BÚSQUEDA SEMÁNTICA EN UN REPOSITORIO DE PROCESOS DE NEGOCIO

DANIEL FELIPE RIVAS BURBANO
DAVID SANTIAGO CORCHUELO CASTRO

ANEXO No 1.

ESTUDIO Y SELECCIÓN DE LOS LENGUAJES PARA EL MODELADO E IMPLEMENTACIÓN DE LOS PROCESOS DE NEGOCIO

Director: Dr. JUAN CARLOS CORRALES

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

DEPARTAMENTO DE TELEMÁTICA

POPAYÁN

2010

ANEXO No 1. ESTUDIO Y SELECCIÓN DE LOS LENGUAJES PARA EL MODELADO E IMPLEMENTACIÓN DE LOS PROCESOS DE NEGOCIO

El presente anexo reúne un estudio de los principales lenguajes utilizados para el modelado e implementación de los procesos de negocio que son comúnmente usados a nivel académico y empresarial. A partir de este estudio se hace la selección de un lenguaje que satisfaga los requerimientos de adición semántica a los flujos de control de los procesos de negocio.

1. Notaciones de Procesos de Negocio

1.1 Introducción

BPM es una forma estructurada, coherente y consistente de entender, documentar, modelar, analizar, simular, ejecutar e intercambiar continuamente de extremo a extremo procesos de negocio y todos los recursos involucrados, con el propósito de contribuir al mejoramiento de los negocios [1]. BPM quiere identificar los procesos de negocio principales dentro de una organización, automatizarlos con una integración eficiente hacia los sistemas y las personas y así maximizar la eficacia de los procesos.

BPM se refiere al diseño y ejecución de los procesos de negocio, también llamado Modelado de Procesos de Negocio. BPM se refiere al diseño y ejecución de procesos de negocio y es a veces llamado Business Process Modeling (Modelado de Procesos de Negocio). El modelado de procesos es la captura de una secuencia ordenada de las actividades de negocio e información de apoyo, un proceso de negocio describe cómo una empresa intenta conseguir sus objetivos [2]. Este documento está enfocado en el análisis de los estándares de modelado de procesos de negocio, ventajas y desventajas con el fin de elegir un lenguaje específico de acuerdo a nuestros propósitos. Existen diferentes niveles de modelado de procesos:

- Mapas de Procesos (diagramas de flujo simples de las actividades)
- Descripciones de Procesos (diagramas de flujo extendidos con información adicional, pero no suficiente para definir completamente el desempeño actual).
- Modelos de Procesos (diagramas de flujo extendidos con suficiente para que el proceso pueda ser analizado, simulado y /o ejecutado).

Consideremos algunas de las notaciones de procesos de negocio más destacadas que permiten el modelado de procesos de negocio.

2. BPMN (Business Process Modeling Notation)

2.1 Visión General

BPMN es una notación para definir Procesos de negocio basado en diagramas de flujo y soporta cada uno de los niveles descritos anteriormente, fue desarrollado por The Business process Management Initiative (BPMI) como una notación estándar para el modelado de Procesos de Negocio (BPMN)[1]. El primer objetivo de los esfuerzos de BPMN fue proveer una

[1] Actualmente, BPMI y OMG han fusionado sus actividades de investigación en Gestión de Procesos de Negocio (BPM) para proveer estándares para esta creciente industria. El grupo combinado se ha denominado así mismo: Business Modeling & Integration (BMI) Domain Task Force (DTF). Disponible en internet: http://bmi.omg.org/

notación que sea fácilmente entendible por todos los usuarios de negocio, desde los analistas de negocio quienes crean los borradores iniciales de los procesos, hacia la responsabilidad técnica de los desarrolladores para desarrollar la tecnología que realizarán estos procesos, y finalmente, hacia las personas de negocio quienes van a gestionar, monitorear estos procesos. Así, BPMN crea un puente estandarizado para la brecha existente entre el diseño de procesos de negocio y la implementación de procesos (Figura 1).

BPMN fue desarrollado para realizar un simple mecanismo de creación de procesos, y al mismo tiempo posibilitar el manejo de la complejidad inherente a los procesos de negocio. Por lo tanto, para manejar estos dos requerimientos conflictivos fue necesario organizar los aspectos gráficos de notación en categorías específicas. Las cuatro categorías de elementos son:

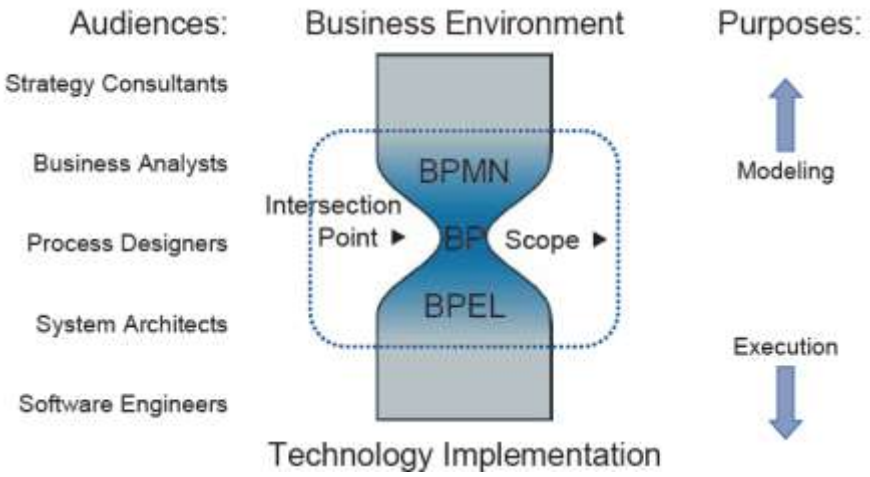


Figura 1. El reloj de arena de BPM

- **Flow Objects** Están divididos en Events, Activities and Gateways.

  Event: Define algo que "pasa" durante el curso del proceso de negocio.
  Activity: Es un termino genérico de trabajo que la compañía desempeña.
  Gateway: Es utilizado para el control de las divergencias o convergencias del flujo de secuencia (Sequence Flow).

- **Connecting Objects** Están divididos en Sequence Flow, Message Flow y Associations.

  Sequence Flow: Es utilizado para mostrar el orden (la secuencia) en que las actividades se ejecutarán dentro del proceso.
  Message Flow: Es utilizada para mostrar el flujo de los mensajes entre dos Participantes de Procesos separados.
  Association: Es utilizada para asociar datos, texto y otros Artefactos con flujos de objetos.

- **Swimlanes** Están divididas en Pool y Lanes.

  Pool: Representa un Participante en un Proceso. Este también actúa como un contenedor gráfico para particionar un conjunto de actividades.
  Lane: Es una sub partición dentro de una Pool y se extenderá a toda la longitud de la misma Pool.

- **Artifacts** Estan divididos en Data objects, Groups y Annotations.

  Data Object: Son un mecanismo para mostrar como los datos son requeridos o producidos por las actividades.
  Group: Puede ser utilizado para propósitos de documentación o análisis, pero no afecta el flujo de secuencia (Sequence Flow).
  Annotation: Provee información de texto adicional para el lector de un diagrama BPMN.

Además, BPMN es una notación rica que simplifica el modelado de procesos de negocio complejos, define un Diagrama de Procesos de Negocio (BPD), el cual se basa en técnicas de mapas de flujo que se ajustan a los modelos gráficos creados de las operaciones de los procesos de negocio. Un modelo de proceso de negocio, es entonces, una red de objetos gráficos, que son actividades, (p.e. trabajo) y el flujo controla que define el orden de ejecución de las actividades (Figura 2). Los analistas de negocio (BA) y desarrolladores construyen diagramas de procesos de negocio (BPDs). Un BPD da a conocer en imágenes lo que BPML y BPEL codifica en XML.

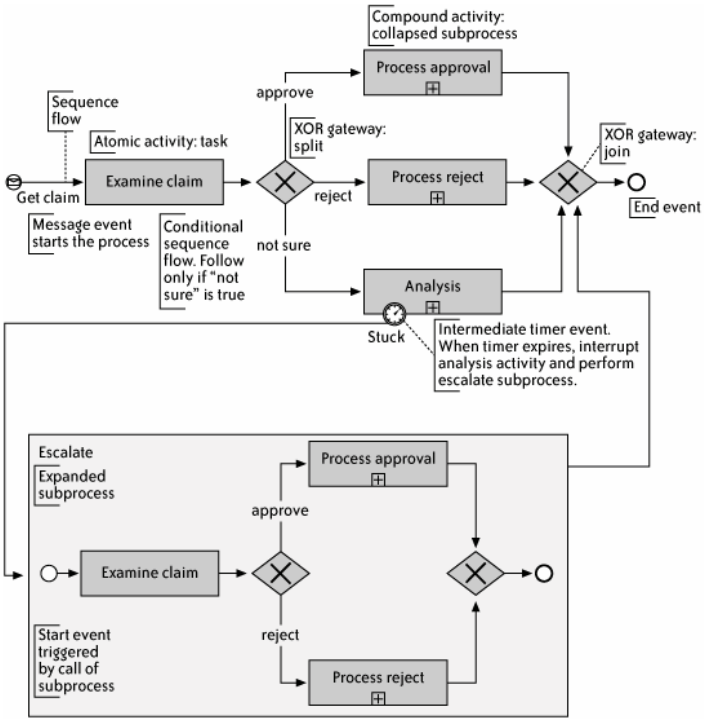


Figura 2: Proceso de reclamo de seguro en BPMN**.**

Descripción (Figura 2):

El proceso recibe un reclamo (*Get claim*), lo examina (*Examine claim*), y entonces los divide en una de tres direcciones, dependiendo de si el reclamo ha sido aprobado (*Process approval*), rechazado (*Process reject*), o dejar para un posterior análisis (*Analysis*). La opción de análisis tiene el mismo limite; si no es ejecutado rápidamente, es abortado (*Stuck*), y un especial proceso de ascenso (*Escalate*, cuyos pasos están incluidos en la caja *Escalate*) se pone en funcionamiento. El proceso de escalamiento se comporta mucho como su padre: este empieza por examinar el reclamo, luego cualquiera de los dos lo aprueba o lo rechaza. No se permiten otros análisis en un escalamiento. El proceso padre se completa cuando sus condiciones path-approval, rejection, analysis, o escalation se completan [4].

