Una propuesta usando Restricciones para la toma de decisiones en la tolerancia a fallos en procesos de negocio

M. T Gómez-López and Rafael Martínez Gasca and Diana Borrego Núñez

Departamento de Lenguajes y Sistemas Informáticos, Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla, España,
{maytegomez, gasca, dianabn}@us.es

Resumen Cuando un proceso de negocio no obtiene el objetivo que se propone será necesario realizar una diagnosis y detección de errores, descubriendo qué servicios están funcionando de forma incorrecta para su sustitución. El objetivo de este artículo es describir los pasos necesarios para buscar otro servicio o actividad que pueda sustituir al que está fallando de una forma eficiente. Para automatizar la búsqueda y sustitución de servicios, es posible describir la funcionalidad de dichos procesos mediante restricciones, lo cual facilitaría la identificación de sus posibles sustitutos. También se analiza en este artículo cómo adaptar el traspaso de información que se realiza entre los servicios mediante mensajes en XML, a la descripción del comportamiento de dichos servicios mediante restricciones.

1. Introduction

Los procesos de negocio son un conjunto de actividades o servicios relacionados mediante un flujo de trabajo y/o datos para obtener un objetivo definido. Cuando un proceso de negocio se somete a una monitorización y diagnosis, pueden ser detectados errores en algunas de sus actividades. En este trabajo nos centraremos en los procesos de negocio que están formados por un conjunto de servicios web, donde uno de ellos falla para cualquier instancia de ejecución, o su funcionamiento es defectuoso.

La diagnosis de fallos ha sido una cuestión analizada en otros trabajos [1], ya que la conversión de las técnicas de inteligencia artificial [2][3] al modelado mediante procesos de negocio no son automáticas. Uno de los mayores problemas de la diagnosis de procesos viene derivada de que el modelo está distribuido y no se tiene una visión ni conocimiento global de su comportamiento.

Más relacionado con la diagnosis de procesos de negocio existen trabajos relacionados con la detección de conflictos (CDM - Conflict Detecting Mechanism), donde la especificación de los servicios descrita en XML se utiliza para diseñar meta-procesos [4] para la detección de inconsistencias en las actividades. En el ámbito de CDM, los conflictos se definen como la causa que viola y modifica el comportamiento normal o esperado en un determinado estado de la ejecución

del proceso de negocio. Para la detección de estos errores, el área de procesos de negocio también se ayuda de la monitorización de procesos (BPMod-Business Process Monitoring), que es un lenguaje de consulta para la monitorización propuesto en [5].

Cuando los procesos de negocios están formados por servicios web, es muy utilizado el estándar BPEL (Business Process Execution Language [6]), que provee de un lenguaje basado en XML, el cual describe tanto la interfaz del proceso como sus operaciones lógicas y su línea de ejecución. La utilización de este lenguaje también ha sido llevaba al campo de la tolerancia a fallos [7].

Una vez detectado el servicio que ha fallado, será necesario decidir qué tareas se tendrán que llevar a cabo dependiendo de qué servicio está fallando. Para sustituir un servicio, se tendrá que encontrar otro que incluya su especificación y preferiblemente en el menor tiempo posible. Para automatizar este proceso, es necesario conocer y tener especificado de una manera formal su comportamiento, junto a un repositorio de posibles servicios sustitutos. Una forma de describir dicho funcionamiento es mediante restricciones, que permiten la formalización de contratos y existen técnicas para optimizar la búsqueda de servicios consistentes con otros. La utilización de restricciones facilitaría la toma de decisiones ante la búsqueda de un sustituto, ya que existen técnicas desarrolladas en este ámbito [8][9] para optimizar la localización de restricciones mediante consultas a una Base de Datos de Restricciones.

