Productos como referentes explícitos en el modelado de procesos de negocios

Germán Regis y Nazareno Aguirre

Departamento de Computación
Facultad de Ciencias Exactas, Físico-Químicas y Naturales
Universidad Nacional de Río Cuarto
Ruta 36 Km. 601, Río Cuarto (5800)
Córdoba, Argentina
{gregis,naguirre}@dc.exa.unrc.edu.ar

Resumen La necesidad de eficiencia y organización en un mundo cada vez más competitivo, ha impulsado, en la última década, la investigación y desarrollo de distintos métodos y formalísmos (lenguajes) en el area de procesos de negocios; particularmente ha tenido un gran auge la especificación y verificación de modelos de procesos de negocios. Entre los lenguajes más difundidos podemos mencionar *Business Process Modeling Notation*(BPMN), para el cual se han aportado numerosos trabajos para el análisis formal de sus modelos. Sin embargo, como sucede en la gran mayoria de este tipo de lenguajes de modelado, no proveen como parte del lenguaje un objeto que permita especificar *productos* (una referencia a un objeto empírico con propiedades) como un referente explícito del modelo.
En este trabajo analizamos diferentes trabajos en el área de métodos formales que tienen como objetivo brindar soporte formal a diferentes lenguajes para el modelado de procesos de negocios. Retomamos un formalismo para especificar procesos de negocios, que provee la especificación de productos como referentes explícitos del modelo y extendemos algunas de sus características con el fin de ampliar su campo de aplicación. También proponemos una sintaxis de alto nivel para el mismo, que facilita la especifiación de sus modelos.

Palabras Clave: Métodos Formales, Real-Time, Autómatas Temporizados, Procesos de Negocios

1 Introducción

El constante esfuerzo de distintas organizaciones por el perfeccionamiento de sus procesos en busca de eficiencia y control, ha impulsado el actual auge de diversos métodos y lenguajes para el modelado de procesos de negocios. Si bien existen diversos lenguajes como *Business Process Modeling Language*(BPMN) entre otros, en su gran mayoría están orientados a servicios y en general no proveen una semántica formal que permita realizar análisis automático de sus modelos. No obstante se pueden encontrar, en el área métodos formales, numerosos trabajos

que aportan técnicas y herramientas con el fin de formalizar dichos lenguajes. En general, estas técnicas adoptan algún lenguaje para modelar procesos de negocios o parte de ellos, como ciertos diagramas que en ellos se definen, y le brindan una semántica formal con soporte en algún formalismo. Luego, haciendo uso de las herramientas automáticas o semi-automáticas de dichos formalismos, permiten analizar propiedades que cumplen los modelos originales. Generalemente estas propiedades son expresadas en algúna lógica temporal. En [14] podemos encontrar un exhaustivo análisis de éstos trabajos clasificados según los formalismos en los cuales se basan.

Si bien el análisis, mediante modelos de procesos de negocios, es cada vez más utilizado para la evaluación de organizaciones o para el diseño y control en etapas tempranas de desarrollo, la gran mayoria de los lenguajes actuales para tal fin, no contienen un objeto que permita especificar *productos* (una referencia a un elemento empírico con propiedades) como un referente explícito del modelo. La norma internacional ISO-9001 define un proceso como "una actividad que utiliza recursos, y que se gestiona con el fin de permitir que los elementos de entrada se transformen en resultados" [9]. Creemos que en modelos reales, por ejemplo, en caso de procesos industriales o modelos en los cuales no exista un intercambio directo de información entre los procesos, la necesidad de contar con la especifiación de productos en el modelo, es tan necesario como las especificaciones de los procesos que los manipulan. De esta manera, el conjunto de productos del modelo representan una visión estructural del estado del sistema a través del tiempo, es decir, una especificación explícita de los estados del modelo previos y posteriores a la realización de uno o mas procesos que lo componen. Además, la especificación de los productos no sólo ayudan a describir condiciones mas complejas de composición de procesos, sino también, nos brindan la posibilidad de analizar el comportamiento de los mismos durante todo el proceso del sistema; por ejemplo la verificación de propiedades que deban reunir los productos al final de una serie de procesos.