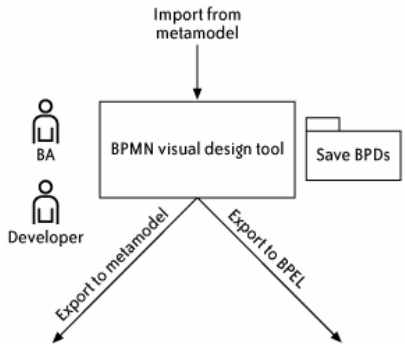


Figura 3: Uso de BPMN.

BPMN está soportado con un modelo interno que permite la generación de BPEL ejecutable [3], también soporta un metamodelo que importa y exporta Modelos de Procesos de Negocio Semánticos (BPSM), permitiendo a la herramienta BPMN el intercambio de procesos que fueron implementados en otras herramientas (Figura 3).

En [5] es propuesto un metamodelo para evaluar los diferentes lenguajes de modelado de procesos (BPMLs) basados en un amplio rango de conceptos. Este metamodelo esta basado en un un marco conceptual de Curtis y otros [6].Es un framework que tiene cuatro perspectivas: organizacional, funcional, comportamiento e información, además, este framework es extendido por otra perspectiva denominada contexto de procesos de negocio que captura información importante como los objetivos de proceso y medidas. Este metamodelo genérico es inspirado por la teoría de procesos de negocio, patrones de workflows y la Workflow Management Coalition (WfMC) [6].

Functional Perspective: Representa los elementos de procesos que están siendo ejecutados. Los elementos básicos de un proceso de negocio son las Actividades (*Activities*).

Organizational Perspective: Representa donde y por quien (que agentes) elementos de procesos están siendo ejecutados.

Behavioral Perspective: Representa cuando los elementos de proceso son ejecutados (p.e. secuencia), así como los aspectos de cómo ellos son ejecutados a través de lazos de

realimentación, iteraciones, toma de decisiones complejas a través de condiciones, criterios de entrada y salida, y así sucesivamente. El flujo de datos (*Data Flow*) conecta actividades atómicas con fuentes de información.

Informational Perspective: Representa las entidades de información producidas o manipuladas por un proceso. Estas entidades incluyen datos, artefactos, productos (intermedios y finales), y objetos.

Business Process Context Perspective: Presenta los procesos de negocio desde una amplia perspectiva [7]. Este provee una visión general del proceso y describe las características principales de los procesos de negocio, así como los objetivos y sus medidas, los entregables, el usuario del proceso, el tipo de proceso y el cliente. A partir de estos conceptos los autores producen ventajas y desventajas acerca de esa evaluación. Algunos de ellos se presentan a continuación en este modelo y en otros modelos.

2.2 Ventajas

- La especificación BPMN define semántica y es sumamente elevado, esto permite que se pueda mapear a múltiples especificaciones de lenguajes de bajo nivel (BPEL, BPML).
- BPMN es gráficamente muy rico en cuanto a flujo de control.
- BPMN es cada vez más utilizado y se asocia directamente a BPEL como estándar de referencia.

2.3 Desventajas

- BPMN es debil en el contexto organizacional.
- No se enfoca en elementos de metamodelos.
- Hay ambigüedad y confusión cuando se comparte modelos BPMN.

3. EPC (Event-driven Process Chains)

3.1 Visión General

Las cadenas de procesos dirigidas por eventos son un lenguaje gráfico de descripción de procesos introducido por Keller, Nüttgens y Scheer en 1992 dentro del framework de la arquitectura del Sistema Integrado de Información (ARIS) para modelar procesos de negocio [7]. El lenguaje apunta a la descripción de procesos en el nivel de su lógica de negocio, no necesariamente en un nivel formal de especificación, y es fácil de entender y utilizar por las personas de negocio. La metodología hereda el nombre de los tipos de diagramas que se muestran en (Figura 4). Así como los diagramas muestran en la estructura del flujo de control de los procesos como una cadena de eventos y funciones, esto es, una cadena de procesos dirigida por eventos [8]. Una cadena de procesos dirigida por eventos tiene los siguientes elementos:

- **Funciones:**
  Los bloques de construcción básicos son funciones. Una función corresponde a una actividad (tarea, etapa de proceso) que necesitan ser ejecutados.

- **Eventos:**
  Un evento describe la situación antes y/o después de que una función es ejecutada. Las funciones son vinculadas por eventos. Un evento podría corresponder a una post condición de una función y actúa como una precondición de otra función.
- **Conectores Lógicos:**
  Los conectores pueden ser utilizados para conectar actividades y eventos. De esta manera el flujo de control es especificado. Hay tres tipos de conectores "^" (AND), XOR (exclusive or) y "v" (OR).

EPC es un modelado orientado a procesos donde el principal enfoque es modelar un sistema en el proceso dentro del sistema. Un proceso consiste de secuencias de eventos que dan inicio a actividades/funciones. Los eventos en sí mismos son el resultado de otras funciones aparte de eventos iniciales que provocan el proceso completo. Por la introducción de operadores lógicos, esta estructura de control dirigida por eventos puede ser expandida a un flujo de control complejo ilustrando decisiones relevantes y potenciales para la cooperación que sucede en el proceso. Este modelado orientado a procesos es la base para que EPC se haya tomado como un estándar para el modelado de procesos de negocio en una empresa. El modelo EPC básico puede ser extendido con componentes semánticos que ilustran los elementos participantes en el proceso, así como objetos de información y unidades de organización [9].

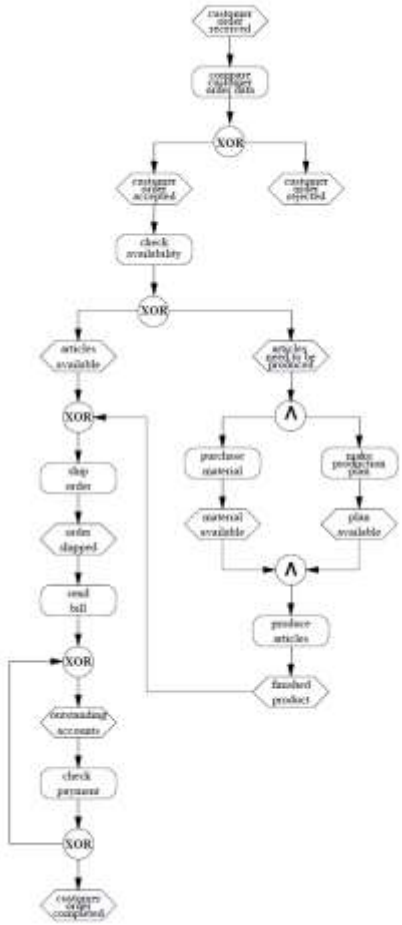


Figura 4: Modelando un proceso de negocio utilizando cadenas de procesos dirigidos por eventos.

Descripcion (Figura 4):

El proceso modelado en la Figura 1 modela el proceso de una orden de compra. El proceso empieza con el evento *customer order received*. Primero los datos de la orden de compra son comprobados y como resultado de esta función la orden es rechazada o aceptada. El conector XOR modela el hecho de que después de la ejecución de la función *compares customer order data* uno de los dos eventos (*customer order accepted* or *customer order rejected*) se mantienen. Si la orden es rechazada, el proceso para. Para cada orden aceptada, la disponibilidad es comprobada. Si los artículos no están disponibles, entonces dos funciones son ejecutadas en paralelo: *purchase material* y *make production plan*. Despues de ejecutar ambas funciones, los artículos son producidos. Si cualquiera de los eventos *articles available* ó *finished product* se mantiene, entonces la función *ship order* puede ser ejecutada. Después de que la orden ha sido enviada, el pago es realizado. Después de que el dinero es enviado, se comprueba si el pago ha sido hecho (función *check payment*). Si el chequeo tiene un resultado positivo, el proceso es completado (evento *customer order completed*) [8].

3.2 Ventajas

- EPC cubre más aspectos así como una descripción detallada de las unidades de organización de negocio junto con sus respectivas funcione y también información y recursos materiales.
- EPC es relativamente rica en el modelado de dominio y en el modelado organizacional.
- Las perspectivas Funcionales y de Comportamiento con bien representadas en EPC.

3.3 Desventajas

- Ni la sintaxis ni la semántica de un EPCesta bien definida. Es necesario realizar un mapeo de EPCs (sin tener en cuenta los conectores tipo ”v”) en redes de Petri.
- Las perspectivas Organizacional y de Información están parcialmente soportadas en EPC.

4. Petri Nets

4.1 Visión General

Las redes de Petri se han vuelto en una herramienta popular para el modelado y análisis de los sistemas concurrentes. Hay muchos factores que contribuyen a su éxito: naturaleza gráfica, habilidad para modelar procesos paralelos y distribuidos en una manera natural, la simplicidad de modelo y las firmes bases matemáticas sobre las que se fundamentan. Sin embargo, los modelos clásicos de redes de Petri no son adecuadas para modelar varios sistemas encontrados en logística, producción, comunicaciones, manufactura flexible y procesamiento de información. Además, las redes de Petri tienen una semántica formal y proveen unas abundantes técnicas de análisis y descripción de los sistemas en tiempo real que tienden a ser complejos y extremadamente extensos [10]. En general, una red de Petri tiene los siguientes símbolos:

Place (Lugar, posición): Dibujado como un círculo, es un punto de parada en un proceso, representa (en muchos casos) el logro de un evento.

Transition: Es un rectángulo que representa un evento o acción.