La búsqueda de un nuevo servicio o un conjunto de ellos en una base de datos conlleva tres problemas fundamentales a solucionar:

- Representación de los servicios mediante un lenguaje de especificación cercano a las restricciones que pueda describir su comportamiento.
- Almacenamiento de dicha descripción en una base de datos para hacer la información persistente y facilitar una búsqueda eficiente.
- Búsqueda eficiente del mejor servicio, o combinación de ellos, para sustituir el detectado como incorrecto.

A lo largo de este artículo, vamos a analizar estas tres tareas y cómo mejorarlas para que la combinación de ellas ayuden a la toma de decisiones en la orquestación de los procesos ante la detección de errores.

2. Representación de servicios web mediante restricciones

El estándar BPEL (Business Process Execution Language [6]) permite describir tanto la interfaz del proceso como sus operaciones lógica y su línea de ejecución. La notación que describe BPEL se basa en la descripción del comportamiento específico de procesos basados en servicios web, que se puede encontrar en la bibliografía como BPEL4WS. Los procesos en BPEL4WS exportan e importan funcionalidad usando exclusivamente las interfaces de los servicios web. Los procesos de negocio se pueden describir de dos formas:

- Modelos de procesos de negocio ejecutables con un determinado comportamiento.

- Protocolos de negocio, usando descripción de negocios que especifican el intercambio mutuo de mensajes sin revelar su comportamiento interno. La descripción de los protocolos de negocio se denomina descripción de *procesos abstractos*.

El lenguaje BPEL4WS se puede usar para modelar tanto los procesos ejecutables como los abstractos, definiendo un lenguaje tanto para la especificación formal como para el intercambio de mensajes para un protocolo determinado.

Las siguientes especificaciones describen el espacio de los Servicios Web: *SOAP*, *Web Services Description Language (WSDL)*, y *Universal Description, Discovery, and Integration (UDDI)*. SOAP define un protocolo de mensajería en XML para la interacción entre servicios básicos. WSDL introduce una gramática para la descripción de servicios. UDDI provee de la infraestructura necesaria para publicar y descubrir servicios de una forma sistemática. Estas tres especificaciones combinadas permiten una funcionalidad completa en los servicios web con un modelo independiente de la plataforma.

Cuando en este artículo se hace referencia al término *Restricción*, nos referimos a un conjunto de variables definidas sobre un dominio y relacionadas entre sí. Donde dicha relación entre variables se puede describir de una manera compacta mediante una combinación con operadores lógicos de ecuaciones e inecuaciones.

La implementación M de un servicio en un determinado proceso tiene que satisfacer un Contrato C descrito mediante una precondición y una postcondición, lo que se puede representar mediante restricciones que se deben cumplir al terminar el servicio. También hay que tener en cuenta que dicho servicio tendrá un conjunto de variables de conexión que serán públicas y estarán relacionadas con otros servicios mediante puertos, mientras que otro conjunto de variables serán privadas. Si M satisface el contrato C y están definidas sobre los mismos puertos, se dirá que $M \subseteq C$. Para sustituir M por otro servicio, será necesario encontrar otra restricción M' que está definida sobre los mismos puertos y que también satisfaga el aserto $M' \subseteq C$.

Para mostrar un ejemplo de representación de servicios web con restricciones utilizaremos el mostrado en la figura 1. Este ejemplo utiliza un conjunto de servicios que representan la obtención de ganancias haciendo distintos tipos de inversiones. Partiendo de una cantidad determinada se utilizará un servicio (*División Capital en distintas Estrategias*) que desglosará la cantidad disponible en distintas partidas. Las distintas estrategias son: *Inversión en Bolsa*, *Inversión Inmobiliaria*, e *Inversión en empresas de I+D+I*. Cada una de estos servicios obtendrá unos beneficios en función de la cantidad invertida y un margen de inversión a dicha cantidad. Esto significa que cada servicio recibirá un par de valores <inversión, margen>, de forma que la inversión total en cada uno de los servicios será de inversión±margen. Como variable de salida del servicio se obtendrá una gananciaTotal. Si por ejemplo tras un proceso de diagnosis se detecta que el servicio *Inversión en Bolsa* no funciona de manera correcta, será necesario buscar un nuevo servicio que haga las veces de éste. Para esto es necesario tener la descripción de dicho servicio, de forma que sea posible la sustitución por otro. Un ejemplo de la descripción de la tarea *Inversión en Bolsa* es la sigu-