Otra característica deseable de un lenguaje para el modelado de procesos de negocios es la posibilidad de contar con referencias al tiempo, es decir, poder especificar, en alguna medida temporal, restricciones o condiciones para los procesos o productos modelados. Como se puede observar en [10] pag. 3 y [16] Section 1.7 entre otros, es clara la necesidad de contar con la especificación de cotas temporales para los procesos. Esta información temporal, además de reflejar características del modelo real, nos brinda la posibilidad de analizar (verificar) condiciones (propiedades) expresadas en algúna lógica temporal, por ejemplo, si cierto producto es fabricado respentando límites de tiempo para garantizar la calidad de producción del cierto proceso industrial.

Product Process Modeling Language PPML [11] es un formalismo para el modelado de procesos de negocios con una semántica basada en sistemas de transición de estados temporizados [8]. Sus modelos se construyen mediante la especificación y composición de *procesos* y su interacción con los *productos* que manipulan. PPML ofrece una variedad de elementos, en particular la asignación de cotas temporales a los procesos, que lo hacen adecuado para la especificación

formal de procesos en los cuales existan restricciones temporales, sincronización entre componentes, concurrencia en las tareas, etc. Si bien PPML fué definido en sus comienzos con el objetivo de ser aplicado a la industria, no posee una sintaxis flexible y amigable para la definición de sus modelos.

Con el objetivo de ampliar el campo de aplicación y facilitar la especificación procesos de negocios con PPML, proponemos una sitaxis de alto nivel para el mismo. Con ella podemos brindar soporte automatizado para el análisis de sus modelos, como *parsers*, etc., y asistir mediantes entornos de especificación, la creación de los mismos. Además, como se describe en [15], podemos verificar propiedades termporales de sus modelos, expresadas en *Computational Tree Logic*, mediante la utilización de UPPAAL (model checker) [1] como herramienta de soporte.

El trabajo procede de la siguiente manera: Primeramente analizamos diferentes propuestas de formalizaciones de lenguajes y metodologías para la especificación de procesos de negocios. Luego presentamos una descripción de PPML y proponemos una sintaxis de alto nivel para el mismo, aplicándola en un caso de estudio que revela la necesidad de contar con productos como referentes explícitos del modelo, .

El caso de estudio abordado es un simple sistema de parquímetros para el estacionamiento medido. En él, los individuos que desean estacionar su vehículo deben tener un llavero magnético que registra su saldo disponible entre otra información. La operatoria del caso de estudio es la siguiente: cuando arribamos al estacionamiento deseado, dejamos el auto, colocamos el llavero junto al parquímetro y se inicia el alquiler del espacio, esto es, el parquímetro registra en el llavero los datos pertinentes, como la hora de inicio e identificador del parquímetro (este puede medir más de un espacio). Cuando nos retiramos, colocamos nuevamente el llavero junto al parquímetro y éste finaliza la transacción, descontando del saldo el importe producto del perío de tiempo por el costo del mismo.

1.1 Trabajos relacionados

Existen en la actualidad diversas propuestas de formalizaciones para lenguajes para el modelado de procesos de negocios con el fin de poder verificar formalmente propiedades de sus modelos. En [14] podemos encontrar un estudio detallado de diferentes trabajos para verificar formalmente procesos de negocios y una clasificación de los mismos según su soporte formal: *automatas*, *redes de Petri* y *álgebra de procesos*. Otro estudio de trabajos para el modelado y análisis de procesos de negocios es presentado en [5] [6], donde se muestra una traducción de modelos de servicios web en UML a FSP, para luego ser analizados usando la herramienta LTSA.

También existen trabajos con el objetivo de proveer de una semántica formal a BPMN (Business Process Modeling Notation), como por ejemplo los presentados en [17] [18]. Dicha semántica permite el análisis de compatibilidad entre los procesos de negocios a nivel de diseño.

En [12] una traducción semi-automática de BPD (Diagramas de procesos de negocios) a TLA+ es presentada. Con ella, se pueden verificar, mediante el TCL *model checker*, propiedades de los procesos de negocios expresadas en fórmulas TLA.

Como propuestas para la formalización y análisis de *workflow languages*, por ejemplo, para diagramas de actividades de UML, podemos citar [7], en la cual éstos diagramas son traducidos a PROMELA (lenguaje de especifiación de SPIN *model checker*). Además de éste, existen numerosos trabajos que proponen una semántica formal para *workflow languages* basados en redes de Petri y sus extensiones, como redes de Petri temporizadas.