Token: Es un punto negro encontrado en una posición (place). Colectivamente, un conjunto de *tokens* representa el estado actual del proceso. Durante la ejecución del proceso, los *tokens* se mueven de un lugar a otro.

Arc: es un enlace desde una *transition* a un *place* o viceversa.

A continuación consideramos algunas extensiones de las redes de Petri llamadas Redes de Workflows (Workflows Nets) y Patrones de Workflows (Workflows Patterns).

4.2 Workflow Nets (Redes de Workflows)

4.2.1 Visión General

La mayoría de WFMS[1] (Sistemas de Gestión de Workflows) están basados en procedimientos. Un procedimiento es el método de operación utilizado por un proceso de negocio para procesar *casos*. Ejemplos de casos son órdenes, reclamos, gastos de viaje, declaraciones de impuestos, etc. El procedimiento especifica el conjunto de tareas requeridas para procesar estos casos en forma exitosa. Además. El procedimiento especifica la orden (parcial) en que estas tareas tienen que ser ejecutadas. El objetivo de un procedimiento es manejar casos eficiente y apropiadamente. Para alcanzar este objetivo, el procedimiento debería ser puesto a punto para el entorno siempre cambiante de los procesos de negocio. Un WFMS soporta la definición y modificación de procedimientos [11]. Además este muestra que no es difícil mapear un procedimiento a una red de Petri. Resulta que uno puede restringir a una subclase de redes de Petri. Estas subclases son llamadas redes de workflows (WF-Nets). Una WF-Net es una red de Petri con dos posiciones (*places): "i " y "o"*. Estos lugares son usados para definir el inicio y el final de un procedimiento, mirar (Figura 5). Las tareas son modeladas por transiciones y el orden parcial de tareas es modelado por posiciones (*places*) que conectan estas transiciones. El procesamiento de un caso empieza en el momento en el que colocamos un *token* en el lugar en una posición *"i "* y termina en el momento que un *token* aparece en la posición "*o* ".

En general una red de Petri que modela una definición de procesos de workflow (p.e. el ciclo de vida de un caso en aislamiento) es llamado una red de Workflow (WF-Net) [13]. En la (Figura 6) se presenta un ejemplo de una red de Petri que puede ser vista como una red de Workflow también.

[1] WFMS, Workflow Management Systems, son sistemas que definen, crean y gestionan la ejecución de workflows a través del reuso de software, ejecutándose en uno o más motores de workflows, los cuales interpretan la definición del procesos, interactúan con los participantes de workflows y, donde son requeridos, invocan el uso de herramientas IT y aplicaciones [12].

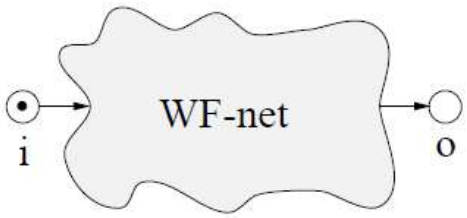


Figura 5: Un procedimiento modelado por una WF-Net

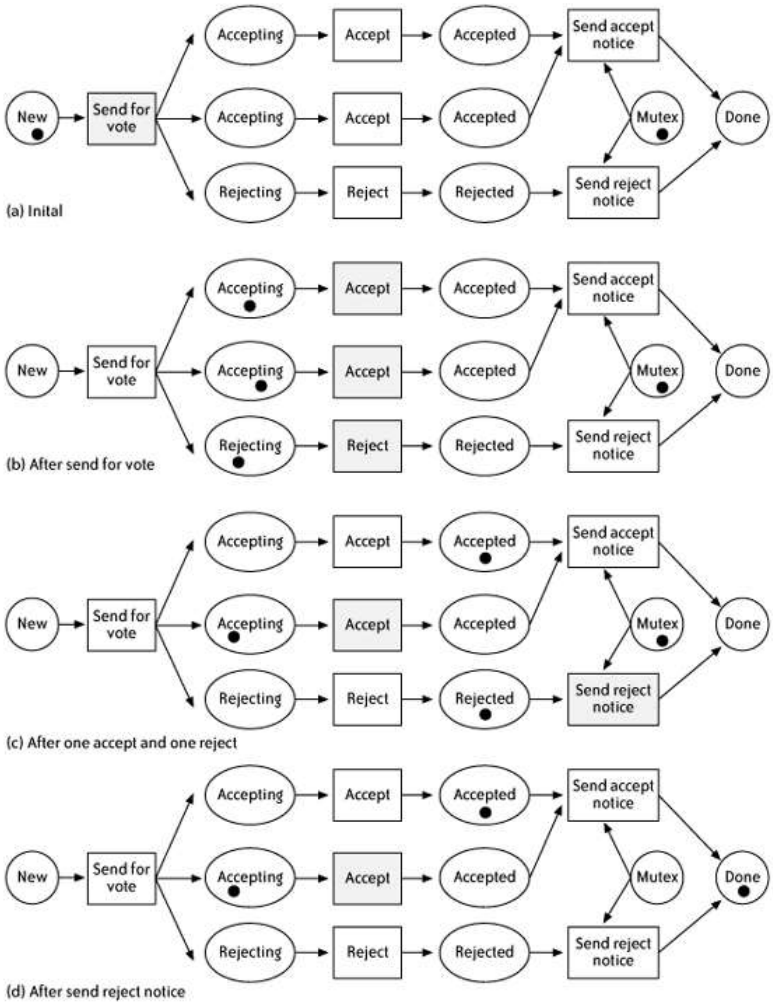


Figure 6: Example Petri net process for vote

Descripción (Figura 6):

El proceso describe que un editor de una revista podría votar así como aceptar un artículo para publicarlo. Cuando el artículo llega. Este es distribuido para su revisión y es votado por un número de editores. La votación se completa cuando cualquiera de los editores la acepta o la rechaza. En el primer caso, al autor le es enviada una noticia de aceptación, en la segunda un rechazo. El comportamiento del proceso de votación es mejor entendido siguiendo el trayecto

de *tokens* a través de la red. Paso (a) muestra la posición inicial: un *token* en *New* y otro en *Mutex*. El estado anterior de la transición *Send for vote* indica que está habilitado; este está habilitado por que su única posición de entrada, *New*, tiene un *token*. Cuando *Send* se activa, el token es es removido de *New* y lso tokens son puestos en los lugares *Rejecting* y *Accepting*, es mostrado en (b). El paso (c) bosqueja el estado de la red después de una aceptación y un rechazo. El token superior del bloque *Accepting* tiene ahora que moverse a *Accepted*; asi mismo *Rejecting* tiene la transición a *Rejected*. De las dos noticias de transiciones enviadas, solo *Send reject notice* se dispara en este paso (d), los tokens de *Rejected* y *Mutex* desaparecen, y ahora un token es movido a *Done* [4].

4.2.2 Ventajas

- Las redes de workflows son una extensión de las redes de Petri y la semántica ha sido definida formalmente a pesar de la naturaleza gráfica, son semánticamente bien definidas.
- Las redes de workflows son basadas en estados en lugar de eventos.
- Las redes de workflows es menos que una notación directa (excepto para académicos masoquistas) y más de un modelo común de comportamiento ha sido trasladado desde y hacia las notaciones industriales.

4.2.3 Desventajas

- Las redes de workflows no soportan las perspectivas organizacional y de información del todo.
- No hay sistemas con una interfaz de usuario robusta.
- Redes de workflows son usualmente modeladas independiente de los datos.

4.3 Workflow Patterns (Patrones de Workflows)

4.3.1 Visión General

Muchos lenguajes han sido propuestos para el diseño y especificación de procesos de workflows, algunos de ellos basados en técnicas de modelado como redes de Petri y mapas de estado. Las funcionalidades convencionales de workflows como task sequencing (secuencias de tareas), split parallelism (divisiones paralelas), join synchronization (uniones sincronizadas), e iterations (iteraciones) han probado la efectividad para la automatización de los procesos de negocio y han sido ampliamente soportadas en los productos actuales de workflows, pero desde el 2000, nuevos requerimientos para los workflows son encontrados en la práctica, abriendo graves incertidumbres acerca de las extensiones para los lenguajes actuales. Diferentes conceptos, aunque exteriormente parezcan ser más o menos lo mismo, son basados en diferentes paradigmas, tienen fundamentalmente diferencias semánticas y diferentes niveles de aplicación más especializados para el modelado o más generalizados para postulados de motores de workflows [14].

Por tal razón en el año 2000 fueron establecidas las iniciativas de Patrones de workflows donde el principal objetivo fue diseñar los requerimientos fundamentales que surgen durante el

modelado de procesos de negocio. Con el fin de describir estos requerimientos una estrategia basada en patrones fue tomada tratando de proveer independencia de la tecnología de implementación y al mismo tiempo independencia de los requerimientos esenciales del dominio en que ellos intentaron ser dirigidos. Un patrón “es una abstracción de una forma concreta que mantiene la aparición periódica en contextos específicos no arbitrarios” [15].

Teniendo en cuenta lo citado en [16] el autor establece algunas especificacionces de workflows vistas desde diferentes perspectivas. La perspectiva del flujo de control (o perspectiva del proceso) describe las actividades y su orden de ejecución a través de diferentes constructores que permiten el control de ejecución, p.e. sequence (secuencia), choice (elección), parallelism (paralelismo) y join synchronization (uniones sincronizadas). En la perspectiva de datos residen los datos de negocio y procesamiento sobre la perspectiva de control. Los documentos de negocio y otros objetos que fluyen entre las actividades, y las variables locales de workflows, calificadas en efectos de pre y post condiciones de actividades de ejecución. La perspectiva de recursos provee una estructura organizacional aferrada al workflow en la forma de roles responsables tanto humanos como de componentes para la ejecución de actividades. La perspectiva operacional describe las acciones elementales ejecutadas por las actividades, donde las acciones se mapean en aplicaciones fundamentales.