iente, donde se describen el precio de cada una de las acciones disponible en este servicio PA_1, ..., PA_n, la cantidad disponibles de cada una de ellas C_1, ..., C_n, y el porcentaje de ganancia tras un determinado tiempo en cada una de ellas $PG_1 \ldots PG_n$. Toda esta información será obtenida por el servicio de forma independiente. Para una cantidad de inversión dada y un determinado margen, se comprarán un conjunto de acciones, tantas de cada tipo como se representan con las variables N_1, ..., N_n. La descripción del comportamiento del servicio será:

$$N_1{*}PG_1 + N_2{*}PG_2 + \ldots N_n{*}PG_n = \mathsf{gananciaTotal}$$
$$\wedge$$
$$PA_1{*}N_1 + PA_2{*}N_2 + \ldots PA_n{*}N_n \leq \mathsf{inversión} + \mathsf{margen}$$
$$\wedge$$
$$PA_1{*}N_1 + PA_2{*}N_2 + \ldots PA_n{*}N_n \geq \mathsf{inversión} - \mathsf{margen}$$
$$\wedge N_1 \leq C_1 \wedge N_2 \leq C_2 \wedge \ldots \wedge N_n \leq C_n$$

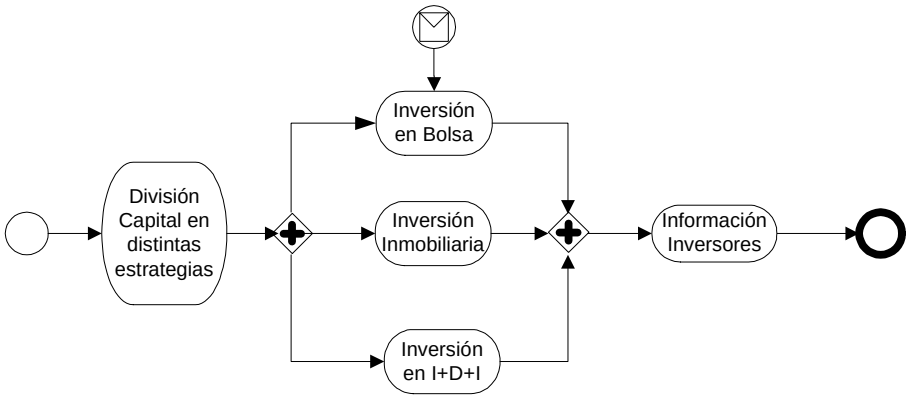

Figura 1. Ejemplo de Proceso de negocio

La restricción que describe el funcionamiento del servicio *Inversión en Bolsa* define la postcondición de dicho servicio, mientras que la precondición también deberá ser representada mediante una restricción.

Como el intercambio de información entre los servicios web se realiza mediante un protocolo de mensajería, la información que se intercambia en dichos mensajes marcará qué variables son conocidas entre los diferentes procesos y cuáles son privadas. En el ejemplo de la inversión en bolsa, existen dos tipos de mensajes, uno de recepción de datos y otro de envío de resultados. En este caso son las variables inversión, margen y gananciaTotal, y los mensajes serán de la forma:

```
<message name="DatosInversionBolsa">
  <part name="inversion"type="xsd:integer"/ >
  <part name="margen"type="xsd:integer"/ >
< /message>

<message name="SalidaInversionBolsa">
  <part name="gananciaTotal"type="xsd:integer"/ >
< /message>
```

3. Almacenamiento de especificaciones de servicios web en bases de datos

Cuando un sistema combina diferentes servicios en un proceso para obtener un objetivo, es común que cada una de las fases de dicha producción trabaje con un conjunto de datos almacenados en diferentes estructuras de almacenamiento y de diferente naturaleza. Como se ha analizado en la sección anterior, el comportamiento de los servicios puede ser mapeado a restricciones, por lo que la forma natural de almacenarlas será en una Base de Datos de Restricciones (CDB - Constraint Database).