2 Descripción de PPML

PPML es un formalismo para modelar procesos de negocios, está basado en el método descripto en [13]. El método provee tres tipos de construcciones básicas: *productos*, *procesos* y *puertas*. Los *productos* son entidades representadas por un conjunto de atributos. Los *procesos* son entidades que pueden manipular, transformar productos. Éstos son caracterizados por su comportamiento en el tiempo teniendo como restricción, sólo un producto de entrada y uno de salida. Como mecanismo adicional para la interconexión de procesos, PPML provee *puertas* que permiten empaquetar, desempaquetar o sincronizar productos.

2.1 Productos

Los *productos* capturan las características relevantes del modelo empírico, por ejemplo altura, peso, color, etc. Estas características pueden ser directamente observables o calculables a partir de otras mediante funciones, además son determinadas en base a una escala apropiada. Debido al requerimiento de poder empaquetar productos, necesitamos una clase de *productos compuestos*; éstos no representan ningún referente empírico, sólo es un mecanismo del método para tal fin. Otra clase de productos que proporciona PPML, son los *productos estructurados*, los cuales pueden ser *refinados* (especializaciones de otros productos) o *agregados* (contienen productos como sus componentes).

Definimos un *producto atómico* como una tupla $\langle codigo, nombre, tiempo, atributos, \langle \Sigma, A \rangle \rangle$ donde: *codigo* y *nombre* son constantes de los conjuntos $Codigo$ y $Nombre$ respectivamente, *tiempo* es una constante del conjunto $Time$, *atributos* es un conjunto de constantes de Σ. Los atributos pueden ser *directos* o *derivados*, éstos últimos obtienen su valor a partir de otros por medio de *axiomas* especificadoes en A. Llamamos a $\langle \Sigma, A \rangle$ la teoría de presentación del producto, que consta de la *signatura* Σ, que contiene todas las constantes definidas en el producto y A el conjunto de axiomas que rigen su comportamiento. Definimos una instancia de un producto como una interpretación en el sentido lógico de $\langle \Sigma, A \rangle$.

En caso de un *producto compuesto*, o bien, es un par $\langle codigo \otimes (P_{c1}, .., P_{cn}) \rangle$ donde $\otimes(P_{c1}, .., P_{cn})$ es una inyección de los componentes P_{ci} en el producto

cartesiano; o bien, un es par $\langle codigo, \iota(\mathcal{P})\rangle$ donde $\mathcal{P}$ es un conjunto finito de productos y $\iota(\mathcal{P}) \in \mathcal{P}$. Éste último utilizado en situaciones donde el producto de entrada es elegido en base a una condición.

Un *producto estructurado* puede ser *especializado* (se extiende uno definido agregando nuevas constantes, funciones y relaciones) o *agregado* (permite combinar varios productos como constituyentes de un nuevo producto). Los atributos de los productos agregados pueden ser atributos de sus constituyentes o nuevos atributos.

Como ejemplo, supongamos que queremos modelar parte del caso de estudio propuesto, en particular, el producto que formaliza un llavero magnético. El llavero contiene como propiedades: los posibles estados en los que se puede encontrar (Parked, Not-parked), el saldo (importe) disponible, la hora de comienzo de estacionamiento, el identificador del parquimetro en donde está estacionado y el costo por hora del mismo.

$$\langle Code, KeyChain, Time, [status, balance, start\_time, parkingID, cost\_hour], \langle \Sigma, A\rangle\rangle$$

Figura 1. Producto *KeyChain*

Donde $Code$ es un código para identificar a este producto, $KeyChain$ es el nombre del producto y $Time$ registra el momento en que fueron actualizados por última vez sus atributos. El conjunto de atributos $status, balance, start\_time$, $parkingID, cost\_hour$ representan las propiedades que deseamos modelar. En $\langle \Sigma, A\rangle$, A tiene como axioma $status \rightarrow balance > 0$, especificando la necesidad de contar con saldo positivo en el llavero para poder estacionar (especificamos el estado como dos posibles valores de *status*, "verdadero" está estacionado y "falso" no lo está).