De acuerdo a estas perspectivas los patrones originales de worflows fueron enfocados en la perspectiva del flujo de control, sin embargo algunos autores como [17] dice que una descripción comprensiva de un proceso de workflow también requiere la consideración de los datos y las perspectivas de recursos. En [18] el autor quiere direccionar los requerimientos de workflows, de básicos a complejos con el propósito de identificar constructores útiles y especificar patrones encaminados a los requerimientos de negocio en un muy importante estilo de expresión de workflows, pero desprendiéndose de lenguajes de workflows específicos. Para demostrar soluciones para los patrones, su recurso es un mapeo a constructores de lenguaje de workflows existentes. En algunos casos soportados desde un motor de workflows que ha sido indentificado, y así este brevemente diseña las estrategias del nivel de implementación.

En [17] se indica los requerimientos para los lenguajes de workflows a través de patrones de workflows y son descritos como los patrones relacionados a los patrones de flujo de control. Hay cuarenta y tres patrones y estos fueron aplicados con el propósito de examinar las capacidades de los lenguajes de modelado de procesos de negocio así como BPMN, Diagramas de Actividades de UML y EPCs, asi como también los lenguajes de composición de servicios como WCSI, y los lenguajes de procesos de negocio BPML, XPDL, BPEL y otras perspectivas de workflows. Consideremos los resultados obtenidos en las (Tablas 1 y 2) de un análisis detallado de los patrones del flujo de control del siguiente artículo.

Pattern	Staffware	WebSphere MQ	FLOWer	COSA	iPlanet	SAP Workflow	FileNet	BPEL	WebSphere BPEL	Oracle BPEL	BPMN	XPDL	UML ADs	EPC
1 (seq)	+	+	+	+	+	+	+	+	+	+	+	+	+	+
2 (par-spl)	+	+	+	+	+	+	+	+	+	+	+	+	+	+
3 (synch)	+	+	+	+	+	+	+	+	+	+	+	+	+	+
4 (ex-ch)	+	+	+	+	+	+	+	+	+	+	+	+	+	+
5 (simple-m)	+	+	+	+	+	+	+	+	+	+	+	+	+	+
6 (m-choice)	–	+	+	+	+	–	+	+	+	+	+	+	+	+
7 (s-sync-m)	–	+	+	–	–	–	+	+	+	+	+	+	–	+
8 (multi-m)	–	–	+/–	+/–	+	–	+	–	–	–	+	+	+	–
9 (s-disc)	–	–	–	–	+	+/–	–	–	–	–	+/–	+/–	+/–	–
10 (arb-c)	+	–	–	+	+	–	+	–	–	–	+	+	+	+
11 (impl-t)	+	+	+	–	–	–	+	+	+	+	+	+	+	+
12 (mi-no-s)	+	–	+	+	+	+/–	+	+	+	+	+	+	+	–
13 (mi-dt)	+	–	+	–	–	+	–	–	–	+	+	+	+	–
14 (mi-rt)	+	–	+	–	–	+	–	–	–	+	+	+	+	–
15 (mi-no)	–	–	+	–	–	–	–	–	–	–	–	–	–	–
16 (def-c)	–	–	+	+	–	–	+/–	+	+	+	+	+	+	–
17 (mi-par)	–	–	+/–	+	–	–	–	+/–	+/–	–	–	–	–	–
18 (milest)	–	–	+/–	+	–	–	–	–	–	–	–	–	–	–
19 (can-a)	+	–	+/–	+	+	+	+	+	+	+	+	+	+	–
20 (can-c)	–	–	+/–	–	–	+	+	+	+	+	+	+	+	–

Tabla 1: Evaluación de herramientas de modelado con los patrones de referencia (1/2).

Pattern	Staffware	WebSphere	FLOWer	COSA	iPlanet	SAP Workflow	FileNet	BPEL	WebSphere BPEL	Oracle BPEL	BPMN	XPDL	UML ADs	EPC
21 (str-l)	–	+	+	–	+	+	+	+	+	+	+	+	+	–
22 (recur)	+	+	–	+	+	+	–	–	–	–	–	–	–	–
23 (t-trig)	+	–	–	+	–	+	–	–	–	–	–	–	+	–
24 (p-trig)	–	–	+	+	–	+	+	+	+	+	+	+	+	+/–
25 (can-r)	–	–	–	+/–	–	–	–	+/–	+/–	+/–	+/–	+/–	+	–
26 (can-mi)	+	–	–	–	–	+	–	–	–	+	+	+	+	–
27 (comp-mi)	–	–	+/–	–	–	–	–	–	–	–	–	–	–	–
28 (b-disc)	–	–	–	–	–	–	–	–	–	–	+/–	+/–	+/–	–
29 (c-disc)	–	–	–	–	–	+	–	–	–	–	+	+	+	–
30 (s-pjoin)	–	–	–	–	+	+/–	–	–	–	–	+/–	+/–	+/–	–
31 (b-pjoin)	–	–	–	–	–	–	–	–	–	–	+/–	+/–	+/–	–
32 (c-pjoin)	–	–	–	–	–	+	–	–	–	–	+/–	+/–	+	–
33 (g-and-join)	–	–	–	–	–	–	+	–	–	–	+	+	–	+/–
34 (st-pjoin-mi)	–	–	–	–	–	–	–	–	–	–	+/–	+/–	–	–
35 (c-pjoin-mi)	–	–	–	–	–	–	–	–	–	–	+/–	+/–	–	–
36 (dyn-pjoin-mi)	–	–	–	–	–	–	–	–	–	–	–	–	–	–
37 (a-sync-m)	–	+	+	+	–	–	–	+	+	+	–	–	+/–	+
38 (g-sync-m)	–	–	–	–	–	–	+	–	–	–	–	–	–	–
39 (crit-sec)	–	–	+/–	+	–	–	–	+	+	+	–	–	–	–
40 (mi-rout)	–	–	+/–	+	–	–	–	+	+	–	+/–	+/–	–	–
41 (tm)	–	–	–	–	–	–	–	+/–	+/–	+/–	+	+	+	–
42 (ta)	–	–	–	–	–	–	–	+/–	+/–	+/–	+	+	+	–
43 (exp-t)	–	–	–	+	+	+	–	–	–	–	+	+	+	–

Tabla 2: Evaluación de las herramientas de modelado con los patrones de referencia (2/2).

4.3.2 Ventajas

- En los patrones de workflows se puede representar semántica a través de redes de Petri.
- Patrones complejos son fáciles de construir.
- Es fácil modelar splits (divisiones) y combinaciones.

4.3.3 Desventajas

- Es una notación independiente del vocabulario común de las expresiones del flujo de control (y luego del flujo de datos) de los lenguajes de workflows.
- Algunos patrones no se mapean bien sobre redes de Petri de alto nivel.

5. YAWL (Yet Another Workflow Language)

5.1 Visión General

Los patrones de workflows permitieron el desarrollo de YAWL, es un acrónimo de *Yet Another Workflow Language.* A diferencia de otros esfuerzos en el area de BPM, YAWL busco proveer un lenguaje de modelado comprensible para procesos de negocio basado en unos fundamentos formales. El contenido del lenguaje YAWL es una adaptación de las redes de Petri introducidas por los patrones de workflows. Uno de sus mayores aspiraciones era mostrar que un conjunto relativamente pequeño de constructores podría ser utilizado para soportar directamente la mayoría de los patrones de workflows identificados. Este también intentaba ilustrar que ellos podrían coexistir dentro de un framework común. Con el propósito de validar que ese lenguaje era capaz de una promulgación directa, el sistema YAWL fue desarrollado, y sirve como referencia de implementación del lenguaje. Con el tiempo, el lenguaje YAWL y el sistema YAWL de forma creciente se han convertido en sinónimos y han generado un amplio interés de los desarrolladores y la comunidad académica [19].

Las versiones iniciales de YAWL se enfocaron en la perspectiva del flujo de control [16] y dieron una completa implementación de diecinueve de los veinte patrones descritos en la sección anterior (Tabla 1). Ediciones siguientes incorporaron soporte limitado para datos seleccionados y recursos de patrones; sin embargo este esfuerzo fue entorpecido por una falta de una descripción formal completa de los patrones en estas perspectivas. Además, una revisión reciente [17] de la perspectiva del flujo de control identifica veintitrés patrones adicionales que ilustran un número común de constructores de flujo de control utilizado, en muchos de los cuales YAWL no puede proveer un soporte directo, incluyendo join parciales, triggers transitorios y persistentes, iteración y recursión.

En un esfuerzo para gestionar los defectos conceptuales de YAWL con respecto al alcance de los patrones de workflow que ahora han sido identificados, una revisión substancial del lenguaje es propuesta y se denomina newYAWL el cual apunta dar soporte al amplio rango de patrones de workflow en cuanto al flujo de control y las perspectivas de datos y recursos. newYAWL provee una descripción formal comprensiva de los patrones de workflow, que hasta hoy solamente ha sido formalizado parcialmente. Este tiene una completa sintaxis abstracta que identifica las características de cada uno de los elementos del lenguaje. Asociado a este

es un modelo semántico, ejecutable para newYAWL presentado en forma de redes coloreadas de Petri (Coloured Petri Net) que define la semántica en tiempo de ejecución de cada uno de los constructores de lenguaje. Los siguientes párrafos muestran una visión general de las características de newYAWL en la perspectiva del fujo de control.

pattern	XPDL	high-level Petri nets	YAWL
1 (seq)	+	+	+
2 (par-spl)	+	+	+
3 (synch)	+	+	+
4 (ex-ch)	+	+	+
5 (simple-m)	+	+	+
6 (m-choice)	+	+	+
7 (sync-m)	-	–	+
8 (multi-m)	-	+	+
9 (disc)	-	–	+
10 (arb-c)	+	+	+
11 (impl-t)	+	–	–
12 (mi-no-s)	-	+	+
13 (mi-dt)	+	+	+
14 (mi-rt)	-	–	+
15 (mi-no)	-	–	+
16 (def-c)	-	+	+
17 (int-par)	-	+	+
18 (milest)	-	+	+
19 (can-a)	-	+/–	+
20 (can-c)	-	–	+

Tabla 3: Una comparación de las redes de Petri de alto nivel y YAWL utilizando los patrones.