Las Bases de Datos de Restricciones tuvieron su origen a principios de 1990 con el artículo de Kanellakis, Kuper y Revesz [10], ampliándose más tarde en los trabajos [11][12]. Originalmente, el objetivo de las CDBs fue definir una versión de bases de datos a la que se le añadió funcionalidad relacionada con la Programación Lógica con Restricciones (Constraint Logic Programming - CLP).

Las CDBs fueron definidas en base a la lógica y modelo de primer orden, tanto para la descripción de restricciones como para la evaluación de las consultas. El fundamento de las CDBs se apoya en la dualidad de representar un conjunto de datos mediante una restricción (fórmula) sobre un conjunto de variables libres x_1, ..., x_m, junto a la utilización de atributos clásicos del álgebra relacional a_1, ..., a_n que sólo pueden tomar un valor en cada tupla. Esta ampliación de las bases de datos de restricciones, desarrollada más ampliamente en [9], permite almacenar y consultar las restricciones como un dato cualquiera como los enteros, cadenas...

- Una *k-tupla restricción* con las variables $x_1, \ldots, x_k$ es una conjunción finita de la forma $\varphi_1 \wedge \ldots \wedge \varphi_N$ donde cada φ_i, para $1 \leq i \leq N$, es un *Atributo Clásico o Univaluado* o un *Atributo Restricción*. Un atributo univaluado es de la forma $\{x_j$=*Constante*$\}$ donde $x_j \in \{x_1, \ldots, x_k\}$. Un atributo restricción es una relación representada mediante una restricción entre las variables $x_1, \ldots, x_k$ no pertenecientes a un atributo univaluado.
- Una *relación restrictiva* está definida como un conjunto de *Atributos Univaluados* y un conjunto de *Atributos Restricción*. Una *relación restrictiva de aridad k*, es un conjunto finito $r = \{\psi_1, \ldots, \psi_M\}$, donde cada ψ_j para $1 \leq j \leq M$ es una k-tupla restricción sobre las variables $x_1, \ldots, x_k$ tal como se muestra en la figura 2. La fórmula correspondiente es la disjunción de

la forma $\psi_1 \vee \ldots \vee \psi_M$, tal que $\psi_j = \varphi_1 \wedge \ldots \wedge \varphi_N$ donde cada φ_i para $1 \leq i \leq N$ es una *k-tupla restricción*. Si existe en cada $\psi_j \in r$ una φ_i de la forma $\{x{=}Constante\}$, donde x sea la misma variable en todos los φ_i pertenecientes a distintos ψ_j y x no aparece en el resto de los φ_i de una misma ψ_j, se dirá que x es un **atributo univaluado**, mientas que el resto de variables formarán parte en uno o varios **Atributos Restricción**.

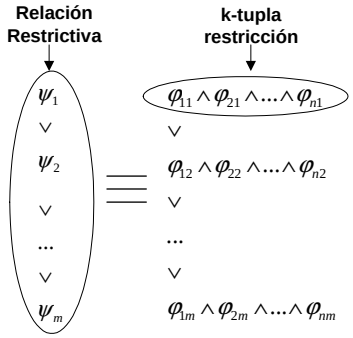

Figura 2. Representación de k-tuplas restricción y relaciones restrictivas

- Por lo tanto, una *Base de Datos de Restricciones* es una colección finita de relaciones restrictivas formadas por **atributos univaluados** y **atributos restricción**.

La idea es que se tendrían todos los servicios definidos en una CDB mediante sus correspondientes contratos, precondición y postcondición, y cuando un servicio es detectado como erróneo, se buscará en la o las bases de datos de restricciones pertinentes qué nuevo servicio podría sustituirlo porque tenga la misma especificación.

