Implementación de la comunicación entre el Workflow y las Aplicaciones Externas con Servicios Web y Reglas de Transformación de Grafos

Daniel Riesco
Universidad Nacional de San Luis - Dpto. Informática-Te: +54(0)2652–424027
driesco@unsl.edu.ar

Paola Martellotto **Marcela Daniele**
Universidad Nacional de Río Cuarto - Facultad de Ciencias Exactas, Fco-Qcas y Naturales - Dpto. de Computación - Te/Fax: + 54 (0) 358 – 4676235 - {paola, marcela}@dc.exa.unrc.edu.ar

Abstract

A process imposes a specified order in the work activities throughout the space and the time, with a principle, an end and entrances and exits clearly specified. The main objective of the organizations is to become more agile, more competent and adapt quickly to the continuous changing market in which they operate. At present, the automation of business processes is a central objective for many organizations. From the Workflow Reference Model, proposed by the Workflow Management Coalition (WfMC), web services and rules transformation graphs, an efficient solution is obtained to optimize the invocation of external applications that are more suitable according to the requirements from workflow.
In this work, the rules transformation graphs are applied to achieve a precise specification of semantic web service, and its application is considered on a specific case study, the process of software development OpenUP/Basic. This implementation allows to the workflow engine to match the user's requirements and the web services available, and to choose the best at the time of invocation.

Key Words: Workflow Reference Model, Invoked Applications Interface, Web Services, Rules Tansformation Graph, OpenUP/Basic

Resumen

Un proceso impone un orden especificado en las actividades de trabajo a lo largo del espacio y el tiempo, con un principio, un fin y entradas y salidas claramente especificadas.
El objetivo principal de las organizaciones es tornarse más ágiles, más competentes y adaptarse rápidamente a los continuos cambios del mercado en el que operan. En la actualidad, la automatización de los procesos de negocio es un objetivo central para muchas organizaciones.
A partir del Modelo de Referencia de Workflow, propuesto por la Workflow Management Coalition (WfMC), los servicios Web y las reglas de transformación de grafos, se obtiene una solución eficiente para optimizar la invocación de las aplicaciones externas que resultan más adecuadas de acuerdo a los requerimientos del workflow.
En este trabajo se aplican reglas de transformación de grafos para lograr una especificación semántica precisa de un servicio web, y se estudia su aplicación a un caso de estudio específico, el proceso de desarrollo de software OpenUP/Basic. Esta implementación le permite al motor workflow establecer una correspondencia entre los requerimientos del usuario y los servicios web disponibles, y elegir el mejor en el momento de la invocación.

Palabras Clave: Modelo de Referencia de Workflow, Interfaz de Aplicaciones Invocadas, Servicios Web, Reglas de Transformación de Grafos, OpenUP/Basic

1. INTRODUCCIÓN

En un entorno con demandas crecientes de competitividad, calidad y servicio al cliente; las organizaciones se ven obligadas a mejorar continuamente sus procesos de negocio; requiriendo mejor coordinación, comunicación, y cooperación; agilidad y flexibilidad de los sistemas que soportan el negocio. Un sistema de administración de Workflow permite describir y automatizar dichos procesos de negocio.

El modelo de referencia de Workflow, desarrollado por la WfMC [25], define un marco genérico para la construcción de sistemas de administración de Workflow, permitiendo la interoperabilidad entre ellos y con otras aplicaciones involucradas. Los sistemas de gestión de workflow contienen componentes genéricos que interactúan de forma definida. El Modelo de Referencia de Workflow define cinco interfaces, que permiten a las aplicaciones del Workflow la comunicación a distintos niveles. En particular, la Interfaz de las Aplicaciones Invocadas [26] se define para la invocación de las aplicaciones externas, y requiere conocer la información acerca de la aplicación y su invocación en tiempo de desarrollo. Si bien en muchos sistemas de gestión de workflow se conocen a priori las aplicaciones que se desea utilizar, también se dan muchos casos donde varias aplicaciones que brindan un mismo servicio, podrían ser requeridas por el motor workflow.

Los Servicios Web son aplicaciones auto-contenidas, auto-descritas que pueden ser publicadas, localizadas e invocadas a través de la Web, sin la necesidad de conocer la ubicación exacta de los mismos [20]. Con el propósito de mejorar la selección de aplicaciones en tiempo de ejecución, en [6] se plantea una especificación de la Interfaz de Aplicaciones Invocadas utilizando Servicios Web. Así, el usuario del Workflow no necesita conocer la ubicación de la aplicación que desea invocar, y cualquier aplicación puede cambiar su ubicación en la red sin que esto implique ningún cambio en su invocación.