2.2 Procesos

Un proceso modela un referente empírico que transforma un producto de entrada en uno de salida. Al igual que los productos, los procesos pueden ser **atómicos** (sin estructura interna o subprocesos) o **estructurados**. Los procesos modelan transformaciones, para ello, por cada proceso existe una *máquina virtual* que interpreta sus comandos básicos. Esta máquina virtual es un objeto con métodos, por ejemplo, asignaciónes a variables. Para especificar un proceso, definimos las transformaciones que éste realiza usando acciones básicas o combinaciones de estas por medio de estructuras de control. La formalización del concepto de *máquina virtual* [4] consiste en un *framework lógico* sobre sistemas de transición de estados llamado RETOOL [3].

La especificación de un proceso es una *transacción*, un segmento de computación de la máquina virtual subyacente. Consta de la *condición inicial* q (estados que satisfacen las precondiciones para llevar a cabo el proceso), la *condición final* p (estados que satisfacen la condición final del proceso), el *invariante* I (sentencia que debe evaluarse verdadera en todos los estados de la traza) y las *cotas inferior y superior* (l, u) que especifican el tiempo mínimo y máximo que puede

demorar el proceso en completarse. Formalmente, dada una secuencia de estados temporizados $\langle \sigma, T \rangle$, donde σ una traza posiblemente infinta de estados y T es una secuencia de tiempos correspondientes a σ, e i un estado de la misma (estado actual), especificamos un proceso con la fórmula $(q, I)_l \Delta^u p$, y su semántica: $\sigma, T, i \vDash (q, I)_l \Delta^u p$ si y sólo si para algún j, k tal que $i + 1 \leq j < k \leq i + u$, tenemos para cada $i \leq m \leq j$ que $\sigma, T, m \vDash q$; $\sigma, T, k \vDash p$, y para cada $j \leq n \leq k$ tenemos $\sigma, T, n \vDash I$.

Definimos un *proceso atómico* como un par $P = \langle proc, VM \rangle$, donde $VM = \langle \theta, \Theta \rangle$ y $proc = \langle codigo,\ nombre,\ P_{in}, P_{out}, (q, I)_l \Delta^u p \rangle$, donde *codigo* y *nombre* identifican y dan nombre al proceso respectivamente, además son fijos, es decir, no cambian durante las trazas de cómputo. P_{in}, P_{out} son los productos de entrada y salida, y $(q, I)_l \Delta^u p$ es la especificación del comportamiento del proceso.

Como ejemplo, se muestra el proceso estacionar (Fig.2), es decir, iniciar el alquiler el espacio de estacionamiento. Este proceso recibe como entrada el llavero electrónico y el parquímetro y devuelve el llavero con la modificaciones correspondientes: el estado ahora es *estacionado* (*Parked*), $parkingID$ contiene el identificador del parquímetro seleccionado y *cost_hour* el costo por hora del mismo.

$\langle Code, Start\_Panking, \otimes\langle KeyChain, ParkingMeter\rangle, KeyChain, (q, I)_l \Delta^u p \rangle$
$q : Keychain.status = NotParked \wedge Keychain.balance > 0.$
$p : Keychain.status = Parked \wedge Keychain.start\_time = ParkingMeter.current\_time$
$\wedge Keychain.parkingID = ParkingMeter.id.$ I : true. l : 2. u : 10.

Figura 2. Proceso que especifica el inicio de un alquiler de estacionamiento

La pre-condición exige que el llavero tenga un saldo positivo y que su estado indique la disponibilidad de estacionar. Como pos-condición el llavero queda afectado al estado de "estacionado" y se especifica en el la hora de comienzo de alquiler. Cabe destacar que, si bien el método provee acciones o variables controladas por el ambiente, en este caso, asumimos que todos los parquímetros tienen sincronizados un reloj interno indicando la hora actual. Las cotas l y u especifican que el parquímetro tardará almenos 2 unidades de tiempo en realizar la transacción y no demorará mas allá de 10 unidades de tiempo, (e.g. para iniciar el estacionamiento no podemos esperar, por parte del usuario, más de 10 segundos sin perder la paciencia).