Esta dualidad permite almacenar los datos que describen el comportamiento de los servicios web como restricciones, y sus variables públicas y puertos como atributos clásicos. Un ejemplo de representación de los servicios descritos en la figura 1 es el mostrado en la figura 3. En este ejemplo aparece dos atributo clásico que serán el identificador del servicio web (IDSW) y la descripción textual (Descripción). Por otra parte también existen dos atributos restricción que definen el contrato de cada servicio, la precondición y la postcondición.

Para conectar los distintos servicios, se utilizarán las variables definidas como públicas, al representar la información mediante restricciones es posible realizar operaciones de proyección sobre ellas para modificar sus variables. Esto haría posible que un mismo servicio mostrara diferentes precondiciones y postcondiciones en función de las variables que hiciera públicas en cada momento, o dependiendo del servicio con el que se relacionara.

IDSW	Precondición	Postcondición	Descripción
2354	margen≥0∧inversión≥0	*N1*PG1 + N2*PG2 +...Nn*PGn* = gananciaTotal ∧...	Inversión en bonos
3487	1.000.000≥inversión∧inversión≥120.000	Inversión*1,8-2.500=gananciaTotal	Inversión Inmobiliaria
5561	18.000≥inversión∧inversión≥0 ∧700≥gastoFijo≥500	Inversion*1,4-gastoFijo=gananciaTotal	Inversión en I+D+I
...	...	...	...

Figura 3. Ejemplo de tuplas con Servicios Web

Para que las consultas a la CDB sean más eficientes, cuando se crea una Base de Datos de Restricciones se crean también tres tablas (*Restricciones*, *Variables* y *Restricciones/Variables*) mostradas en la figura 4, que mantienen las relaciones entre las variables y las restricciones. Estas tablas disminuyen la complejidad computacional de las consultas ya que con una sola consulta se puede conocer qué restricciones tienen qué variables y viceversa. Estas tablas permiten identificar cada restricción (tabla *Restricciones*), cada variable (tabla *Variables*) y establecer la relación entre ambas (tabla *Restricciones/Variables*), lo que ayuda a detectar las restricciones relacionadas con una consulta sin necesidad de tratarlas todas. Estas tablas no son visibles ni accesibles por el usuario de una forma directa, pero sí de una manera implícita cuando se añaden, modifican, borran o consultan restricciones de la base de datos. La tabla *Restricciones* almacena el *idRestricción* que corresponde al identificador del objeto (OID) generado por el sistema, y la *Etiqueta* acorde con el tipo de restricción que representa (Polinómica, lineal, de igualdad o inecuación). La tabla *Variables* almacena, para cada variable, su identificador, su nombre y el dominio (Natural, Entero o Flotante) al que pertenece. La tabla *Restricciones/Variables* almacena la relación entre las variables y las restricciones, y sobre qué rango (*Rango_Inicio*, *Rango_Fin*) están definidas cada una de las variables para cada una de las restricciones en las que participan.

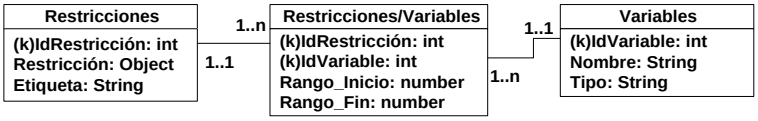

Figura 4. Indexación entre variables y restricciones

4. Búsqueda eficiente de Servicios Web sustitutos

En la bibliografía hay trabajos que se han centrado en las consultas relacionadas con servicios web [13], [14]. Más concretamente, La propuesta de este trabajo se centra en buscar de forma eficiente servicios que puedan sustituir a

uno dado, para ello debemos tener en cuenta como ya dijimos en la sección 2 que una implementación M de un servicio de un proceso satisface un contrato C si satisface la postcondición, sujeto a una precondición dada.