Los servicios web son registrados a través del protocolo denominado UDDI [17] para ser localizados y usados por aplicaciones. Este protocolo requiere la descripción de la aplicación o servicio concreto que se quiere invocar y si hay más de una aplicación que brinda el mismo servicio, es necesario definir en tiempo de desarrollo cual se invocará.

En este trabajo, el estudio está centrado en permitir que el motor Workflow pueda seleccionar una aplicación entre varias que ofrecen el mismo servicio. Se propone optimizar esta selección de servicios usando reglas de Transformación de Grafos [2].

Las reglas de transformación de grafos permiten proveer una especificación semántica precisa de un servicio web, y establecer una correspondencia entre los requerimientos del usuario y los servicios web disponibles, facilitando así su descubrimiento automático, en tiempo de ejecución.

Este trabajo está organizado como sigue: la sección 2 describe el estado del arte relacionado a la presente propuesta, en la sección 3 se presentan los conceptos fundamentales de reglas de transformación de grafos, la sección 4 ilustra el uso de servicios web para la especificación de la Interfaz de Aplicaciones Invocadas del Modelo de Referencia de Workflow, la sección 5 grafica la aplicación de reglas de transformación de grafos a la definición de servicios web, con un ejemplo de aplicación de la propuesta. Por último, la sección 6 presenta las conclusiones abordadas.

2. ANTECEDENTES

Existen diversos trabajos que presentan alternativas para implementar la comunicación del Workflow con las aplicaciones de clientes.

En [27] la WfMC presenta una propuesta de especificación de las Interfaces 2 y 4 basada en el uso de IDL y *bindings* con OLE como alternativas a las especificaciones existentes C y MIME. En [19] se presenta la especificación *JointFlow*, que define las interfaces que soportan la interacción en tiempo de ejecución entre los componentes de Workflow, y permite la interoperabilidad de los componentes de

negocio a través de los dominios de negocio y el monitoreo de los procesos. Otros trabajos interesantes son los de [28], [16], [4] y [14], donde los autores presentan otras alternativas, usando Grid Computing, definiendo un workfow composer agent, una interfaz que encapsula las funcionalidades de una organización y las publica como servicios web y un lenguaje de Workflow que usa los Servicios Web como componentes.

En [6] los autores proponen aprovechar los beneficios de los Servicios Web, utilizándolos para especificar las funciones de la Interfaz de las Aplicaciones de Cliente, facilitando la comunicación del Workflow con las aplicaciones.

Pero actualmente el descubrimiento y selección de los servicios web presenta algunos inconvenientes dados por el hecho de que el registro de los mismos no tiene en cuenta su semántica, lo cual dificulta encontrar todos los servicios que resuelven el mismo requerimiento, aunque su descripción no sea exactamente la misma. Se han propuesto distintas alternativas para solucionar estos inconvenientes.

En [12] se presenta un enfoque formal para la selección automática de servicios web, basado en la correspondencia de los requerimientos del solicitante del servicio con la descripción de los servicios ofrecidos por el proveedor. Focaliza en el chequeo del comportamiento funcional de los contratos de las operaciones, especificando las precondiciones y los efectos de las operaciones requeridas y las ofrecidas. Utiliza reglas de transformación de grafos con condiciones positivas de aplicación como una notación visual y formal para definir los contratos, y poder determinar la compatibilidad semántica y sintáctica entre los requerimientos del solicitante y las ofertas el proveedor.

En [5] los autores de [12] avanzan en su propuesta de formalizar la selección automática de servicios web, usando reglas de transformación de grafos con condiciones positivas y negativas, y se concentran en chequear la compatibilidad del comportamiento. A partir de la relación semántica encontrada entre el solicitante y el proveedor, avanzan en probar la correctitud y completitud de esta relación.

En [9] se propone solucionar el problema de la selección dinámica y en tiempo de ejecución de los servicios web usando reglas de transformación de grafos. El proceso de descubrimiento depende del entendimiento semántico de los servicios ofrecidos, pero actualmente, este descubrimiento se desarrolla manualmente y en tiempo de desarrollo. Las reglas de transformación de grafos proveen una especificación semántica precisa, necesaria para automatizar el descubrimiento de los servicios de una manera visual e intuitiva.

