [2] [5].

Aplicaciones basadas en composición de servicios

En este ambiente de arquitecturas de aplicaciones empresariales, se desarrollan nuevas aplicaciones basándose en un sistema de manejo de relaciones con el cliente, en un sistema de manejo de canal de suministro y un sistema de planificación de recursos, caracterizándose estos sistemas por brindar servicios vía interfaces estandarizadas. Estas nuevas aplicaciones que se desarrollan reciben el nombre de aplicaciones de composición, las cuales se caracterizan por invocar la funcionalidad de los sistemas de soporte. En el tope de esta composición encontramos a los usuarios que interactúan con interfaces gráficas amigablemente diseñadas.

Los avances tecnológicos han posibilitado la apertura de los servicios empresariales; donde la base de estos desarrollos son el conjunto de aplicaciones empresariales disponibles en la actualidad. Encontramos entonces un *middleware* muy potente y productos de integración de aplicaciones empresariales que se pueden usar para realizar una integración técnica, muchos de los cuales poseen un componente de *workflow* dedicado a la representación de los procesos.

Un elemento importante a considerar para la integración de múltiples aplicaciones es la transformación de datos, para lo cual se usa en la mayoría de los casos el lenguaje XML. Igualmente, a pesar de esto, aún se debe definir la lógica de trabajo en cada proyecto de integración.

Los procesos son la clave de la realización de aplicaciones de composición, debido a que proveen la unión entre los procesos de alto nivel y el sustrato de tecnología de la información.

Muchas veces la estructura de una aplicación de composición se puede expresar como un proceso de negocio, donde las actividades del mismo son implementadas mediante la invocación de servicios empresariales. Además podemos encontrar restricciones de ejecución como ejecuciones condicionales, las cuales pueden ser representadas mediante modelos de procesos, y ejecutadas mediante orquestación de procesos.

Los servicios empresariales se pueden usar también para realizar interacciones de negocio entre múltiples organizaciones. Cuando hablamos de escenarios multinivel, se requieren interacciones entre los sistemas de cada una de las partes intervinientes. Además, mientras que la visión de servicios empresariales incluye procesos B2B, actualmente la mayoría de los servicios empresariales son usados en actividades intra-empresariales.

Servicios empresariales y arquitecturas orientadas a servicios

Muchas veces los roles dentro de una arquitectura orientada a servicios pueden no estar completamente definidos. La especificación es hecha típicamente por el proveedor del servicio; el registro del servicio se instala localmente, y no se suele permitir que lo accedan otras compañías. Además, cuando se desarrollan nuevas aplicaciones, se buscan en este registro local servicios que puedan ser útiles para cumplir la funcionalidad requerida.

Usualmente esta búsqueda suele ser un proceso manual, y en muchos casos asistido por una taxonomía y una descripción textual del servicio.

Existen, entonces, una serie de problemas irresueltos en la actualidad: uno de ellos es el alcance del servicio. Básicamente nos estamos refiriendo a la funcionalidad provista por una o más aplicaciones y que es adecuada para un servicio empresarial. Si la granularidad es baja, entonces el grado de reutilización es bajo también, debido a que se van a necesitar varios servicios para lograr alcanzar la funcionalidad deseada. Si la granularidad es alta, entonces debería haber pocos escenarios donde el servicio encaja bien y tenga sentido. La adaptación de servicios de alta granularidad no es una opción ya que esto obstaculiza la reutilización. Suele no encontrarse una respuesta para este problema, la elección de la granularidad dependen del escenario particular, de las propiedades de las aplicaciones a integrar y de las aplicaciones compuestas a desarrollar.

En las arquitecturas de servicios empresariales, cada servicio suele estar asociado con exactamente un sistema. Esto es una limitación ya que efectuar un servicio sobre una serie de aplicaciones involucra integración, y así se simplifica la reutilización. Consideremos por ejemplo un servicio de órdenes de compra en el cual una orden entrante debe ser almacenada en múltiples sistemas de *back-end*. En este caso el servicio se puede usar con facilidad, ya que es invocado una vez por una aplicación de composición, y automáticamente provee la integración del sistema de *back-end* almacenando la orden de compra en el destino respectivo.

También la integración de sistemas de recursos se puede realizar dentro de un servicio empresarial. Esto permite usar los servicios con un alto grado de granularidad y así el trabajo de integración puede ser reutilizado por múltiples aplicaciones de composición [1] [2].

Bus de servicios empresariales

En un proceso de integración se necesita un *middleware* que provea interfaces estándar a las aplicaciones que se busca integrar. En el caso de utilizar un bus, cada una de las aplicaciones se asocia a este, el cual actúa como un componente centralizado. Es importante destacar que al introducir interfaces se logra ocultar la heterogeneidad de las aplicaciones. Generalmente, para las interfaces se utiliza la metodología de adaptadores, yendo así un paso más allá del *middleware* de integración de aplicaciones empresariales.

Composición de servicios

Para la realización de aplicaciones de composición en un ambiente de orientación a servicios, se utilizan técnicas de composición de servicios.

El *middleware* de integración de aplicaciones en general, y el *middleware* de bus de servicios empresariales en particular, proveen una base técnica aceptable para realizar composición de servicios, debido a que proveen interfaces estándar que pueden ser utilizadas en desarrollos de composición. El *middleware* típico para la integración de aplicaciones empresariales presenta un componente de *workflow* de sistema, que puede o bien usar un código

propietario o bien usar código BPEL (Lenguaje de ejecución de procesos de negocio para Web Services) [1] [2] [5].

Arquitecturas para gestión de procesos de negocio (BPM)

En este rubro, encontramos distintas arquitecturas que favorecen a constituir la infraestructura necesaria para así conformar el componente de BPM.

Arquitecturas de gestión de *workflow*

Tiempo de construcción y tiempo de ejecución: es esencial en este tipo de arquitecturas realizar una separación de ambos tiempos. Durante el tiempo de construcción se completa la especificación de un modelo de *workflow*, típicamente utilizando alguna herramienta gráfica.

Estos modelos de *workflow* constituyen el plan original para los procesos de negocio en un sistema de gestión de *workflow*, ya que deben estar en línea con los modelos de proceso que capturan los negocios operacionales. Luego se los extenderá con mayor información para volverlos ejecutables.

La etapa de modelado en el *workflow* se termina cuando un modelo satisface todos los requerimientos impuestos por los procesos de negocio. Estos modelos pueden representarse vía script en el lenguaje del sistema de *workflow*, pudiendo también ser almacenados en un repositorio de modelos.

Cuando un proceso para el cual se definió un *workflow* que lo implemente se inicia, se crea una instancia de *workflow*, la cual se basa en el modelo predefinido. Luego, la instancia se inicia y así entramos en tiempo de ejecución. El tiempo de construcción se finalizó una vez que el modelo de *workflow* estaba completamente terminado.

Las instancias de *workflow* viven típicamente en la memoria del motor de procesos, el cual ejerce un control sobre las mismas. Este motor decide para cada instancia que actividades deben ser iniciadas, y se comunica con los clientes de aplicaciones *workflow*, los cuales son accedidos a través de los participantes de los procesos.

Siendo este un escenario de *workflow* tradicional, no existe una unión entre una instancia de *workflow* y el modelo que fue usado para crearla. Esto implica que cambiar el modelo no afecta a las instancias en ejecución. Esta es una de las principales razones para separar el tiempo de construcción del de ejecución [1].

Arquitecturas de sistemas de gestión de *workflow*

Estas arquitecturas involucran a los subsistemas encargados del diseño y representación, tanto en el *workflow* del sistema como en los *workflows* de interacción humana.

En estas arquitecturas encontraremos:

- El subsistema de modelado de *workflow* permite modelar los aspectos técnicos de los procesos implementados. Por cada actividad que se

realiza por software en un proceso operacional, se tiene una especificación completa del ambiente requerido.

- El repositorio de modelos de *workflow*, el cual contiene la totalidad de modelos de *workflow* desarrollados en la compañía.
- El motor de *workflow*, responsable de la representación de los procesos de *workflow*. Si ocurre un evento correspondiente a un proceso para el cual se ha desarrollado un modelo, el motor crea una nueva instancia de *workflow* basada en el modelo definido. Si se crea un *workflow* de sistema, el motor usa el modelo para llamar a las aplicaciones invocadas en dicho *workflow*, las cuales serán hechas de acuerdo a la estructura definida. También se encarga de transferir datos entre llamadas a diferentes aplicaciones.

El motor usa también información organizacional sobre los participantes del proceso, lo cual posibilita ofrecer trabajo a los trabajadores con conocimiento, otorgándoles capacidad para ejecutar estas tareas.

- El subsistema de interfaz gráfica permite las interacciones con los humanos.

Gestión flexible de *workflow*

La gestión de *workflow* comúnmente interviene en el control de ejecución de los procesos acorde a los modelos predefinidos, su estructura y su realización técnica. Debido a la separación de tiempos antes citada, una vez que las instancias fueron iniciadas no hay un punto de unión claro entre ellas y los modelos.

Esta estructura se adecua bien al soporte de procesos con estructuras de control estáticas, es decir aquellos que fueron modelados una vez y ejecutados en forma rutinaria. Pero en ambientes dinámicos esta condición puede no darse con facilidad. La necesidad de responder rápidamente a los requerimientos plantea un esquema de flexibilidad.

La adaptación dinámica no fue siempre de preocupación en la gestión de *workflow*, ya que muchas veces esto no está provisto en sistemas de *workflow* de difusión comercial.

Si el proceso de *workflow* no puede ser continuado como fue especificado en el modelo, muchas veces los participantes ejecutan actividades que están fuera de su alcance, pudiendo ocurrir entonces errores en el código fuente.

Web services y su composición en relación a la metodología de *workflow*

Los *web services* son la realización actual de la computación orientada a servicios. Si bien hay aspectos que no están completamente cubiertos por los servicios web tales como el macheo y ligado dinámico de servicios, son un elemento fundamental en la orientación a servicios.

Elementos que caracterizan a los *web services* son su autocontención, autodescripción, y que se puede localizar, publicar e invocar aplicaciones modulares en la Web.

Los servicios ejecutan funciones que pueden ir desde tareas simples a complicados procesos de negocio. Una vez completado el *deploy* del *web*

service, el mismo puede ser descubierto e invocado por otras aplicaciones, incluso otros *web services*. Generalmente esta interacción se da en formato XML.

Es fundamental en una arquitectura de orientación a servicios que la comunicación entre componentes se efectúe mediante mecanismos estándar, los cuales son promulgados por la W3C. Los tres roles intervinientes son el proveedor del servicio, el cliente y el registro de servicios.

- SOAP define un protocolo de mensajes basado en XML para la comunicación de servicios. Este usa los estándares para convertir un mensaje SOAP a una invocación a un servicio, y luego traducir los resultados a un mensaje SOAP.
- WDSL (Lenguaje de descripción de Web Services) introduce un formato para especificar *web services*, permitiendo la definición de interfaces estándar. Además presenta una serie de extensiones para representar la falta de un conocimiento centralizado para transporte y direccionamiento, tales como puntos de servicio, requeridos para la invocación de los servicios.
- UDDI (Integración, descubrimiento y descripción universal) provee una infraestructura para publicar información sobre los servicios y sus proveedores. La interfaz UDDI provee el acceso a información del registro, por ejemplo, para registrar un servicio o para buscar por un determinado servicio o proveedor.

WDSL se usa para indicar como utilizar un determinado servicio, lo cual puede estar en un contrato lógico y uno o más contratos físicos. El contrato lógico define una interfaz pública del servicio, la cual es independiente de la implementación del servicio, y de los formatos y protocolos utilizados. Esto solo es considerado en el contrato físico. Los distintos contratos físicos pueden utilizar SOAP, o también protocolos de mail u otros protocolos de transporte.

El proveedor del servicio es responsable de la preparación del archivo WDSL, al cual accederán los potenciales clientes vía un registro de servicio.

El cliente crea un mensaje SOAP utilizando la información dentro del contrato lógico de la especificación WDSL. La información dentro del contrato físico se utilizará para determinar la codificación del mensaje y el protocolo adecuado.

El mensaje se envía al punto de servicio, el cual se especifica en el contrato físico. El servidor recibe el mensaje e invoca al software que lo implementa. Si estuviera definido un mensaje de respuesta entonces se envía un mensaje SOAP al cliente, completando así la transacción [1] [5].

Composición de web services en pos de los procesos

La composición de servicios es una idea de especial interés para el desarrollo de nuevas aplicaciones, basándose en funcionalidad ya existente. Así, la composición describe la forma en que se relacionan los distintos servicios, es decir que se están describiendo estructuras de proceso. Como resultado una

composición posee un conjunto de servicios, cada uno de los cuales realiza una actividad.

La composición de *web services* es una realización concreta de este concepto, aunque también podríamos considerar la implementación de *workflows* del sistema en ambientes orientados a servicios, basados en Web Services.

La composición de servicios es un concepto recursivo: cada composición puede ser expresada como un nuevo *web service*, utilizando WDSL. De esta manera podemos proseguir con la composición armando una jerarquía.

El lenguaje estándar para la composición de *web services* es BPEL, el cual surgió de unir WSF de IBM con XLANG de Microsoft.

En primera instancia, el caso de WSF de IBM se puede considerar como una serialización XML de un lenguaje de definición de flujo, el lenguaje utilizado por el componente *workflow* de IBM. Está basado en un lenguaje gráfico donde las actividades están ordenadas en forma acíclica por flujos de control.

Las dependencias de datos se manejan como flujos de datos entre actividades. El comportamiento del proceso se define como un conjunto de condiciones de transición adjuntas a los flujos de control. Esto indica que no hay un comportamiento de unión y separación definido, aunque se puede hacer un comportamiento de división mediante el adjuntado de condiciones.

En segunda instancia, XLANG es un lenguaje estructurado por bloques usado en BizTalk, un software de integración, enfocado en la integración de aplicaciones de sistemas *back end* heterogéneos usando procesos. Como rasgo característico se usan bloques de anidamiento de flujos de control. Esta es una característica luego presente en BPEL, el cual usa estructuras de bloques para organizar las composiciones de servicios, pudiendo usar links para definir estructuras similares a grafos.

El lenguaje se puede utilizar para caracterizar procesos concretos y abstractos: los procesos abstractos describen el comportamiento externamente visible de un proceso de negocio. Generalmente sirven a propósitos de comunicación por los que se omiten detalles operacionales. Por otro lado, los procesos concretos contienen información requerida para ejecutar los *web services* de la composición.

En BPEL encontramos los siguientes tipos de actividades:

- Invocación (*invoke*): es una operación ofrecida por un *web service*, aunque la misma puede o no tener una respuesta.
- Recepción (*receive*): espera por la llegada de un mensaje.
- Respuesta (*reply*): envía una respuesta a un mensaje enviado.
- Espera (*wait*): se aguarda por un período fijo de tiempo.
- Asignación (*assign*): se asignan valores recibidos, por ejemplo, de mensajes recibidos a variables del proceso.
- *Throw* (arrojar): indica que un error ha ocurrido, generalmente utilizado para el manejo de excepciones.
- Terminación (*terminate*): completa la ejecución del proceso.

Además, las actividades se pueden relacionar entre ellas mediante estructuras de flujo de control, las cuales son:

- Secuencia (*sequence*): define un bloque que consiste de una secuencia de actividades.

- Switch: basándose en una expresión, selecciona una actividad particular entre un conjunto de alternativas posibles.
- Pick: espera por un mensaje determinado o por el cumplimiento de un plazo. En el caso que se reciba el mensaje o se cumpla el plazo, se comienza con una actividad definida.
- *While*: se ejecuta un conjunto de actividades mientras una condición sea evaluada como verdadera.
- *Flow*: ejecuta concurrentemente un conjunto de actividades.
- Link: representa restricciones de ejecución entre actividades [1] [2].

Composición de servicios en la integración de aplicaciones empresariales

BPEL permite directamente implementar la estructura del proceso en la cual los servicios subyacentes se orquestan. Generalmente se utilizan herramientas visuales para la definición de dicha estructura, a partir de la cual se genera código ejecutable BPEL que representa una versión ejecutable del proceso.

En tiempo de ejecución, BPEL realiza el control de ejecución del servicio compuesto. Esto se logra mediante la invocación de los *web services* acorde a la orquestación del proceso, reaccionando ante situaciones de error, juntando respuestas y demás, de acuerdo a como esté especificado en el archivo BPEL.

Se requieren varios pasos para la definición y representación de una composición de servicios utilizando BPEL. Tendríamos varios *web services* cuyas especificaciones WSDL están almacenadas en un registro. El diseñador del servicio compuesto usa un editor BPEL para ensamblar los *web services*.

Las especificaciones WSDL de un *web service* proveen una especificación del proceso ejecutable. En una primera instancia, el editor provee esta información desde el registro. Una vez que el diseñador ha finalizado la composición del servicio, se genera el archivo BPEL. A partir de ahora el servicio compuesto se puede poner disponible mediante su almacenamiento en el archivo WSDL correspondiente, el cual describe como se puede utilizar este nuevo servicio.

Cuando se ejecuta el servicio compuesto, un motor BPEL lee la especificación correspondiente, haciendo que durante la representación del servicio compuesto los servicios se invoquen en secuencia.

Composición avanzada de servicios

Para mejorar el entendimiento de los diagramas de procesos a un nivel no técnico, se asocian explicaciones textuales a los modelos, orientados por ejemplo a eventos. En caso de ambigüedades dentro de un modelo de procesos, o ante la existencia de secciones no claras, los participantes pueden preguntar al diseñador sobre el significado deseado. Muchas veces estas dudas pueden derivar en un refinamiento del modelo.

Nos interesan en particular los procesos de negocio que son representados vía software, ya que los servicios se pueden componer en forma correcta solo si operan sobre los mismos conceptos de dominio.

WSDL provee un lenguaje de descripción de interfaces, detallando los tipos de datos de los parámetros entrantes y salientes. Como los *web services* involucrados en una composición pueden estar desarrollados en forma

independiente, muchas veces los tipos de datos no coinciden. Típicamente estas diferencias se identifican y son los arquitectos y desarrolladores quienes las solucionan, usando por ejemplo técnicas de mapeos de datos. Puede ocurrir que por estas diferencias surjan errores de comparación semántica en los datos. Por este tipo de inconvenientes se fomenta el área de investigación de Semántica web, donde se anotan datos en la web, y se especifican correctamente estas semánticas. Así los datos se pueden integrar automáticamente [1].

Ontologías y mapeos de datos

Las ontologías de dominios se pueden considerar como modelos de datos sobre los cuales todos los participantes han acordado. En la ciencia de la computación se caracterizan como modelos de datos que representan un conjunto de conceptos dentro de un dominio, además de considerar las relaciones entre estos conceptos.

Obviamente, la ontología está relacionada con un conjunto de interesados, los cuales acuerdan sobre el dominio. Generalmente este tipo de acuerdos son necesarios al momento de realizar integraciones de sistemas de gestión de relaciones con el cliente (CRM) y los sistemas de planeamiento de recursos empresariales (ERP), donde obviamente las estructuras de datos pueden ser muy diferentes.

Si los campos de datos de las aplicaciones son mapeados a la ontología de dominio, entonces la integración se puede resolver automáticamente en tiempo de ejecución.

De acuerdo a los distintos conjuntos de datos con los que se puede trabajar, es necesario definir el manejo de casos. Este tipo de sistemas se puede implementar mediante el uso de formularios, los cuales son un conjunto de campos donde cada uno de los cuales representa un objeto de datos. Además, conjuntamente a la definición de formularios se definen actividades.

Los formularios pueden contener objetos de datos que son obligatorios para actividades subsecuentes. Esta característica permite la representación flexible de los procesos, debido a que el usuario puede ingresar valores de datos en etapas tempranas.

El manejo de casos también tiene una faceta organizacional, presentándose distintos roles:

- *Execute*: este rol se usa para ejecutar una instancia de una actividad.
- *Skip*: las personas que pueden ejecutar una actividad no siempre están autorizadas para omitir su ejecución, con lo cual dicho permiso se asigna mediante este rol.
- *Redo*: puede haber instancias de actividades que se puedan volver a realizar, asignando dicho permiso mediante este rol. Hay que tener en cuenta que permitir repetir una actividad conlleva permisos para repetir actividades anteriores, o para reconfirmar datos que ya han sido ingresados y que son necesarios para repetir dicha actividad.

A través de lo que hemos expuesto anteriormente, vemos que los servicios son un componente fundamental, sobre todo desde el punto de vista de la

integración, para lograr aplicaciones cuya funcionalidad esté orientada a servicios. El paradigma de orientación a servicios constituye un tópico actual y de gran potencia en la Ingeniería de Software, con lo cual analicemos un poco más en detalle este tipo de arquitecturas [1].

Volviendo sobre el concepto de SOA

Un Modelo de Referencia es un marco de referencia para entender las relaciones más significativas dentro del dominio de un problema concreto y facilitar el desarrollo de estándares o especificaciones. En el caso de SOA, hablamos, por un lado, de un marco de referencia para la creación y utilización de servicios, de la definición de la infraestructura que permite a distintas aplicaciones el intercambio de datos, y finalmente la participación en los procesos de negocio, independientemente de los sistemas operativos, los lenguajes de programación y de su residencia o no dentro de una misma organización.

El conjunto de conceptos y sus relaciones, que han sido definidos en el modelo de referencia SOA deben ser la base para describir la arquitectura de referencia. Así, la arquitectura SOA será el resultante de aplicar la arquitectura de referencia desarrollada en base al modelo de referencia y a los patrones de la arquitectura, así como los requerimientos necesarios, incluyendo los “impuestos” por los entornos tecnológicos. Claramente, estamos buscando una arquitectura que tenga en cuenta objetivos, motivaciones y requerimientos del sistema a resolver.

Es común hallar que las arquitecturas se desarrollan en el contexto de un entorno predefinido como protocolos, perfiles, especificaciones y estándares. Una plataforma SOA combina todos estos elementos con el propósito de obtener un producto operativo.

En este marco de orientación a servicios, encontramos el siguiente conjunto de conceptos fundamentales, que son:

- Descripción de los servicios (*service description*): a través de la misma se accede a la información que un consumidor necesita para usar un servicio determinado. A través de la misma el consumidor conoce que el servicio existe y es alcanzable (*reachability*), puede ver cuáles son las funciones que cumple un servicio (*functionality*), las restricciones y normas que se aplican para su uso (*policies related*) y como se debe interactuar con el servicio (por su interface) en cuanto a formato y secuencia.
- Contratos y normas (*contract and policy*): cada contrato representa un acuerdo entre dos o más parte definiendo condiciones de uso del servicio. Recordemos que ya hicimos con anterioridad referencia a los contratos lógicos y físicos. Las normas, de esta manera representan las restricciones de uso y despliegue de los servicios.
- Contexto de ejecución (*execution context*): está compuesto por un conjunto de elementos técnicos y de negocio que conforman la vía para que proveedores de servicios y consumidores puedan interactuar.

De esta manera, en SOA no hay conceptos radicalmente nuevos, sino que surge gracias a la orientación a servicios que se ha aplicado durante años,

además del hecho que los servicios se aplican a entornos concretos creando sistemas aislados. Las principales diferencias que introduce radican en la forma de comunicarse, en que los servicios dejan de utilizarse en forma aislada, en que prevalece la utilización de estándares, y se utiliza en gran parte muchas técnicas experimentadas en el pasado.

Entre los beneficios que encontramos por la utilización de una arquitectura SOA vemos:

- Adaptabilidad: la arquitectura se adapta fácilmente a los cambios del negocio, facilita la integración de sistemas y es independiente de la plataforma y los lenguajes de programación.
- Es ágil, pues resulta más fácil añadir nueva funcionalidad y hacerla llegar a los canales, ya sean estos nuevos o existentes. Por otro lado, facilita la creación de nuevos servicios haciendo uso de elementos existentes (propios o ajenos).
- Permite una baja en los costos de producción, ya que facilita la reutilización de inversiones y reduce los costos de desarrollo.

El componente principal de esta arquitectura es el bus de servicios, el cual es una plataforma de software que incorpora interfaces estandarizados para la comunicación, conectividad, transformación, portabilidad y seguridad. En la actualidad, para el mismo existen dos tendencias de implementación que son, por un lado aquellas que necesitan un servidor de aplicaciones y por otro lado las que son totalmente distribuidas. Funcionalmente, cualquiera de las dos aproximaciones tiene ventajas e inconvenientes, los cuales se remiten por sus similitudes a las distintas variantes de una arquitectura cliente servidor.

El bus está compuesto de mecanismos de comunicación que hacen que todos los elementos conectados al ESB puedan entenderse entre si sin la necesidad de conocerse unos con otros. Así, el bus es el elemento central de la arquitectura, independientemente de su arquitectura interna. Con el uso del mismo se gana en flexibilidad haciendo únicamente que las aplicaciones se comuniquen con el bus en forma exclusiva.

Entre las características principales del bus encontramos que es una implementación propia de SOA, usa 100% estándares, normalmente suele ser multiplataforma y multilenguaje, y es distribuido (no se necesita conocer la locación física de los servicios, y el mismo servicio puede existir en más de un lugar simultáneamente). Por otro lado, la comunicación está basada en mensajes, posee enrutamiento inteligente de mensajes (basado en el contenido de los mismos, además de seguir ciertas reglas), se logra una orientación más próxima a los procesos ya que las características tecnológicas quedan ocultas, es robusto y posee un modelo de seguridad estandarizado. A su vez, la mayor parte de los elementos de un ESB solo necesitan ser configurados, permitiendo así el manejo de distintos eventos. Es útil a su vez la prestación que permite monitorear cada uno de sus componentes en tiempo real, donde a su vez, la mayor parte de los elementos que lo componen, incluidos los servicios desarrollados sobre el, se pueden poner en producción con mínimas interferencias en los sistemas en producción.

Dos características esenciales las encontramos en, por un lado, la comunicación, que es el mecanismo por el cual se constituye el bus. Normalmente está orientado a un mecanismo de MOM (*Message Oriented*

Middleware), aunque en general los ESB soportan distintos mecanismos de intercambio de mensajes y protocolos de comunicación. Por otro lado, el segundo rasgo es la conectividad, mecanismo por el cual se permitirá hacer pleno uso de las funcionalidades de los elementos externos al ESB en forma de servicios. Aquí es donde cumplen un rol preponderante los adaptadores o conectores.

Siguiendo esta línea de análisis, también encontramos los siguientes aspectos:

- Transformación de datos: como se encuentran distintas aplicaciones integradas, es común que las mismas requieran la misma información pero en distintos formatos.
- Coordinación de flujos de mediación: se debe coordinar los flujos que son el punto de interacción entre los procesos y los sistemas IT.
- Herramientas de administración y gestión de elementos del bus: básicamente para mantener el control de los distintos componentes del ESB.