2.3 Puertas

Usaremos el concepto de **puertas** como herramienta que provee el formalismo para interconectar procesos, i.e. para especificar situaciones complejas de conexión entre los procesos, por ejemplo, cuando un proceso manipula varios productos, para distribuir varios productos, o para *sincronizar* productos bajo cierta condición. Existen tres tipos básicos de puertas: *multiplexor*, *demultiplexor* y *semáforo*, cada uno de ellos con su interpretación esperada, gráficamente:

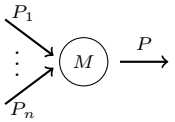

Figura 3. Multiplexor

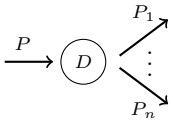

Figura 4. Demultiplexor

Figura 5. Semáforo

2.4 Composición de procesos

En el proceso de modelar sistemas, al igual que en programación, necesitamos herramientas para poder componer procesos. Para ello, PPML provee las siguientes estructuras de procesos: Sean $p1 = \langle p1_{code}, p1_{name}, p1_I, p1_O, (q_{p1}, I_{p1})_{l_{p1}} \Delta^{u_{p1}} p_{p1} \rangle$ y $p2 = \langle p2_{code}, p2_{name}, p2_I, p2_O, (q_{p2}, I_{p2})_{l_{p2}} \Delta^{u_{p2}} p_{p2} \rangle$ dos procesos,

- *Composición secuencial*: Denotada por $p_1; p_2$, combina dos procesos en un nuevo proceso $p = \langle code, name, p1_I, p2_O, (q_{p1}, I_p)_{l_{p1}} \Delta^{u_{p1}+u_{p2}} p_{p2} \rangle$. Donde *code* y *name* tienen nuevos valores para identificarlo y el invariante I_p está definida por $q_1 \mathcal{U}_{>l_{p1}} (I_{p1} \mathcal{U}_{\leq u_{p1}-l_{p1}} (p_{p1} \Rightarrow true\ \mathcal{U} (q_{p2} \mathcal{U}_{>l_{p2}} (I_{p2} \mathcal{U}_{\leq u_{p2}-l_{p2}} p_{p2}))))$.
- *Composición condicional*: Denotada por $p1;_s p2$, es análoga a la composición secuencial, pero con el invariante
  $q_{q1} \mathcal{U}_{>l_{p1}} (I_{p1} \mathcal{U}_{\leq u_{p1}-l_{p1}} (p_{p1} \Rightarrow true \mathcal{U} ((q_{p2} \wedge S_s) \mathcal{U}_{>l_{p2}} (I_{p2} \mathcal{U}_{\leq u_{p2}-l_{p2}} p_{p2}))))$,
  donde S_s es una expresión condicional que debe ser verdadera para que $p2$ pueda comenzar.
- *Composición Paralela*: Sean $m_O = \langle multiplexer\_code, \mathcal{P}_O, P_O, F_O \rangle$ y $d_I = \langle demultiplexer\_code, P_I, \mathcal{P}_I, F_I \rangle$ multiplexor y demultiplexor, cada uno definido sobre los productos de entrada y salida de $p1$ and $p2$ respectivamente. La composición paralela p de $p1$ y $p2$ respecto de m_O y d_I, denotada por $[p_1; p_2](d_I, m_O)$, está definida por: *codigo* y *nombre* análogos a las definiciones anteriores. La entrada de p es la del demultiplexor d_I, la salida es la del multiplexor m_O. La función de transferencia τ de p es $(q_{p1} \wedge q_{p2}, I_p)_{max((l_{p1}, l_{p2}))} \Delta^{max((u_{p1}, u_{p2}))} (p_{p1} \wedge p_{p2})$, donde el invariante I_p está definido por $(q_{p1} \wedge q_{p2}) \Rightarrow (\tau_1[(p_{p1} \mathcal{U} (p_{p1} \wedge p_{p2}))/p_{p1}] \wedge (\tau_2[(p_{p2} \mathcal{U} (p_{p1} \wedge p_{p2}))/p_{p2}]))$ (donde $\tau[(p \mathcal{U} q)/r]$ denota el reemplazo de la condición final r en τ por la condición $(p \mathcal{U} q)$). Esto especifica que cuando los procesos están habilitados para comenzar, realizan su trabajo en paralelo y luego de que uno de ellos termina, debe esperar que el segundo finalice su tarea.

3 Sintaxis y extensiones de PPML

En trabajos previos de PPML, todos los elementos que forman parte de la especificación de un modelo, son definidos formalmente. Con el objetivo de brindar un modo de especificar modelos PPML en un nivel menos riguroso, proponemos una sintaxis para los productos, procesos y puertas. Esta sintaxis tiene dos objetivos: proveer una forma más flexible y accesible de escribir especificaciones en PPML (como en otros lenguajes de modelado de procesos de negocios) y estandarizar el

lenguaje con el objetivo de poder construir herramientas para el lenguaje como parsers, analizadores, asistentes, etc.