La Tabla 3 indica para cada patrón si las redes de Petri de alto nivel y/o YAWL ofrecen soporte directamente (indicado por un ‘’+’’), soporte directo parcial (indicado por un “+/-”), o no lo soporta directamente (indicado por un “-”). Para la comparación, también se incluyo a XPDL [20].

Perspectiva del flujo de Control

La figura 7 identifica el conjunto de elementos de lenguaje que comprenden la perspectiva del flujo de control de newYAWL. Todos los elementos de lenguajes en YAWL han sido mantenidos y ejecutan las mismas funciones. Muchos constructores nuevos han sido adicionados basados en el amplio rango del los patrones de workflows que ahora han sido identificados [19]. Estos son:

- Los constructores Thread split (división de hilos) y Thread merge (combinación de hilos), que permite al hilo de control ser dividido en múltiples hilos concurrentes o distintos hilos que son integrados en un solo hilo de control respectivamente. El numero de hilos creados/combinados es especificado por el constructor en el modelo de tiempo de diseño. La Figura 8(a) ilustra estos constructores. Después de hacer la tarea del bloque, doce hilos de control son creados asegurando que todas las tareas de botella se ejecuten doce veces antes de que el paquete de tareas en bloques pueda ejecutarse (combinando estos hilos antes que comiencen).

- La unión parcial (también conocida como la unión m-out-of-n) permite una series de divisiones entrantes sean combinadas así como el hilo de control es dirigido a la división subsecuente cuando m de las n divisiones entrantes son habilitadas, entonces la tarea cancel booking es habilitada (y cualquier tarea precedente que están todavía ejecutándose en la región de cancelación asociada son apartadas).

- El ciclo estructurado (que soporta los ciclos o bucles while, repeat y combination) permite una tarea (o una secuencia de tareas en la forma de subprocesos) para ejecutar repetidamente basados en tests condicionales en el inicio y/o al final de cada iteración. El ciclo es estructurado en forma y este tiene una sola entrada y un punto de salida. La Figura 8(c) ilustra un ciclo repetitivo para la tarea check backup que se ejecuta repetidamente hasta que todos los backups han sido verificados (es decir, se trata de un post-prueba del ciclo repeat).

- La terminación de la región soporta la finalización de las tareas que estas abarcan. En la figura 8(c) la tarea de prueba que recupera completamente es forzada completamente una sola vez (todas las iteraciones de) la tarea check backup ha finalizado. Esto permite que la tarea issue review report sea inmediatamente establecida.

- Las tareas “señales de persistencia” (Persistent triggers) y “señales transitorios” (Transient triggers) soportan la aplicación de una tarea dependiendo de una señal que está siendo recibida de un entorno operativo. Estos pueden durar o ser transitorios respectivamente. La Figura 8(d) ilustra una señal de persistencia (asumiendo que está asociada con alguna forma de alarma) que permite a la tarea deadline, ser habilitada cuando esta es recibida. Así como esta señal es de tipo duradera, es retenida para un uso futuro si es recibida antes que el hilo de control llegue a la tarea deadline.

- El constructor Disablement arc permite múltiples instancias de tareas para prevenir la creación de más instancias que se permita para cada una de las instancias que se están ejecutando en ese momento para que se completen normalmente. La Figura 8(d) tiene un constructor disablement arc asociado con la tarea deadline que impide que nuevos documentos sean aceptados una vez que se han completado.

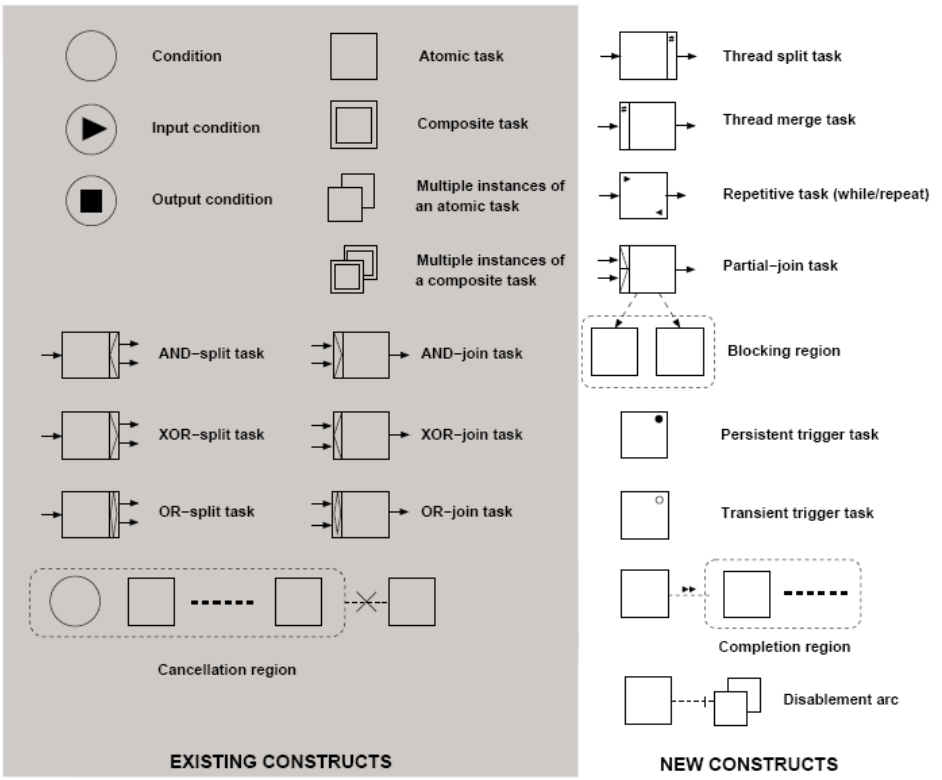


Figura 7: Conjunto de los elementos del lenguaje newYAWL.

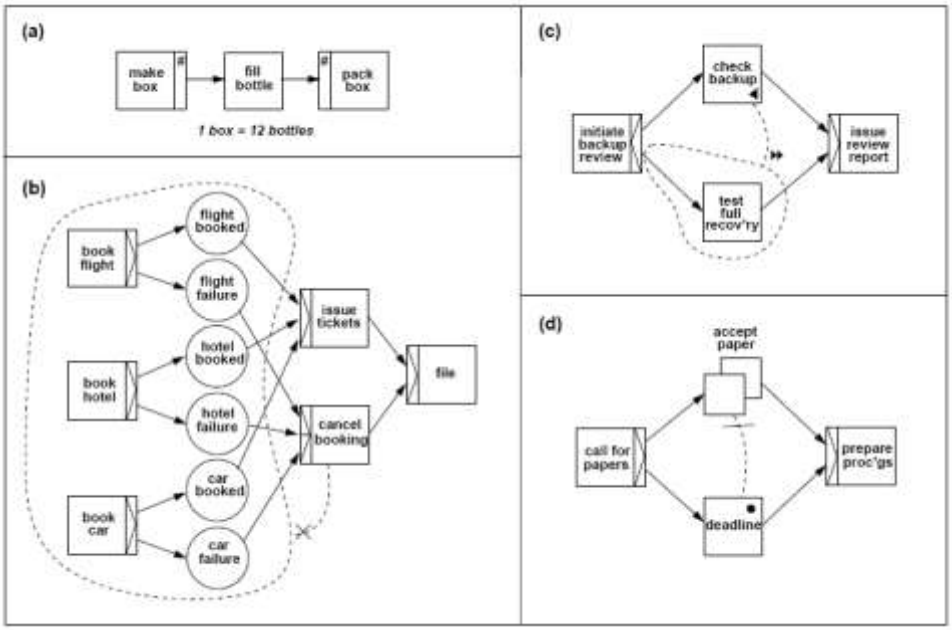
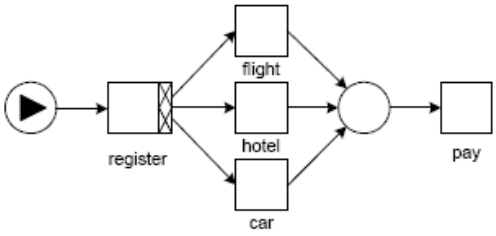
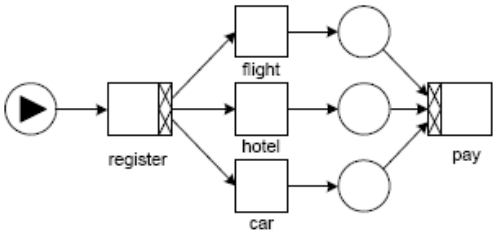


Figura 8: Ejemplos de los constructores del flujo de control de newYAWL.

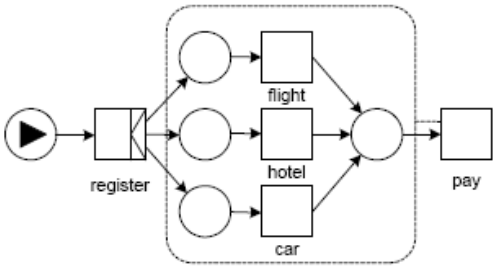
El ejemplo dado en esta sección muestra que YAWL resuelve muchos de los problemas indicados en la (Tabla 3) [20].



(a) Task pay is executed each time one of the three preceding task completes (Pattern 8).



(b) Task pay is executed only once, i.e., when all started tasks have completed (Pattern 7).



(c) Task pay is executed only once, i.e., when the first task has completed (Pattern 9).

Figure 9: Algunos ejemplos que ilustran la manera en que YAWL trata con patrones avanzados de sincronización.