La arquitectura opta por el uso del ESB ante la serie de beneficios que este provee, entre los cuales encontramos la mayor flexibilidad y adaptabilidad de los cambios, el hecho de estar basado en estándares, la posibilidad de implantación en forma progresiva, la mayor facilidad de integración con respecto a los sistemas existentes, y la mayor alineación de la topología ESB con las políticas del negocio. Si bien optar por esta metodología acerca los distintos sistemas de información al negocio, se requiere adaptación del personal de tecnología (desarrolladores, analistas y arquitectos) como así también del personal del negocio (analistas y directores).

En esta arquitectura encontraremos entonces, distintos tipos de servicios conviviendo, los cuales constituyen la arquitectura de referencia. Estos son:

- Los servicios de interacción, que constituyen el enlace con el “lado de las personas” en SOA. Proveen las capacidades requeridas para brindar funciones de IT y datos, adecuándose a las preferencias de los distintos usuarios.
- Los servicios de proceso: se encargan de suministrar los servicios de control requeridos para gestionar el flujo y las interacciones de múltiples servicios, de forma tal de implementar procesos de negocio.
- Servicios de información, encargados de proveer las capacidades requeridas para distribuir, replicar y transformar las fuentes de datos que pueden ser implementadas en distintas formas.

Por otro lado, de acuerdo a los cambios que pueden surgir en los requerimientos del negocio, necesitamos capacidad de reutilización. Las áreas clave en cuanto a este concepto son:

- Los servicios de acceso: son servicios que permiten establecer un puente entre las distintas aplicaciones heredadas y empaquetadas, los almacenes de datos corporativos y el propio ESB.
- Los servicios asociados, encargados de proveer la documentación, los protocolos y las capacidades de gestión requeridas por los procesos de

negocio, los cuales realizan interacciones con asociados y proveedores externos.

- Servicios de aplicaciones de negocio: proveen los servicios de tiempo de ejecución (*runtime*) utilizados para incluir nuevos componentes de aplicaciones en el sistema integrado.

En los sitios laterales de la arquitectura vemos los servicios de desarrollo, como herramientas esenciales en el proceso de integración, más los servicios de gestión IT, los cuales incluyen capacidades referidas a la escalabilidad y el rendimiento. Un ejemplo lo constituyen los servicios de *clustering* o de virtualización, que permiten un uso eficiente de los recursos computacionales basados en ciertos patrones de carga [1] [2].

Conceptos relacionados con SOA

SCA (*Service Component Architecture*) es un estándar propuesto para componer y hacer *deploy* de aplicaciones orientadas a servicios. La diferencia con un enfoque tradicional radica en que nos centramos en la lógica del negocio, ya que SCA nos permite evitar escribir los detalles tecnológicos específicos.

Además, SCA permite la invocación de *web services*, invocar aplicaciones tradicionales por medio de adaptadores, y acceder a colas de mensajes. También libera al desarrollador de tener que añadir lógica que distinga entre un petionario y otro durante el proceso conversacional. Mediante la colocación de valores fuera del código, la organización puede inicializar variables, elegir un protocolo de transporte y puntos finales en una invocación de código, o incluso especificar capacidades de Calidad de Servicio (QoS).

Podemos ver a través de los conceptos anteriores que SCA facilita la reutilización de componentes mediante una forma simple de componer los servicios a partir de otros. Así, un analista de negocios puede hacer el ensamblado, reemplazando un servicio por otro si las interfaces son similares, o asignando un intermediario entre ellos si los mismos son incompatibles.

Un dominio es una unidad lógica de las aplicaciones ejecutables, o al menos, un agrupamiento coherente de componentes que interactúan juntos, conectados entre si usando las conexiones de SCA. Así un compuesto se instancia cuando es usado en un entorno SCA.

Los dominios pueden variar en tamaño, donde uno muy pequeño puede ser uno dentro del entorno de pruebas en un IDE; uno de tamaño medio puede ser un único servidor o un pequeño cluster que soporta una única aplicación; y un gran dominio aquél que describe todos los servicios dentro de un departamento de la organización.

Cada uno de los componentes se comunican, ya se de forma intra-proceso, inter-proceso o inter-máquinas, pudiendo esto darse en formas diferentes para cada vendedor SCA. Independientemente de la elección que se haya hecho, los compuestos no se extienden más allá de los límites del dominio.

Aunque un compuesto SCA se ejecuta en el entorno de un único vendedor, puede comunicarse con aplicaciones que están fuera de su dominio. Para hacer esto, el componente SCA puede hacerse accesible utilizando un

protocolo inter-operable tal como los *web services*. En realidad, una aplicación SCA que se comunica con otra situada en otro dominio, ve a esta como una aplicación no SCA. El uso de SCA no es visible fuera del dominio.

SDO (*Service Data Objects*) complementa la fortaleza que ofrece SCA para simplificar el desarrollo de soluciones basadas en SOA. SCA maneja la composición de redes de servicios, mientras que SDO encara la simplificación del manejo de datos. La arquitectura SDO es componible, como oposición a antecedentes monolíticos, y además la especificación principal de SDO suministra las APIs básicas que son aplicables a todos los tipos de fuentes de datos. De esta manera, la especificación SDO no asume ningún lenguaje de interrogación particular o tipo de almacenamiento. La filosofía de la arquitectura SDO es usar, en la medida de lo posible, facilidades comunes y permitir también facilidades de tipos de fuentes de datos específicos cuando es necesario. La especificación SDO constituye el esqueleto que hace posible esta flexibilidad y simplicidad.

La arquitectura se basa en el concepto de grafos de datos desconectados. Aquí, un cliente busca en un grafo de datos (constituido como una colección de datos estructurados en forma de árbol o grafo), modifica el grafo de datos y puede luego aplicar los cambios en el mismo a la fuente de datos.

Frecuentemente la actualización se realiza con una semántica de concurrencia optimista, o sea que si los datos subyacentes fueron modificados antes que el cliente aplique los cambios, se rechaza la actualización y la aplicación debe tomar las acciones correctivas. La semántica de concurrencia optimista es una semántica natural que sirve a la mayoría de las aplicaciones de negocio.

El acceso a las fuentes de datos se realiza a través de un servicio de acceso de datos (DAS), responsable de buscar en las fuentes de datos, crear grafos que contienen objetos de datos y aplicar cambios a los grafos en las fuentes de datos.

De esta manera, necesitamos una serie de componentes en la arquitectura SDO, la cual abarca:

- El núcleo SDO: contiene los componentes primarios con los cuales trabajan normalmente los programadores, incluyendo objetos de datos y grafos de datos. También provee una API a metadatos que permite a los clientes tomar introspección del modelo de datos. Esta API permite a las herramientas y marcos trabajar uniformemente con fuentes heterogéneas de datos. Las APIs de metadatos soportan conceptos esenciales de otros metamodelos, como el W3C XML *Schema*, el modelo SQL relacional y la EMOF (Facilidad Esencial de Objetos Meta).
- Servicios de acceso a datos: proveen acceso a las fuentes de datos. Estos servicios crean grafos de datos leyendo datos de las fuentes finales y pueden modificar las mismas en base a los cambios realizados sobre los grafos de datos.
- Herramientas y marcos: encontramos las herramientas posibilitadoras de SDO que abarcan los generadores de código, convertidores de metamodelos, convertidores de esquemas, modeladores de datos, modeladores de esquemas, etc. Por otro lado hallamos los marcos y *runtimes* posibilitadores de SDO, los cuales funcionan con varios

componentes en SDO para realizar una variedad de tareas, tales como la ligadura de datos con componentes de interfaz de usuario.

- Objetos de datos: alojan a los datos reales, incluyendo los valores primitivos, y las referencias a otros objetos de datos. Los mismos también tienen referencias a sus metadatos, lo cual permite que puedan realizarse introspecciones sobre la información de los tipos de datos, sus relaciones y restricciones.
- Grafos de datos: son un conjunto de datos que proveen la unidad de transferencia entre los distintos componentes o capas. En definitiva es un conjunto con múltiples raíces de objetos de datos. El grafo registra nuevos objetos de datos, objetos modificados y objetos eliminados.
- Meta-datos: permiten a las herramientas de desarrollo y los marcos de *runtime* realizar introspecciones a los datos, incluyendo tipos de datos, relaciones y restricciones. SDO suministra una API común de metadatos a través de tipos fuentes de datos para ayudar a las herramientas genéricas y marcos.

Hemos analizado hasta aquí el estándar de BPM como la nueva metodología para satisfacer los objetivos de nuestra organización a través de la gestión de procesos de negocio. Hemos visto a su vez, que dicho estándar no plantea una visión completamente renovadora y de empezar de cero, sino que es necesario contemplar todo el trabajo realizado con anterioridad dentro de la empresa, de manera de apuntar a la integración. Aquí es donde se insertan los conceptos de SOA, *web services* y BPEL, de manera tal de lograr construir una aplicación real, integrada e insertada en un entorno B2B, y por sobre todo, orientada a procesos.

Analicemos con un poco más de detalle como implementar la orientación a procesos dentro de nuestra organización, de manera tal de facilitar el cumplimiento de las metas de negocio propuestas [1].

Metodología para implementar un paradigma de orientación a procesos

Para desarrollar una orientación a procesos debemos considerar una serie de fases necesarias:

- Estrategia y organización: es independiente de una instancia particular de proceso, porque se relaciona con la identificación de la estrategia de negocio, y sus objetivos asociados. En esta fase se determinan los objetivos estratégicos como así también los operacionales, de manera que la organización es estructurada de forma tal de que los procesos puedan ser implementados en forma completa dentro de la compañía.
- Estudio: es la primera fase relevante para los procesos individuales, y para post-proyectos que implementan a los mismos. Aquí se definen los objetivos del proyecto, se establece el equipo para el mismo, y se reúne información para el ambiente. También es esencial en esta fase el uso de una ontología de dominio que permita un entendimiento común de los términos y conceptos que componen el dominio.

Además de estudiar el dominio del negocio, se estudia también el ambiente de los procesos, por las implicancias que este posee en la implementación final.

- **Diseño:** se reúne la información obtenida y se la analiza, consolida y representa a través de los modelos de procesos. Dichos modelos serán una base de comunicación entre los distintos interesados.
- **Selección de la plataforma:** se utiliza la información sobre el ambiente organizacional y técnico para decidir la plataforma en la cual los procesos se representarán. Aquí se considerarán distintas alternativas en cuanto a *middleware* de integración de aplicaciones empresariales, arquitecturas orientadas a servicios, etc. También deben considerarse plataformas no técnicas, que sirven para establecer políticas del negocio por parte de personal no técnico interesado.
- **Implementación y test:** los modelos desarrollados anteriormente se vuelven aquí procesos ejecutables. Se considera el desarrollo de prototipos, y se procuran mejoras mediante interacciones con empleados con conocimiento del dominio. También es necesario considerar aspectos no funcionales, como optimizaciones de performance y robustez.
- **Deploy:** se despliega la implementación lograda en el ambiente de destino. Se deben tener en cuenta por un lado aspectos técnicos, para asegurar el correcto funcionamiento en producción, y por otro lado también es necesario observar aspectos organizacionales, como por ejemplo capacitación o ampliación de los recursos humanos requeridos.
- **Operación y control:** la aplicación ya está en ejecución en el ambiente, de manera tal que se hace necesario reunir información propia de la ejecución y utilizarla para realizar mejoras evolutivas.

Debemos tener en cuenta que el orden de estas fases puede variar, y se pueden dar dependencias entre cada una ellas. Además, la metodología tiene un enfoque iterativo e incremental, donde cada iteración produce resultados que retroalimentan a las siguientes.

El patrón evolutivo está definido por la recolección de información acerca de la ejecución de los procesos y su ambiente, además de su clasificación según características comunes, con su consecuente validación y refinamiento [1] [2] [5].

Estrategia y organización

Desarrollar una estrategia apunta a mejorar la competitividad a largo plazo de la organización de una manera sustentable, donde se tienda a minimizar los impactos en la infraestructura de la organización que puedan provocar los cambios en el mercado.

BPM apunta a establecer un lazo entre los objetivos estratégicos y el trabajo que se realiza dentro de la organización en forma cotidiana, ya que los procesos favorecen a alcanzar los objetivos del negocio, los cuales se relacionan a su vez con la estrategia de negocio.

Consideremos ahora en detalle cada una de las fases de la estrategia:

- Estudio: los objetivos principales de esta etapa son la recolección y organización de la información referente a la gestión de procesos dentro del dominio. El dominio está determinado por los procesos organizacionales, sobre los cuales luego se efectúan los procesos operativos. Los procesos organizacionales se identifican durante la etapa de estrategia y planificación.

Se deben considerar las posibles restricciones que ofrezca la infraestructura técnica de la organización, pudiendo esto impactar en los sistemas que se desee integrar. Además es fundamental la intervención de los trabajadores con conocimiento ya que esto impacta sobre la calidad final del modelo.

El mayor esfuerzo radica en la recolección de información sobre aspectos organizacionales y técnicos, con el fin de volcarla en documentación generalmente de carácter textual. La fase culmina cuando ya se ha depurado la información y todos los miembros del equipo acuerdan sobre la documentación resultante.

- Diseño: esta fase usa la documentación generada en la etapa anterior con el fin de identificar los procesos, y decidir los que se realizarán durante el proyecto. Existen ciertas reglas para identificar los procesos, las cuales se pueden resumir en:

- ❖ Cada proceso comienza y termina con un cliente que requiere un producto y recibe el mismo como un resultado del proceso.
- ❖ Cada proceso tiene un propietario, el cual es asignado como responsable.
- ❖ En cada proceso, los objetos se procesan en forma completa.
- ❖ Se utilizan restricciones de ejecución para ordenar la ejecución de actividades, de forma tal de emplear en forma eficiente los recursos empresariales, y al mismo tiempo alcanzar los objetivos de negocio.
- ❖ Contratar los proveedores de los procesos de negocio.

Se deben procurar modelos organizacionales y técnicos desarrollados con una perspectiva evolutiva e iterativa, para poder representar en forma explícita las dependencias entre los aspectos organizacionales y técnicos dentro del proceso.

- Selección de la plataforma: se busca escoger la plataforma sobre la cual los procesos se volverán ejecutables. Para el caso en que el proceso es realizado íntegramente vía software, la plataforma adecuada será un sistema de integración por *workflow*. Si se pensara en basar los sistemas de información en arquitecturas orientadas a servicios, los procesos se pueden realizar usando técnicas de composición de servicios. En cambio, si se requiere interacción humana la plataforma adecuada sería un sistema de *workflow* con manejo de casos.

Muchas veces elegir un componente de *workflow* en etapas tempranas suele ser un inconveniente ya que en esas instancias no se posee mucha información sobre los procesos y su ambiente de ejecución.

El proceso de selección comienza con la definición del criterio de selección en si mismo, de acuerdo a los procesos definidos en la etapa de diseño. Los distintos criterios a utilizar son:

- ❖ Criterio de integración: especifica aspectos de integración de aplicaciones y datos. Es un criterio importante ya que el éxito de un proyecto de *workflow* recaerá sobre la integración de aplicaciones existentes, las cuales comúnmente son desarrolladas en forma independiente. Buenas soluciones en esta línea son la orientación a servicios, y la composición de servicios en particular.
- ❖ Criterio de interacción: se basa en la inspección de las interfaces del sistema de *workflow* con el usuario, los mecanismos de notificación y la cantidad de entrenamiento requerido para la utilización del sistema de *workflow*.
- ❖ Criterio de diseño de procesos: incluye la expresividad del lenguaje de procesos. Aquí se requiere que el flujo de control establecido en el proceso se pueda mapear al modelo de *workflow*. En particular se debe poder expresar el flujo de control, las estructuras de datos y el flujo de las mismas.
- ❖ Criterio de simulación y test: apuntan a validar el proceso y su realización técnica.
- ❖ Criterio de tiempo de ejecución: el sistema debe proveer un mecanismo adecuado para seguir las instancias de los procesos a un nivel técnico, como así también ejecutivo. También debe soportar la carga necesaria para la aplicación, *workflows* flexibles y dinámicos, entre otras prestaciones deseables.
- ❖ Criterio general: ¿está el sistema disponible según la infraestructura de información general de la organización? También se tendrán en consideración cuestiones monetarias, de disponibilidad, etc.

Se utilizan estos criterios y un análisis de mercado para seleccionar un sistema de gestión de *workflow*, pudiéndose rediseñar el criterio si no se encuentran sistemas disponibles que satisfagan las condiciones planteadas. En caso que se lo encuentre, se lo instala y testea, y luego se toma una decisión basada en este análisis.

- Implementación y testeo: esta fase comienza con la implementación de los modelos de procesos usando la plataforma seleccionada, la cual puede ser, por ejemplo, un sistema de gestión de *workflow*. La implementación provee una representación de los procesos operacionales en el lenguaje provisto por la plataforma. Las actividades en el proceso operacional se mapean en actividades en el nivel de *workflow*. Se representan además restricciones de ejecución introducidas por el proceso.
En el caso en que se requiera de interacción humana para la realización del proceso, se deben representar los aspectos organizacionales en adición a los aspectos del proceso. Dependiendo de la funcionalidad que provea la plataforma, se reingresa la información organizacional en el

sistema, o se debe establecer una interfaz para algún sistema existente que maneje este tipo de información.

También se debe considerar la integración de aplicaciones externas. Dependiendo del soporte que provea la plataforma elegida, puede llegar a necesitarse mucho trabajo de programación y testeo. En el caso del testeo debe considerarse la simulación de laboratorio y el testeo de campo. El objetivo principal de la fase de testeo es obtener información sobre la estabilidad y la capacidad de uso que posee la solución en el ambiente de destino.

Es fundamental en el caso del testeo la definición de escenarios, cada uno de estos con restricciones temporales. Se definen además lotes de datos para efectuar las pruebas. Una vez construidos los casos de prueba, se ejecutan los mismos y se analizan los resultados. Pueden requerirse también redefiniciones de los escenarios, aunque si los resultados son de interés, esto puede derivar en una nueva iteración en el proceso.

En el caso que la simulación muestre que la aplicación implementada no es adecuada, se debe realizar entonces una remodelación del modelo de *workflow*. En caso contrario, se puede empezar con el testeo de campo.

El testeo de campo apunta a confirmar que la aplicación se maneja correctamente ante casos del mundo real. Este tipo de problemas en general no se puede prever con facilidad en un ambiente de laboratorio. Se deben definir los objetivos del test, y seleccionar los procesos a testear. Para cada uno de estos procesos se debe tener en cuenta una solución de backup, por cualquier error que pudiera haber surgido ante la representación en el ambiente de *workflow*. Además, en el caso de requerir interacción humana hay que realizar tareas de capacitación. Una vez que se ha completado la capacitación, y la solución de backup se prueba extensamente y se determina como estable, se puede iniciar entonces el testeo de campo. Una vez realizado el mismo se analizan los datos generados, pudiendo requerir revisar los objetivos del test. Obviamente este proceso se vuelve iterativo hasta alcanzar resultados aceptables.

- Operación y control: esta fase contiene las subfases de instalación y ejecución, como así también la configuración, en la cual se provee el ambiente técnico para el *deploy* de la aplicación.

La subfase de instalación incluye migración de datos, capacitación de usuarios y *deploy* del sistema en el ambiente de destino de la aplicación de *workflow*. En el caso de finalizar exitosamente la migración y la capacitación, entonces puede realizarse el *deploy* en la organización. La subfase de ejecución se caracteriza por la ejecución diaria de los negocios propios de la organización, utilizando la nueva aplicación de *workflow*. Esta aplicación se monitorea, y se obtienen datos de ejecución, la cual es importante para la mejora continua de la aplicación y de los procesos subyacentes [1].

Conclusión

A través de los conceptos precedentes hemos expuesto la importancia de los procesos dentro de una organización, y la influencia de los mismos en el alcance de los objetivos de negocio. Hemos apreciado a su vez como los mismos presentan una integración con ciertos recursos técnicos actuales como las arquitecturas orientadas a procesos, así como también guardan una relación con conceptos anteriores, tales como las arquitecturas de *workflow*. En el capítulo siguiente presentaremos un caso concreto de BPMS que permitirá efectuar una gestión completa de los procesos a lo largo de su ciclo de vida.

CAPITULO 2: ANÁLISIS TÉCNICO Y METODOLÓGICO DE IBM WEBSHERE BPM

Descripción de la infraestructura IBM Websphere BPM.

La infraestructura IBM Websphere BPM es un modelo complejo basado en un esquema de múltiples niveles. Esto responde a la arquitectura cliente-servidor en n capas, donde las capas superiores consumen los recursos producidos por las etapas inferiores. (Ver Figura 1)

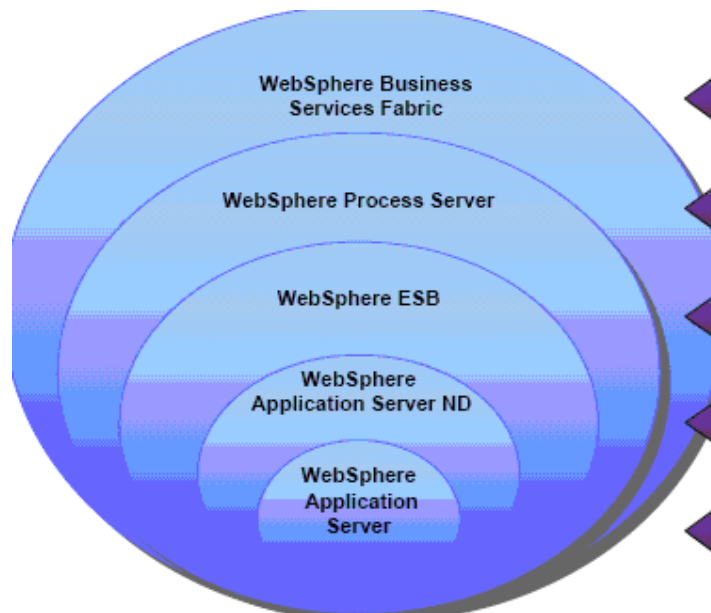


Figura 1. Descripción de la arquitectura Websphere

Además contamos con un conjunto de aplicaciones que poseen distintas relaciones entorno a la arquitectura arriba descrita (Figura 2). Describiremos a continuación cada uno de los componentes considerados [3].

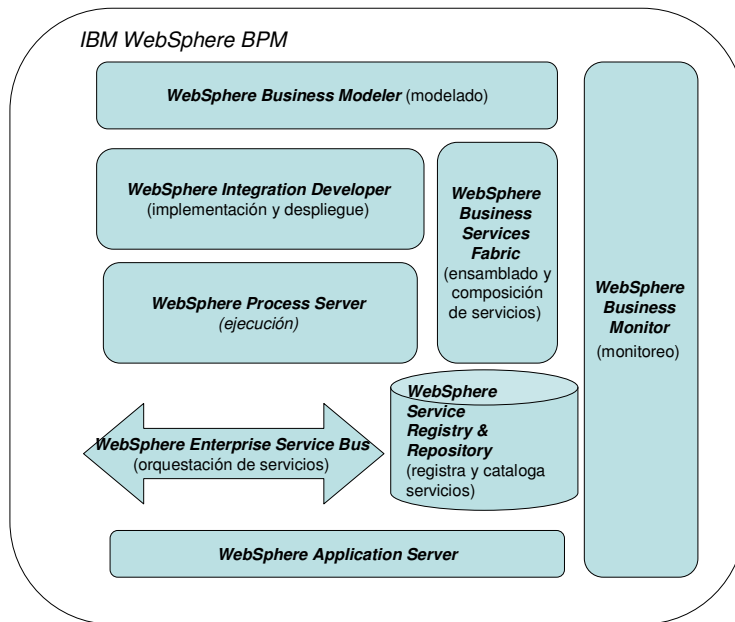


Figura 2: Componentes de la arquitectura IBM Websphere BPM.

WebSphere Application Server (WAS)

El WebSphere Application Server (WAS) se presenta como una alternativa a la planteada por el Apache Tomcat, para la publicación de aplicaciones Java de plataforma J2EE, siendo el primero de tipo propietario y el segundo de tipo Open Source. En ambos casos encontramos una serie de productos complementarios que conforman una infraestructura alrededor del servidor de aplicaciones. En el caso de WebSphere, el corazón de esta infraestructura es el servidor de aplicaciones, constituyéndose este en la capa más baja del modelo. El mismo es un servidor de aplicaciones J2EE genérico basado en un formato de nodo principal y células dependientes, entre cuyas funcionalidades principales encontramos:

- Publicación y despliegue de aplicaciones J2EE: En esta instancia se permite insertar las aplicaciones en el servidor, desplegarlas, publicarlas y habilitar el posterior acceso a los usuarios (esto se logra mediante un componente llamado *WebSphere Application Server Network Deployment*, accesible como un módulo dentro del mismo WAS).
- Acceso a fuentes de datos: mediante su configuración permite conectarse con la mayoría de los RDBMS existentes en el mercado. El mecanismo de conexión es JDBC. Es importante notar que muchas opciones de configuración solo se encuentran disponibles para DB2, siendo este último el motor de base de datos propietario de IBM.
- Manejo de servicios web mediante componentes SCA (Service Component Architecture). Esta adhesión a los estándares facilita la integración de aplicaciones en un esquema SOA (Arquitecturas orientadas a servicios).
- Manejo de EJB y JMS: el uso de *Enterprise Java Beans* (EJB) para la construcción de aplicaciones y el manejo de eventos a través de *Java Messaging Service* (JMS) hacen de este servidor una alternativa válida

para el desarrollo de aplicaciones J2EE, y en particular aquellas basadas en SOA.

- Configuración de recursos: permite establecer parámetros de configuración asociados a distintos aspectos como son:
 - Seguridad: permite crear roles, y asociar grupos de usuarios a los mismos. Esto permite establecer permisos de ejecución y visibilidad sobre las aplicaciones desplegadas en el servidor, como así también en los web services residentes en el mismo. Además permite manejar SSL (*Security Socket Layer* o Capa de sockets seguros), el cual proporciona comunicaciones seguras entre los procesos de servidor remoto o puntos finales. La seguridad SSL se puede utilizar para establecer comunicaciones entrantes y salientes de un punto final. Para establecer comunicaciones seguras, se debe especificar un certificado y una configuración SSL para el punto final.
 - Manejo de adaptadores: se gestionan adaptadores de recursos, que proporcionan la interfaz fundamental para conectar aplicaciones a un sistema de información empresarial (EIS – *Enterprise Information System*). El adaptador de recursos relacional de Websphere está incorporado en el producto para proporcionar acceso a bases de datos relacionales.
 - Bus de integración de servicios: Un bus de integración de servicios da soporte a aplicaciones mediante arquitecturas basadas en mensajes y orientadas a servicios. Un bus es un grupo de servidores y clústeres interconectados que se han añadido como miembros del bus. Las aplicaciones se conectan con un bus en uno de los motores de mensajería asociados con sus miembros del bus. En próximas secciones haremos un mayor hincapié sobre este componente al hablar del ESB.
 - Personalización de web services: el administrador puede realizar optimizaciones sobre elementos de comunicación de servicios, tales como JAX-RPC.