En [10] los autores presentan un enfoque visual basado en el uso de modelos de software y transformación de grafos, que permite la descripción de los servicios y además provee un concepto preciso de correspondencia. Presentan también una implementación usando estándares y herramientas disponibles en la actualidad.

En [13] los autores plantean las dificultades que incorporar la prueba automática para validar los servicios web antes de permitir su registro. Primero, el descubrimiento automático de un servicio web genera casos de prueba de conformidad a partir de la descripción del servicio a proveer, luego se corren estos casos de prueba sobre el servicio web considerado, y solamente si la prueba es pasada satisfactoriamente, el nuevo servicio web se registra. De esta manera, los clientes obligan a lo servicios web ofrecidos a proveer una signatura compatible, un comportamiento adecuado, y una implementación de alta calidad.

3. LAS REGLAS DE TRANSFORMACIÓN DE GRAFOS

La transformación de grafos [2] es una técnica gráfica, formal, declarativa y de alto nivel muy usada para transformación de modelos, simulación de modelos, chequeos de consistencia entre modelos o vistas y optimización de diversos contextos.

La transformación de grafos es un mecanismo formal para la manipulación de grafos basado en reglas. En analogía a las gramáticas de Chomski sobre cadenas de caracteres, las reglas de grafos están formadas por una parte izquierda y una parte derecha que contienen grafos.

La transformación de grafos es una modificación de estos grafos basada en las reglas, como se muestra en la Figura 1.

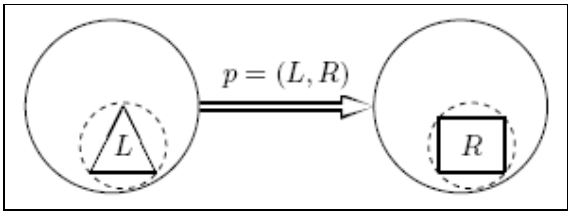

Figura 1. Modificación de grafos basada en reglas

Intuitivamente, una regla de transformación de grafos o producción p = (L, R) contiene un par de grafos (L, R), denominados la parte izquierda L y la parte derecha R. Aplicar la regla p = (L, R) significa encontrar una ocurrencia de L en el grafo anfitrión y reemplazarla por R, obteniendo el grafo destino. Para aplicar una regla al grafo anfitrión, se debe encontrar un morfismo de correspondencia entre la parte derecha de la regla y el grafo. Si dicho morfismo es encontrado, por un proceso de derivación, la regla se aplica sustituyendo la imagen de morfismo encontrado en el grafo, por la parte derecha de la regla. Dicho de otro modo, una regla de transformación de grafos, también llamada producción, (L, K, R) consiste en tres grafos L, K y R. Una producción (L, K, R) es aplicable a un grafo G si G contiene un subgrafo que es una imagen de L. Una producción (L, K, R) es aplicable a un grafo G si L es un subgrafo de G. El grafo L constituye la parte izquierda de la producción, y formula las condiciones bajo las cuales resulta aplicable la producción, o sea, es un subconjunto del conjunto de objetos (nodos y arcos) del universo. El grafo de contacto K, que generalmente es un subgrafo de L y de R, describe los subobjetos de la parte izquierda que se deben preservar en el proceso de aplicación de la producción. Por lo tanto, la diferencia entre L y K: L\K, contiene a todos aquellos subobjetos que se deben eliminar al aplicar la producción ((1) Figura 2). Análogamente, la diferencia R\K contiene a todos aquellos subobjetos que se deberán agregar al aplicar la producción ((2) Figura 2). Este grafo intermedio K describe el contexto en el cual se integran los subobjetos agregados.

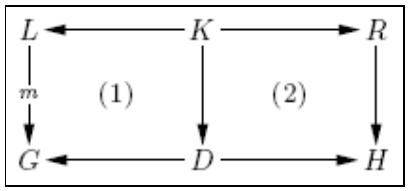

Figura 2. Transformación de Grafos

3.1. Las Reglas de Transformación de Grafos en el Modelado Orientado a Objetos

En el modelado orientado a objetos los grafos ocurren a dos niveles: a nivel de tipos (los diagramas de clases) y a nivel de instancia (diagramas de objetos). Esta noción se puede generalizar en el concepto de grafos tipados, los cuales pueden verse como una representación abstracta de los diagramas de clases del Unified Modeling Language (UML) [3]. Los diagramas de objetos son grafos equipados con una correspondencia (que preserva la estructura) a los grafos tipados, formalmente expresada como un homomorfismo de grafos.