Sean $\mathtt{S}_W\mathtt{(Pre)}$ y $\mathtt{S}_W\mathtt{(Post)}$ la representación de la precondición y la postcondición del contrato correspondiente a un servicio $\mathtt{S}_W$ que implementa una actividad de un proceso de negocio. Para poder obtener otro servicio S_b que pueda sustituir el que trabaja de forma incorrecta S_a, será necesario encontrar otro servicio que cumpla las siguientes asertos:

- $S_b\mathtt{(Pre)} \supseteq S_a\mathtt{(Pre)}$. Lo que significa que todos los posibles valores de entrada del servicio S_a están recogidos en el nuevo servicio S_b.
- $S_b\mathtt{(Post)} \subseteq S_a\mathtt{(Post)}$. Lo que significa que los valores de salida que devuelva el servicio S_b son equivalentes o están incluidos en los valores de salida que devolvía el servicio S_a.

Nuestra propuesta define la implementación de esta comprobación mediante la construcción de Problemas de Satisfacción de Restricciones, más concretamente con el operador de Selección de las CBDs.

Un problema de satisfacción de restricciones (Constraint Satisfaction Problem - CSP) es una representación de un conjunto de variables, dominios y restricciones, ligado a uno o varios esquema de razonamiento para resolverlo. Formalmente, está definido por la tripleta <X, D, C> donde, X = $\{x_1, x_2, \ldots, x_n\}$ es un conjunto finito de variables, D = $\{d(x_1), d(x_2), \ldots, d(x_n)\}$ es el conjunto de dominios de cada una de las variables, y C = $\{\mathrm{C}_1, \mathrm{C}_2, \ldots, \mathrm{C}_m\}$ un conjunto de restricciones. Cada restricción C_i está definida como una relación R de un conjunto de variables V = $\{x_i, x_j, \ldots, x_k\}$ a lo que se denomina *ámbito de la restricción*.

Para hacer más eficiente la búsqueda de servicios, la implementación de CDBs que se propone utilizar es la descrita en [15]. Esta propuesta busca la eficiencia, por lo que intenta evitar la construcción y resolución de CSPs en la medida de lo posible. Para ello hacemos un análisis de los rangos de las variables, que se almacenarán tal como se ha descrito en la sección anterior. Existen casos donde el análisis de dichos rangos evita la construcción de CSPs. Analizando el mínimo y máximo valor que puede tomar una variable, es posible inferir si las restricciones involucradas en la comparación cumplen o no una condición. Por ejemplo si se comparan dos restricciones C_x y C_y, ambas definidas sobre las variables v_1 e v_2, donde los rangos son $C_x(v_1{:}[5..15], v_2{:}[20..30])$ y $C_y(v_1{:}[20..25], v_2{:}[40..55])$. En este caso, el predicado fuera $C_x \subseteq C_y$ seguro que no se cumple, y el servicio no serviría como sustituto. Volviendo al ejemplo anterior, si los dominios de las variables fueran $C_x(v_1{:}[5..15], v_2{:}[20..30])$ y $C_y(v_1{:}[2..25], v_2{:}[10..55])$, y el predicado $C_x \subseteq C_y$, pese a que los dominios de las variables en C_x están incluidas en los dominio de las variables en C_y, no se puede asegurar que se cumple el predicado para las restricciones. Un ejemplo donde los rangos de las variables están incluidos dentro de otra, pero no sus soluciones, es el mostrado en la figura 5.