Como ya dijimos anteriormente, este servidor de aplicaciones constituye la base en el diseño de la estructura, y todos los demás componentes se basan en la funcionalidad provista por el WAS.

Websphere Enterprise Service Bus (ESB)

En la arquitectura de capas, por encima del WAS encontramos al Websphere ESB (Enterprise Service Bus). Se observa en esta última década que los servicios han adquirido una importancia fundamental en el desarrollo de aplicaciones web, debido a características como la portabilidad, reusabilidad, multiplicidad de plataformas, etc. Debido a este hecho, las aplicaciones acceden a componentes de servicios con mayor frecuencia, necesitando servidores que permitan establecer un bus para acceder a los mismos. Es

decir, no sólo necesitamos un servidor que nos permita publicar las aplicaciones empresariales, sino que a su vez requerimos de un bus, esto es, un medio común al cual acceden todas las aplicaciones con el objetivo fundamental de comunicación. Las aplicaciones residentes en el bus entienden la misma API (*Application Program Interface* o interfaz de comunicación) de manera tal de posibilitar la interacción entre las mismas.

Al poner como disponible una aplicación en un bus de servicios empresariales posibilitamos una comunicación bidireccional entre dicha aplicación y las demás que cohabitan el bus.

En la construcción del IBM Websphere ESB Server se observa una fuerte adhesión al estándar SCA (*Service Component Architecture*), el cual es un conjunto de especificaciones que describen un modelo para construir aplicaciones y sistemas basados en una Arquitectura Orientada a Servicios (*Service-Oriented Architecture* o SOA). El propósito de este proyecto es doble:

- Definir en forma única un modelo de componentes de servicios, tanto para la provisión como para el consumo de los mismos. Lo novedoso es que esta filosofía procura trascender al lenguaje natural de los servicios como es Java (por una cuestión histórica en su definición), para abarcar también otros lenguajes tales como C++, COBOL, PHP o incluso lenguajes basados en XML como BPEL, XSLT o XQuery.
- Definir en forma única la manera de ensamblar esos componentes, y de referenciarlos para poder construir aplicaciones orientadas a servicios.

SCA no sólo no se limita al lenguaje Java, ni tampoco son los web services con XML el único mecanismo de comunicación. Si bien este último parece ser el estándar de facto, SCA también puede ejecutarse bajo otros estándares de mensajería como CORBA o IIOP.

El borrador 0.9 de la especificación de SCA para Java, hace recordar mucho al modelo de programación de *Windows Communication Foundation* (WCF). Por ejemplo, la forma de exponer un componente como servicio es añadiendo la anotación `@Service`, similar al atributo `ServiceContract` de *WCF*.

Es importante notar que las aplicaciones requieren de un mecanismo de comunicación para posibilitar el intercambio de notificaciones a través del bus. En el caso del IBM Websphere ESB el intercambio se realiza a través de JMS (*Java Messaging Service*). Esta API permite a los componentes de aplicaciones basados en J2EE crear, enviar, recibir y leer mensajes. Esto posibilita una comunicación débilmente acoplada, confiable y asíncrona.

Para la invocación de servicios web, el servidor de IBM utiliza como protocolo de comunicación a JAX-RPC. Este es un mecanismo de invocación desarrollado en Java basado en el mecanismo XML de RPC (Remote procedure call). El mecanismo de comunicación es el siguiente:

- Un programa Java invoca un método en un stub (un objeto local que representa el servicio remoto).
- El stub invoca rutinas en el sistema de ejecución JAX-RPC.

- Este sistema convierte la invocación del método remoto a un mensaje SOAP.
- El sistema transmite el mensaje como un requerimiento http.

La descripción de los servicios se basa en el estándar WSDL (*Web Service Definition Language*), un formato XML que se utiliza como descripción para web services. En la actualidad la versión oficial es la 2.0.

WSDL describe fundamentalmente la interfaz pública de los servicios web, estableciendo la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje. Así, WSDL se usa a menudo en combinación con SOAP y XML Schema. Un programa cliente que se conecta a un servicio web puede leer la descripción del servicio a través de su definición WSDL y determinar qué funciones están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

El servidor permite también establecer parámetros de seguridad para el intercambio de peticiones entre los servicios. Esto se logra a través de componentes de WS (*Web Security*).

Recapitulando hasta el momento, hemos descrito el servidor de aplicaciones J2EE genérico (WAS), el cual se vale de un componente llamado WAS Network Deployment para la publicación y despliegue. Este último componente está situado un escalón más arriba en la arquitectura. Además, hemos detallado las características del bus de servicios, capaz de permitir la comunicación de aplicaciones basadas en SOA.

Por lo tanto, hasta el momento contamos con la infraestructura necesaria para publicar aplicaciones web que accedan a servicios existentes en el bus [3] [17].

Websphere Process Server (WPS)

El enfoque fundamental de nuestro trabajo se basa en BPM, con lo cual necesitamos analizar como insertar la gestión de procesos en un entorno orientado a J2EE y SOA.

IBM Websphere lo logra a partir del *Process Server*, el cual es un servidor que se apoya sobre la funcionalidad provista por la interacción de los componentes descritos con anterioridad, es decir, del WAS y el ESB. Se encarga del despliegue y ejecución de procesos que orquestan servicios (ya sean recursos humanos, información, etc.) en un entorno que puede o no estar basado en SOA.

Lo interesante del *Process Server* radica en el hecho de que no es un gestor de procesos exclusivamente, sino que al estar basado en un servidor de aplicaciones permite publicar aplicaciones que hagan uso de procesos BPEL, junto a todo tipo de aplicaciones J2EE (incluyendo implementaciones de web services), y con la consecuencia de que toda aplicación publicada está automáticamente siendo introducida en un bus de servicios empresariales, ya que a su vez está montado sobre el ESB. Esto facilita en un alto grado la

integración de aplicaciones basadas en procesos al circuito de aplicaciones ya existentes dentro de la organización, ya sea que se basen o no en SOA.

En cuanto a la funcionalidad propia del mismo, presenta los mismos aspectos que el WAS en cuanto a publicación, configuración de seguridad y administración de recursos (complementos de EJB, JMS, JDBC), junto con el agregado fundamental que es la capacidad de desplegar código BPEL.

En torno a este servidor encontramos un conjunto importante de componentes que permiten cubrir las distintas etapas del ciclo de vida del proceso, y que interactúan entre si consumiendo los recursos producidos por la etapa anterior de dicho ciclo [3] [17].

Websphere Business Process Modeler (WBPM)

Uno de los componentes existentes en torno al servidor de procesos es el *Websphere Business Process Modeler*. Se trata de una aplicación basada en la IDE Eclipse para el desarrollo de aplicaciones Java. Permite el modelado de procesos en una notación similar a BPMN (*Business Process Management Notation*), aunque no adhiere completamente a dicho estándar. En este modelador se pueden crear proyectos y dentro de estos modelos de procesos, los cuales son representados mediante categorías, las cuáles son:

- Elementos empresariales
- Procesos
- Recursos
- Organizaciones
- Clasificadores
- Informes
- Consultas
- Servicios empresariales
- Objetos de servicios empresariales

De esta manera se pueden documentar ampliamente los procesos, considerando aspectos como sus entradas, salidas, componentes de servicios a los que accede, asignación de recursos (vista por costo, por área, por roles).

Las diferencias fundamentales que encontramos en torno a la notación que usa el modelador de procesos con respecto al estándar de BPMN son las siguientes:

- **Eventos de inicio, fin e intermedios:** son elementos que permiten indicar los hitos de comienzo, finalización o algún hito intermedio que se desarrolle dentro del proceso, con su causa asociada. El modelador de IBM solo presenta eventos de inicio, fin y paradas de tipo básico, y no incluye así los inicios por evento, por mensaje, cronológicos y otros tipos que presenta BPMN. Esto le saca poder de expresividad a la notación, ya que es necesario documentar en forma textual detalles tales como el tipo de evento, y luego en la práctica no resultan fácilmente implementables las características de los mismos.

- **Tareas:** concentran las actividades que se ejecutan dentro del proceso. El modelador de IBM nuevamente no diferencia los tipos de tarea propios de BPMN. Estos son por ejemplo, tareas de script, humanas, de envío de datos, de recepción de datos, de servicio, entre otras. El modelador solo presenta tareas de tipo básico y tareas humanas, con lo cual, si bien todo tipo de tarea es representable por tratarse de la raíz de la jerarquía con respecto a los demás tipos de tareas, se vuelve a perder expresividad ante la necesidad de aclaraciones documentales.
- **Subprocesos:** sirven para representar un conjunto de tareas dentro del proceso que se ejecutan indefectiblemente como un bloque. Mediante su uso se gana en legibilidad y reusabilidad durante la actividad documental, disminuyendo el impacto de las modificaciones ya que se minimiza el acoplamiento entre componentes dentro del proceso. Igualmente es necesario notar que luego al traducir al mismo a su equivalente grafo en BPEL para lograr su versión ejecutable, se pierde la semántica lograda al no poder representar el anidamiento.
- **Compuertas:** representan las estructuras de control propias del flujo de ejecución. El modelador presenta compuertas básicas (o sea, la representación gráfica de una estructura tipo *if then else* en el que ante la evaluación de una condición tenemos una posibilidad de ejecución ante un valor verdadero y otra ante el valor falso de la misma) y compuertas compuestas en donde puede haber más de una opción válida de ejecución ante una condición verdadera. El estándar de BPMN presenta varios tipos más de compuertas que son por ejemplo las basadas en eventos, en datos, complejas o con paralelismo, que le agregan al modelo mayor grado de expresividad. Igualmente hay que notar que con los elementos disponibles en el modelador igualmente se pueden representar la mayor parte de problemas típicos.
- **Uniones, bifurcaciones y fusiones:** estos elementos en BPMN se presentan como parte del inciso de compuertas. Sirven para representar alteraciones en la linealidad del flujo de control del proceso. Aquí el modelador las utiliza como facilitadores para lograr el inicio de actividades de paralelismo mediante la bifurcación y luego sincronizar las actividades mediante la fusión o la unión, dependiendo de la semántica necesaria. Estos elementos suelen tener un gran impacto en los parámetros de entrada que reciben las actividades, pudiendo resultar por momentos algo engorroso. Las compuertas de bifurcación por ejemplo reciben un parámetro con un tipo asociado y lo diseminan en ambas tareas que disparan. Una fusión puede recibir un tipo de parámetro distinto de cada una de las tareas que une, y luego puede enviar ambos, quedando a criterio del diseñador elegir a qué tareas asignar cada uno. En ocasiones puede ser útil crear tareas intermedias que faciliten dicho intercambio de parámetros.
- **Bloque *while*:** si bien BPMN no contempla este elemento ya que es posible de ser representado mediante las compuertas, el modelador presenta además esta estructura de control, lo cual facilita en gran medida la representación de bucles, sin la necesidad de uso de *if's* accesorios. Presentan las variantes de condición al inicio del bucle (estructura de control *while* en su forma clásica), al final (similar al *repeat*

until de Pascal), y la variante de repetición un número fijo de veces (similar al for).

En la tabla 1 podemos observar una comparación entre la notación utilizada por el *Websphere Business Modeler* y el estándar BPMN (detallado luego en el Anexo 1). A través de la misma podemos ver que en realidad el producto de IBM opta por una versión más acotada de la misma [3] [4] [17].

Tabla 1: Comparación entre la notación utilizada por Websphere Business Modeler y BPMN [21] [22] [23] [24]

Elemento notacional	Descripción	Diferencia con BPMN
Evento de inicio, fin e intermedio	Sirven para modelar la condición de inicio y finalización del proceso. Los eventos intermedios representan hitos alcanzables durante la ejecución del proceso.	WBPM solo utiliza inicios y fines de tipo clásico (raíz de la jerarquía) y no considera eventos intermedios más que Paradas (pueden representar sincronización). BPMN por el contrario presenta los siguientes eventos de inicio: <ul style="list-style-type: none"> ➤ Inicio clásico o vacío ➤ Inicio por mensaje ➤ Inicio por regla ➤ Inicio por temporizador ➤ Inicio por señal ➤ Inicio por eventos múltiples. Los eventos de fin que utiliza BPMN son los siguientes: <ul style="list-style-type: none"> ➤ Fin clásico o vacío ➤ Por mensaje ➤ Por error ➤ De compensación ➤ De señal ➤ Por cancelación ➤ Por múltiples causas Los eventos intermedios son del mismo tipo que los de finalización.
Tareas	Sirven para modelar las actividades que se desarrollan dentro del proceso.	WBPM solo distingue entre tareas de tipo genérico y de tipo humanas. BPMN por el contrario distingue entre los siguientes tipos de tareas: <ul style="list-style-type: none"> ➤ Clásicas ➤ De script o automáticas ➤ Humanas ➤ De envío ➤ De recepción
Subprocesos	Sirven para modularizar conjuntos de actividades que pueden ejecutarse una o más veces dentro del proceso	BPMN considera los subprocesos de manera idéntica.
Compuertas	Sirven para representar los distintos flujos de control dentro de la ejecución del	WBPM utiliza las compuertas clásicas (if then else) además de las que pueden tener

	proceso.	varias opciones válidas (similar al switch). BPMN considera además las siguientes compuertas: <ul style="list-style-type: none"> ➤ Exclusiva basada en datos ➤ Exclusiva basada en eventos ➤ Inclusiva basada en datos ➤ De paralelismo ➤ Complejas
Uniones, bifurcaciones y fusiones	Sirven para representar divisiones en el flujo de control, y la posterior unificación del mismo para permitir continuar la ejecución.	WBPM utiliza uniones, bifurcaciones y fusiones en forma genérica. BPMN representa a todas ellas mediante las compuertas. La bifurcación se representa mediante un disparador de compuerta de paralelismo, y las uniones y fusiones se representan mediante la misma compuerta (paralela), utilizada para reunir los flujos divididos con anterioridad.
Bloque while	Sirve para representar iteraciones de tipo condicional	Si bien WBPM lo utiliza, BPMN no lo considera en forma explícita sino a través de las compuertas.

WebSphere Service Registry and Repository

Un componente importante a la hora de modelar procesos son los documentos que permiten establecer descripciones de los objetos que el proceso consume, ya sean objetos de datos empresariales, servicios, etc. Websphere maneja un componente de publicación y registro llamado *Service Registry and Repository* que centraliza como repositorio las descripciones de todos los recursos que se manejan en la plataforma. Se pueden publicar descripciones de objetos de datos (en archivos xsd para la definición de esquemas) y de componentes de servicios (en formatos wsdl), entre otras funcionalidades. A través del modelador, el desarrollador puede conectarse con dicho repositorio e importar descripciones de objetos de datos y/o servicios que serán consumidos por el proceso. De esta manera, este componente resulta de gran utilidad para centralizar todas las descripciones de recursos existentes en desarrollo, y controlar de esta manera sus accesos y modificaciones, sirviendo así como control de consistencia.

Una vez modelado el proceso, con todas sus actividades, las definiciones de entradas y salidas, y las condiciones de implementación de las compuertas en caso que las hubiera, se puede exportar el proyecto e incorporarlo a la etapa siguiente del ciclo de vida, relacionada con obtener una versión ejecutable del proceso. Es importante notar que el modelador se encuentra desarrollado bajo un esquema de interfaces amigables cuyo objetivo es minimizar el esfuerzo de

programación, de manera tal de lograr establecer la mayor cantidad de reglas de negocio y procesamiento a través de manipulaciones gráficas (tal es el caso de la definición de condiciones para las compuertas, o la definición de estructuras para los objetos de negocio) en las que casi no es necesario introducir líneas de código.

Websphere Integration Developer (WID)

En el siguiente paso dentro del desarrollo encontramos el Websphere Integration Developer (WID). Es otra IDE de desarrollo basada en Eclipse al igual que el modelador. Incorpora una serie de complementos llamados *plugins* que permiten el desarrollo de procesos implementados en código BPEL. Esta herramienta concentra tareas que corresponden al diseño de los procesos y avanza en la ejecución y despliegue de los mismos.

Permite la importación de los proyectos generados mediante el modelador de procesos. Aquí se transforma el proceso de notación BPMN en un grafo BPEL, donde se aprecia la secuencialidad de las actividades, y las bifurcaciones producidas por la ejecución de las estructuras de control.

En dicho árbol se puede acceder a cada uno de los componentes e indicar el grupo de sentencias Java que permitirán implementarlo. Es importante notar que al utilizar código Java de posibilidades abiertas, el proceso puede interactuar con infinidad de recursos, como servicios web, bases de datos, modelos de objetos existentes, mecanismos de comunicación remotos, etc.

Recordemos brevemente los tipos de elementos que podemos encontrar en un árbol de representación BPEL:

- Invocación (*invoke*): es una operación ofrecida por un web service, aunque la misma puede o no tener una respuesta. Aquí se refleja el hecho de que BPEL fue diseñado con una clara orientación a un paradigma de servicios.
- Recepción (*receive*): espera por la llegada de un mensaje proveniente de un servicio invocado.
- Respuesta (*reply*): envía una respuesta a un mensaje enviado.
- Espera (*wait*): se aguarda por un período fijo de tiempo. Permite un esquema de sincronización dentro del proceso.
- Asignación (*assign*): se asignan valores recibidos, por ejemplo, de mensajes recibidos a variables del proceso.
- *Throw* (arrojar): indica que un error ha ocurrido, generalmente utilizado para el manejo de excepciones. En el caso del *Websphere Process Server*, las excepciones generadas se comunicarán mediante JMS.
- Terminación (*terminate*): completa la ejecución del proceso.

Además, las actividades se pueden relacionar entre ellas mediante estructuras de flujo de control, las cuales son:

- Secuencia (*sequence*): define un bloque que consiste de una secuencia de actividades. La ejecución de las mismas se desarrollará una detrás de la otra.

- *Switch*: basándose en una expresión, selecciona una actividad particular entre un conjunto de alternativas posibles.
- *Pick*: espera por un mensaje determinado o por el cumplimiento de un plazo. En el caso que se reciba el mensaje o se cumpla el plazo, se comienza con una actividad definida. Esta estructura de control permite establecer el cumplimiento de un determinado evento, como puede ser el alcance de una fecha, o el cumplimiento de algún evento de procesamiento.
- *While*: se ejecuta un conjunto de actividades mientras una condición sea evaluada como verdadera.
- *Flow*: ejecuta concurrentemente un conjunto de actividades.
- *Link*: representa restricciones de ejecución entre actividades.

El proceso de esta manera es editable dentro del modelador, lo cual se efectuará en su versión de BPMN, o es posible editar directamente el grafo BPEL. Es importante recalcar que si bien esta última opción es posible, sería deseable evitarla ya que en caso contrario estaríamos rompiendo con la complementación de etapas en el ciclo de vida. La consecuencia de ello es perder consistencia entre las versiones modeladas de los procesos y las versiones que finalmente están siendo ejecutadas en el servidor de procesos.

La interfaz del WID presenta facilidades para la integración de servicios web ya existentes en el entorno y lograr así invocarlos dentro del proceso, como así también para colocar al proceso como un servicio web para que pueda ser accedido desde otras aplicaciones.

En relación a esto último, se presentan generadores de código automático para ciertas actividades. Por ejemplo, al querer invocar un proceso desde otra aplicación, es necesario crear el *stub* que sirva para representar al proceso e invocarlos como un web service. El mismo editor de código genera las líneas de código necesarias para ubicar e invocar el servicio desde la aplicación y hacerlo de manera transparente. Luego haremos más hincapié en este tema al hablar de la conversión del proceso en un web service.

La definición de las actividades del proceso se hace accediendo al mismo árbol BPEL presentado por la aplicación. Así, se definirán manejadores asociados a cada actividad. Cada uno de estos manejadores son clases Java, dentro de las cuales se define el código necesario para manejar las reglas de negocio.

Es importante señalar que sólo se programa el código concerniente al procesamiento dentro de las tareas, mientras que el flujo de control entre las actividades ya es generado por el mismo código BPEL.

Una vez culminada en forma completa la programación del proceso, el mismo puede publicarse a través de un enlace con el Process Server ubicado dentro de la misma interfaz de desarrollo. La publicación y despliegue se hace por interfaz gráfica, con lo cual se evitan labores engorrosas relacionadas con consolas de comandos presentes en otros servidores.

Se genera de esta manera una aplicación asociada al proyecto de descripción y definición de los procesos, la cual se publicará junto con el mismo en el servidor. Además, se deberán publicar todos los proyectos complementarios con los que el proceso presente dependencia [3].

Business Process Coreographer (BPC)

Una vez publicado el proceso, podemos acceder al Explorador de Coreografía de Procesos (BPC, *Business Process Coreographer*). Es una aplicación basada en JSF (Java Server Faces) que permite navegar los procesos publicados en el servidor, crear instancias de los mismos, analizar el ciclo de ejecución, acceder a las actividades humanas que estos presenten y obtener estadísticas personalizadas sobre los procesos ejecutados y sus actividades. Podemos ver por esto que ejerce un rol importante en la labor de monitoreo de los procesos creados y ejecutados.

Las tareas humanas son un tipo especial de actividades contemplado por BPMN: son tareas en cuya ejecución se da la interacción del usuario con una aplicación. Así, en caso que el proceso incluyera este tipo de tareas, el servidor trata a las mismas como un tipo especial, para las cuales el usuario deberá completar un formulario con los datos de entrada, y así poder proseguir la ejecución del proceso. En el caso del BPC, hay un apartado especial a través del cual se puede acceder a las tareas humanas que hubieran sido definidas en el proceso y para las cuales el usuario registrado en el sistema tuviera acceso de acuerdo al esquema de permisos definido.

Si bien este explorador constituye una herramienta muy útil, hay que notar su importancia más que nada para el ambiente de desarrollo y análisis de los procesos, no así para el ámbito de ejecución de los procesos destinados a usuarios reales. Es muy útil contar con un explorador de procesos en la interfaz interna del ambiente de programación, ya que esto facilita las pruebas y la depuración de los procesos. Ahora bien, esto resulta algo rudimentario al pensar en orientar los procesos a usuarios finales. Las interfaces que se autogeneran para la interacción con el proceso son algo primitivas para presentarlas ante un usuario promedio, donde por ejemplo no se visualizan en forma automática listas desplegables, o validaciones de campos mandatorios en los formularios. Por eso es importante notar el hincapié que haremos posteriormente con la necesidad de lograr una interacción plena de los procesos con aplicaciones reales existentes, y no necesariamente circunscribir la ejecución de los procesos generados a este entorno de exploración, de manera tal de suplir estas limitaciones.

Gestor de reglas de negocio (*Business Rules Monitor*)

Continuando con la descripción de componentes relacionados con el servidor de procesos, encontramos al Gestor de Reglas del Negocio, otro componente al cual podemos acceder mediante el WID. Como su nombre lo indica, permite explorar reglas de negocio. Las mismas se pueden definir ya sea dentro del Modelador (acompañando a la descripción del proceso en notación BPMN) como así también dentro del ambiente de programación del proceso, al momento de generar el código java asociado a las tareas. El gestor permite observar como es el desenvolvimiento de dichas reglas en una instancia real del proceso [3].

Conclusiones

Vemos de esta manera que contamos con una serie de herramientas que nos permiten hacer la gestión integral de las fases del ciclo de vida del proceso:

- Análisis y diseño: se inicia a través del *Websphere Business Modeler* y se complementa con el *Websphere Integration Developer*.
- Configuración y representación: se logra a través del mismo *Websphere Process Server* (con su consola administrativa), del *Service Registry and Repository* y del *Websphere Integration Developer*.
- Evaluación y administración: a través del BPC y de la consola administrativa del *Websphere Process Server*.

Un aspecto importante de recalcar en cuanto a la infraestructura descrita es la facilidad que presenta para incorporar elementos de una etapa del ciclo de vida a la siguiente: una vez desarrollado el proceso en BPMN es muy intuitiva su exportación y posterior incorporación al entorno de programación que permitirá obtener una versión ejecutable del proceso en código BPEL. Por otro lado, resulta sencilla la incorporación de servicios dentro del proceso, y su despliegue dentro del bus de aplicaciones, permitiendo así que el mismo sea accedido, o bien por otros procesos, o incluso por aplicaciones genéricas que deban consumirlo como un elemento de servicios. Así mismo es fácil de comprender el mecanismo de acceso al servidor de procesos para lograr así la publicación y despliegue de los procesos generados, ya que todo se ejecuta a través de accesos directos dentro de la interfaz gráfica, no necesitando así el uso de consolas para introducir líneas de comando. El hecho de concentrar gran cantidad de herramientas dentro del entorno de programación acorta los tiempos de desarrollo.

Por otro lado, es notable la unicidad de criterio en la representación visual de los servidores: el *Websphere Process Server* posee la misma interfaz y ubicación de componentes que el WAS, al igual que el *Websphere ESB Server*. Esto, a los fines prácticos acorta los tiempos de producción y termina dando mayor facilidad de uso sobre el sistema al desarrollador.

Análisis funcional aplicado

A continuación describiremos el caso de uso utilizado para analizar la funcionalidad de la herramienta IBM Websphere BPM.

Caso de uso: solicitud de crédito

Este caso de uso consiste de un proceso en el cual un cliente y una entidad crediticia se conectan ante un requerimiento de un crédito.

El cliente realiza la solicitud de crédito, para la cual la entidad crediticia solicita una serie de datos de su interés. A partir de allí, la entidad realiza una

búsqueda de antecedentes internos, para el caso en que el cliente ya presente algún tipo de deuda con la propia entidad, además de realizar una verificación externa para verificar la liquidez del cliente. Esto significa que el mismo no posea deudas o juicios con otras entidades y esto lo convierta en un caso de escasa credibilidad para solicitar el posterior pago del crédito. En caso de cumplir con los requisitos de ambas validaciones, se analizará el caso concreto de la solicitud, acerca de la adecuación del monto con el destino solicitado. En caso de cumplir correctamente estas condiciones el crédito será otorgado, denegándose el mismo en caso contrario.

Describiremos a continuación cada una de las etapas necesarias para cubrir el ciclo de vida propio de los procesos. El mismo contemplará el modelado del proceso, el análisis y diseño del modelo de datos; configuración, desarrollo y despliegue del proceso, incorporación de tareas humanas, invocación del proceso como servicio y finalmente la interacción del usuario final con nuestra aplicación.

Modelado del proceso

A fines prácticos que expondremos a posteriori, el proceso fue desdoblado en dos partes, la primera corresponde a la solicitud del crédito propiamente dicha, la cual incluye el proceso de validación del cliente. La segunda etapa es el estudio de la solicitud, hasta la decisión final y la comunicación de la misma al cliente. En la figura 3 se visualiza la primer parte del proceso, comenzando con el ingreso de los datos correspondientes a la solicitud de crédito, el registro de la misma, las validaciones sobre las condiciones del cliente para poder otorgarle o no el crédito, el ingreso al padrón de clientes en caso de corresponder, y finalmente la comunicación al usuario del estado de su solicitud. Luego, en la figura 4 visualizamos la segunda etapa del proceso, la cual se inicia realizando el estudio de las solicitudes pendientes, y en base al mismo la decisión de otorgamiento o denegación del crédito, con la consecuente información al usuario.