Descripción (Figura 9):

Una unión OR puede ser interpretada de múltiples maneras. La Figura 9 muestra tres posibles interpretaciones usando el registro de un viaje de negocio como ejemplo. La primera especificación de workflow 9(a) empieza con una unión OR *register* que permite las tareas *flight*, *hotel y*/o *car*. La tareas *pay* es ejecutada para cada tiempo una de las tres tareas. (esto es, *flight*, *hotel*, and *car*) se completen. Este constructor corresponde al patrón *Multi merge* (Patron 8 Tabla 3). La segunda especificación de workflow que se muestra en la Figura 8(b) es similar pero combinca los pagos individuales en un solo pago. Además, ésta espera hasta que cada una de las tareas habilitadas por el *register* se complete. Note que si solamente un vuelo es reservado, no hay sincronización. Sin embargo, si el viaje contiene dos o aún tres elementos, la tarea *pay* es retardada hasta que todas se hayan completado. Este constructor corresponde al patrón *synchronizing merge* (Patrón 7 Tabla 3). La tercera especificación de workflow 8(c) permite las tres tareas (esto es, *flight*, *hotel*, and *car*) pero paga después de que la primera tareas se ha completado. Después del pago todas las tareas que se están

ejecutando son canceladas. Aunque este constructor no tiene sentido en este contexto se ha adicionado para ilustrar como el patrón *Discriminator* puede confirmase (Patrón 9 Tabla 3) asumiendo que todos los hilos que se ejecutan son cancelados en el momento en que el primero se ha completado.

YAWL en términos de instanciación semántica

En [21] es propuesto un framework llamado CASU framework y su investigación está enfocada en el problema de la instanciación y su representación en modelos de procesos. Este trabajo contribuye con una descripción sistemática de alternactivas de diseño para la especificación de instanciación semántica de lenguajes de modelado de procesos. CASU significa: Creación de instanciación (C), Activación del hilo de control (A), Subscripción de eventos (S), y Desuscripción (U). La tabla 4 muestra los casos de patrones establecidos para esta investigación.

YAWL se concentra en el control y el flujo de control entre las instancias de procesos. Hay una distinguida "condición inicial" por modelo de proceso, sin embargo, las definiciones de cómo y cuando la instanciación toma lugar no parte de modelos de YAWL. Las nociones de condiciones iniciales o de eventos iniciales no están presentes. Por consiguiente, este no soporta C-1 hacia C-4, A-3 hacia A-5, ninguno de los patrones S hacia los patrones U. El estado inicial de la instancia de un proceso es implícitamente dada por: hay exactamente un punto de partida que recibe una señal (token) sobre una instanciación. (A-2) (Tabla 4).

De esta manera, YAWL es similar a un UAD (Diagrama de Actividades UML) en términos de instanciación semántica con una adicional restricción a exactamente un punto de partida. Las redes de Workflow comparten el mismo perfil de instanciación con YAWL.

Patterns	oWFN	EPCs	UAD	YAWL	BPEL	BPMN
C-1 Ignorance	+	+	+	+	+	+
C-2 Single Condition Filter	–	+	–	–	–	–
C-3 Multi Condition Filter	–	+	–	–	–	–
C-4 Single Event Trigger	–	+	–	–	+	+
C-5 Multi Event Trigger	–	+	–	–	–	+
A-1 Initial State	+	–	–	–	–	–
A-2 All Start Places	–	–	+	+	–	–
A-3 True Conditions	–	+	–	–	–	–
A-4 Occurred Events	–	+	–	–	+	+
A-5 Occurred Events plus Cond.	–	+	–	–	–	–
S-1 All Subscriptions	+	∅	–	–	+	–
S-2 No Subscriptions	–	∅	–	–	–	+
S-3 Reachable Subscription	–	∅	–	–	+	–
U-1 Until Consumption	–	∅	–	–	+	–
U-2 Until Termination	+	∅	–	–	+	–
U-3 Timer-based	–	∅	–	–	+	–
U-4 Event-based	–	∅	–	–	+	–
U-5 Proper Completion	–	∅	–	–	–	–

Tabla 4: Una comparación de instanciación en diferentes lenguajes de modelados de procesos

En [22] es propuesto un software de integración en el nivel de modelos de procesos de negocio y el objetivo es crear la descripción de comportamiento de sistemas integrados que vayan de acuerdo con las descripciones de comportamiento de los sistemas locales originales para ser integrados. Los procesos de negocio son unos buenos candidatos para modelar el comportamiento de los sistemas que pueden automáticamente ser derivado de sistemas existentes a través de minería de procesos. La idea detrás del método de integración es identificar las dependencias, también llamadas interdependencias de procesos, entre procesos de negocio locales existentes con la creación de procesos de negocio globales.

YAWL en términos de interdependencias de procesos

Identificación de interdependencias de procesos: La Tabla 5 contiene todas las interdependencias de procesos que son consideradas prácticas en un mundo real de procesos de negocio. Cada dependencia es explicada a través de un ejemplo donde “a” y “b” son actividades de procesos de negocio origen y destino, respectivamente, y “s” representa el estado en el origen. Las dependencias están agrupadas de acuerdo a sus efectos en “b” [22].

DEPENDENCY	EXAMPLE
Enabling dependencies ($\curvearrowright$b):	
$^{\bullet}a \curvearrowright b$	When a sports competition starts (a), access to the team statistics (b) is enabled.
$\overset{\bullet}{a} \curvearrowright b$	During travel booking (a), the optional reservation of a car is enabled (b).
$a^{\bullet} \curvearrowright b$	When booking a travel (a) finishes, printing the booking (b) is enabled.
$^{\bullet}s \curvearrowright b$	When an order enters state “all parts delivered” (s), reclaiming the order is enabled (b).
$\overset{\bullet}{s} \curvearrowright b$	While a student is in state “is PhD candidate” (s), applying for an internship (b) is enabled.
$s^{\bullet} \curvearrowright b$	When the temperature of cooling water leaves a critical state (s), the experiment is enabled to start again (b).
Triggering dependencies ($\rightarrow b$):	
$^{\bullet}a \rightarrow b$	When the backup of all computers starts (a), the backup of a single computer (b) is invoked.
$\overset{\bullet}{a} \rightarrow b$	While a sensor measures values (a), a backup sensor is activated (b) for value comparison.
$a^{\bullet} \rightarrow b$	When a security officer finishes his shift (a), another security officer must start his shift (b).
$^{\bullet}s \rightarrow b$	When an operating system enters state “logged off” (s), an external backup is started (b).
$\overset{\bullet}{s} \rightarrow b$	While a power plant is in state ”in operation” (s), a monitoring system (b) is running.
$s^{\bullet} \rightarrow b$	When a order leaves state ”in archive” (s), a notification service (b) is invoked.
Disabling dependencies ($\not\curvearrowright b$):	
$^{\bullet}a \not\curvearrowright b$	When an inspection of a power plant starts (a), turning on the power plant (b) is disabled.
$\overset{\bullet}{a} \not\curvearrowright b$	During the booking of a bargain travel (a), the cancellation of the hotel (b) is disabled.
$a^{\bullet} \not\curvearrowright b$	When a payment process finishes successfully (a), cancelling the payment (b) is disabled.
$^{\bullet}s \not\curvearrowright b$	When a hotel enters state “booked out” (s), booking a room is disabled (b).
$\overset{\bullet}{s} \not\curvearrowright b$	While a room resides in state “occupied” (s), the cleaning service of the room is disabled (b).
$s^{\bullet} \not\curvearrowright b$	When a book leaves state ”in archive” (s), changing the delivery address of the book (b) is disabled.
Cancelling dependencies ($\not\downarrow b$):	
$^{\bullet}a \not\downarrow b$	When an experiment with high priority starts (a), a running experiment with lower priority (b) is cancelled.
$\overset{\bullet}{a} \not\downarrow b$	When a experiment with higher priority starts (a), an ongoing experiment with lower priority (b) is cancelled and restarted after a has finished execution.
$a^{\bullet} \not\downarrow b$	When the supervisor leaves the experiment (a), a participating assistant must stop working (b).
$^{\bullet}s \not\downarrow b$	When the temperature of cooling water enters a critical state (s), the running experiment (b) is stopped.
$\overset{\bullet}{s} \not\downarrow b$	When the temperature of cooling water enters a critical state (s), the running experiment (b) is stopped and restarted after the temperature has fallen below the critical level.
$s^{\bullet} \not\downarrow b$	When a hotel leaves state “rooms available” (s), a currently running hotel booking (b) is cancelled.

Tabla 5: Ejemplos de interdependencias de procesos

Dependencias Enabling and triggering: en contraste a una dependencia WF-Net •a $\curvearrowright$ b puede ser modelada en YAWL sin que se asuma que “a” deba ser una tarea automática por que dos tareas pueden ser conectadas directamente. Esto signiica que el estado entre AND-Split y “a” en la Figura 10 que modela la versión de WF-net de dependencia •a $\curvearrowright$ b puede ser

removida y “a” es ejecutada inmediatamente despueés de un AND-Split. Las dependencias restantes son soportadas en la misma manera como WF-nets, desde que ninguna extensión este provista que incrementaría su soporte.

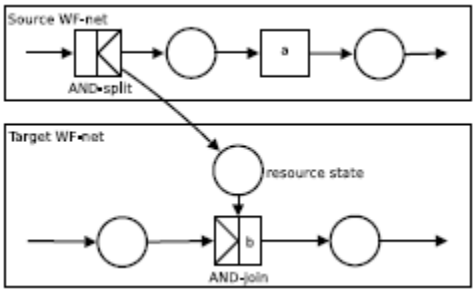


Figure 10: Dependency •a ↷ b.

Dependencias Disabling: Están indirectamente ó solo bajo ciertas suposiciones soportadas por simples WF-nets. YAWL mejora el soporte en algunos casis tomando ventajas de la funcionalidad remover símbolo (remove token) introducida recientemente. La ventaja yace en la expresividad del lenguaje. Esto hace posible modelar una actividad de tal manera que pueda ser ejecutada mientras que “b” está ejecutándose y ésta puede ser parte de una ejecución y un bucle de ejecución porque puede ser ejecutada muchas veces. Estas dos circunstancias no pueden ser realizadas por las WF-nets. La dependencia •a ↶ b puede ser modelada un poco diferente a WF-nets simples como se describe en la Figura 11.