Por cuestiones de espacio, mostraremos la sintaxis aplicándola al caso de estudio propuesto. Para los productos, la sintaxis se muestra en la Fig. 6.

```
Product KeyChain {
   boolean status,
   float balance,
   time start_time,
   int parkingID,
   float cost_hour
   axioms:  status imply ( balance > 0)
}
```

Figura 6. Producto *KeyChain*

Cabe destacar que si bien se proponen los tipos para cada atributo, la actual versión de traducción [15] para verificar propiedades de los modelos no contempla tipos como *float*. En caso de ser un producto estructurado, es decir, un producto definido a partir de otros como sus componentes, se declara el atributo con el nombre del producto como su tipo, por ejemplo, si el llavero tuviera como componente un *led* para indicar su saldo actual (color verde para saldo positivo y rojo para negativo), deberíamos agregar en la definición anterior "*led* indicator " junto con la especificación del producto *led*. Para modelar el comportamiento del mismo deberíamos agregar el axioma, " ((balance >= 0) and (indicator.color = 1)) or ((balance ¡ 0) and (indicator.color = 2)) ", donde "1" y "2" representa luz verde y roja respectivamente.

Cuando modelamos procesos de negocios existen situaciones en las cuales cierto producto es procesado en diferentes etapas o cuando alguno de sus componentes no están definidos en determinados momentos del sistema. En estas situaciones es necesario contar con valores nulos que permitan especificarlas.

Para describir la sintaxis de los procesos, especificaremos los procesos de comienzo y finalización del alquiler del estacionamiento. El primero de ellos (Fig.7) espera un producto compuesto que contiene el llavero y el parquímetro, como pre-condición requiere que el saldo sea mayor que cero y que su estado indique su disponibilidad para estacionar. Como resultado, devuelve el llavero actualizado, esto es, con el idenficador del parquímetro, la hora de comienzo y su estado como "estacionado". Los límites de tiempo indican que el proceso demorará almenos 2 unidades de tiempo (segundos) y como máximo 10 unidades.

El proceso de finalización (Fig.8), espera nuevamente como un producto compuesto, un llavero y un parquímetro. Requiere que el llavero indique un estado "estacionado" y que el identificador del parquímetro donde inició el alquiler sea igual a identificador del parquímetro de entrada. Como resultado devuelve el llavero descontando del saldo, el importe correspondiente al alquiler que finaliza y restaurando el estado del llavero para futuros alquileres. Cabe destacar que, si bien el saldo pudiera quedar negativo, inhabilitaría al mismo hasta que se le realice una recarga.

```
Process StartPanking {
  input: [KeyChain KeyC_in; ParkingMeter ParkingM_in]
  output: KeyChain KeyC_out
  invariant: true
  requires: not KeyC_in.status and KeyC_in.balance $>$ 0
  ensure: KeyC_out = KeyC_in and KeyC_in.status = true and
          KeyC_in.start_time = ParkingM_in.current_time and
  KeyC_in.parkingID = ParkingM_in.id
  l_time: 2 u_time: 10
}
```

Figura 7. Proceso que especifica el comienzo de un alquiler de estacionamiento

```
Process EndPanking {
  input: [KeyChain KeyC_in; ParkingMeter ParkingM_in]
  output: KeyChain KeyC_out
  invariant: true
  requires: KeyC_in.status and KeyC_in.parkingID = ParkingM_in.id
  ensure: KeyC_out = KeyC_in and KeyC_in.status = false and
          KeyC_in.balance = KeyC_in.balance - (ParkingM_in.cost_hour *
  (ParkingM_in.current_time - KeyC_in.start_time) / 60)
  l_time: 2  u_time: 10
}
```

Figura 8. Proceso que especifica la finalización del alquiler de un estacionamiento

Como podemos observar, la finalización del alquiler deber realizarse sobre el mismo parquímetro donde se comenzó la misma. Esta restricción podría flexibilizarce permitiendo que las personas que olvidan finalizar el alquiler y se retiran, puedan cerrar la operación en otro parquímetro. Para ello, el proceso *StartParking* debería modificar el atributo "cost_hour" del llavero, así cuando se cierra el alquiler en otro parquímetro, éste puede descontar correctamente el costo.