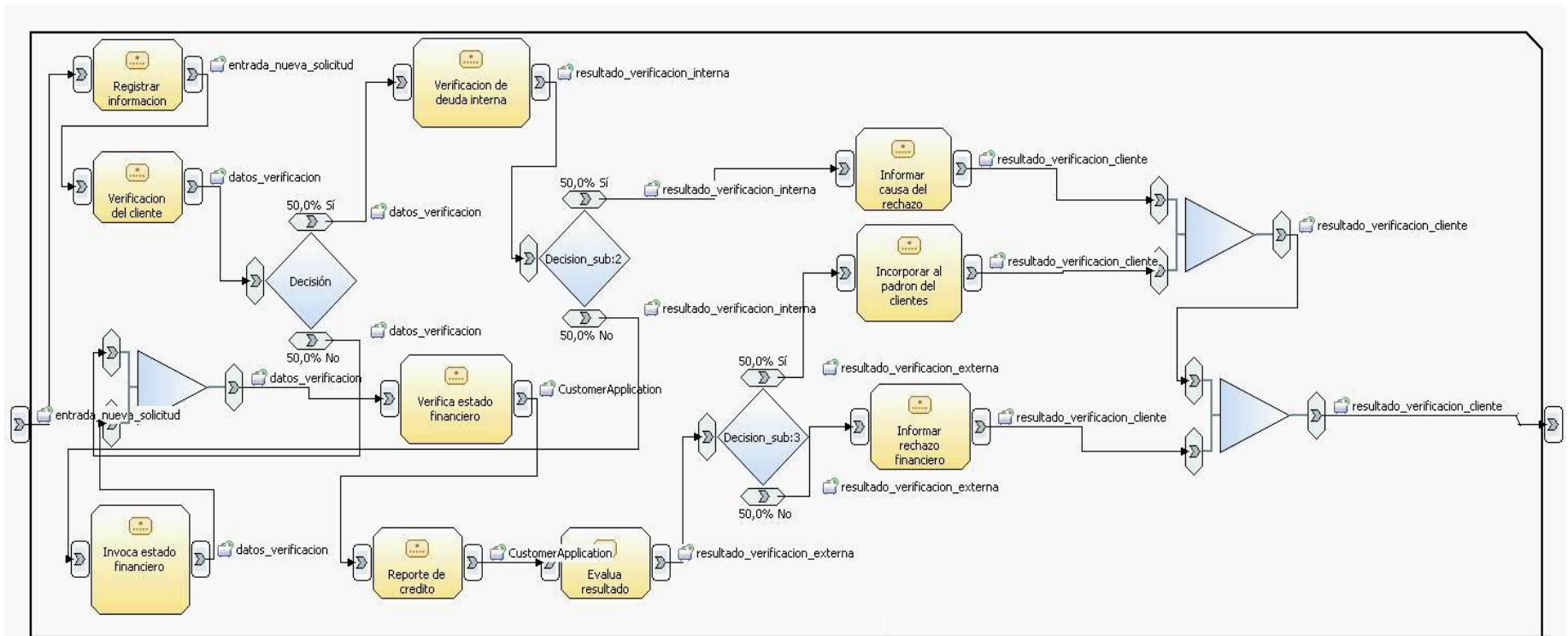


Figura 3. Vista de la etapa 1 del proceso: Solicitud de crédito

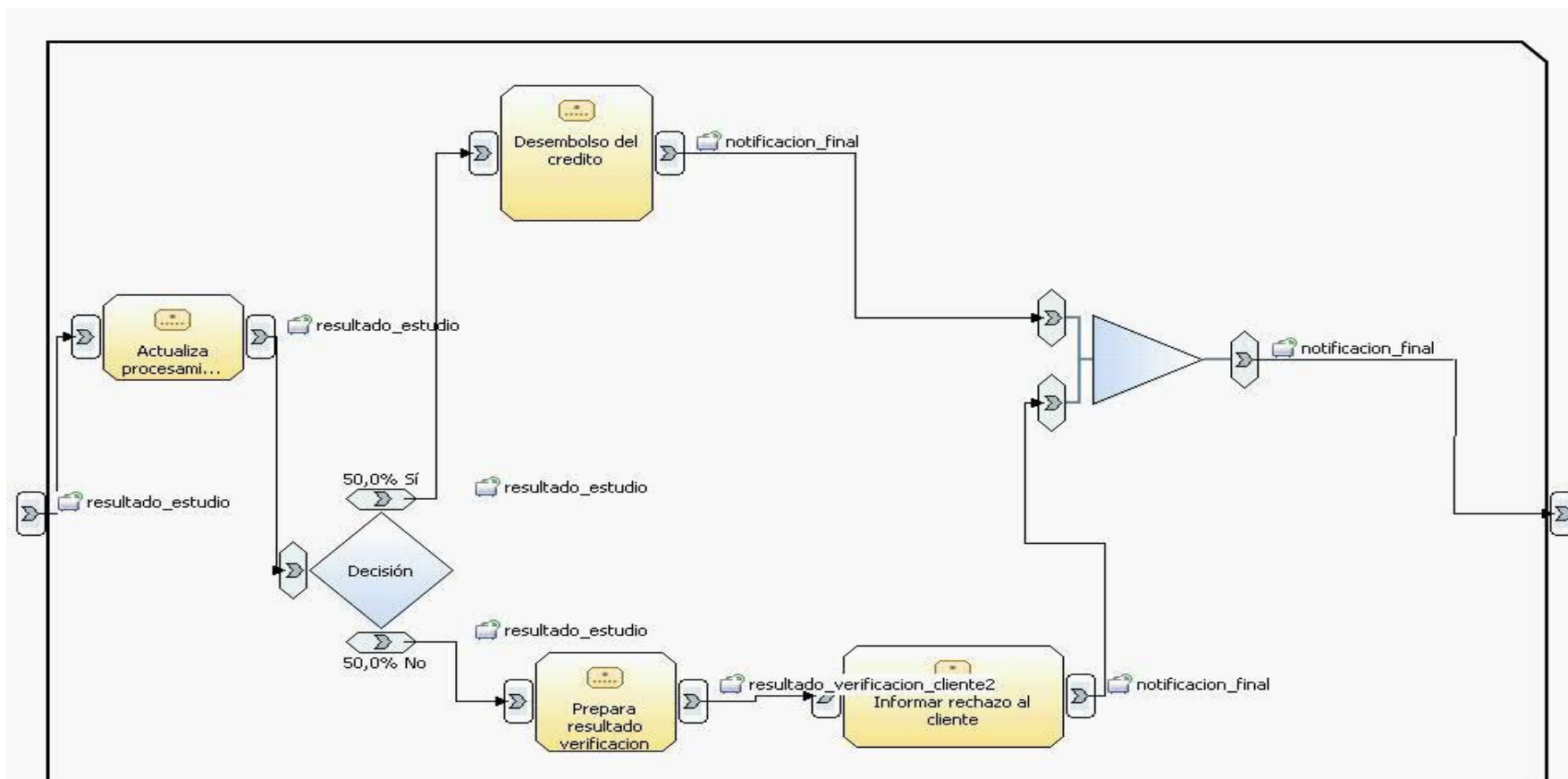


Figura 4. Vista de la etapa 2 del proceso: Estudio de solicitud

Participantes del proceso

Los siguientes participantes están involucrados en el proceso:

- **Clientes:** intervienen en la etapa 1. Generan solicitudes de crédito. Se analizan sus antecedentes crediticios tanto internos como externos respecto de la entidad y el pedido concreto de crédito. Luego se les comunica la decisión que tomó la entidad acerca del otorgamiento.
- **Analista de créditos:** interviene en la etapa 2. Es la persona encargada de supervisar el pedido concreto de crédito efectuado por los clientes, una vez validados los mismos en forma interna o externa. El mismo se encarga de decidir el otorgamiento o denegación del mismo en base a su análisis.

Servicios

- **Servicio de verificación externa:** se invoca en la etapa 1. Se encarga de verificar si la persona solicitada posee algún tipo de juicio o deuda con alguna entidad de manera tal que resulte de posible incobrabilidad ante el hecho de concederle el crédito.

Interfaces

- **Interfaz de cliente:** se despliega al comenzar la etapa 1. El cliente utiliza esta interfaz para comunicarse con la entidad crediticia, de manera de consolidar el pedido de crédito que desea efectuar
- **Interfaz de analista de créditos:** se visualiza al comenzar la etapa 2. El analista hace uso de esta interfaz para analizar las solicitudes de crédito pendientes. En base al destino del crédito, el monto y los avales presentados por el cliente, realiza el análisis acerca del otorgamiento o la denegación de la solicitud correspondiente.

Datos

- **Entrada_nueva_solicitud:** son los datos requeridos para realizar una solicitud de crédito. El cliente completa los siguientes datos: apellido y nombre, tipo y número de documento, CUIT-CUIL, monto y destino del préstamo, datos del garante, declaración jurada y dirección de correo electrónico.
- **Datos_verificacion:** son los datos necesarios para saber que tipo de verificación debe realizar la aplicación. Está compuesto por el identificador de solicitud de crédito, el DNI del cliente correspondiente y un campo booleano que indica si el solicitante ya es cliente de la entidad.
- **Resultado_verificación_interna:** es el resultado de la verificación realizada internamente dentro de la entidad. Está compuesto por el

identificador de solicitud de crédito, el DNI del cliente correspondiente y un campo booleano que indica si el cliente posee o no deuda interna.

- **Resultado_verificación_externa:** es el resultado de la verificación realizada externamente a la entidad. Está compuesto por el identificador de solicitud de crédito, el DNI del cliente correspondiente, y tres campos booleanos que indican si el cliente posee algún juicio, si es solvente para afrontar la deuda y si además posee garantía.
- **CustomerApplication:** es la entrada del web service externo que invoca la aplicación para verificar si el cliente posee juicios o declaraciones de insolvencia. Dentro de el contiene el DNI del cliente sobre el cual se realizará la verificación.
- **Resultado_estudio:** contiene el resultado del analista de créditos. Indica si se otorga o no el crédito, el monto aprobado y observaciones sobre la decisión tomada.
- **Resultado_verificacion_cliente:** contiene los datos necesarios para indicar si el cliente cumple o no cumple las condiciones para otorgarle el crédito una vez superadas las verificaciones internas, externas y el análisis de crédito correspondiente.
- **Notificacion_final:** contiene la información final para comunicar el resultado de análisis de la solicitud. Está formado por los datos personales del cliente, el monto y destino del préstamo y el resultado del análisis.

Flujo de la etapa 1 del proceso: Solicitud de crédito

1. El cliente accede al formulario para completar los datos requeridos en la solicitud de crédito. El mismo contiene datos tales como su apellido y nombre, tipo y número de documento, destino y monto del préstamo solicitado, datos de la garantía, etc.
2. Se toman los datos del cliente y se busca si ya es o no cliente de la entidad, basándose en un historial de operaciones. En caso de serlo se lo somete a una evaluación interna en la que se busca la existencia de deudas con la entidad misma.
Simultáneamente se realiza la verificación externa por medio de la invocación del servicio web que permite saber si el cliente posee algún juicio, y si es solvente para afrontar los pagos del préstamo.
3. Una vez evaluados los resultados de las verificaciones pertinentes se toma la siguiente decisión: si el cliente no posee ningún tipo de deuda u otra condición que pueda impedir la normal cancelación del préstamo, se lo ingresa al padrón de clientes para conservar su información de contacto y se procede al estudio concreto del caso. En caso contrario se le comunica la imposibilidad de otorgarle el crédito por las condiciones presentadas.

Flujo de la etapa 2 del proceso: Estudio de solicitud de crédito

4. El analista realiza el estudio de las solicitudes pendientes, las cuales son aquellas cuyos solicitantes pasaron las verificaciones internas y externas, y han quedado almacenadas en la base de datos. Se estudia la relación entre el destino solicitado y el monto pedido. Se plasma el resultado del análisis y se colocan observaciones pertinentes al mismo.
5. En base al resultado del estudio se informa al cliente el otorgamiento o no del préstamo. En caso que el resultado haya sido positivo, entonces se crea una cuenta interna para el mismo, depositándose en ella el monto del préstamo. Se le informa al cliente para que retire la suma depositada.

En caso de ser negativo, simplemente se le informa la causa por la cual no se le otorga el mismo. En este caso puede tratarse o bien porque la línea de créditos para ese destino esté momentáneamente suspendida, o bien porque está solicitando montos mayores a los permitidos para el tipo de préstamo requerido.

Modelo de datos

La aplicación se comunicará con una base de datos relacional implementada sobre MSSQL 2000, con la cual los procesos se interactuarán a través de JDBC. En la figura 5 visualizamos el modelo de la base de datos.

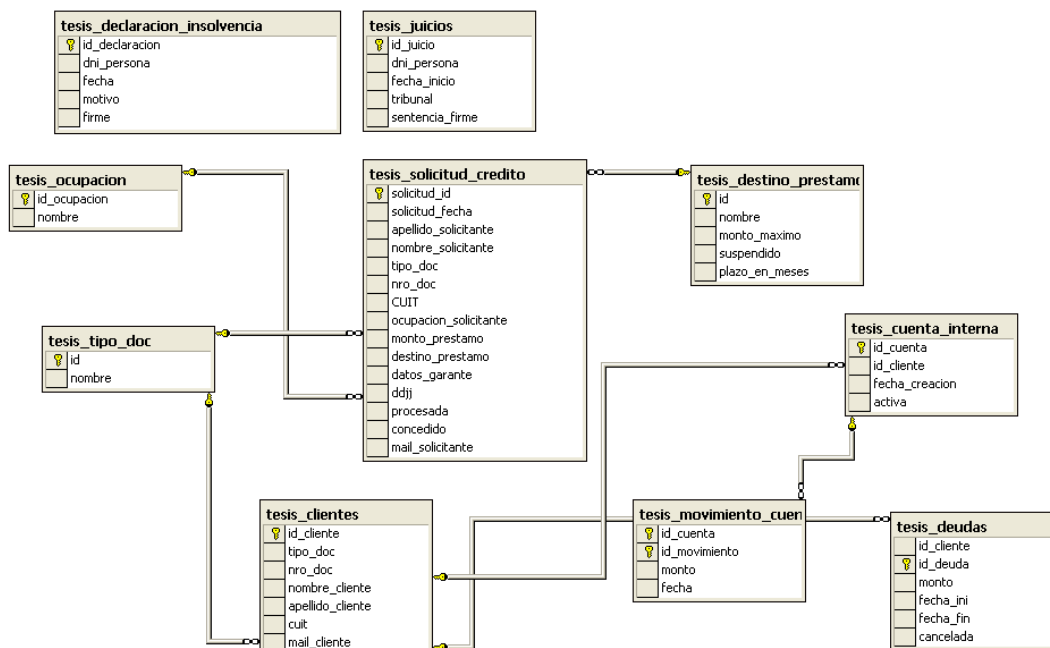


Figura 5: Modelo de datos de la aplicación, construido sobre el motor MSSQL 2000

En esta base de datos se guarda información correspondiente a las solicitudes de crédito generadas por los clientes, los clientes ingresados en el padrón de la organización, cuentas internas de los clientes, junto con los movimientos generados en las mismas, deudas que los clientes pudieran tener con la entidad, tablas de referencia (tipo de documentos, ocupación, destinos de préstamos). Además se guarda información sobre los juicios y declaraciones de insolvencia que son utilizadas por el *web service* que realiza las verificaciones externas de los clientes.

Despliegue del proceso

Una vez modelado y documentado el proceso en sus dos etapas dentro del modelador BPMN, procedemos a la exportación del mismo y a su incorporación en la plataforma de desarrollo, esto es el *Websphere Integration Developer*. Veamos la versión BPEL de ambas etapas del proceso, donde visualizamos el grafo emanado directamente de su conversión a partir del modelo BPMN. Dicha conversión es efectuada en forma automática por el entorno de desarrollo al importar el proyecto generado con el modelador. La figura 6 corresponde a la primer parte del proceso, la solicitud del crédito, donde visualizamos un flujo de control análogo al modelo BPMN pero convertido en su equivalente BPEL, y la figura 7 se asocia a la segunda fase correspondiente al estudio de dicha solicitud.

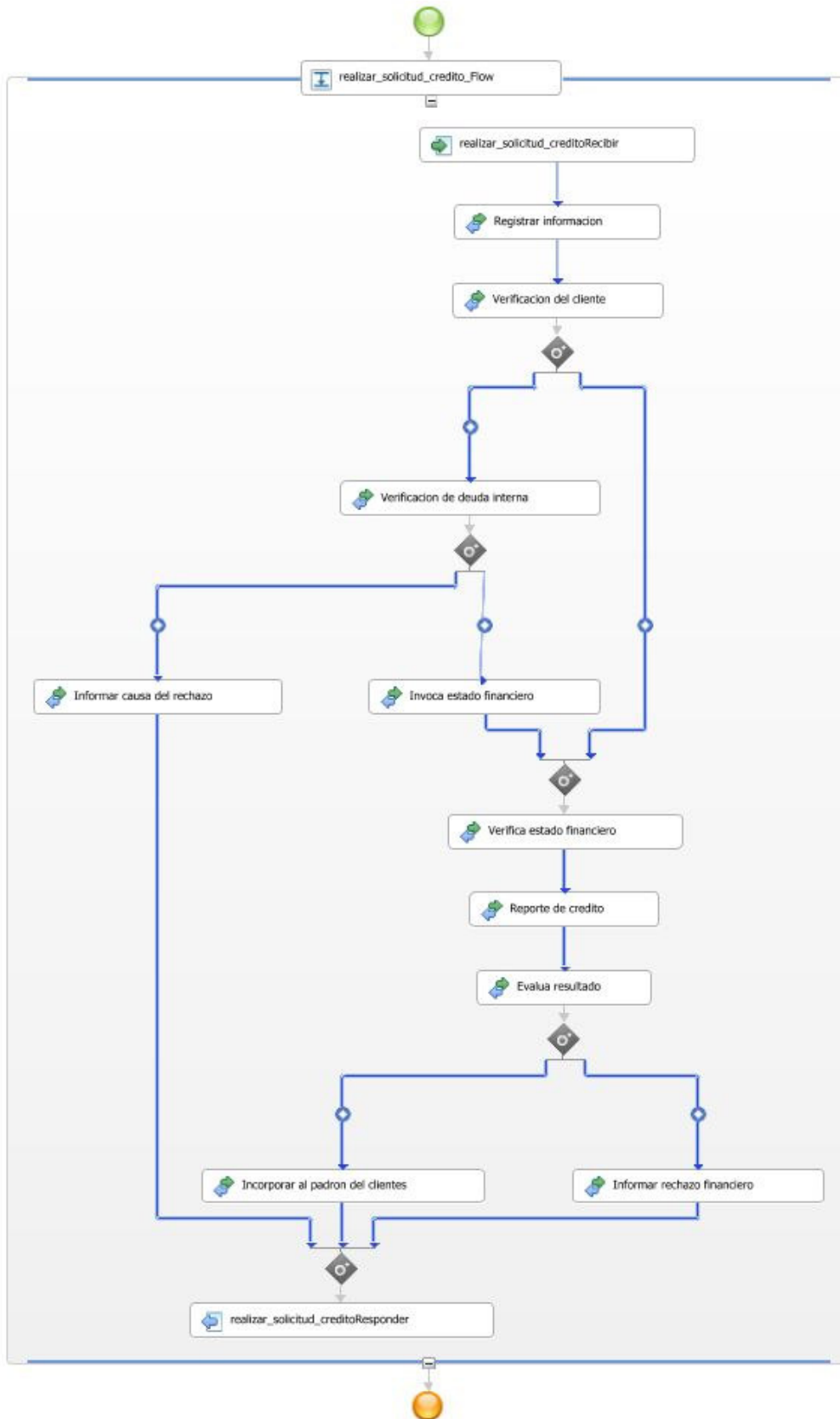


Figura 6. Versión BPEL de la etapa 1 del proceso: Solicitud de crédito

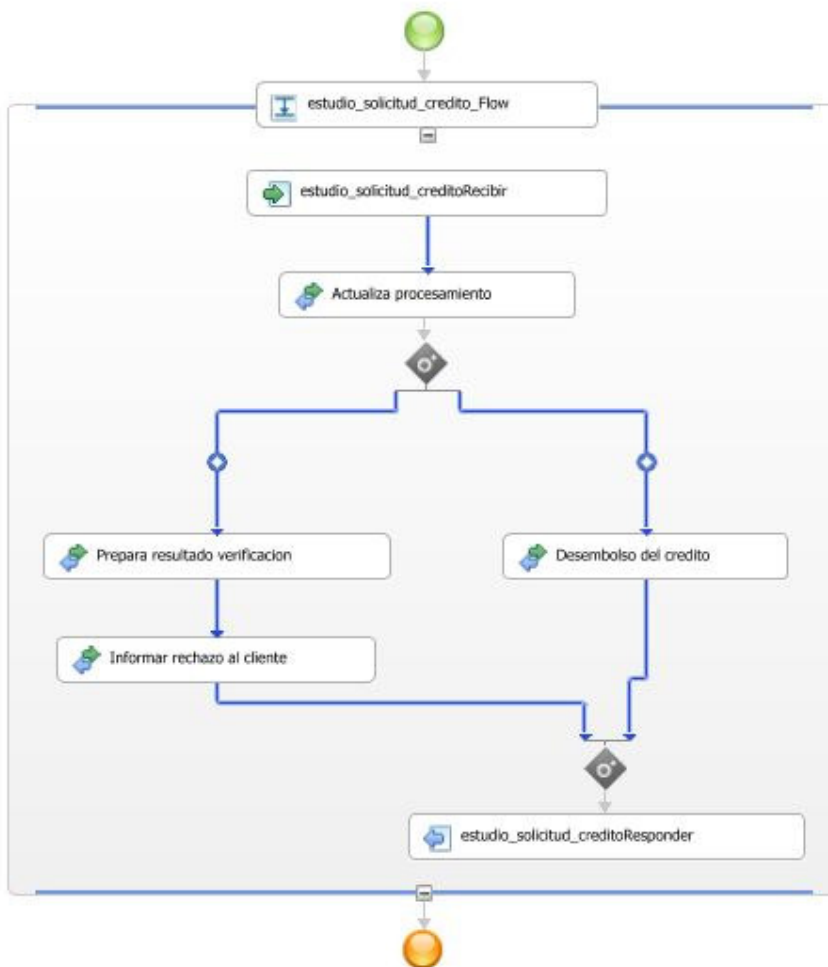


Figura 7: versión BPEL de la etapa 2 del proceso: Estudio de solicitud de crédito

Esta herramienta de desarrollo permite la posibilidad de generar manejadores en código Java para cada una de las actividades del proceso. En ellas hemos implementado las líneas que nos permiten analizar los datos recibidos como entrada, realizar las invocaciones necesarias para generar las validaciones de los clientes, y asentar los clientes y sus solicitudes en caso que corresponda.

Incorporación de tareas humanas

Ya con anterioridad vimos que BPMN soporta un tipo de tareas llamadas “tareas humanas”. Estas tareas son aquellas en las que para su resolución es necesaria una interacción del usuario con el sistema, es decir que no se resuelven en forma automática sino que necesitan de la intervención del usuario. A los fines prácticos, en BPEL este tipo de tareas son implementadas de manera tal que requieren de completar un formulario con la entrada necesaria para la resolución de la misma. Una vez que el usuario completa dicho formulario el proceso sigue con su ejecución.

Volviendo al problema particular del proceso de solicitud de crédito, vemos que en teoría no sería necesario dividirlo en dos partes, sino que el flujo de control podría extenderse en forma única desde el ingreso de la solicitud hasta la decisión de otorgamiento o negación del mismo por parte de la entidad. Pero nuestra decisión de dividirlo se ve emparentada con este tipo de actividades arriba citadas.

El paradigma de ejecución que propone *Websphere BPM* para los procesos es el siguiente: se inicia el ciclo del proceso mediante la recepción de datos cargados en un formulario, y a partir de allí se desencadena la ejecución del mismo hasta su actividad final. De esta manera las únicas actividades que reciben datos directamente cargados desde formularios visibles por interfaz gráfica son la actividad inicial más todas aquellas actividades declaradas como de tipo humano.

Para aquellas tareas declaradas como humanas, el BPC ofrece un apartado para acceder a los formularios propios de dichas actividades. Es decir, el proceso detiene su ejecución a la espera que el usuario complete los datos correspondientes a las actividades humanas y luego prosigue su ejecución.

Existe entonces en nuestro problema concreto una actividad que cuadra perfectamente en la clase de actividades humanas: aquella correspondiente al estudio de la solicitud para decidir si otorgar o no el préstamo. Esta actividad no es completamente automatizable ya que requiere un análisis racional de los antecedentes del cliente, su declaración jurada, el monto de préstamo solicitado y otros factores que indefectiblemente deben ser analizados en forma humana. Ahora bien, veamos cuáles serían los inconvenientes de implementar dicha actividad como una tarea humana:

- **Interfaces generadas:** hemos dicho que las interfaces generadas en forma automática en la actividad de despliegue del proceso y que son luego visualizadas en el BPC son muy rudimentarias comparadas con interfaces de aplicaciones web reales existentes en la actualidad. Para hacer esta afirmación nos basamos en análisis de parámetros de amigabilidad ampliamente aceptados hoy en día, como son:
 - la recepción de mensajes de error entendibles por el usuario ante errores en el ingreso de los datos
 - la mayor validación posible de errores en el lado del cliente (client side) para minimizar así los accesos al servidor (por ejemplo, por medio de *JavaScript*)
 - el uso de listas desplegables para campos de dominios taxativos
 - uso de tecnologías que optimicen la cantidad de recargas de páginas por el uso de acceso asincrónicos al servidor (como lo es *AJAX*), entre otros.

De esta manera, tanto las interfaces para el inicio del proceso como aquellas generadas para las actividades humanas no cumplen con estas características, con lo cual se dificulta abordar al postulado enunciado en nuestra propuesta: llegar a una aplicación orientada al paradigma de BPM, pero con características que permitan insertarla en un universo de aplicaciones reales, accionadas por usuarios reales.

- **Personalización de interfaces de usuario:** existe un mecanismo para personalizar las interfaces de los procesos dentro de la infraestructura de *IBM Websphere BPM* y es a través del Portal Websphere y el uso de *doclets* (productos creados por IBM para ser aplicados a interfaces gráficas de aplicaciones web). El problema fundamental ante el uso de estos elementos es que no tienen una asociación directa con el proceso, con lo cual cualquier alteración en el mismo no se ve reflejada en las interfaces, sino que hay que obligadamente modificarlas una por una en caso de ser necesario. Esto no sólo tiene un impacto en el tiempo de desarrollo de la aplicación, sino que a su vez se rompe directamente con la relación que habíamos conseguido establecer entre cada fase del ciclo de vida del proceso. Es decir, modificando elementos que forzosamente se encuentran inconexos no se puede continuar de manera natural con el ciclo de vida, ni tampoco asegurar entonces la ausencia de inconsistencias entre los resultados de las distintas etapas.
- **Entorno de exploración de procesos:** para esto entonces nos vemos forzados a salir momentáneamente del entorno de exploración propuesto por el BPC y buscar un mecanismo que nos permita invocar a nuestro proceso desde una aplicación real donde sea posible hacer validaciones y optimizaciones propias de aplicaciones web en la actualidad.