Para cada una de las variables de las restricciones de los servicios que se quieren analizar, existen diferentes posibilidades entre los rangos de las variables,

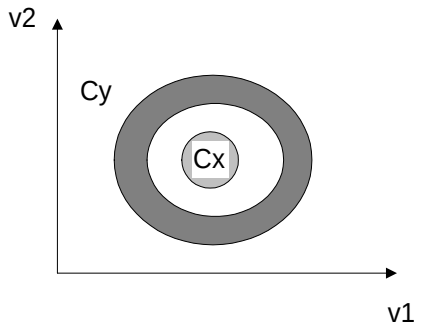

Figura 5. Ejemplo de $C_x \nsubseteq C_y$

mostradas en la figura 6. En función de cada una de las relaciones entre los rangos y la resolución de CSP, se puede concluir que:

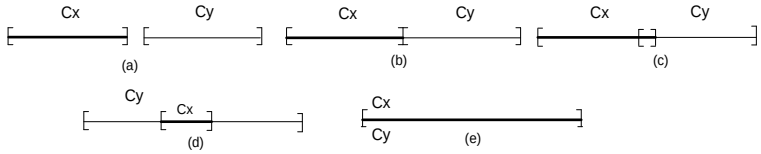

Figura 6. Tipos de relaciones entre los rangos de las variables de dos restricciones

- (a): $C_x \subseteq C_y$ es falso
- (b): $C_x \subseteq C_y$ es falso
- (c): $C_x \subseteq C_y$ es falso
- (d): $C_x \subseteq C_y$ es necesario crear un CSP para saber si es cierto o falso
- (a): $C_x \subseteq C_y$ es necesario crear un CSP para saber si es cierto o falso

La creación del CSP se basa en que $C_x \subseteq C_y$ será cierto si todas las soluciones de C_x lo son también de C_y. Para la inclusión, es necesario que las restricciones estén definidas sobre las mismas variables, de forma que sean C_x y C_y dos restricciones donde $X = \{x_1, x_2, \ldots, x_n\}$ son las variables de C_x y C_y, $C_x \subseteq C_y$ es igual que la implicación $(C_x \rightarrow C_y)$ [16]. Esta operación determina si todas las soluciones de C_x son también soluciones de C_y, aunque es posible que C_y tenga soluciones que no pertenezcan a C_x.

Para evitar analizar todas las soluciones de C_x, comprobando que éstas también lo son de C_y, se buscará una solución donde esto no ocurra, de forma que la evaluación de la condición $C_x \subseteq C_y$ corresponde con la fórmula:

$$\neg(\exists_{x_i \in X}(C_x \ \wedge \ \neg C_y))$$

Y el CSP que se creará es:

$$X = \{x_1, x_2, \ldots, x_n\}$$
$$C_x \wedge \neg C_y$$

Si se encuentra alguna solución para el CSP devolverá *falso*, y *cierto* si no encuentra ninguna.

Cuando no se encuentra un único servicio, será necesario buscar una combinación de varios que puedan definir un nuevo servicio. Para la composición de servicios existen trabajos previos relacionados [17], [18], [19].

5. Conclusions and Future Work

Este artículo presenta los pasos necesarios para la búsqueda eficiente de servicios que implementen las actividades de un proceso de negocio que permita su sustitución cuando éstas fallen. El uso de CDBs permite crear indexación entre las restricciones y las variables, para que la evaluación de la selección sea lo más eficiente posible.

Como trabajo futuro se quieren analizar el resto de operaciones de servicios que se describen en el lenguaje BPEL, tanto las básicas como las estructurales, dando un análisis completo de todas las posibilidades.

También se plantea como trabajo futuro la búsqueda de restricciones para combinar distintos servicios web y que se pueda obtener servicios más eficientes. Esto implicará cómo saber si pueden sustituir un servicio determinado, cuál es el mejor y cuál es la opción más eficiente en caso de que existieran varios servicios o combinación de ellos para sustituir el erróneo. Esto puede ser analizado con el operador de proyección de las Bases de Datos de Restricciones.

6. Agradecimientos

Este trabajo ha sido parcialmente subvencionado por la Junta de Andalucía, mediante la Conserjería de Innovación, Ciencia y Empresas (P08-TIC-04095) y por el Ministerio de Ciencia y Tecnología de España (DPI2006-15476-C02-01) y las ayudas Europeas al desarrollo (Fondo FEDER).