El diagrama de objetos UML (representando un grafo de instancia) se puede corresponder al diagrama de clases (representando un grafo tipado) definiendo que *type(object) = Class* para cada instancia *object : Class* en el diagrama de objetos. Extendiendo la idea a los enlaces, la preservación de la estructura significa que, por ejemplo, un enlace entre los objetos *object1* y *object2* debe corresponderse a una asociación en el diagrama de clases entre type(*object1*) y type(*object2*). Por

el mismo mecanismo de compatibilidad estructural se puede asegurar que un atributo de un objeto está declarado en la clase correspondiente, etc.
En este contexto una ***regla de transformación de grafos*** p : L → R consiste de un par de grafos de instancia tipados L, R tal que la unión L ∪ R está definida. La parte izquierda L , representa la precondición de la regla y la parte derecha R representa la poscondición.
Una transformación de grafos desde el pre-estado G al pos-estado H está dada por un homomorfismo o : L ∪ R → G ∪ H, llamado ocurrencia, tal que:

- o(L) ⊆ G y o(R) ⊆ H, es decir, la parte izquierda de la regla está incluida en el pre-estado y la parte derecha de la regla está incluida en el pos-estado
- o(L \ R) = G \ H y o(R \ L) = H \ G, es decir, la parte de G que se borra es la que se corresponde con los elementos de L que no están incluidos en R y, simétricamente, la parte de H que se agrega es la que se corresponde con los elementos nuevos en R.

Operacionalmente, la transformación se realiza en tres pasos:

1) Encontrar una ocurrencia o de la parte izquierda L en el grafo G.
2) Remover todos los vértices y arcos de G que no se corresponden por L \ R. Asegurarse que la estructura resultante D := G \ o(L \ R) continúa siendo un grafo legal, es decir, que no se dejaron sueltos arcos debido a la supresión de su origen o destino.
3) Pegar D con R \ L para obtener el grafo derivado H.

Los lenguajes de modelado gráfico estándar, como UML, poseen muchas ventajas, son fáciles de entender y de usar, proveen un cierto nivel de formalismo, son de propósito general y están soportados por diversas herramientas. Los diagramas de clases de UML son ampliamente aceptados para visualizar conceptos y estructuras de un dominio particular, y proveer un entendimiento común a los desarrolladores. Así, pueden utilizarse para representar conceptos codificados en un lenguaje basado en texto, como por ejemplo WSDL, resultando muy útiles para describir los dominios que son interés para los proveedores y solicitantes de los servicios web.

4. ESPECIFICACIÓN DE LA INTERFAZ DE APLICACIONES INVOCADAS CON SERVICIOS WEB

La Interfaz de las Aplicaciones Invocadas definida por la WfMC, permite que el motor workflow pueda invocar aplicaciones externas, como por ejemplo servicios de e-mail, fax, administración de documentos, o aplicaciones de usuario. Dicha interfaz define un conjunto de *Workflow Application Programming Interfaces* (WAPIs), que son usadas por el Sistema de Workflow para controlar los dispositivos de aplicaciones especializadas. Estas herramientas son las que finalmente se encargan de comenzar y terminar las aplicaciones, pasar la información relevante del workflow y de la aplicación "a" y "desde" la aplicación y controlar el estado a nivel de ejecución de la aplicación.
Como esta interfaz debe manejar requerimientos bi-direccionales, la interacción con los dispositivos de aplicaciones especializadas depende de la interfaz y arquitectura de la aplicación, lo cual restringe la selección dinámica de las aplicaciones. Para permitir el requerimiento y la actualización de datos de la aplicación y otras funcionalidades importantes en tiempo de ejecución, se propone especificar estas WAPIs con servicios web.
Los Servicios Web son servicios autónomos e independientes que se ofrecen mediante la web. Su principal beneficio es que permiten que las aplicaciones sean más modulares y desacopladas, facilitando su reutilización en distintas plataformas o lenguajes de programación.
Para la descripción de un Servicio Web se utiliza WSDL (Web Service Description Language) [23] [24] basado en XML [21]. Además, es necesario describir los mensajes entre las aplicaciones y el servicio web, y la forma en que los mismos serán transportados a través de la web. SOAP (Simple Object Access Protocol) [22] es el protocolo más conocido basado en mensajes, que es utilizado para