Existe una forma de invocar al proceso como si fuera un web service, es decir, como una caja negra a la cual se le envían todos los parámetros necesarios y este devuelve un resultado. Pero entonces, al salir del entorno previsto por defecto para la navegación, ¿qué ocurre con las tareas humanas que hubieran dentro del proceso?, ¿cómo acceder a los formularios que nos permiten enviarles las entradas necesarias a las mismas si ya no estamos inmersos en el entorno de ejecución del BPC? No nos queda otra alternativa que renunciar al uso purista de las mismas, manejarlas como tareas de inicio de un proceso y que reciban los datos desde un formulario de inicio.

De aquí la decisión que hemos tomado de partir nuestro proceso en dos etapas, de manera tal de permitir que la tarea de estudio siga respondiendo a un esquema de una tarea no completamente automatizable sino que requiere de la interacción del usuario, y además lograr invocar nuestro proceso desde una aplicación real.

Invocación del proceso como un servicio

Describamos ahora el procedimiento que nos permite invocar el proceso como un web service.

Para *IBM Websphere BPM* los conceptos de web service y proceso están sumamente emparentados. Se hace uso implícitamente del concepto de web service desde la definición del proceso. Aún tratándose de procesos en los que no necesariamente va a haber invocación de web services, la tarea inicial del proceso es definida como una interface (de manera tal de definir un punto de entrada), y la especificación de la misma se hace a través de un archivo wsdl (lenguaje de definición propio de los servicios web). De allí

que la noción de puertos y URL no son totalmente ajenas a la definición de un proceso, sino que están asociadas al mismo desde el momento de su creación. Igualmente, hasta el momento, el proceso no es accesible desde el exterior ya que su URL es de tipo simbólico, con lo cual la URL termina siendo de uso privado del BPC para acceder al proceso.

Siguiendo con el mecanismo, es necesario hacer una exportación del proceso como un enlace de servicio web, en la cual el *Websphere Integration Developer* genera un *stub*, es decir, un objeto asociado al proceso que sirve de representación del web service, al cual se le asignan una URL con un puerto determinado para permitir la invocación del mismo. El *stub* lleva asociado su propia definición *wsdl*.

Así, hasta el momento tenemos un proceso definido, y asociado a este un *stub* que permitirá invocarlo como web service. Necesitamos definir ahora un cliente que efectivamente consuma el servicio que hemos definido.

Para esto hace falta crear lo que se denomina un Web Service Client, el cual será una aplicación J2EE basada en servlets y jsp. Este es el caso de nuestra aplicación (llamada *tesis_joseClient*), la cual posee varias páginas jsp que sirven de interfaz para la recolección de los datos necesarios para la invocación del proceso. Una vez recolectados los mismos, se invoca al proceso haciendo uso del *stub*. Al crear al cliente web, el WID genera una serie de clases basadas en el *wsdl* de definición del servicio y que sirven para el enlace e invocación efectiva del mismo. La invocación se realiza en un servlet propio. Luego de realizada la misma, se toman los resultados devueltos por el proceso y se los muestra nuevamente en una página jsp. Un esquema idéntico es utilizado para invocar a la segunda parte del proceso correspondiente a la etapa de estudio y decisión sobre el otorgamiento del préstamo.

Una vez culminado el proceso de programación de ambos clientes web, aparece entonces un hecho no menor que es dónde publicar nuestra aplicación. En este momento surgen preguntas tales como

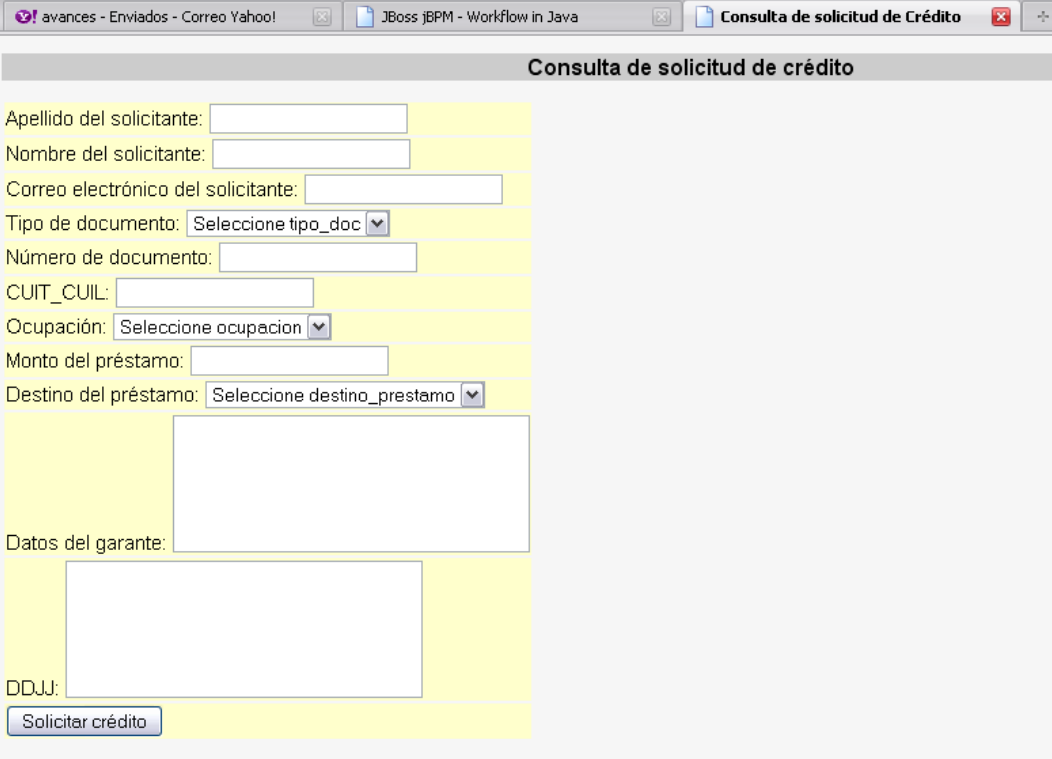
- ¿habrá problemas de compatibilidad al invocar el web service desde una aplicación externa?
- ¿surgen problemas de acceso o de conversión de datos?
- ¿Cómo ejecutar nuestro web service, si este contiene código BPEL?

Websphere soluciona todos estos problemas de una manera notable: la publicación se realiza siempre sobre el mismo servidor, en nuestro caso el Process Server. Este, como ya habíamos dicho, no es solo un servidor que soporta BPEL sino que a su vez es un servidor genérico de aplicaciones, con lo cual se puede publicar sin problemas en él procesos BPEL, aplicaciones J2EE, implementaciones de web services, etc. Esto le otorga un potencial enorme al servidor en torno al abanico de soluciones que son posibles de efectuar y publicar en él [17] [18] [19] [20].

Vemos en las siguientes figuras el detalle de interfaces de nuestros clientes web, las notificaciones recibidas y el detalle de invocación de los *stubs* correspondientes.

En la figura 8 observamos una captura de la interfaz que permite al usuario ingresar la solicitud de crédito. En ella, se encuentra con una página jsp la cual contiene un formulario HTML. En el mismo se encuentran los datos para completar dicha solicitud. Existen una serie de campos, tales como el tipo de documento, la ocupación del solicitante o el destino del préstamo cuyo dominio es taxativos, es decir, no es válido que el usuario ingrese cualquier opción. Además, los mismos están relacionados con tablas de referencia dentro de la base de datos. De allí la utilidad de visualizarlos como listas desplegables (recordar la imposibilidad de hacerlo en las interfaces generadas para visualizar en el BPC).

Por otro lado, en caso que el usuario no completara los datos que son considerados mandatorios para poder procesar la solicitud, la aplicación le solicita el llenado de los mismos a través de mensajes de alerta en *JavaScript*, siendo esto mucho más amigable que un mensaje de error visualizado una vez que se enviaron los datos al servidor.



The screenshot shows a web browser window with the title 'Consulta de solicitud de crédito'. The browser tabs include 'avances - Enviados - Correo Yahoo!', 'JBoss jBPM - Workflow in Java', and the current page. The form itself is titled 'Consulta de solicitud de crédito' and contains the following fields:

- Apellido del solicitante:
- Nombre del solicitante:
- Correo electrónico del solicitante:
- Tipo de documento:
- Número de documento:
- CUIT_CUIL:
- Ocupación:
- Monto del préstamo:
- Destino del préstamo:
- Datos del garante:
- DDJJ:

At the bottom of the form is a button labeled 'Solicitar crédito'.

Figura 8: Captura de la interfaz para realizar la solicitud de crédito

Una vez completada la solicitud, el cliente recibe un mensaje informándole el estado de la misma, y en caso de haber sido ingresada se le comunica que la misma está siendo sometida a estudio y el resultado se informará a la brevedad (Figura 9).

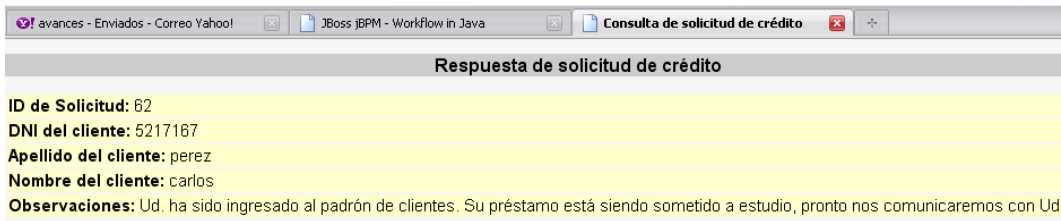


Figura 9: Captura de la interfaz de comunicación una vez ingresada la solicitud

Una vez que la solicitud ha sido ingresada, comienza la fase de estudio en la que un analista de la entidad decidirá de acuerdo a los datos completados con anterioridad si es posible o no otorgar el crédito.

Para esto, la aplicación le presenta la posibilidad de seleccionar las solicitudes pendientes de estudio. Una vez seleccionada la misma, se visualizarán los datos que la componen y el analista podrá ingresar el monto autorizado para otorgar, y algún tipo de observación que considere necesario. (Figuras 10 y 11).

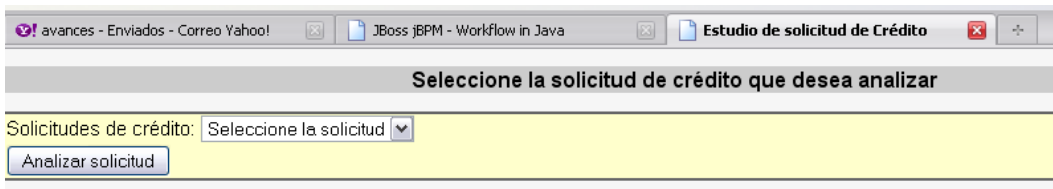


Figura 10: Selección de solicitudes para efectuar análisis

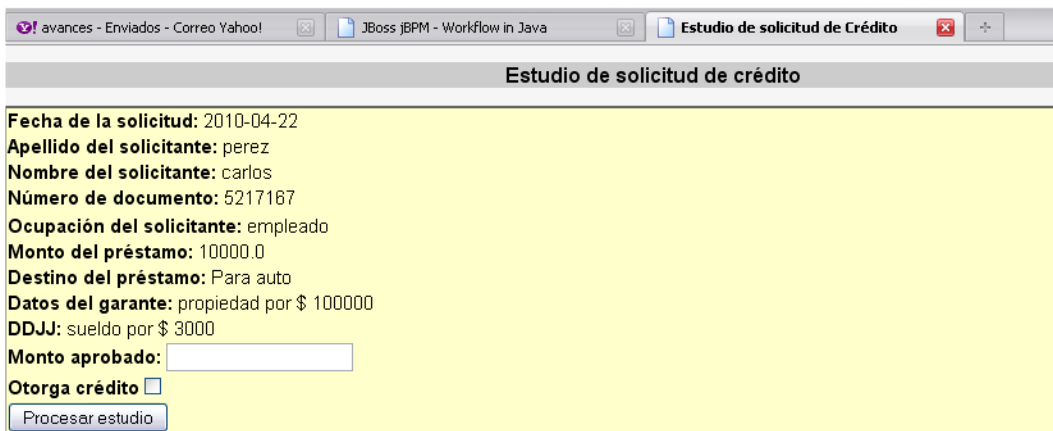


Figura 11: Visualización de solicitud a analizar

Una vez que el analista ha completado la interfaz de la Figura 9, se muestra una pantalla final con el resultado obtenido, siguiendo luego la comunicación vía correo electrónico con el cliente para comunicarle la decisión tomada (Figura 12)

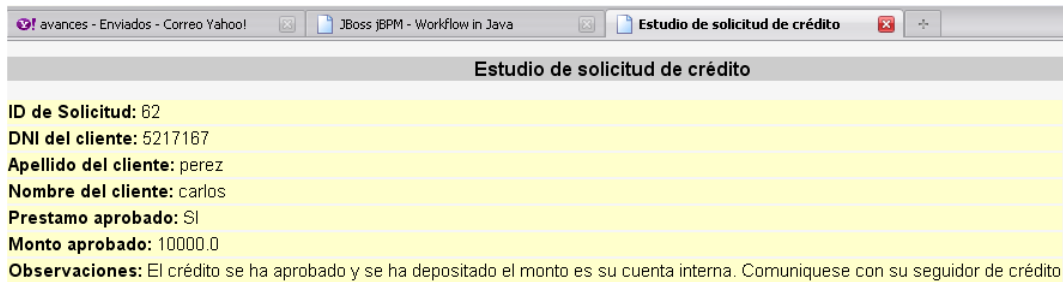


Figura 12: Pantalla de información luego del análisis

Vemos que hemos logrado crear una aplicación web que implementa el patrón MVC (*Model View Controller*), usando tecnologías de visualización y control propios de aplicaciones web reales, y que interactúa con un modelo completamente implementado en BPM con elementos de SOA.

Podemos describir todas las etapas desarrolladas como un proceso de negocio (propio por ejemplo del área técnica informática de la organización) en el que cubrimos todas las fases del desarrollo de la aplicación. En la figura 13 visualizamos el mismo en donde tendremos:

- Análisis del problema y diseño del modelo BPMN con *Websphere Business Modeler*.
- Exportación del proyecto generado.
- Incorporación del proyecto generado al *Websphere Integration Developer* y realizar la codificación necesaria.
- Si el proceso estará destinado a usuarios familiarizados con el entorno, podrá utilizarse el entorno de exploración BPC.
- En caso contrario, crear el stub propio del proceso como web service para poder invocarlo desde una aplicación real.
- Desarrollar el cliente web que permitirá invocar el stub.
- Desplegar la aplicación en el *Websphere Process Server*.

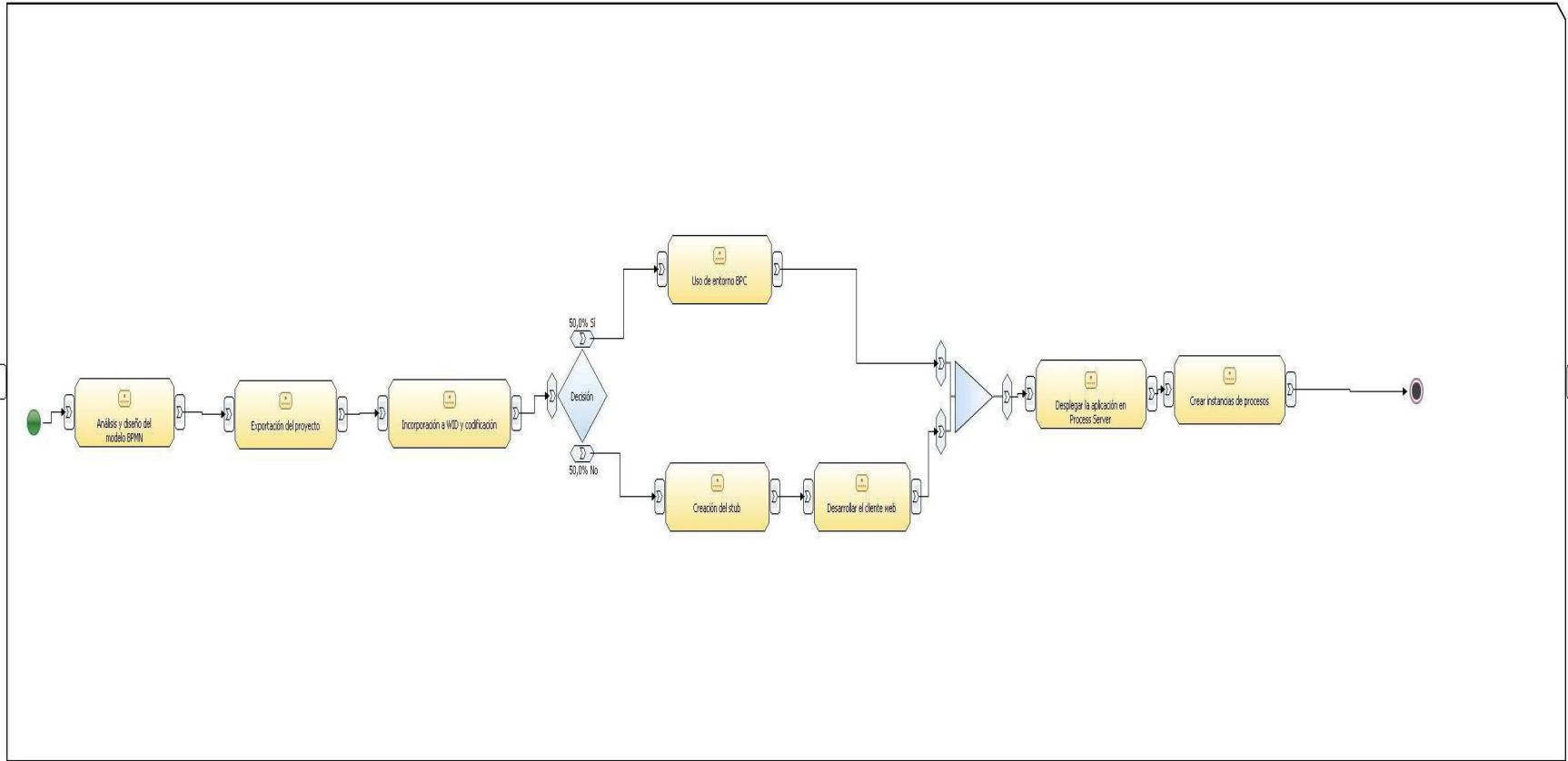


Figura 13: Modelo BPMN del proceso de desarrollo

Conclusiones

A través del capítulo precedente hemos descripto la funcionalidad alcanzada con la herramienta IBM Websphere BPM en un caso concreto de Gestión de procesos de negocio. Hemos visto el grado de cobertura que desarrolla la misma con respecto al ciclo de vida de los procesos, y como permite desde distintas perspectivas la integración de recursos existentes, así como la obtención de aplicaciones capaces de ser accedidas por usuarios finales.

En el próximo capítulo veremos distintas alternativas existentes en el mercado que brindan funcionalidad de BPMS desde una perspectiva *open source*. Analizaremos la posibilidad de cubrir con las mismas la funcionalidad obtenida con IBM Websphere BPM, junto con las ventajas y desventajas del uso de una u otra herramienta.

CAPÍTULO 3: ANÁLISIS TÉCNICO Y METODOLÓGICO DE LOS EQUIVALENTES OPEN SOURCE A IBM WEBSHERE BPM

Fundamentos

Para realizar el presente análisis nos basaremos en los puntos conceptuales que debemos cubrir para intentar igualar la funcionalidad de la herramienta IBM Websphere BPM. Recordemos que el análisis de la misma fue realizado considerando el ciclo de vida de los procesos (según [1]). De esta manera, los aspectos a considerar son:

- **Análisis y diseño de procesos:** la herramienta que propongamos nos debe permitir realizar modelos de nuestros procesos (preferentemente en notación estándar BPMN) y facilitar de esta manera la comunicación interdisciplinaria que propone BPM durante la fase de Análisis y Diseño del problema.
- **Configuración:** una vez que hemos realizado los modelos de procesos, necesitamos configurarlos, es decir, poder aplicar sobre los mismos las reglas de negocio que impone la organización subyacente. Sería deseable contar con una herramienta que integre las reglas propuestas al resultado ejecutable del proceso, y no que las mismas queden simplemente en un documento de tipo textual.
- **Despliegue:** Cuando los procesos se encuentran diseñados y configurados es necesario llegar a una versión ejecutable de los mismos. De esta manera requerimos una herramienta que permita efectuar las tareas de programación necesarias, y luego un servidor en el cual desplegar la versión final de nuestros procesos.
- **Exploración y ejecución:** Una vez desplegados los procesos, necesitamos un entorno de exploración en el cual los usuarios puedan crear instancias e interactuar con los procesos.
- **Monitoreo y administración:** Sobre los rastros dejados por las ejecuciones deberíamos poder realizar actividades de monitoreo que permitan establecer indicadores y puntos críticos que habiliten a efectuar actividades de mejora continua sobre los procesos desarrollados [1].

A través de las premisas anteriores estamos buscando una herramienta que cubra el ciclo de vida completo de los procesos, y que además, en caso de valerse de aspectos técnicos, los considere de manera estándar. Este último aspecto es muy importante, sobre todo en una metodología reciente como lo es BPM, ya que el uso de estándares nos garantizará la perdurabilidad de nuestro trabajo.

La primera de las herramientas que propondremos como candidata “equivalente” a IBM Websphere BPM es jBPM. Veremos a través del siguiente análisis si la misma cumple con las premisas propuestas.

jBPM

Es una herramienta basada en *workflow* para el trabajo con BPM. Surge como una iniciativa *open source* dentro del proyecto JBoss (www.jboss.org/jbpm), uno de los servidores de aplicaciones J2EE más utilizados dentro de la industria informática.

Esta herramienta posee licenciamiento LGPL, con lo cual estamos habilitados a modificar el código de la misma en pos de enriquecer el proyecto. Obviamente cualquier cambio sobre las mismas debe ser declarado.

La arquitectura del *framework* es la siguiente (figura 14):

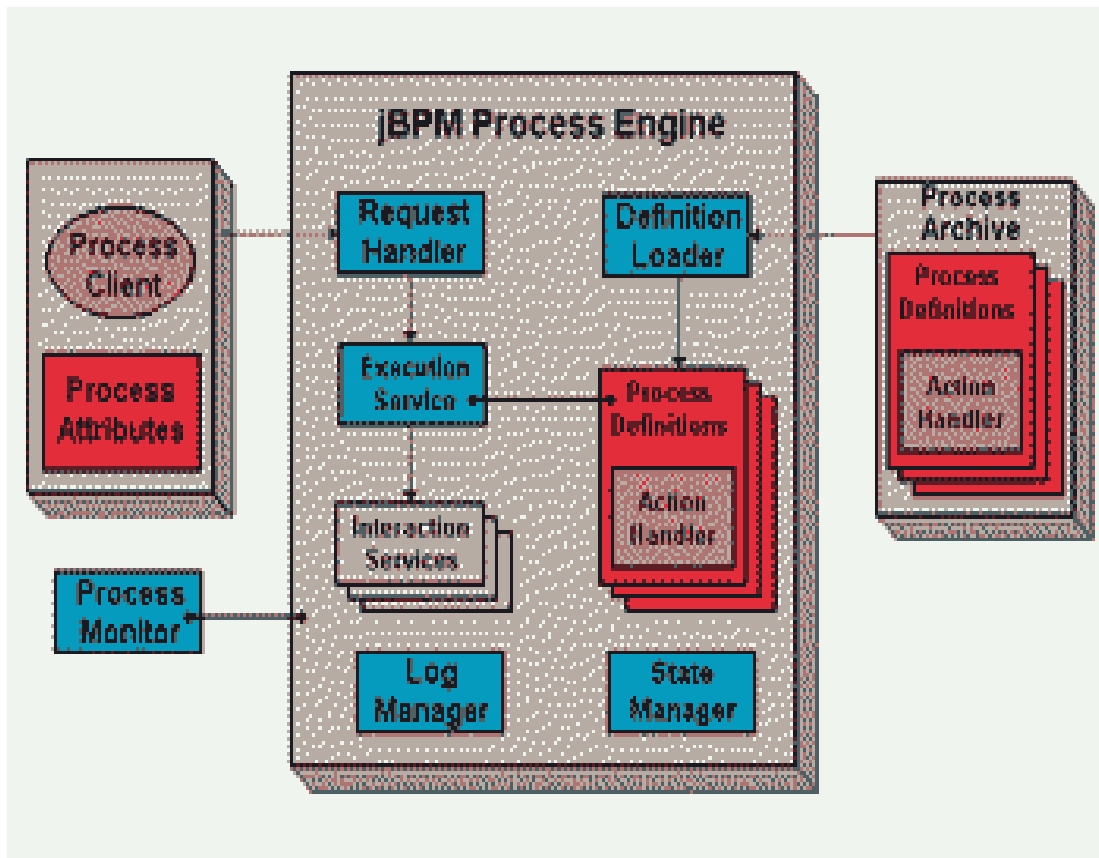


Figura 14: Arquitectura de jBPM

La arquitectura jBPM se inserta como un módulo dentro del servidor de aplicaciones JBoss. Este último es el entorno de compilación más natural para

el *framework* debido al origen del mismo, pero también se pueden lograr compilaciones válidas en otros servidores J2EE tales como *Apache Tomcat*.

Podemos ver en la figura 14 que el principal componente de toda la arquitectura es el *Process Engine*. Este componente se encarga de las siguientes tareas:

- Ejecución de las acciones propias de los procesos definidos.
- Mantenimiento del estado de los procesos.
- Almacenamiento de todos los eventos de los procesos mediante los siguientes componentes de delegación:
 - Un manejador de requerimientos (*request handler*)
 - Un manejador de estados (*state manager*).
 - Un manejador de registros para el almacenamiento de los eventos ejecutados (*log manager*).
 - Un cargador de las definiciones de los procesos (*definition loader*).
 - Un servicio de ejecución, encargado de correr la definición de los procesos generados (*execution service*).

Además del *jBPM Process Engine* encontramos los siguientes componentes:

- **Monitor de procesos (*Process Monitor*):** se encarga de encaminar, auditar y reportar el estado de los procesos ejecutados.
- **Lenguaje de procesos (*Process Language*):** el lenguaje de definición de los procesos es *jPDL (jBPM Process Definition Language)* y se encuentra basado en *GOP (Graph Oriented Programming* o Programación orientada a grafos). Se compilará a través de su *runtime*.
- **Servicios de interacción:** estos servicios exponen aplicaciones *legacy*, tales como funciones o datos que serán utilizadas en las ejecuciones de los procesos.