4 Conclusiones y Trabajos Futuros

Hemos analizado exhaustivamente el estado del arte del área de métodos formales que persigue como objetivo brindar soporte formal a la especificación de procesos de negocios, particularmente notamos que la gran mayoría de lenguajes para tal fin, no provee elementos dentro del lenguaje que permita modelar productos (entidades del modelo). Por ello, retomamos un formalismo para la especifiación de procesos de negocios y lo adaptamos para facilitar y extender su aplicación. Focalizamos nuestro interés en investigar las ventajas de contar con "productos" como elementos lenguaje y para ello mostramos un caso de estudio que revela dicha necesidad. Actualmente estamos trabajando en un prototipo de entorno para asistir la creación de modelos PPML, para luego integrar en él diferentes herramientas de análisis para los mismos. Por otra parte, estamos

investigando la posibilidad de dotar a PPML con transacciones, es decir, poder especificar en sus modelos, que diferentes procesos que realizan cambios en los productos que manipulan, o bien, se concretan los exitosamente, o bien los productos intervinientes deben quedar en su estado inicial, previo al primero de dichos procesos. Para ello se están estudianto diferentes técnicas formales para su tratamiento como por ejemplo *compensaciones* [2].

Referencias

1. J. Bengtsson, et. al. . *UPPAAL- a tool suite for the automatic verification of real-time systems*. In Proc. of Hybrid Systems III. LNCS 1066. pp. 32-243. 1996.
2. R. Bruni, H. C. Melgratti, U. Montanari. *Theoretical foundations for compensations in flow composition languages*. POPL, pp.209-220, 2005.
3. S. Carvalho, J. Fiadeiro, E. Haeusler. *A Formal Approach to Real-Time Object Oriented Software*. In: Proceedings of the Workshop on Real-Time Programming. IFAP/IFIP, 1997, pp91-96, Lyon, France.
4. J. Fiadeiro, T. Maibaum, *Temporal Theories as Modularisation Units for Concurrent System Specification*. Formal Aspects of Computing. 1992, 4(3):239-272
5. H. Foster, et al. . *LTSA-WS: A Tool for Model-based Verification of Web Service Compositions and Choreography*. ICSE 2006, pp. 771-774, 2006.
6. H. Foster, et al. . *WS-Engineer: A Model-Based Approach to Engineering Web Service Compositions and Choreography*. Test and Analysis of Web Services, pp. 87-119, 2007.
7. N. Guelfi, A. Mammar. *A Formal Semantics of Timed Activity Diagrams and its PROMELA Translation*, APSEC, pp. 283-290, 2005.
8. T. A. Henzinger, Z. Manna, A. Pnueli, *Timed Transition Systems*,1996.
9. Holger Hinrichs and Thomas Aden, *An ISO 9001:2000 Compliant Quality Management System for Data Integration in Data Warehouse Systems*, In Proc. DMDW'01, 2001.
10. J. Koehler, G. Tirenni, S. Kumaran, *From Business Process Model to Consistent Implementation: A Case for Formal Verification Methods*. 6th International Enterprise Distributed Object Computing Conference (EDOC 2002), EEE Computer Society, pp. 96, 2002.
11. T. S. E. Maibaum, *An Overview of The Mensurae Language: Specifying Business Processes*, Rigorous Object-Oriented Methods, BCS, 2000.
12. C. Masalagiu, et al. . *A Rigorous Methodology for Specification and Verification of Business Processes*. Formal Aspects of Computing, Springer London, 2009.
13. M. Myers, A. Kaposi, *A First Systems Book: Technology and Management*, ISBN 978-1860944321, Imperial College Press, 2 edition, 2004.
14. S. Morimoto, *A Survey of Formal Verification for Business Process Modeling*, ICCS (2), pp.514-522, 2008
15. G. Regis, N. Aguirre, *Verificación de propiedades en PPML*. Anales del XIV Congreso Argentino de Ciencias de la Computación, CACIC 2008. Universidad Nacional de Chilecito, Chilecito, La Rioja, Argentina. Octubre 2008
16. W3C, *Web Service Choreography Interface (WSCI)* 1.0. http://www.w3.org/TR/wsci, 2002.
17. P. Y. H. Wong, J. Gibbons.*A Process Semantics for BPMN*. ICFEM 2008, pp. 355-374, 2008.
18. P. Y. H. Wong, J. Gibbons. *Property Specifications for Workflow Modelling*, IFM 2009, pp. 56-71, 2009.