describir la interacción de las aplicaciones con los web services. Por su parte, el protocolo de transporte más usado es HTTP (Hiper Text Transport Protocol). Y por último, es necesario registrar y localizar el servicio web, para lo cual se define un directorio de Servicios Web distribuido y basado en Web que permite que se listen, busquen y descubran. Por lo general este directorio es definido UDDI (Universal Description, Discovery and Integration) [17]. Las Figuras 3 y 4 especifican los elementos ***types*** e ***interface*** del servicio web de la operación que permite invocar una aplicación externa.

```
<types>
    <xs:element name="InvokeApp" type="tInvokeApp"/>
    <xs:complexType  name="tInvokeApp">
      <xs:sequence>
        <xs:element name="appReqeriments" type="xs: tAppReq"/>
          <xs:complexType  name=" tAppReq ">... </xs:complexType>
        <xs:element name="procInstId" type="xs: string"/>
        <xs:element name="workItemId" type="xs: string"/>
        <xs:element name="parmeters" type="xs: tParams"/>
          <xs:complexType  name="tParams"> ... </xs:complexType>
        <xs:element name="appMode" type="xs: integer"/>
      </xs:sequence>
    </xs:complexType>
    <xs:element name="AppResult" type="tAppResult"/>
    <xs:complexType  name="tAppResult">
      <xs:sequence>
        <xs:element name="appName" type="xs: string"/>
        <xs:element name="appLocation" type="xs: string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:element name="WorkflowException" type="xs: tException"/>
    <xs:complexType  name="tException">
      <xs:sequence>
        <xs:element name="error" type="xs: tError"/>
        <xs:element name="message" type="xs: String"/>
      </xs:sequence>
    </xs:complexType>
 </xs:schema>
</types>
```

Figura 3: Descripción del elementos *types* del servicio web *InvokeApplication*

```
<interface name = "InvokeAppInterface" >
<fault name = "InvokeAppFault_WorkflowClientException" element = "xs:
WorkflowException"/>
        <operation name="opInvokeApp"
pattern=http://www.w3.org/2004/03/wsdl/in-out>
            <input messageLabel="In" element="xs: InvokeApp" />
            <output messageLabel="Out" element="xs: AppResult" />
             <outfault ref="tns: InvokeAppFault_WorkflowException"
             messageLabel="Out"/>
        </operation>
</interface> ...
```

Figura 4: Descripción del elementos *interface* del servicio web *InvokeApplication*

De esta manera quedan especificados los elementos *types* e *interface* para la definición de la operación InvokeApplication utilizando SOAP y WSDL. Esta especificación con Servicios Web mejora la comunicación de las Aplicaciones Invocadas con el motor workflow, favoreciendo al usuario dado que no necesita conocer la ubicación exacta de la aplicación a invocar, y a las aplicaciones porque pueden variar su ubicación en la red sin implicar ningún cambio en su invocación.

5. APLICACIÓN DE LAS REGLAS DE TRANSFORMACIÓN DE GRAFOS A LA SELECCIÓN AUTOMÁTICA DE SERVICIOS WEB

En la sección anterior se presentó una especificación del servicio web *InvokeApplication*, que permite la invocación de una aplicación externa al sistema de administración de workflow, independizándose de la ubicación exacta de la misma y facilitando así la distribución de las aplicaciones en la red.
El objetivo ahora es analizar la compatibilidad de los servicios web provistos en la red y los requeridos por un usuario del workflow.
Uno de los elementos que debe definirse al especificar un servicio web es la *interface*, la cual describe la secuencia de mensajes que el servicio envía o recibe, agrupándolos en una colección de operaciones, que son accesibles desde la red a través de mensajes XML estandarizados.
En base a los conceptos antes mencionados sobre reglas de transformación de grafos en el modelado orientado a objetos y las propuestas de [5] y [9], se analiza la compatibilidad del comportamiento de estas operaciones que constituyen las interfaces de los servicios web. La Figura 5 muestra la interface, en notación UML, del servicio web *InvokeApplication*.

Figura 5: Descripción del elementos *interface* del servicio web *InvokeApplication*

Esta interface sólo contiene información sobre la estructura de la operación. El comportamiento de la misma puede representarse mediante ***contratos***, expresados usando reglas de transformación de grafos. Un contrato consiste de una precondición especificando el estado del sistema antes de que el comportamiento sea ejecutado, y una poscondición especificando el estado del sistema después de la ejecución del comportamiento.
Usando UML como notación visual para describir los contratos, y reglas de transformación de grafos para establecer la correspondencia, se describe el servicio web ofrecido por un proveedor y la formulación del requerimiento de un usuario del workflow, expresados como grafos con atributos (Figuras 6 y 7). El grafo de la parte izquierda de la regla representa la precondición de un servicio, es decir, el estado del sistema antes de la ejecución del servicio web. El grafo de la parte derecha de la regla representa la poscondición, es decir, la situación después de satisfacerse la ejecución del servicio web.