Referencias

1. Stefano Bocconi, Claudia Picardi, Xavier Pucel, Daniele Theseider Dupré, and Louise Travé-Massuyès. Model-based diagnosability analysis for web services. In *AI*IA '07: Proceedings of the 10th Congress of the Italian Association for Artificial Intelligence on AI*IA 2007*, pages 24–35, Berlin, Heidelberg, 2007. Springer-Verlag.

2. R. Reiter. A theory of diagnosis from first principles. volume 1, pages 57–96, 1987.
3. R. Greiner, B. A. Smith, and R. W. Wilkerson. A correction to the algorithm in reiter's theory of diagnosis. volume 41, pages 79–88. Elsevier Science Publishers Ltd, 1989.
4. Shi-Ming Huang, Yuang-Te Chu, Shing-Han Li, and David C. Yen. Enhancing conflict detecting mechanism for web services composition: A business process flow model transformation approach. *Inf. Softw. Technol.*, 50(11):1069–1087, 2008.
5. Catriel Beeri, Anat Eyal, Tova Milo, and Alon Pilberg. Monitoring business processes with queries. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 603–614. VLDB Endowment, 2007.
6. Business Process Execution Language for Web Services. http://www.ibm.com/developerworks/library/ws-bpel/, 2003.
7. Glen Dobson. Using ws-bpel to implement software fault tolerance for web services. In *EUROMICRO '06: Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 126–133, Washington, DC, USA, 2006. IEEE Computer Society.
8. María Teresa Gómez-López, Rafael M. Gasca, Carmelo Del Valle, and F. Fernando de la Rosa T. Querying a polynomial constraint object-relational database in model-based diagnosis. In *International Conference on Database and Expert Systems Applications-DEXA*, volume 3588, pages 848–857. Lecture Notes in Computer Science, Springer, 2005.
9. María Teresa Gómez López, Rafael Ceballos, Rafael M. Gasca, and Carmelo Del Valle. Developing a labelled object-relational constraint database architecture for the projection operator. *Data Knowl. Eng.*, 68(1):146–172, 2009.
10. G. M. Kuper P. C. Kanellakis and P. Z. Revesz. Constraint query languages. *Symposium on Principles of Database Systems*, pages 299–313, 1990.
11. G. Kuper, L. Libkin, and J. Paredaes. *"Introduction", Constraint Databases, G. Kuper and L. Libkin and J. Paredaes.* Springer, 2000.
12. P. Revesz. Datalog and Constraints. *Constraint Databases*, pages 155–170, 2000.
13. D. Martin S. Narayanan R. Waldinger G. Denker, J. Hobbs. Querying and accessing information on the semantic web. In *Proc. Semantic Web Workshop, in conjunction with 10th international Worldwide Web Conference*, 2001.
14. Catriel Beeri, Anat Eyal, Simon Kamenkovich, and Tova Milo. Querying business processes. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 343–354. VLDB Endowment, 2006.
15. María Teresa Gómez López. Lorcdb: Gestor de bases de datos objeto-relacionales de restricciones. Ph D thesis, 2007.
16. Kim Marriott and Peter J. Stuckey. *"Programming with Constraints. An introduction", Simplification, Optimization and Implication.* The Mitt Press, 1998.
17. Singh M.P. Cheng, Z. and M.A. Vouk. Composition constraints for semantic web services. In *Proceedings of WWW2002 Workshop on Real World RDF and Semantic Web Application*, 2003.
18. Srini Narayanan and Sheila A. McIlraith. Simulation, verification and automated composition of web services. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 77–88, New York, NY, USA, 2002. ACM.
19. Rajesh Thiagarajan and Markus Stumptner. Service composition with consistency-based matchmaking: A csp-based approach. In *ECOWS '07: Proceedings of the Fifth European Conference on Web Services*, pages 23–32, Washington, DC, USA, 2007. IEEE Computer Society.