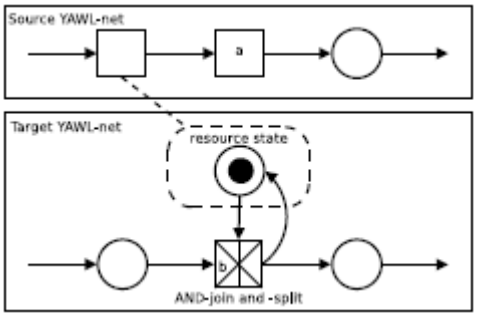


Figura 11: Dependencia •a ↶ b.

Sin embargo, tenemos que asumir que la tarea que remueve los símbolos (tokens), no toma tiempo y esa “a” es disparada (triggered) inmediatamente después de que los símbolos son removidos, asi que deshabilitar “b” sucede en el momento en que “a” comienza la ejecución. Estas suposiciones son necesarias porque no está exactamente definido cuando los símbolos son removidos. El reporte técnico de YAWL afirma “…el momento en que la tarea ejecuta todos los símbolos en esta área son removidos” [20]. Pero esto no determina si estos son removidos al principio, al final, ó durante la ejecución. Las dependencias restantes pueden ser modeladas de una manera similar como se muestra en la Figura 10, asumiento que las tareas remueven los simbolos que no consumen tiempo. Debido a las suposiciones hechas por el modelado de las dependencias, los resultados son los mismos para WF-nets.

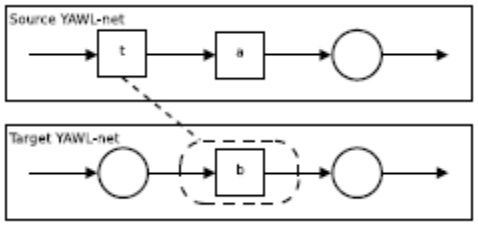


Figure 12: Dependency •a ↯ b.

Las dependencias restantes •s ↯ b y s• ↯ b son modelados por una tarea adicional donde “s” es el estado posterior y el estado anterior de la tarea, respectivamente. Esta tarea permite la extracción del símbolo de “b”. Para •s ↯ b, se debe asumir que el símbolo es removido cuando la tarea finaliza la ejecución. Para s• ↯ b, se debe asumir que el símbolo removido sucede al final de la ejecución.

El modelado de la dependencia “b” con la funcionalidad de remover símbolo de YAWL podría inducir a un comportamiento no deseado desde un punto de vista orientado a objetos. Si un símbolo representa una instancia de proceso de negocio o un identificador de un objeto, no está definido que pasa con esa instancia o ID del objeto una vez es removido. En caso de una instancia de proceso, esta podría significar que un caso de negocio no es procesado más o es abortado sin ejecutar una tarea de compensación. Una solución práctica sería dejar el símbolo en la red y quitar (rollback) la tarea destino de la dependencia. Sin embargo, YAWL no incluye una transición de tipo rollback comparada al comportamientode bajo nivel de una tarea de WF-net [22].

La Tabla 6 contiene un análisis de cada lenguaje de modelado para determinar si soporta las dependencias identificadas. Los posibles resultados fueron, (1) el lenguaje soporta la dependencia directamente (escrito con “+”), (2) soporta la dependencia indirectamente la dependencia “(+)”, (3) soporta la dependencia bajo ciertas suposiciones (escrito con “ ∼”), ó (4) no soporta la dependencia (escrito con “−”). El soporte directo significa que un solo nodo o arco puede expresar la dependencia mientras que el soporte indirecto significa que una red de elementos es necesaria para modelarlo [22].

DEP.	WF-net	YAWL	UML 2.0 Activities	BPMN	EPC
Enabling dependencies ($\curvearrowright b$):					
$^{\bullet}a \curvearrowright b$	$\sim$	(+)	(+)	(+)	−
$\overset{\bullet}{a} \curvearrowright b$	(+)	(+)	(+)	$\sim$	−
$a^{\bullet} \curvearrowright b$	(+)	(+)	(+)	(+)	(+)
$^{\bullet}s \curvearrowright b$	(+)	(+)	(+)	(+)	(+)
$\overset{\bullet}{s} \curvearrowright b$	(+)	(+)	$\sim$	$\sim$	−
$s^{\bullet} \curvearrowright b$	$\sim$	$\sim$	(+)	(+)	−
Triggering dependencies ($\rightarrow b$):					
$^{\bullet}a \rightarrow b$	$\sim$	(+)	(+)	(+)	−
$\overset{\bullet}{a} \rightarrow b$	(+)	(+)	$\sim$	(+)	−
$a^{\bullet} \rightarrow b$	(+)	(+)	(+)	(+)	(+)
$^{\bullet}s \rightarrow b$	(+)	(+)	−	(+)	(+)
$\overset{\bullet}{s} \rightarrow b$	(+)	(+)	−	(+)	−
$s^{\bullet} \rightarrow b$	$\sim$	$\sim$	−	(+)	−
Disabling dependencies ($\not\curvearrowright b$):					
$^{\bullet}a \not\curvearrowright b$	(+)	(+)	$\sim$	$\sim$	−
$\overset{\bullet}{a} \not\curvearrowright b$	(+)	(+)	$\sim$	$\sim$	−
$a^{\bullet} \not\curvearrowright b$	(+)	(+)	$\sim$	$\sim$	−
$^{\bullet}s \not\curvearrowright b$	$\sim$	$\sim$	$\sim$	$\sim$	−
$\overset{\bullet}{s} \not\curvearrowright b$	$\sim$	$\sim$	$\sim$	$\sim$	−
$s^{\bullet} \not\curvearrowright b$	(+)	(+)	$\sim$	$\sim$	−
Cancelling dependencies ($\lightning b$):					
$^{\bullet}a \lightning b$	−	$\sim$	$\sim$	(+)	−
$\overset{\bullet}{a} \lightning b$	−	$\sim$	$\sim$	$\sim$	−
$a^{\bullet} \lightning b$	−	$\sim$	$\sim$	(+)	−
$^{\bullet}s \lightning b$	−	$\sim$	−	(+)	−
$\overset{\bullet}{s} \lightning b$	−	$\sim$	−	(+)	−
$s^{\bullet} \lightning b$	−	$\sim$	−	(+)	−

Tabla 6: Comparación de lenguajes de modelado: + soportado directamente, (+) soportado indirectamente, $\sim$ soportado bajo ciertas condiciones, − no soportado.

5.2 Ventajas

- Soporte par adaptación dinámica de modelos de workflows a través de nociones de worklets
- Sofisticado modelo de validación de características de workflows (e.g. detección de punto muerto en el tiempo de diseño).
- Modelo basado en XML ara la definición y manipulación de datos basado en XML Schema, XPath y XQuery.
- Interfaces basadas en XML para el monitoreo y control de las instancias de workflows y para el acceso a los registros de ejecución.
- Plugins de interfaces basados en XML que permiten conectar servicios web de terceros al sistema, incluyendo listas de trabajo/ tareas de terceros.
- Generación automática de formularios, a partir de XML Schema.

5.3 Desventajas

- Terminación implicita no es soportada por que el diseñador es forzado a identificar un único nodo final. Cualquier modelo con multiples nodos finales pueden ser transformados en una red con un único nodo (simplemente se usa un synchronizing merge). Esto no ha sido adicionado a YAWL para forzar al diseñador a pensar en la realización satisfactoria del caso. Este requerimiento permite la de detección de terminaciones no exitosas.
- Es un intento académico para realizar una notación bien definida que no se trabaja en los Patrones de Workflows.
- No es ampliamente adoptada.
- Poca semántica implementada.
- Si un símbolo representa una instancia de un proceso de negocio o un identificador de un objeto, no se define que pasa con la instancia o el ID del objeto una vez es removido.

6. BPMO (Business Process Modeling Ontology)

6.1 Visión General

El modelado de procesos de negocio es la primera etapa en el ciclo de vida de la gestión de procesos de negocio. Los últimos son preferidos por los usuarios de negocio. Sin embargo, un gran esfuerzo ha sido invertido en varias clases de formalizaciones y transformaciones. Métodos anteriores estuvieron basados en XML como sintaxis de serialización. La interpretación de modelos de procesos se enfocaba en ese entonces en técnicas como Extensible Style sheet Language Transformations (XSLT). Los esfuerzos de estandarización podrían ser tomados para reducir de alguna manera la necesidad de muchas traducciones. XPDL, primero diseñado como lenguaje de intercambio de workflows, es un estándar para el intercambio de modelos de workflows entre diferentes sistemas de gestión de workflows, sin información en tiempo de ejecución [23].

Para facilitar a los usuarios de negocio en el entendimiento de los diagramas de procesos de negocio, varias notaciones fueron introducidas. Entre ellas BPMN, una Notación de Modelado de Procesos de Negocio que ha ganado importancia muy rápidamente. El verdadero desafio, sin embargo, es unificar las IT y la visión de expertos de negocio. En muchas aproximaciones una única sintaxis es deseada. Sin embargo, no solo la sintaxis necesita ser homogeneizada per sobre todo la semántica si lo necesita. Uno de las ideas iniciales fue combinar los servicios Web semánticos, o más generalmente las tecnologías semánticas y BPM para crear la Gestión de Procesos de Negocio Semánticos (SBPM) [24].

En el proyecto SUPER (Semantics Utilized for Process Management within and between Enterprises), el rol de unificación para varias notaciones de modelado de procesos es provista por BPMO (Business Process Modeling Ontology). Es posible modelar procesos directamente en el modelador BPMO. Sin embargo, considerando el legado de procesos en empresas uno esperaría importar herramientas que ayuden a reducir la brecha entre “interoperabilidad de

modelos de procesos de negocio sintáctico" y haciendo uso de nuevos métodos de tecnologías semánticas [25].