Tal como se muestra en la figura 14, las definiciones de procesos *jBPM* que contienen manejadores de acción (*Action Handlers*) son cargados y ejecutados por el *Process Engine*. Cuando este último encuentra un nodo en el proceso que posee un manejador de acción asociado, se invocan entonces todos los manejadores que estuvieran involucrados. Estos manejadores son instancias de código Java que pueden interactuar con sistemas externos al proceso en caso de ser necesario.

El objetivo fundamental del *Process Engine* es orquestar las interacciones entre aplicaciones y servicios. Está construido bajo una tecnología *SOA*, pudiendo interactuar con la gran mayoría de las tecnologías de integración *J2EE*, tales como *web services*, *Java Messaging Service (JMS)*, conectores *J2EE*, *JDBC (Java Database connectors)* y *EJB's (Enterprise Java Beans)*. Una vez que los

procesos se activan, el *Process Engine* se encarga de manejar estados, variables y formularios de tareas, dejando registros en los archivos de logs. En este punto es fundamental recalcar una diferencia notoria entre el *Websphere Process Server* y el *jBPM Process Engine*: mientras que en el primero podíamos desplegar todo tipo de aplicaciones relacionadas con la plataforma J2EE además de aplicaciones de procesos con código BPEL, en el segundo sólo podremos desplegar aplicaciones jBPM con código jPDL, debido a que el mismo se encuentra embebido como un módulo del *JBoss Application Server*.

Sobre este último podremos efectivamente desplegar aplicaciones de tipo J2EE, tales como *web services*, pero su manejo es completamente ajeno e independiente al funcionamiento de jBPM. De esta manera, los procesos jBPM podrán acceder a componentes externos a través del código Java, pero los mismos se deberán encontrar en un espacio de ejecución ajeno al *Process Engine*. Esto hace que las tareas de integración de recursos en un ESB sean mucho más complicadas con la presente arquitectura que con los servicios propuestos por IBM [6] [8] [10] [12].

jPDL (jBPM Process Definition Language)

jBPM utiliza para sus procesos un mecanismo de definición basado en grafos dirigidos. Los mismos se componen de nodos, transiciones, un estado inicial y uno final. Cada tipo de nodo tiene asociado un comportamiento que se desplegará al momento de la ejecución del proceso.

jPDL (descrito en Anexo 2) como lenguaje consta de los siguientes elementos:

- **Definición de los procesos:** dentro de los procesos se documentan aspectos tales como:
 - **Nombre de los procesos.**
 - **Swimlanes:** sirven para indicar roles dentro del proceso a los cuales se asocian las actividades del mismo.
 - **Evento de inicio**
 - **Conjunto de nodos:** representan el conjunto de actividades asociadas al proceso.
 - **Manejadores de acciones:** son bloques de código Java que se ejecutan ante una determinada acción dentro del proceso. Pueden estar asociados a actividades, transiciones, incluso a excepciones y al ambiente global del proceso.
 - **Estado del proceso:** durante su ejecución el proceso atraviesa diferentes estados, tales como creado, en ejecución, abortado, suspendido y terminado.

- **Nodos:** representan las actividades a ejecutar dentro del proceso. Pueden tener un manejador de acción asociado.
- **Eventos de inicio:** indican la condición por la que se desencadena el flujo de ejecución del proceso. De los mismos se considera:
 - **Nombre del nodo**
 - **Tarea de inicio del proceso**
 - **Transiciones salientes:** al tratarse de un evento de inicio es mandatorio la existencia de las mismas.
 - **Listado de manejadores:** pueden existir un conjunto de manejadores asociados que se ejecutarán ante las excepciones definidas.
- **Evento de fin:** representa la condición de finalización del proceso. Dentro de ellos encontramos:
 - **Nombre del nodo**
 - **Transiciones entrantes:** en este caso son de tipo obligatorio.
 - **Listado de manejadores asociados:** se ejecutarán ante determinadas excepciones.
- **Estados:** sirven para representar nodos dentro de un proceso, pero que poseen semántica asociada, similar a una máquina de estados.
- **Nodos de tarea:** todo nodo definido dentro del proceso puede tener su tarea asociada, la cual a su vez puede tener definido un formulario. En ese caso, se representa una tarea de tipo humana ya que para completar su ejecución el usuario deberá completar dicho formulario.
- **Fork y Join:** sirven para representar bifurcaciones y uniones en el flujo de ejecución del proceso. Son útiles a la hora de modelar paralelismo.
- **Decisiones:** se utilizan para representar los flujos condicionales que se realizan dentro del proceso. Las mismas pueden tener asociado:
 - **Un action handler**
 - **Listado de transiciones salientes:** son el conjunto de transiciones que salen de la condición y se conectan con las tareas correspondientes.
- **Events:** representan los eventos que pueden originarse durante la ejecución del proceso.

- **Transiciones:** sirven para modelar el flujo de ejecución dentro del proceso. Las mismas constituyen las direcciones dentro del grafo. De las mismas se considera:
 - **Nombre de la transición**
 - **Nodo de destino**
 - **Condición de guarda:** es una expresión que se asocia al paso del flujo de control por la transición.
 - **Manejadores de excepción asociados:** son un listado de manejadores de excepción que se ejecutarán ante la ocurrencia de la misma.
- **Action:** sirven para modelar código asociado a la ejecución de un determinado componente dentro del grafo. Tendrán asociado código Java, lo cual abre un gran abanico de posibilidades para el acceso a recursos dentro del proceso.
- **Subprocesos:** sirven para representar conjuntos de tareas que pueden ejecutarse más de una vez dentro del proceso. Facilitan la modularidad y la reutilización.
- **Asignaciones:** sirven para dar valor a las variables del proceso.

jPDL constituye de esta manera el lenguaje propio de jBPM tanto para la definición documental de los procesos, así como también para el formato ejecutable de los mismos.

Aquí encontramos entonces otra diferencia notable entre IBM Websphere BPM y jBPM: ***los lenguajes para el modelado y ejecución de los procesos son diferentes en uno y otro caso.***

Analicemos las implicancias que tiene este aspecto:

- IBM Websphere BPM usa una versión acotada de BPMN para el modelado de los procesos, y BPEL para la ejecución de los mismos. En ambos casos estamos tratando con estándares.
- jBPM usa jPDL tanto para el modelado como para la ejecución de los procesos. En este caso no solo no estamos tratando con un lenguaje estándar sino que además la semántica propuesta por el mismo no es la misma que los estándares previamente nombrados.

Consideremos entonces las principales diferencias entre jPDL y BPMN, las cuales visualizamos en la Tabla 2.

Tabla 2: Comparación entre BPMN y jPDL.

Concepto	BPMN	jPDL
Definición	Constituye el estándar elaborado por la OMG como notación para la Gestión de procesos de negocio (BPM).	Es el lenguaje propuesto por el proyecto jBPM para la gestión de procesos de negocio.
Semántica asociada	Su núcleo lo constituyen las actividades, basándose en los diagramas de actividad de UML.	Si bien permite modelar actividades, tiene una orientación a representar máquinas de estados, una semántica difícilmente representable mediante BPMN, la cual no considera en sí misma "estados del proceso", tal como lo hace jPDL.
Expresividad en la notación	Presenta gran cantidad de subclasificaciones en elementos tales como eventos de inicio, fin, eventos intermedios, tareas y compuertas.	Solo considera los tipos básicos de tareas (básicas y humanas), compuertas básicas (solo del tipo <i>if then else</i>) y eventos de inicio y fin representados mediante estados.
Descripción de datos asociados	Permite describir datos asociados a la ejecución de las actividades, los cuales se intercambian en el flujo de ejecución.	No permite modelar flujos de datos en la ejecución, sino que los mismos forman parte del entorno global del proceso.

Por lo anteriormente enunciado vemos entonces que jPDL constituye una alternativa con la cual podemos llegar a resultados similares, pero a través de una semántica literalmente diferente.

Anteriormente dijimos que jPDL no sólo es un lenguaje para modelado de procesos, sino que también es utilizado por el *Process Engine* para llegar a una versión ejecutable de los mismos. Esto lo logrará a través del *runtime* propio de jBPM.

Esto último se contrapone con el estándar BPEL utilizado por el IBM Websphere BPM. En la Tabla 3 vemos cuáles son los aspectos que diferencian un lenguaje de otro.

Tabla 3: Comparación entre BPEL y jPDL.

Concepto	BPEL	jPDL
Definición	BPEL es un lenguaje de procesos diseñado para trabajar sobre <i>web services</i> , debido a estar basado en WSDL. Este último podría potencialmente mapear adecuadamente a Java Beans, pero lo hace mejor a <i>web services</i> .	Al estar basado en Java y permitir programación en el entorno, permite invocar naturalmente Java Beans, pero requiere del uso de <i>stubs</i> para invocar <i>web services</i> .
Relación con <i>web services</i>	Desplegar un proceso BPEL se emparenta en gran medida con desplegar un <i>web service</i> .	Un proceso jPDL no tiene un emparentamiento directo con los <i>web services</i> . Puede convivir con ellos pero no está basado ontológicamente

		en los mismos.
Estandarización	Debido a la evolución de los web services en la última década y a la concepción de definición de los procesos a través de los servicios que este invoca, BPEL se constituyó en un estándar para la definición de procesos.	jPDL no es un estándar sino la versión propuesta por jBPM .
Uso de variables	Las variables de BPEL son snippets en un código XML.	Las variables de jPDL son variables de un programa Java, basado en tecnología J2EE (web container).
Metodología de diseño	Se orienta a un grafo dirigido para modelar la ejecución de los procesos.	Utiliza la noción de máquina de estados para representar semánticamente a los procesos.

A través de esta comparación podemos concluir que no existe un lenguaje que sea mejor que otro en forma incondicional, sino que debemos considerar los pros y contras de cada uno de ellos aplicados al problema concreto que debemos resolver. Cuando hablamos de una orientación a servicios en forma preponderante y debemos convivir con una arquitectura SOA, entonces sin duda alguna BPEL es la mejor alternativa. Si por el contrario, necesitamos un entorno de programación robusto y multifuncional como lo es Java, y necesitamos reflejar en nuestra solución la orientación a estados, entonces será mejor inclinarse por jPDL.

El *Process Engine* utiliza una base de datos de tipo HSQL (*Hibernate SQL*) para guardar información referente a los modelos y que no se coloca sobre el documento de definición de los procesos. Esta base se conoce como esquema (*schema*) y es portable a distintos RDBMS [9] [10] [11] [12].

jBPM Modeler

Para las tareas de desarrollo y despliegue, jBPM propone el *jBPM Modeler*, basado en la IDE Eclipse. Es un *plugin* en el que se cuenta con un espacio de diagramación junto a la paleta de componentes. En el mismo se puede modelar el proceso y detallar los aspectos documentales del mismo utilizando jPDL como lenguaje para el modelo, y Java para los manejadores de acciones.

Una vez que el proceso se encuentra desarrollado, es posible desplegarlo dentro del servidor de procesos. Las tareas de despliegue pueden realizarse dentro del mismo modelador de manera tal de agilizar la tarea de publicación, sin necesidad de salir del ambiente de desarrollo.

En este punto debemos notar una diferencia sustancial con respecto al Websphere Business Modeler: si bien jPDL es un lenguaje que puede resultar útil a la comunicación interdisciplinaria parece estar más orientado al desarrollo que al modelado en sí mismo. Es más cercano al área técnica informática que al área gerencial. Recordemos que BPM propone como metodología una participación fluida de las áreas gerenciales de la organización en la generación de los modelos de procesos. De esta manera, las ventajas que lográbamos teniendo en forma separada el ambiente de diseño con el ambiente

de desarrollo se pierden en el caso de jBPM al unir ambas tareas en una misma IDE.

Monitoreo de los procesos

JBPM posee una consola desplegable sobre el servidor de aplicaciones JBOSS. A través de la misma los usuarios pueden acceder a los procesos definidos, crear instancias, ejecutarlas y monitorear los resultados alcanzados. El abanico de reportes y salidas obtenibles sobre las instancias ejecutadas no es muy amplio, sino que más que nada se orienta a tareas de administración y visualización de estados y variables propias del proceso.

Es necesario recalcar que las interfaces gráficas que se generan para la interacción de los usuarios con los procesos son bastante pobres con respecto a estándares actuales, al igual que lo eran en el BPC de IBM Websphere. En jBPM no es posible realizar una personalización directa de las interfaces sino que las mismas son generadas al momento de crear los formularios de tarea. En caso que fuera posible la personalización de interfaces, se permitiría que los usuarios finales pudieran interactuar directamente con la consola de ejecución propuesta por el ambiente.

En cambio, para plantear soluciones orientadas a usuarios finales es necesario entonces considerar algún mecanismo alternativo que permita presentar interfaces más amigables.

Consideremos ahora un caso de uso concreto aplicado a la herramienta jBPM donde intentaremos cubrir los siguientes puntos funcionales:

- Diseño del proceso en notación jPDL (aplicada de una manera funcionalmente similar a BPMN). En esta etapa buscaremos aplicar reglas del negocio y crear formularios para la interacción del usuario con el proceso.
- Crear una versión ejecutable del proceso diseñado.
- Desplegar el proceso en el *Process Engine*.
- Monitorear el proceso desplegado de manera de buscar indicadores acerca de nodos críticos u otros aspectos susceptibles de mejora.

El caso de uso que consideraremos será idéntico al estudiado con IBM Websphere: una solicitud de crédito efectuada por un cliente genérico a una entidad financiera. A los fines prácticos utilizaremos la misma base de datos que la utilizada con el IBM Websphere BPM.

Modelo de procesos

El modelo generado con el jBPM Modeler se presenta en la Figura 15. En la misma observamos un grafo que posee un flujo de control similar al presentado por el *Websphere Business Modeler*, aunque aquí hemos decidido adoptar la semántica de máquina de estados para hacer uso de dicha potencialidad ofrecida por jBPM.

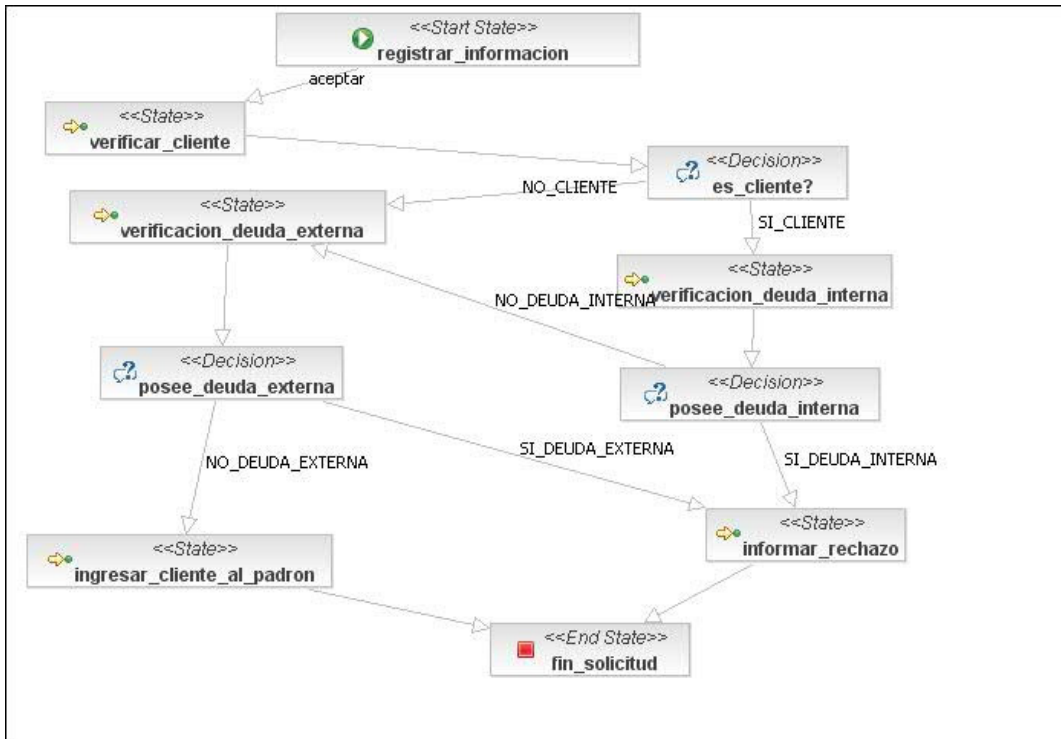


Figura 15: Representación en jPDL del proceso “Realizar solicitud de crédito”

El código generado en formato XML es el siguiente:

```

<?xml version="1.0" encoding="UTF-8"?>

<process-definition xmlns="urn:jbpn.org:jpd1-3.2"
name="solicitud_credito_estados">

  <start-state name="registrar_informacion">
    <task name="registrar_informacion"></task>
    <transition to="verificar_cliente" name="aceptar">
      <action name="action"
class="com.sample.action.FormActionHandler">
        </action>
    </transition>
  </start-state>

  <state name="verificar_cliente">
    <transition to="es_cliente?">
      <action name="action"
class="com.sample.action.Verifica_clienteActionHandler">
        </action>
    </transition>
  </state>

  <decision name="es_cliente?"
expression="#{solicitante_cliente}">
    <transition to="verificacion_deuda_interna"
name="SI_CLIENTE"></transition>
    <transition to="verificacion_deuda_interna"
name="NO_CLIENTE"></transition>
  </decision>

```

```

        <state name="verificacion_deuda_interna">
            <transition to="posee_deuda_interna">
                <action name="action"
class="com.sample.action.Verifica_deuda_internaActionHandler">
                    </action>
                </transition>
            </state>

        <state name="verificacion_deuda_externa">
            <transition to="posee_deuda_externa">
                <action name="action"
class="com.sample.action.Verifica_deuda_externaActionHandler">
                    </action>
                </transition>
            </state>

        <decision name="posee_deuda_interna"
expression="#{posee_deuda_interna}">
            <transition to="informar_rechazo"
name="SI_DEUDA_INTERNA"></transition>
            <transition to="verificacion_deuda_externa"
name="NO_DEUDA_INTERNA"></transition>
        </decision>

        <state name="informar_rechazo">
            <transition to="fin_solicitud"></transition>
        </state>

        <decision name="posee_deuda_externa"
expression="#{posee_deuda_externa}">
            <transition to="informar_rechazo"
name="SI_DEUDA_EXTERNA"></transition>
            <transition to="ingresar_cliente_al_padron"
name="NO_DEUDA_EXTERNA"></transition>
        </decision>

        <state name="ingresar_cliente_al_padron">
            <transition to="fin_solicitud">
                <action name="action"
class="com.sample.action.Ingresa_clienteActionHandler">
                    </action>
            </transition>
        </state>

        <end-state name="fin_solicitud"></end-state>

</process-definition>

```

Aspectos destacables del presente código:

- La semántica de máquina de estado es completamente distinta que la adoptada por BPMN. Aquí la ejecución de los manejadores se realiza en las transiciones, mientras que en BPMN el grueso de la funcionalidad se realiza en el cuerpo de las tareas.

- Así, vemos que cada una de las transiciones tiene asociado el manejador de acción correspondiente. Los mismos son segmentos de código Java que serán invocados por el servidor de procesos en el momento de la ejecución. En ellos es donde se realiza el grueso de la funcionalidad, tal como la conexión con la base de datos y las consultas sobre el cliente.
- Existen muchos aspectos documentales que no están directamente reflejados en el documento XML, como son las descripciones de roles y actividades y formularios de actividades. Estos se reflejan en el esquema de jBPM manejado en una base de datos externa.

Una vez que hemos modelado el proceso, creado todos los manejadores de acción y documentado las reglas de negocio, entonces es momento de desplegar el proyecto sobre el servidor de procesos. Dicha tarea la realizaremos desde el mismo Eclipse, en el área etiquetada como *Deployment*. Recordemos la premisa considerada al analizar el BPC de *Websphere*: si el proceso va a interactuar con usuarios finales, tal como es este caso, necesitamos de interfaces que respondan a las normas comúnmente adoptadas para amigabilidad y accesibilidad de aplicaciones.

La consola de procesos de jBPM despliega formularios rudimentarios en los que no se pueden usar mecanismos de verificación de errores sobre los datos, tales como Javascript asociados al *Client Side*, o técnicas para presentar los campos del formulario que poseen dominios taxativos a través de listas desplegables. Por esto es que necesitamos construir una aplicación propia que presente un formulario con el cual el usuario pueda interactuar en forma amigable, y luego invocar al proceso desde nuestra aplicación.

Dicho perfil de aplicación lo hemos efectuado nuevamente con una aplicación web basada en *servlets* y páginas *jsp*.

Las páginas *jsp* nos sirven para presentar el formulario de solicitud de crédito y la respuesta a la solicitud, y el *servlet* nos sirve para interactuar con nuestro modelo, que en este caso es un proceso desarrollado en jPDL. Cabe aclarar que el modo de interacción de nuestro proyecto web con el proceso es a través del modo de librerías, es decir que el proceso jBPM es una librería más de nuestra aplicación web.

En la figura 16 visualizamos nuevamente el formulario de ingreso a la aplicación.

Figura 16: Formulario de solicitud de crédito

En la figura 17 visualizamos la respuesta que presenta la aplicación una vez realizada la solicitud

Figura 17: Respuesta de solicitud de crédito

En este punto debemos preguntarnos qué es lo que hemos conseguido:

- Desarrollamos un modelo de proceso en la notación jPDL, que nos permite tener una versión documentada y ejecutable de nuestro proceso.
- Al ser una aplicación destinada a usuarios finales desarrollamos una aplicación web que presenta un formulario de ingreso y que interactúa con nuestro proceso previamente generado.

Es decir que hasta el momento hemos conseguido prácticamente lo mismo que con la herramienta de IBM, salvo por los siguientes detalles:

- El proceso no pudo ser modelado en BPMN, con el agravante que tiene esto en no permitir una fácil intervención de las áreas no técnicas en el modelado de los procesos.
- El proceso no pudo ser invocado como un web service debido a la incapacidad del *Process Engine* de desplegar web services. Esto obligaría a usar el método de librerías cada vez que se quiera invocar el proceso, en vez de la portabilidad ofrecida por el mecanismo de invocación provisto por un *stub*.

En este punto deberíamos preguntarnos si es suficiente para nuestra organización adoptar la solución propuesta o debería seguir en la búsqueda de productos que nos ofrezcan una solución más fiel a lo que hemos obtenido con la herramienta de IBM [13] [14] [15].

jBPM BPEL

Estudiando el mercado hemos encontrado una versión de jBPM que utiliza BPEL como lenguaje de especificación de procesos en lugar de jPDL. La misma se vale de un servidor de procesos propio que, al igual que la versión jPDL corre sobre el servidor JBoss.

Dicho *framework* se vale de un *plugin* de Eclipse para el modelado de los procesos en formato BPEL. Recordemos que si bien BPEL es un lenguaje para la especificación de procesos, es esencialmente un lenguaje técnico y de desarrollo. No contempla la posibilidad de documentar reglas de negocio, roles o restricciones temporales tal como lo hace BPMN.

Aquí podríamos plantear el siguiente interrogante:

- Si utilizáramos jPDL como lenguaje para la especificación de nuestros procesos gozaríamos de un modelo lo suficientemente expresivo como para representar problemas del mundo real, con la flexibilidad provista por la ejecución de código Java en los manejadores. Ahora bien, si pudiéramos tomar el código jPDL generado y convertirlo a BPEL entonces tendríamos un mecanismo para especificar procesos y lograr una versión ejecutable de los mismos en formato estándar.

¿Existe entonces alguna manera de pasar de jPDL a BPEL sin perder información?

La respuesta es **NO**.

Existen herramientas como Visual Paradigm 7.2 que permiten convertir un documento jPDL en un documento XPDL. Esta última herramienta no es *open source*.

Por otro lado, existen líneas de investigación actual para transformar de XPDL a BPEL, como la propuesta por la herramienta EnhydraJawe. Pero es casi imposible no perder información al pasar de un modelo al otro, debido a lo disímiles que son ambos paradigmas [16].

De esta manera, aunque propusiéramos jBPM BPEL como alternativa a la falta de estandarización de jPDL, no seríamos capaces de continuar con el ciclo de vida propuesto para nuestros procesos, ya que no habría relación directa entre los modelos generados y la versión ejecutable de los mismos. Si se introdujera una modificación sobre la versión BPMN de nuestro proceso, no sería trivial introducirla en forma equivalente en la versión BPEL del mismo.

Por lo tanto esta alternativa queda desechada [11] [12] [13] [15] [18].

Intalio BPMS

Siguiendo con la investigación de productos relacionados con BPM encontramos un BPMS generado por la empresa Intalio.

La misma desarrolla productos basados en BPM que alimentan dos vertientes:

- Una llamada *Community*, de tipo gratuita y *open source* que cuenta con un modelador de procesos basado en Eclipse y un servidor de procesos BPEL creado sobre el proyecto Apache Tomcat
- Otra rama empresarial, en la cual se ofrecen productos relacionados con SOA tales como un ESB empresarial o editores de formularios de procesos los cuales son pagos y no son open source.

Analicemos las prestaciones propuestas por la rama *open source* de manera tal de ver si cubre o no nuestras expectativas.

El modelador de procesos está construido como un *plugin* de Eclipse completamente basado en BPMN. Presenta una única diferencia en torno a que no subclasifica las tareas, sino que sólo presenta tareas de tipo básico. Las compuertas, eventos de inicio, fin e intermedios son todos los propuestos por el estándar.