SUPER ha identificado dos de las notaciones de modelado de procesos más comúnmente utilizadas. Una de ellas es EPC, Event-Driven Process Chains, que ha Ganado una amplia adopción por parte de la industria, especialmente en países de habla Alemana. El segundo es BPMN, un estándar de industria de la OMG. Ambos soportan modelado grafico de procesos. El propósito de [25] es la sintaxis de serialización para BPMN a través de XPDL, el problema de investigación es entonces como se traslada los procesos XPDL existentes en su contraparte mejorada semánticamente en BPMO. El problema fue descomponerlo en dos partes: ontologización de la notación y la alineación semántica, por lo tanto la introducción de la capa intermedia que es llamada Semantic Business Process Modeling Notation (sBPMN), facilitando una validación adicional en un nivel semántico.

BPMN carece de una semántica formal. Por un largo tiempo no había ningún estándar de notación especifico de serialización en XML y muchos proveedores desarrollaron formatos propietarios. En un esfuerzo hacia la estandarización, una ontologización de BPMN fue desarrollada dentro del proyecto SUPER – La ontología sBPMN, representada en WSML (Web Service Modeling Language) –. Teniendo una ontología para solo un método de modelado no resuelve el problema de integración entre procesos colaborativos, además Business Process Modeling Ontology (BPMO) fue desarrollada como una abstracción sobre sBPMN y otras notaciones ontologizadas, e.g. sEPC para Even-driven Process Chains semánticos.

sBPMN consta de jerarquías de conceptos junto con sus atributos y un conjunto de axiomas que permiten chequear automáticamente si un diagrama de procesos de negocio está bien formado. La semántica que se provee en sBPMN da muchas ventajas sobre otras soluciones. Primero que todo, es posible chequear si el modelo serializado diseñado en sBPMN cumple con la especificación BPMN. Si el diagrama es incompleto, la información perdida podría en muchos casos ser inferida. Además, los conceptos podrían ser reclasificados si es necesario.

Usar una ontología intermedia en lugar de traducirla directamente a BPMO, por muchas razones. Primero, es útil para el chequeo de restricciones sobre modelos de procesos traducidos. Un sBPMN refleja la especificación BPMN, el nivel de abstracción es más fácil de entender para diseñadores de herramientas, y facilitaría el desarrollo de filtros para exportar diagramas BPMN. Segundo, la semántica es provista a un nivel en que los usuarios de negocio están familiarizados con ella. Finalmente, esto facilita el chequeo si un diagrama de proceso de negocio (BPD) está bien formado antes de utilizar algunas operaciones de traducción que asumen un BPD bien formado. La adición semántica es entonces hecha dentro de SUPER como una traducción a BPMO.

Una de las filosofías subyacentes centro de SUPER es adoptar un enfoque incierto abierto siempre que sea apropiado. Relacionado a esto la ontología sEPC primero fue creada en OWL, y entonces automáticamente traducida a OCML, el lenguaje de representación subyacente IRS-III, utiliza un traductor automatizado de la Open University (OU). La ontología está disponible

también en WSML el cual usa el traductor automatizado disponible como parte de las APIs del SEE e IRS-III. La estructura principal de la ontología sEPC viene de los principales elementos de EPC.

BPMO representa Business Process Modeling Ontology [26]. La idea detras de BPMO fue crear una ontología que sería capaz de represntar artefactos de varias metodologías de modelado de procesos y proponer una representación unificada y una única interpretación. Esta es la principal ontología de modelado del proyecto SUPER.

BPMO unifica dos métodos de modelado: basado en grafos y basado en bloques. El primero es preferido por los usuarios de negocio; el ultimo es necesario para la traducción a BPEL antes de que la ejecución se realice. Para claridad los elementos a ser usados en los siguientes métodos son puestos bajo Patrones de Grafos y Patrones de Bloques respectivamente. En un proceso uno puede mezclar ambos métodos. Un bloque puede ser tratado como un fragmento de procesos y puede ser conectado en una manera SESE: single entry, single exit, (una sola entrada, una sola salida). Cada flujo de objeto necesita estar contenida dentro de un bloque o un grafo de workflow. Partes del modelo de proceso puede ser traducida si la equivalencia semántica es guardada. Todos los patrones de bloques pueden ser representados como patrones de grafos y solo algunos grafos pueden ser traducidos a bloques.

La idea detrás de BPMO fue crear una ontología que sería capaz de representar artefactos de varias metodologías de modelado de procesos y proponer una representación unificada y una única interpretación. Esta es la principal ontología de modelado del proyecto SUPER1 (Semantics Utilized for Process Management within and between Enterprises), y unifica dos métodos de modelado: basado en grafos y basado en bloques. El primero es preferido por los usuarios de negocio; el último es necesario para la traducción a BPEL antes de que la ejecución se realice. En el proyecto SUPER, el rol de unificación para varias notaciones de modelado de procesos es provista por BPMO (Business Process Modeling Ontology).

Algunas ventajas presentes en el lenguaje BPMO en comparación con los anteriores lenguajes de modelado son las siguientes:

- BPMO es un destino común de traducción y además es un recurso para notaciones industriales.
- BPMO es una representación común sin ser de tan bajo nivel y complicado como las redes de workflows.
- Unas características comunes a través de un vocabulario basado en patrones de workflows.
- Basado en BPMN, en lugar de imponer una nueva notación (así como YAWL).

[1] El proyecto SUPER pretendía llevar la Gestión de Procesos de Negocio (BPM) desde el nivel de las tecnologías de la Información (TI), donde reside ahora, hacia el nivel de negocio donde debe estar. El resultado fue el desarrollo de herramientas que permiten el despliegue de la Gestión de Procesos de Negocio Semánticos (SBPM). Este proyecto fue financiado por la Unión Europea dentro de las prioridades de la Information Society Technologies (IST). Disponible en Internet: http://www.ip-super.org/

- BPMO preserva y está fundamentado en la perspectiva organizacional y en las capacidades del modelado de dominio de EPCs, en lugar de concentrarse solo en el comportamiento.
- BPMO presenta las bases del razonamiento automatizado basado en ontologías.
- Tiene una conexión formal entre la semántica del comportamiento y los procesos ejecutables a través del razonamiento basado en ontologías.

Por medio de las descripciones anteriores de los lenguajes de modelado de procesos de negocio más utilizados y además del análisis presentado en [5] se observa que BPMO es el lenguaje que reúne las mejores características de cada uno de ellos. Además que presenta un enfoque innovador para la representación semántica de procesos de negocio y es un lenguaje que permite realizar anotaciones semánticas, así como también añadir ontologías de dominio para realizar consultas basadas en razonamiento semántico. Así mismo, ofrece los mejores aspectos que se ajustan a los objetivos planteados en este trabajo de grado.

Referencias

[1] Ankhi M., Mahendrababu R., Spoorti P. Implement Business Process Management to realize Cost Savings and High Return on Investments. TATA Consultancy Services. 2009.

[2] Stephen A. White, BPM Architect, IBM. OMG BPMN Tutorial. Introduction to BPMN. IBM Software Group. 2006.

[3] Stephen A. White, BPM Architect, IBM. Introduction to BPMN. IBM Corporation. 2004.

[4] HAVEY, Mike. Essential Business Process Modeling. O'Reilly. 2005.

[5] LIST, Beate., KORHERR, Birgit. An Evaluation of Conceptual Business Process Modelling Languages. Vienna University of Technology. 2005.

[6] Curtis, B., Kellner, M. and Over, J. Process Modeling. Communication of the ACM, Vol. 35, No.9, 1992.

[7] G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.

[8] W.M.P. van der Aalst. Formalization and Verification of Event-driven Process Chains. Department of Mathematics and Computing Science, Eindhoven University of Technology. 2000.

[9] Dr. J.W. Schmidt. A Comparison of Event-driven Process Chains and UML Activity Diagram for Denoting Business Processes. April 2001.

[10] W.M.P. van der Aalst. Projecting multi-dimensional Petri nets. Eindhoven University of Technology. 2000.

[11] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. Eindhoven University of Technology. 2000.

[12] Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1996.

[13] W.M.P. van der Aalst. Verification of Workflow Nets. Eindhoven University of Technology. 2005.

[14] W.M.P. van der Aalst, A. P. Barros, A.H.M. ter Hofstede, B. Kiepuszewski. Advanced Workflow Patterns. 2000.

[15] D. Riehle and H. Z□ullighoven. Understanding and Using Patterns in Software Development. Theory and Practice of Object Systems. 1996.

[16] S. Jablonski and C. Bussler. Workflow Management: Modeling Concepts, Architecture and Implementation. Thomson Computer Press, London, UK, 1996.

[17] Nick Russell, Arthur H.M. ter Hofstede, W.M.P. van der Aalst, Nataliya Mulyar. Workflow Control-Flow Patterns, A Revised View. 2007.

[18] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. July 2003.

[19] N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst. *new YAWL*: *Specifying* a Workflow *Reference* Language *using Coloured Petri* Nets. 2007.

[20] W.M.P. van der Aalst, A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. 2005.

[21] G. Decker, J. Mendling. Instantiation Semantics for Process Models. 2008.

[22] G. Grossman, M. Schrefl, M. Stumptner. Modeling Inter-Process Dependencies with High-Level Business Process Modeling Languages. 2008.

[23] XML Process Definition Language. Version 2.0. Workflow Management Coalition. 2005.

[24] Hepp M, Leymann F, Domingue J, Wahler A, Fensel D. Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. 2005.

[25] J. Domingue et al. Deriverable D4.5. sBPMN and sEPC to BPMO Translation. 2008.

[26] S. Heymans, et al. Process Modelling Ontology and Mapping to WSMO. Deliverable D1.1. SUPER Consortium. 2007.

[27] L. Cabral et al. Deliverable D4.6 BPMO to sBPEL Two Way Translation. SUPER Consortium. 2008.

[28] Z. Yan, E. Cimpian, M. Mazzara, M. Zaremba. BPMO Requirements Analysis and Design 2007.