Analicemos un caso de estudio genérico, en el que invocamos un *web service* externo. En este caso invocaremos un *web service* que nos retorna la hora de cualquier ciudad del mundo que solicitemos. El mismo se encuentra descrito en la figura 18

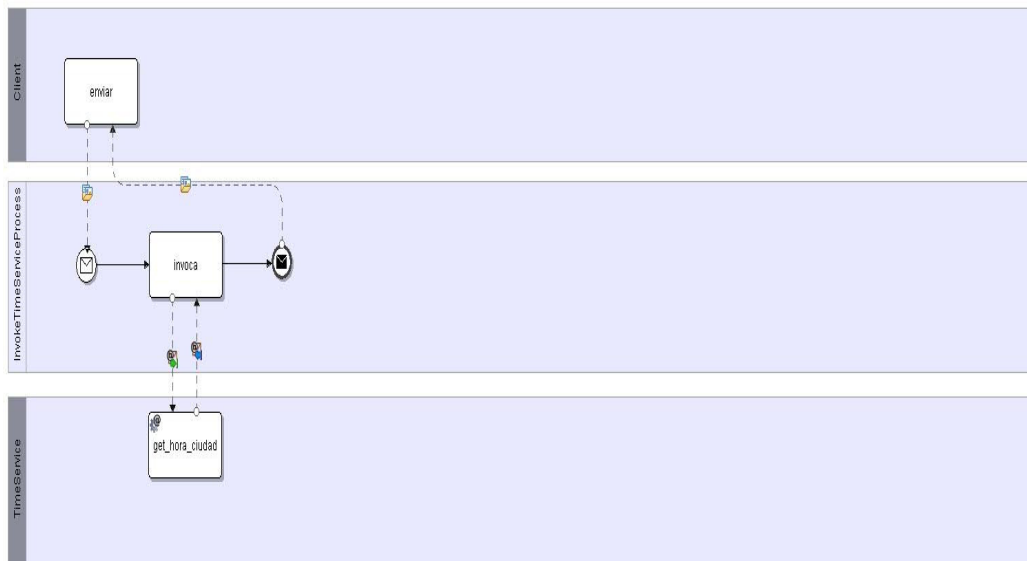


Figura 18: Invocación de web service con Intalio BPMS

Es necesario aclarar que dicho modelador, si bien presenta una alta adhesión al estándar, no permite la generación de ninguna línea de código asociada al proceso. De esta manera, el proceso va a estar definido por los *web services* que se invocan. Se presenta de esta manera una completa fusión con SOA

(Service Oriented Architecture). Esto puede resultar no aconsejable por las siguientes razones:

- Es difícil imaginar una organización real que posea todos los componentes de un proceso en forma de *web service* para poder invocarlos.
- Aún en caso de poseerlos, se vuelve incontrolable la gestión de tanta cantidad de servicios. Perdemos así el beneficio del orden y la granularidad que idealmente logra la metodología BPM al tender a simplificar el flujo de las operaciones.

El modelador, una vez culminado el proceso de modelado permite efectuar el despliegue del proceso en el servidor BPEL.

En el presente caso de estudio, el código generado es el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<bpel:process xmlns:bpel="http://docs.oasis-
open.org/wsbpel/2.0/process/executable"
xmlns:vprop="http://docs.oasis-open.org/wsbpel/2.0/varprop"
xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ode="http://www.apache.org/ode/type/extension"
xmlns:TimeService="http://example.com/invoca_servicio_horario/TimeServ
ice" xmlns:TimeService1="http://ws.intalio.com/TimeService/"
xmlns:this="http://example.com/invoca_servicio_horario/InvokeTimeServi
ceProcess" xmlns:tns="http://www.example.org/GetTime"
xmlns:diag="http://example.com/invoca_servicio_horario"
xmlns:Client="http://example.com/invoca_servicio_horario/Client"
xmlns:xml="http://www.w3.org/XML/1998/namespace"
xmlns:bpmn="http://www.intalio.com/bpms"
xmlns:atomic="http://ode.apache.org/atomicScope"
queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath2.0"
expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath2.0"
bpmn:label="InvokeTimeServiceProcess" name="InvokeTimeServiceProcess"
bpmn:id="_yh2pwLb0Ed2J08YLg1qROw"
targetNamespace="http://example.com/invoca_servicio_horario/InvokeTime
ServiceProcess">
  <bpel:import namespace="http://example.com/invoca_servicio_horario"
location="invoca_servicio_horario.wSDL"
importType="http://schemas.xmlsoap.org/wSDL/"/>
  <bpel:import
namespace="http://example.com/invoca_servicio_horario/InvokeTimeServic
eProcess" location="invoca_servicio_horario-
InvokeTimeServiceProcess.wSDL"
importType="http://schemas.xmlsoap.org/wSDL/"/>
  <bpel:import namespace="http://ws.intalio.com/TimeService/"
location="TimeService.wSDL"
importType="http://schemas.xmlsoap.org/wSDL/"/>
  <bpel:partnerLinks>
    <bpel:partnerLink name="invokeTimeServiceProcessAndClientPlkVar"
partnerLinkType="diag:InvokeTimeServiceProcessAndClient"
myRole="InvokeTimeServiceProcess_for_Client"/>
    <bpel:partnerLink
name="invokeTimeServiceProcessAndTimeServiceForPortTimeServiceSoapPlkV
ar"/>
  </bpel:partnerLinks>
</bpel:process>
```

```

partnerLinkType="diag:InvokeTimeServiceProcessAndTimeServiceForPortTimeServiceSoapPlk" initializePartnerRole="yes"
partnerRole="TimeService_for_InvokeTimeServiceProcess"/>
</bpel:partnerLinks>
<bpel:variables>
  <bpel:variable name="thisEventStartMessageRequest"
messageType="this:EventStartMessageRequest"/>
  <bpel:variable name="thisEventStartMessageResponse"
messageType="this:EventStartMessageResponse"/>
  <bpel:variable name="timeService1GetCityTimeRequestMsg"
messageType="TimeService1:getCityTimeSoapIn"/>
  <bpel:variable name="timeService1GetCityTimeResponseMsg"
messageType="TimeService1:getCityTimeSoapOut"/>
</bpel:variables>
<bpel:sequence>
  <bpel:receive
partnerLink="invokeTimeServiceProcessAndClientPlkVar"
portType="this:ForClient" operation="EventStartMessage"
variable="thisEventStartMessageRequest" createInstance="yes"
bpmn:label="Message_Start_Event" name="Message_Start_Event"
bpmn:id="_BjF0ULb1Ed2J08YLg1qROw"></bpel:receive>
  <bpel:assign name="init-variables-InvokeTimeServiceProcess"
bpmn:id="_BjF0ULb1Ed2J08YLg1qROw">
    <bpel:copy bpmn:label="$thisEventStartMessageResponse">
      <bpel:from>
        <bpel:literal>
<this:EventStartMessageResponse>
  <tns:cityTime></tns:cityTime>
</this:EventStartMessageResponse></bpel:literal>
        </bpel:from>
        <bpel:to>$thisEventStartMessageResponse.body</bpel:to>
      </bpel:copy>
      <bpel:copy bpmn:label="$timeService1GetCityTimeRequestMsg">
        <bpel:from>
          <bpel:literal>
<TimeService1:getCityTime>
  <TimeService1:city></TimeService1:city>
</TimeService1:getCityTime></bpel:literal>
        </bpel:from>
        <bpel:to>$timeService1GetCityTimeRequestMsg.parameters</bpel:to>
      </bpel:copy>
    </bpel:assign>
    <bpel:assign bpmn:label="invoca" name="invoca"
bpmn:id="_B9sssLb1Ed2J08YLg1qROw">
      <bpel:copy>
        <bpel:from>$thisEventStartMessageRequest.body</bpel:from>
      </bpel:copy>
      <bpel:to>$timeService1GetCityTimeRequestMsg.parameters/TimeService1:city</bpel:to>
    </bpel:copy>
  </bpel:assign>
  <bpel:invoke
partnerLink="invokeTimeServiceProcessAndTimeServiceForPortTimeServiceSoapPlkVar" portType="TimeService1:TimeServiceSoap"
operation="getCityTime"
inputVariable="timeService1GetCityTimeRequestMsg"
outputVariable="timeService1GetCityTimeResponseMsg"
bpmn:label="invoca" name="invoca-1"
bpmn:id="_B9sssLb1Ed2J08YLg1qROw"></bpel:invoke>

```



```

    <bpel:assign bpmn:label="Message_End_Event "
name="Message_End_Event " bpmn:id="_H9xYwLb1Ed2J08YLg1qROw">
    <bpel:copy>

<bpel:from>$timeService1GetCityTimeResponseMsg.parameters</bpel:from>
    <bpel:to>$thisEventStartMessageResponse.body</bpel:to>
    </bpel:copy>
</bpel:assign>
    <bpel:reply partnerLink="invokeTimeServiceProcessAndClientPlkVar "
portType="this:ForClient " operation="EventStartMessage "
variable="thisEventStartMessageResponse "
bpmn:label="Message_End_Event " name="Message_End_Event-1 "
bpmn:id="_H9xYwLb1Ed2J08YLg1qROw"></bpel:reply>
    </bpel:sequence>
</bpel:process>

```

En este código podemos observar que se invoca un *web service* externo residente en la misma empresa Intalio, el cual retorna la hora de la ciudad ingresada.

Recapitulando, lo que hemos obtenido con la presente herramienta es lo siguiente:

- Un proceso completamente modelado en BPMN
- Capacidad de invocar *web services*, tanto internos como externos al servidor
- Se presentan posibilidades de reformular las vistas de los formularios, de manera tal de lograr mayor amigabilidad.
- Una versión ejecutable del proceso en código BPEL, lo cual respeta el estándar.

Hasta ahora, pareciera que hemos encontrado una herramienta que cubre exactamente la funcionalidad que habíamos logrado con IBM Websphere BPM. Analicemos las desventajas:

- El servidor BPEL, si bien está basado en Tomcat presenta una serie de diferencias con el servidor de aplicaciones original. Esto imposibilita la capacidad de desplegar nuevos servicios en este servidor, con la consecuente necesidad de plantear una arquitectura de red alternativa para poder acceder a los recursos.
- Existe funcionalidad relacionada con la creación de un ESB, editores de formularios para la personalización de tareas humanas y otros aspectos relacionados con BPM que están sesgadas para la versión *open source*, debido a que dichas funcionalidades son ofrecidas como de tipo propietario.
- Debido a que el proceso es directamente desplegado en formato BPEL y no se permite al usuario introducir código Java, se dificulta de esta

manera la invocación del proceso desde una aplicación genérica de usuario, como por ejemplo una aplicación web.

- La decisión de conservar los componentes actuales como open source puede responder directamente a la voluntad empresarial de Intalio, con lo cual se corre el riesgo de quedar como rehén de un producto que puede convertirse en completamente propietario en algún momento.

De esta manera debemos concluir que Intalio se constituirá en una opción válida para aquellas organizaciones capaces de recolectar todos los *web services* necesarios para implementar cada uno de sus componentes de proceso. Esto les dará la ventaja, en caso de lograrlo, de poseer un proceso funcional implementado con los estándares actuales [7] [11].

Conclusiones

A través del análisis precedente hemos podido notar la existencia de una serie de herramientas de tipo *open source* con funcionalidad que permite trabajar con BPM desde distintas perspectivas. Una como jBPM lo hace desde el punto de vista de una máquina de estados con mucha flexibilidad debido a la incorporación de clases Java. La otra como Intalio, más restringida desde el punto de vista de la implementación se vale solo de estándares para llegar a una versión operativa.

En el próximo capítulo concluiremos el grado de cobertura funcional que podemos efectuar con las mismas con respecto a IBM Websphere BPM, además de plantear las posibles líneas futuras de investigación que permitan cubrir los puntos ausentes en la funcionalidad actual de las herramientas analizadas.

CAPÍTULO 4: CONCLUSIONES Y POSIBLES LÍNEAS DE INVESTIGACIÓN FUTURA

Conclusiones

BPM (*Business Process Management* – Gestión de procesos de negocio) constituye una visión innovadora de la orientación a procesos dentro de las organizaciones. No por el concepto de proceso, el cual ya está arraigado hace décadas en la industria, sino por la metodología propuesta. Los puntos sobresalientes de la misma son:

- La acción interdisciplinaria de las distintas áreas de la organización en pos de la definición y posterior ejecución del proceso.
- El interés por la reutilización de recursos existentes en la organización, tales como aplicaciones *legacy*.
- La capacidad de absorción rápida por parte de las áreas técnicas de los requerimientos fijados por los sectores gerenciales de la organización.

El hecho que una organización tome la decisión operativa de comenzar a gestionar su actividad por medio de la orientación a procesos, y además decida administrar sus procesos troncales por medio de BPM indica un grado de madurez importante. No se trata de un cambio de paradigma, sino una decisión tendiente a mejorar el uso de recursos dentro de la misma, así como a ordenar el flujo de ejecución y la interacción entre sus dependencias durante la ejecución de los procesos que dan vida a la misma.

Es necesario ver que dentro de las organizaciones, los procesos constituyen el corazón del funcionamiento de la misma. Si los procesos fallan, entonces se fallará en el alcance de los objetivos de negocio.

Cuando la organización toma conocimiento en forma clara de los procesos que en ella se ejecutan, esto implica:

- Clarificar las dependencias de la organización que intervienen en un proceso. Esto permite determinar responsables para cada una de las actividades ejecutadas.
- Establecer en forma fehaciente los recursos que el proceso consume, tanto en términos económicos como humanos. Esto posibilitará identificar posibles cuellos de botella y optimizar de esta manera la asignación de recursos dentro de los participantes.
- Especificar en forma concisa los objetivos a los que se pretende llegar, y efectivamente verificar el logro de los mismos. Esto a su vez permite ver el grado de eficacia y eficiencia en la tarea organizacional.

BPM, como toda metodología de trabajo, puede ser adoptada en distintas medidas. Lo ideal sería gestionar los procesos mediante dicha metodología durante todo el ciclo de vida de los procesos. Para poder efectuar esta tarea las organizaciones se valen de los BPMS, de manera tal de automatizar la mayor cantidad posible de tareas, y facilitar el control y reutilización de recursos.

IBM Websphere BPM constituye a nuestro entender uno de los BPMS más robustos de la actualidad. Hemos podido concluir a lo largo de nuestro análisis en el capítulo 2 la cobertura del ciclo de vida completo de los procesos por medio de su arquitectura.

Si bien es una herramienta propietaria, adopta estándares para las distintas etapas. En la fase de modelado, si bien *Websphere Business Modeler* no usa BPMN en forma literal, su aproximación es bastante cercana. Luego, ya para el desarrollo y ejecución de procesos utiliza BPEL, el cual es el estándar adoptado por la industria para el uso de procesos y *web services* en forma conjunta.

Hemos visto también que ofrece capacidades de monitoreo, pero que las mismas se encuentran supeditadas al uso del BPC (*Business Process Coreographer*) como entorno de navegación, lo cual se vuelve un punto cuestionable. Si los procesos están orientados a usuarios finales, dicho explorador de procesos no es una opción válida en cuanto a amigabilidad e interacción visual. La interfaz del mismo está muy orientada al ambiente de desarrollo.

De esta manera, para el caso antes mencionado hemos propuesto la creación de aplicaciones que interactúen con los procesos desde un entorno más amigable. Para efectuar dicha tarea hemos desarrollado una aplicación web que consume al proceso como un *web service*. De esta manera logramos disparar la ejecución del proceso desde un entorno amigable para el usuario y que además es funcionalmente correcto desde el punto de vista de la metodología.

Hay que notar que dicha aplicación fue construida considerando el patrón de diseño MVC (*Model – View - Controller*). En la misma las vistas son las páginas jsp, el controlador está dado por los *servlets* que determinan el flujo de ejecución de la aplicación, y por último el modelo constituido por el proceso de negocio. Mediante el uso de esta aplicación conseguimos los siguientes resultados:

- Independencia del formato de visualización con respecto al modelo de la aplicación.
- Integración del proceso en una arquitectura SOA, ya que el mismo es consumido como un *web service*.
- Invocación del proceso desde un entorno amigable y depurado.

Durante todo el proceso de desarrollo hemos podido vislumbrar la potencia del corazón de la arquitectura: el *Websphere Process Server*. Podemos concluir que el mismo es un servidor robusto, con gran capacidad de integración de recursos, debido a la capacidad de generar la convivencia de aplicaciones J2EE con procesos BPEL, y además de situar a todas ellas en un ESB. Desde este punto de vista no presenta un equivalente completamente funcional en la industria al momento.

También estamos en la obligación de recalcar una serie de puntos factibles de perfección dentro de la arquitectura:

- El *Websphere Process Server* posee una serie de *bugs* que aún se encuentran en perfeccionamiento.

- Las capacidades de ingeniería inversa son muy reducidas: si bien es fácil incorporar un proceso en notación BPMN al ambiente de desarrollo para generar la versión BPEL del mismo, la tarea inversa no es factible de realizar. Esto puede provocar a la larga una desconexión entre los procesos documentados y los que efectivamente se encuentran ejecutándose en el servidor.
- La plataforma Websphere sigue siendo un fuerte consumidor de recursos de hardware, limitando entonces el universo de utilización del mismo.

De esta manera podemos concluir que a pesar de ser una herramienta de tipo propietaria, con las consecuencias económicas y de mantenimiento propias de este tipo de licencia, IBM Websphere BPM constituye una herramienta real y consistente en la Gestión de Procesos de negocio.

Por otro lado, hemos analizado dos BPMS *open source* disponibles en el mercado con el objetivo de intentar cubrir la misma funcionalidad que la alcanzada por medio de la herramienta de IBM.

Los resultados alcanzados fueron dispares por lo siguiente:

- jBPM es una herramienta consistente y flexible para la gestión de procesos, pero lo hace desde una arquitectura y un lenguaje propios. Esto implica que, al ser un proyecto aún con muchos detalles en etapa experimental, no se torne una opción tentadora para herramienta corporativa. Si el proyecto jBPM no prosperara esto ocasionaría serios problemas en la organización que la hubiera adoptado.
- Intalio es un BPMS menos completo que jBPM desde el punto de vista de la funcionalidad provista, pero lo hace desde el uso de estándares. Esto indica la posibilidad de reutilización de los procesos en él generados, ya sea desde el punto de vista notacional como en ejecución. Por otro lado, al no permitir la implementación de componentes fuerza a la organización al uso de *web services* para todos los nodos intervinientes. Esto puede provocar que la organización caiga en una especie de síndrome por el uso de servicios, con lo cual se tiende a perder el control sobre el inventario de servicios disponibles. Por otro lado, si bien el proyecto de Intalio es *open source*, hay componentes que son comerciales dentro de la misma iniciativa. Esto puede generar una serie de costos paralelos para la organización que se decida por esta herramienta ante problemas que resultan insolubles bajo la vista de la edición *community*, tal como la implementación de un ESB.

En ambos casos debemos considerar que si bien hemos alcanzado funcionalidad similar a la propia de IBM Websphere BPM por medio de los casos de uso propuestos como ejemplo, en ninguno de los casos lo hemos podido alcanzar por el mismo camino que el propuesto para IBM. Esto muestra que las herramientas *open source* consideradas (las cuales a nuestro entender son las más desarrolladas en el mercado en su clase) no están en el mismo nivel de evolución que la herramienta de IBM. Para poder estarlo deberíamos tener, en primer lugar un servidor funcionalmente equivalente, y en segundo lugar mecanismos que nos permitan introducir al

proceso en un ESB empresarial, siempre con los estándares como guías de desarrollo.

Por el momento, entonces, IBM Websphere BPM sigue siendo funcionalmente más potente que sus “equivalentes” en *open source*.

Líneas posibles de investigación futura

Para lograr finalmente la equivalencia entre la herramienta IBM Websphere BPM y las herramientas *open source* es necesario plantear una serie de líneas para desarrollos futuros, a saber:

- En el caso de jBPM, el problema más importante para alcanzar la compatibilidad es el uso de jPDL como lenguaje, tanto para la notación como para la ejecución del proceso. En cuanto a la notación, la gravedad no es muy alta porque existen algunas similitudes con respecto a BPMN. Quizás la manera de solucionar este escollo sería seguir conservando la semántica de orientación a estados propia de jPDL, pero a su vez incorporar la clasificación de tareas, eventos y compuertas que hace BPMN. Esto permitiría seguir conservando la funcionalidad actual y enriquecer aún más el modelo rotacional.
- Por otro lado, para solucionar la estandarización del modelo de ejecución de jBPM sería necesario profundizar en la integración con el proyecto existente como jBPM BPEL. Recordemos que en la actualidad no existen posibilidades de convertir los modelos jPDL en modelos de ejecución BPEL en forma directa. Si efectivamente se incorporaran los componentes BPMN a jPDL entonces sería factible realizar una conversión a BPEL tal como lo hacen herramientas basadas en XPD, aunque habría que profundizar aún en la pérdida de información al pasar de un modelo a otro. De esta manera lograríamos un modo de conversión de los modelos a un formato ejecutable.
- Para el caso de Intalio, la limitación más importante se da en el hecho de no poder especificar mediante la misma herramienta los manejadores de tareas. En este caso, a nuestro entender habría que explorar más aún en la integración de dicho modelador con la IDE Eclipse, tal como lo hizo el diseñador de jBPM. Esto permitiría mayor flexibilidad a la hora de implantar un modelo dentro de la organización.
- Como ítem final, es necesario establecer un modelo para realizar ingeniería inversa de procesos. Este aspecto no está cubierto aún siquiera por la herramienta de IBM. Sería deseable poder recorrer el ciclo de vida en forma inversa para asegurar la coherencia entre los procesos ejecutables y las versiones documentales de los mismos.

Referencias

- [1] Mathias Waske – “BUSINESS PROCESS MANAGEMENT - Concepts, Languages, architectures”. Springer-Verlag Berlin Heidelberg 2007.
- [2] Judith Hurwitz, Robin Bloor, Carol Baroudi, Marcia Kaufman – “Service Oriented Architecture for dummies”. Wiley Publishing Inc 2007.
- [3] IBM Corporation – “An IBM Proof of Technology. Discovering the value of Websphere BPM for your organization”. IBM Corporation 2008.
- [4] OMG – “BPMN Final Adopted Specification”. Febrero 2006.
- [5] Material didáctico del curso “SOA-BPM” organizado por SADIO. Noviembre 2008.
- [6] Documentación del producto jBPM www.jboss.com/products/jbpm/
- [7] Documentación del producto Intalio www.intalio.com/products/cloud/saas/bpm/
- [8] Tom Baeyens – “7 Forms of Business Process Management with JBoss jBPM” - <http://community.jboss.org/servlet/JiveServlet/download/12802-30-5866/seven.forms.of.bpm.pdf>
- [9] jBPM Process Definition Language (JPDL) <http://docs.jboss.org/jbpm/v3/userguide/jpdl.html>
- [10] jBPM User Guide http://docs.jboss.org/jbpm/v4/userguide/html_single/
- [11] Process Developments: BPEL compared to jPDL. <http://processdevelopments.blogspot.com/2007/04/bpel-compared-to-jpdl.html>
- [12] Spring integration in jBPM 4 http://n.tomek.bujok.info/index.php/Spring_integration_in_jBPM_4
- [13] Documentación de Hibernate y HSQL <http://hsqldb.sourceforge.net/web/hsqFAQ.html>
- [14] Creating custom GUIs for jBPM <http://community.jboss.org/wiki/SOA-PCreatingCustomGUIsforjBPM>
- [15] Combining jBPM and JSF <http://koentsje.blogspot.com/>
- [16] The BPMN-XPDL-BPEL value chain <http://kswenson.wordpress.com/2006/05/26/bpmn-xpdl-and-bpel/>
- [17] IBM WebSphere Developer Technical Journal: SSL, certificate, and key management enhancements for even stronger security in WebSphere Application Server V6.1

http://www.ibm.com/developerworks/websphere/techjournal/0612_birk/0612_birk.html

[18] ¿BPEL o ESB? <http://blog.continuum.cl/archives/47>

[19] Connecting to WebSphere ESB and WebSphere Process Server
<http://www.websphereusergroup.org.uk/uploadedfiles/ConnectingAppsToESB.pdf>

[20] Transactionally integrate Web services with BPEL processes in WebSphere Process Server
http://www.ibm.com/developerworks/websphere/library/techarticles/0703_xu/0703_xu.html

[21] N. Patel, V. Hlupic. *“Dynamic Business Process Modeling (BPM) for Business Process Change”*. Octubre 2003.

[22] S.A.White. “Introduction to BMPN”. IBM Corporation. Mayo 2004.

[23] S.A.White. “Using BMPN to Model a BPEL Process”. IBM Corporation. Marzo 2005.

[24] S.A.White. “Process Modeling Notations and Workflow Patterns”. IBM Corporation. 2005.

[25] M. Owen, J. Raj. “BMPN and Business Process Management. Introduction to the New Business Process Modeling Standard”. Popkin Software. 2003

Anexo 1 – BPMN y los procesos de negocio

Introducción a BPMN

BPMN es un diagrama llamado *Business Process Diagram* (BPD) que fue creado para facilitar el uso y la comprensión de los procesos de negocio, pero también para modelizar casos complejos. También ha sido diseñado con los *web services* en mente.

BPMN es una de las tres notaciones que desarrollo BPMI (*Business Process Management Initiative*). Las otras dos son BPML (*Business Process Management Language*) y BPQL (*Business Process Query Language*).

Todos fueron desarrollados con sólidas fundamentaciones matemáticas que permiten que un BPD pueda llevarse directamente a BPML, de la misma manera que un modelo físico de base de datos mapea directamente a un DDL. BPMN provee varias ventajas para modelizar procesos respecto de UML. Por un lado ofrece una técnica de modelización de flujo de procesos más conducente a la manera en que lo hacen los analistas de negocio. Además, su sólida fundamentación matemática permite mapear directamente a un lenguaje ejecutable.

Objetivos

BPMN es el nuevo estándar para modelizar flujos de procesos de negocio y *web services*. Sus principales objetivos son:

- proveer una notación que puedan comprender todos los usuarios del negocio. Esto incluye tanto a los analistas del negocio que tienen el requerimiento como a los técnicos que lo implementan.
- Asegurar que los lenguajes XML diseñados para la ejecución de procesos de negocio, tales como BPEL4WS y BPML, puedan expresarse visualmente con una notación común.

BPMN “habilita” a BPM

Dijo Winston Churchill “*Para mejorar hay que cambiar, para ser perfecto hay que cambiar con frecuencia*”.

El BPM tiene que ver con manejar el cambio para mejorar los procesos de negocio. BPM unifica distintas disciplinas anteriores como modelización de procesos, simulaciones, *workflow*, EAI (*Enterprise Application Integration*) y B2B (*Business To Business*), en un estándar.

Para establecer estos estándares se creó BPMI.ORG. Los estándares que se definen tienen que ver con el diseño, implementación, ejecución, mantenimiento y optimización de procesos.

Estos estándares son rigurosos en cuanto a su fundamentación matemática. Son análogos al fundamento matemático de la teoría relacional que apuntala los RDBMS.

BPMI.ORG trabaja con OASIS para los estándares de *e-business*.

OASIS (www.oasis-open.org) es un consorcio global sin fines de lucro que conduce el desarrollo, convergencia y adopción de estándares *e-business* (seguridad, *web services*, transacciones comerciales, publicidad electrónica e interoperabilidad)

Tanto BPML de BMPI.ORG como BPLE4WS (de Microsoft, IBM y otros) han adherido a OASIS. El resultado de este comité se denomina actualmente *Web Services- Business Process Execution Language (WS-BPEL)*







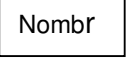
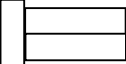

BPMN modeliza los Servicios Web



Poner en funcionamiento un Servicio Web es una tarea de 4 etapas:

1. Diseñar usando BPMN
2. Simular el proceso y modificarlo para lograr eficiencia
3. Poner el servicio a disposición publicándolo usando un lenguaje de ejecución de procesos de negocio
4. Orquestar el servicio web en un flujo de negocio, ensamblándolo y coordinando su comportamiento. (BPMS *Business Process Management System*)

BPMN y BPD

BPMN se utiliza para especificar un diagrama llamado BPD. Un BPD se compone de elementos agrupados en cuatro categorías (2):

Categoría	Elemento	Descripción	Grafica
Flujos	Evento	Es algo que sucede durante el curso del proceso de negocio. Afectan al flujo del proceso. Normalmente tienen una causa (disparador) o un impacto (resultado). Dependiendo de cuando afectan al flujo serán eventos iniciales, intermedios o finales.	
	Actividad	Es un término genérico para el trabajo que realiza una compañía. Puede ser atómica (tarea) o compuesta (sub-proceso). Para indicar la no atomicidad se coloca un signo + en la esquina del símbolo de actividad.	
	Gateway	Se utiliza para controlar la convergencia o divergencia de flujos. Representa una decisión para mezclar o unir caminos.	
Conexiones	Flujo de secuencia	Se utiliza para mostrar el orden o secuencia en que las actividades se realizan en un proceso	
	Flujo de mensajes	Se utiliza para mostrar el flujo de mensajes entre dos participantes separados.	
	Asociación	Se utiliza para mostrar entradas y salidas de actividades.	
Swimlanes	Pool (fondo común)	Representa un participante en un proceso. Actúa como contenedor gráfico para particionar un conjunto de actividades.	
	Lane (sendero)	Es una sub-partición dentro de un pool y puede extenderse a todo lo largo o ancho del pool. Se utilizan para organizar y categorizar actividades.	
Artefactos	Objetos de datos	Mecanismo para mostrar como los datos son requeridos y producidos por las	

		actividades. Se conectan a las actividades por asociaciones.	Nombre (estado)
	Grupos	Se utiliza para documentación o para propósitos de análisis, pero no afecta al Flujo de Secuencias	
	Anotaciones	Mecanismo para que quien esta modelizando provea información adicional para el lector del diagrama.	

Diagramas BPM

La notación BPM permite construir diagramas fáciles de leer y que además manejen la complejidad traduciendo el diagrama en algún lenguaje de ejecución.

Para modelizar un flujo **sólo se modelizan los eventos** que ocurren. Las decisiones entre flujos se modelizan con *gateways*.

Un proceso en el flujo puede contener subprocesos que cuando son atómicos se denominan tareas.

Además, se puede especificar “quien hace que”, ubicando los procesos en *pools* que denotan quien hace la tarea pudiendo particionar el *pool* en *lanes* o senderos. Típicamente un pool representa a toda la organización mientras que el *lane* representa un departamento dentro de la misma. Puede extrapolarse esto mismo a funciones, aplicaciones y sistemas.

Modelización por eventos

La modelización esta centrada en flujos y eventos, pero principalmente en eventos, que son los elementos disparadores de determinadas situaciones.



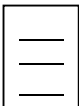
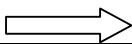





Los eventos se dividen en iniciales, intermedios y finales según se desencadene al inicio, durante o al finalizar el flujo del proceso.

Para mejorar el poder expresivo, a estos tres tipos de eventos básicos se los divide a su vez en eventos más complejos en función de la tarea que realizan. Esta sub-clasificación agrega restricciones al modelo. Ejemplo: un evento “*timer*” nunca finaliza un flujo.

Los distintos sub-tipos de eventos son:

- **Mensajes:** arriba de un participante y dispara el inicio de un proceso o continúa otro en el caso de ser un evento intermedio. El mensaje de fin denota el mensaje generado al finalizar el proceso.
- **Timer:** una hora o ciclo puede establecer para el disparar el inicio de un proceso o continuarlo si es un evento intermedio. No puede ser un evento de finalización.
- **Reglas:** se dispara cuando la condición de una regla se cumple al inicio o durante un proceso. No puede ser un evento de finalización
- **Enlace:** es un mecanismo para conectar el evento final de un proceso con el inicial de otro.

- **Múltiple:** un evento múltiple indica que hay varias maneras de iniciar un proceso o continuarlo en el caso de un evento intermedio. Solamente una de esas maneras será requerida. Los atributos del evento definen cual se aplica. Para la finalización de un evento múltiple hay múltiples consecuencias del proceso, las cuales ocurrirán TODAS. (Ejemplo, se enviaran múltiples mensajes).
- **Excepción:** un evento de excepción final informa a la maquina de ejecución que el proceso finalizo anormalmente o también puede ser captado por un evento intermedio. Un evento de excepción nunca inicia un proceso.
- **Compensación:** un evento de compensación final informa a la máquina de ejecución que el proceso debe ser compensado. Se utiliza como evento intermedio cuando el proceso requiere un roll-back..
- **Cancelación:** un evento de cancelación final indica que el usuario cancelo el proceso.
- **Kill:** un evento “Kill” final indica que un error fatal y que todas las actividades dentro del proceso deben ser finalizadas. El proceso es finalizado con compensación o con manejo de errores.

Tipo de Evento	Grafico	Inicio	Intermedio	Fin
Mensaje		SI	SI	SI
Timer		SI	SI	NO
Regla		SI	SI	NO
Enlace		SI	SI	SI
Múltiple		SI	SI	SI
Excepción		NO	SI	SI
Compensación		NO	SI	SI
Cancelación		NO	NO	SI
“kill”		NO	NO	SI

Ejemplo: el proceso Check Inbox genera un evento mensaje que envía un mensaje a otro proceso..



Procesos, sub-proceso y tareas

El corazón de la modelización de procesos de negocio son los procesos mismos. Estos a su vez son de tres tipos: procesos, sub-procesos y tareas, como tres niveles de abstracción diferentes pero comunicados entre si.

Un **proceso** es una red de “cosas para hacer”. Los procesos que se dibujan en diagramas hijos se denominan **sub-procesos**. El proceso de menor nivel y que no puede ser descompuesto, se considera una **tarea**.

Modelar el flujo de secuencia de un proceso

Los flujos de secuencia muestran el orden de ejecución de los procesos al conectarlos. Muestra la secuencia de procesos **en** una organización o departamento.

Los flujos de mensajes modelizan el ordenamiento de los procesos **entre** organizaciones y departamentos.

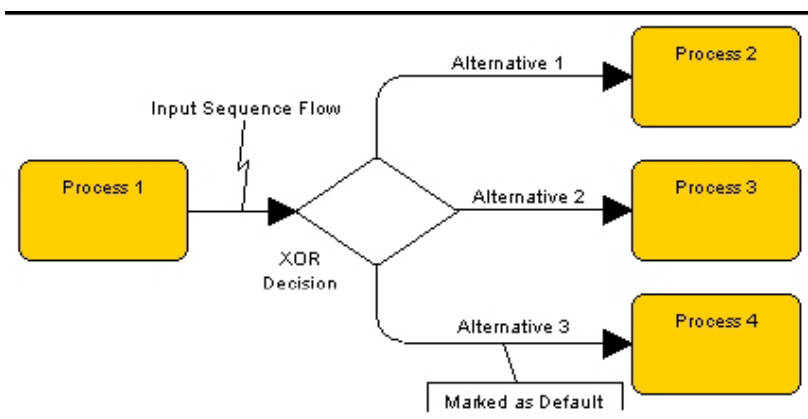
Modelar los puntos de decisión con Gateways

Un *gateway* es una pregunta a ser formulada en algún punto del flujo del proceso. La pregunta posee un conjunto de respuestas definidas con son el efecto de las “puertas”.

Los distintos tipos de *gateway* son:

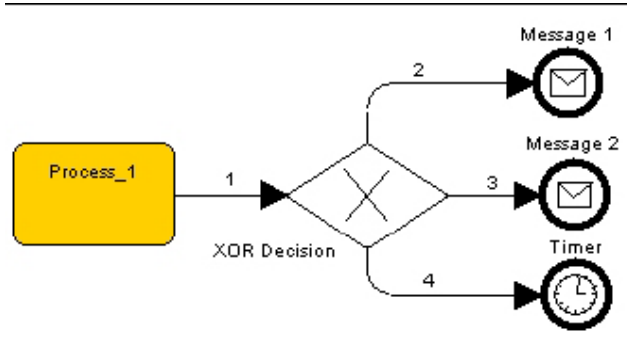
- **Decisión exclusiva XOR basada en datos**

Son los *gateway* más comunes. El dato atraviesa el flujo del proceso y alcanza al *gateway* siguiendo el camino correspondiente según la condición que satisfaga.



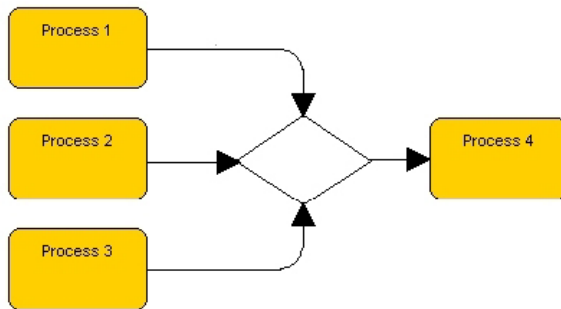
- **Decisión exclusiva XOR basada en eventos**

Representa una ramificación donde las alternativas están basadas en un evento que ocurre en algún punto del proceso. Un evento específico, usualmente el receptor del mensaje, decide el camino a tomar.



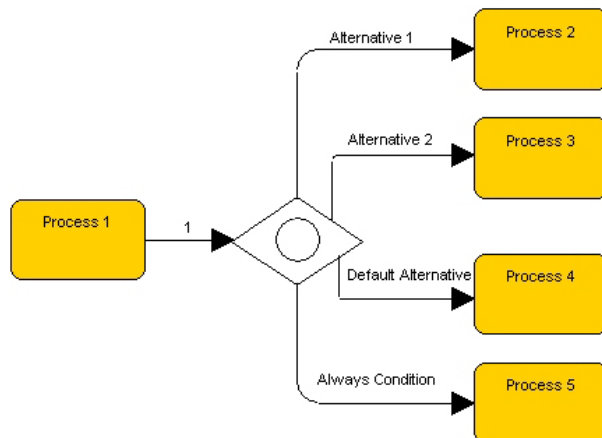
- **Merge exclusivo XOR**

Representan un *merge* basado en datos y en eventos. Lo exclusivo significa que una sola de las entradas se elegirá para la salida.



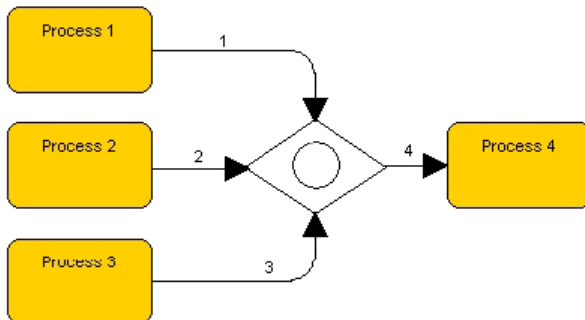
- **OR inclusivo**

Indica que uno o más flujos de secuencias de una decisión pueden tomarse. No puede haber salidas vacías, por lo tanto deben tomarse valores por defecto.



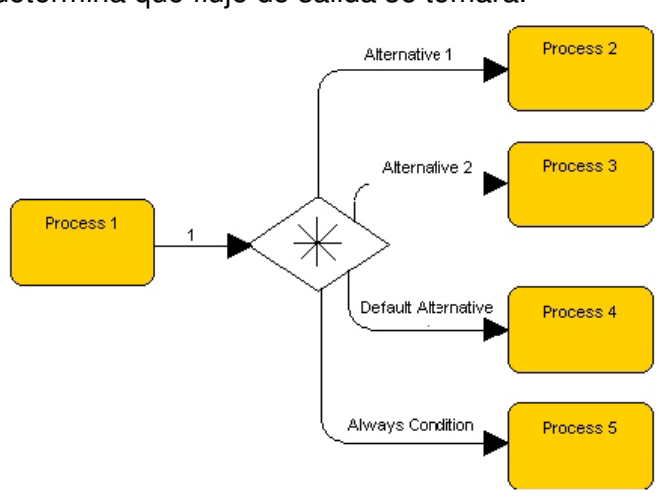
- **Merge OR inclusivo**

Indica que el flujo del proceso continua cuando la primer salida desde cualquiera de los conjuntos de entrada de los flujos de secuencia. Si otras señales llegan mas tarde, no son usadas.



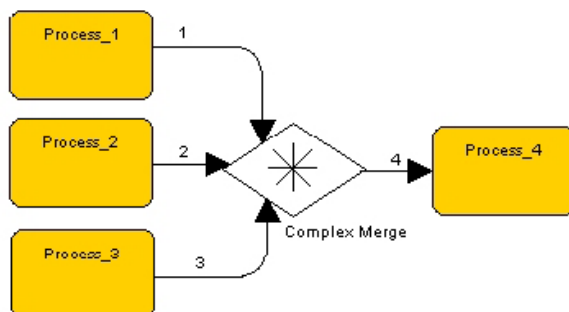
- **Decisión compleja**

Representa nombres de flujos de secuencia en las salidas. La expresión determina que flujo de salida se tomara.



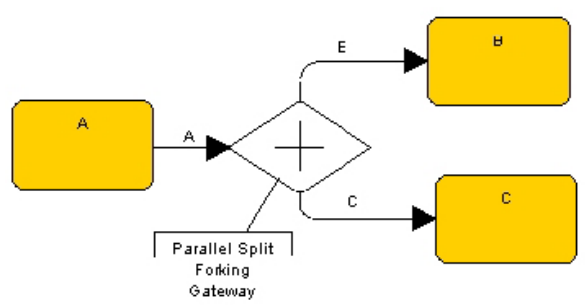
- **Merge Complejo**

Representa nombres de flujos de secuencia en las salidas y/o procesos e datos que ingresan al *gateway*. La expresión cuando se detiene la tarea.



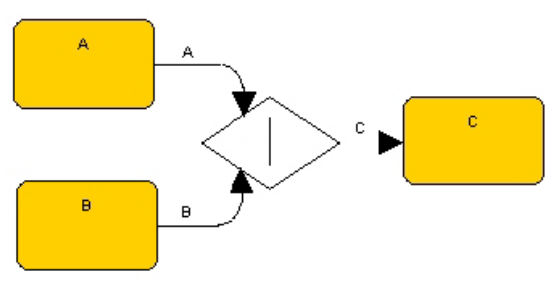
- **Bifurcación paralela**

También llamado *gateway* AND, indica que todos los flujos de secuencia que salen de la secuencia son tomados.



- **Unión paralela**

El *gateway* puede recibir una entrada de TODOS los flujos de secuencia para la salida tomada. El flujo de proceso espera que todas las señales lleguen antes de continuar.



Quien hace que – Pool y Lanes

Los *pool* y *lanes* permiten ordenar los actores y las actividades. Ubicando los procesos dentro de ellos se especifica **quien** hace **que** cosa. Los eventos especifican **cuando** ocurren y los *gateways* determinan **dónde** se toman las decisiones o **quien** las toma.

El *pool* puede considerarse un repositorio común de recursos. Hay ocasiones en que los procesos necesitan saltar a otro *pool* porque necesitan otros recursos para completar su actividad.

Esto es apto para describir procesos B2B. Así como los flujos de secuencia se utilizan para comunicar eventos, procesos y *gateways* dentro de un mismo *pool*, los flujos de mensajes se utilizan para comunicar eventos, procesos y *gateways* entre distintos *pools*.

Un *pool* puede representar varias cosas además de una organización, como por ejemplo:

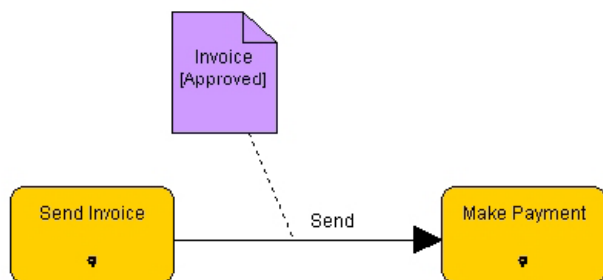
- una **función**: algo que una organización lleva a cabo como venta, capacitación, marketing.
- una **ubicación**: una ubicación física dentro de la compañía
- una **clase**: un módulo de software en un ambiente orientado a objetos
- una **entidad**: una representación conceptual de datos dentro de la base de datos.

El pool representa una única cosa pero lo que representa es de carácter heterogéneo.

Transformación de datos

Los procesos transforman datos en la organización. Se puede modelar la manera en que los datos son transformados durante un flujo de proceso con **objetos de datos**. Los datos se representan como un conjunto de entidades o clases.

Los objetos de datos se vinculan a los flujos de secuencia o de mensajes con líneas punteadas. Pudiendo especificar el estado del objeto de datos entre corchetes.



Lenguajes de ejecución de negocios

Los lenguajes que modelizan procesos de negocio son meta-lenguajes basados en XML. La nueva raza de estos lenguajes incluye un lenguaje de ejecución de procesos de negocio para servicios web (BPEL4WS) creado por BEA, IBM y Microsoft y un lenguaje de modelización de procesos de negocio BMPL, creado por BPMI.ORG.

BPMN mapea directamente a los lenguajes de ejecución por su fuerte fundamentación matemática.

BPMS orquestan los servicios web

Internet es un ambiente heterogéneo de plataformas y aplicaciones que deben trabajar en forma armoniosa. Esta es una de las fuerzas conductoras de la estandarización de servicios web.

Poner en funcionamiento los servicios web involucra al menos las etapas de:

1. Diseñar procesos con BPMN
2. Verificar su eficiencia con simulación
3. Ponerlos disponibles publicándolos
4. Orquestarlos y coordinarlos usando un BPMS

BPMS (*Business Process Management System*) ofrece la posibilidad de transformar disciplinas de Workflow, EAI y B2B en soluciones abiertas, accesibles a la masa de desarrolladores, ágiles y bajo acopladas.

Entre las compañías que desarrollan BPMS se encuentran IBM, BEA Systems, Vitria, Intalio, FileNet, Fuego, Collaxa.

Integración de BPMN con UML

El advenimiento de BPMN, BMPL y BPMS no volvió obsoletas las necesidades de desarrollo de sofá.

El UML es un lenguaje que ayuda a los desarrolladores a especificar, visualizar y documentar modelos de sistemas de software.

UML define un conjunto de diagramas que pueden agruparse en tres categorías:

1. Estructura estática
2. Comportamiento dinámico
3. Gestión y organización de las soluciones de software.

Los diagramas que reflejan un comportamiento dinámico como los casos de uso y los diagramas de actividad, se utilizan para modelizar procesos.

En este sentido BPMN y UML se asemejan en cuanto a que ambos tienen notaciones para los procesos de negocio.

La diferencia fundamental radica en que UML sigue un paradigma orientado a objetos para modelar aplicaciones, mientras que BPMN se centra en los procesos.

Con BPMN los flujos de control y flujos de mensaje se modelizan primero. El modelo de objetos para dicho proceso se modeliza de una manera mas implícita que explícita.

UML carece de una visión de implementación de los modelos. Los desarrolladores modelizan parte de sus aplicaciones con UML. UML no define un meta-modelo de ejecución.

En contraste, BPMN define un tipo simple de diagrama que tiene múltiples visiones derivadas de la ejecución subyacente del meta-modelo.

BPMN puede usarse para conducir soluciones que se ejecuten directamente en un BPMS o usarse como front-end del análisis de negocio para el posterior desarrollo de sistemas utilizando UML. En este escenario los usuarios UML ven a los procesos de negocio simplemente como otro tipo de componente.

Modelizar con BPMN es esencial para comprender y comunicar los procesos de negocio a través de las organizaciones.

La ejecución de procesos de negocio es una alternativa a los paradigmas de desarrollo tradicionales, los cuales, de hecho, no desaparecen.

Anexo 2 – jPDL

jPDL (*jBPM Process Definition Language*) es el lenguaje utilizado por jBPM para la definición formal de los procesos de negocio.

Es un lenguaje de tipo GOP (*Graph Oriented Programming*) al igual que BPMN. La fundamental diferencia con el anterior surge en el hecho de introducir la semántica de máquina de estados. Esto es, el proceso en vez de ser visto simplemente como una sucesión de actividades es considerado como un conjunto de estados en los que los flujos definen las transiciones posibles entre los mismos.

jPDL especifica un *xml schema* y el mecanismo para empaquetar todas las definiciones relacionadas con el proceso en un solo archivo.

Validación

En el *xml schema* se especifican dos condiciones:

- La versión de jpdL que se utilizará para diagramar el proceso (esto especificará la versión del *runtime* utilizada posteriormente para compilar el proceso.)
- El *parser xerces* debe ubicarse en el classpath especificado en el *schema*.

Definición del lenguaje

A continuación haremos una descripción detallada de cada uno de los elementos que conforman jPDL.

Definición del proceso

Nombre	Tipo	Multiplicidad	Descripción
Name	Atributo	Opcional	Es el nombre del proceso
Swimlane	Elemento	[0..*]	Son las áreas usadas en el proceso. Representan roles y sirven para asignar responsabilidades
start-state	Elemento	[0..1]	Es el estado inicial del proceso. Un proceso sin estado inicial no puede ser ejecutado.
{end-state state node task-node process-state super-state fork join decision}	Elemento	[0..*]	Son los nodos en la definición del proceso. Un proceso sin nodos es válido pero no puede ser ejecutado.
event	Elemento	[0..*]	Son los eventos del

			proceso que sirven como contenedores de acciones.
{action script create-timer cancel-timer}	Elemento	[0..*]	Las acciones definidas en forma global pueden ser referenciadas desde eventos y transiciones. Estas acciones deben especificar un nombre para poder ser referenciadas.
task	Elemento	[0..*]	Las tareas definidas en forma global pueden ser utilizadas en las acciones.
exception-handler	Elemento	[0..*]	Sirven para manejar condiciones de error generadas durante la ejecución del proceso.

Nodos

Nombre	Tipo	Multiplicidad	Descripción
{action script create-timer cancel-timer}	Elemento	1	Una acción que representa el comportamiento del nodo
common node elements			(Ver Common node elements)

Common node elements (elementos comunes de nodo)

Nombre	Tipo	Multiplicidad	Descripción
name	Atributo	Requerido	Nombre del nodo
async	Atributo	{ true false }, false es el default	Si está seteado en true, el nodo se ejecutará en forma asincrónica.
transition	Elemento	[0..*]	Son las transiciones salientes. Cada transición que abandona el nodo debe tener un nombre diferente. La primera transición especificada es la transición default.
event	Elemento	[0..*]	Los tipos de evento soportadas: {node-enter node-leave}
exception-handler	Elemento	[0..*]	Sirven para manejar condiciones de error generadas durante la

			ejecución del proceso.
timer	Elemento	[0..*]	Sirve para cronometrar la duración en la ejecución de un nodo.

End-state (Estado final)

Nombre	Tipo	Multiplicidad	Descripción
name	Atributo	requerido	Nombre del estado final
event	Elemento	[0..*]	Tipo de evento soportado: {node-enter}
exception-handler	Elemento	[0..*]	Sirven para manejar condiciones de error generadas durante la ejecución del proceso.

Estado

Nombre	Tipo	Multiplicidad	Descripción
common node elements			Ver common node elements

Task-node (Nodo de tarea)

Nombre	Tipo	Multiplicidad	Descripción
signal	Atributo	Opcional	{unsynchronized never first first-wait last last-wait}, el default es last. Sigal especifica el efecto de completar una tarea en la ejecución del proceso.
create-tasks	Atributo	Opcional	{yes no true false}, el default es true.
end-tasks	Atributo	Opcional	{yes no true false}, el default es false.
task	Elemento	[0..*]	La tarea que debería ser creada cuando el flujo de ejecución arriba al nodo.
common node elements			Ver common node elements

Process State (Estado de proceso)

Nombre	Tipo	Multiplicidad	Descripción
sub-process	Elemento	1	Es el subproceso asociado al nodo
variable	Elemento	[0..*]	Especifica como se deben copiar los

			datos cuando desde el súper proceso se invoca al subproceso.
common node elements			Ver common node elements

Super-State (Súper estado)

Nombre	Tipo	Multiplicidad	Descripción
{end-state state node task-node process-state super-state fork join decision}	Elemento	[0..*]	Son los nodos del súper estado. Los súper estados puede estar anidados.
common node elements			Ver common node elements

Fork (División)

Nombre	Tipo	Multiplicidad	Descripción
common node elements			Ver common node elements

Join (Unión)

Nombre	Tipo	Multiplicidad	Descripción
common node elements			Ver common node elements

Decision (Decisión)

Nombre	Tipo	Multiplicidad	Descripción
handler	Elemento	Es opcional pero debería estar definido.	Es la descripción de la clase org.jbpm.jpdl.Def.DecisionHandler
transition conditions	Atributo o elemento de texto		Son las transiciones que abandonan la condición.
common node elements			Ver common node elements

Transición

Nombre	Tipo	Multiplicidad	Descripción
name	Atributo	Opcional	Es el nombre de la transición. Todas las transiciones deberían tener un nombre distinto.
To	Atributo	Requerido	Es el nombre jerárquico del nodo de destino
Condition	Atributo o elemento de texto	Opcional	Es la condición de guarda.
{action script create-	Elemento	[0..*]	Son las acciones a

timer cancel-timer}			ser ejecutadas al abordar a la transición.
exception-handler	Elemento	[0..*]	Sirven para manejar condiciones de error generadas durante la ejecución del proceso.

Action (acción)

Nombre	Tipo	Multiplicidad	Descripción
name	Atributo	Opcional	Es el nombre de la acción
Class	Atributo	Opcional	Es la clase que implementa la interface <code>org.jbpm.graph.def.ActionHandler</code>
Ref-name	Atributo	Opcional	Nombre de la acción referenciada
expression	Atributo	Opcional	Una expresión jPDL que resuelve a un método.
accept-propagated-events	Atributo	Opcional	{yes no true false}. El default es yes true
config-type	Atributo	Opcional	{field bean constructor configuration-property}. Especifica como la acción debería ser construida.
async	Atributo	{true false}	El default es false, lo que significa que la acción se ejecuta en el tren de ejecución sincrónico.
	{contenido}	Opcional	El contenido puede ser utilizado como información de configuración.

Tarea

Nombre	Tipo	Multiplicidad	Descripción
name	Atributo	Opcional	Es el nombre de la tarea
blocking	Atributo	Opcional	{yes no true false}, el default es false
signalling	Atributo	Opcional	{yes no true false}, el default es true.
duedate	Atributo	Opcional	Es la duración de la tarea expresada en horas.
swimlane	Atributo	Opcional	Referencia a un swimlane definido.
priority	Atributo	Opcional	Una de {highest, high, normal, low, lowest}.
assignment	Elemento	Opcional	Describe una delegación que asignará la tarea a un actor,
event	Elemento	[0..*]	Los tipos de eventos soportados son {task-create task-start task-assign task-end}
exception-handler	Elemento	[0..*]	Sirven para manejar condiciones de error

			generadas durante la ejecución del proceso.
timer	Elemento	[0..*]	Sirve para cronometrar la duración en la ejecución de un nodo.
controller	Elemento	[0..1]	Especifica como las variables del proceso se transforman en parámetros del formulario de tarea.