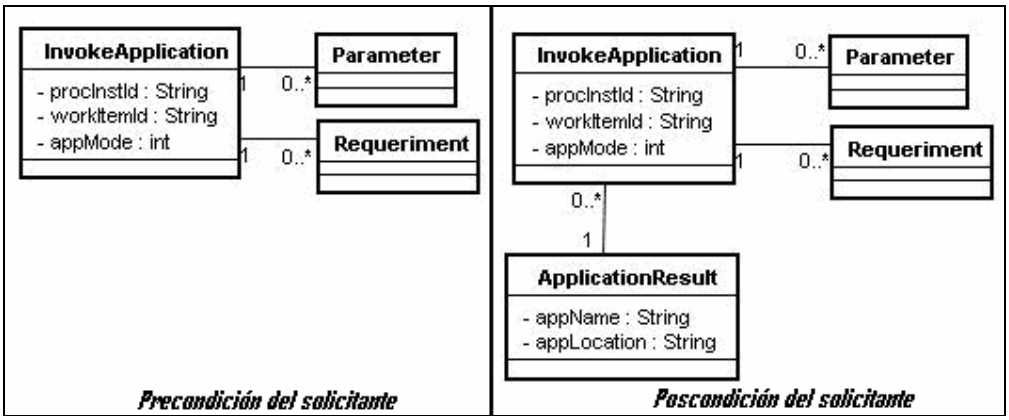

Figura 6: Grafos de la pre y poscondición del servicio solicitado

En los requerimientos modelados en la pre y la poscondición es posible almacenar la información relevante para el motor workflow sobre las características de la aplicación buscada por el usuario, por

ejemplo, la categoría de la aplicación (cliente de correo, editor de texto, planilla de cálculo, etc.), el sistema operativo del usuario, su ubicación geográfica, etc.
Los parámetros almacenan información relevante para la aplicación, por ejemplo, el nombre de un archivo, el identificador de un registro, etc.

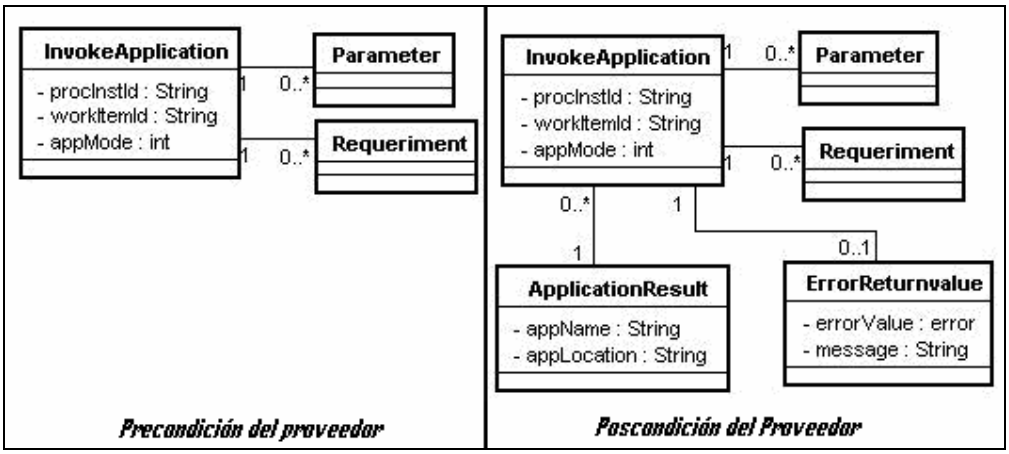

Figura 7: Grafos de la pre y poscondición del servicio del proveedor

Para decidir automáticamente si un servicio del motor workflow satisface la demanda del solicitante, es necesario comparar los grafos generados por el solicitante y el proveedor (Figuras 6 y 7) (motor workflow). Se formalizará usando relaciones de subgrafos. Si la precondición del proveedor es un subgrafo de la precondición del solicitante, entonces el proveedor provee toda la información necesaria para ejecutar el servicio web. Si la poscondición del solicitante es un subgrafo de la poscondición del proveedor, entonces el servicio genera todos los efectos esperados por el solicitante, además de algunos efectos adicionales, como en este caso los errores posibles.
A continuación se presenta un ejemplo de invocación automática de una aplicación externa al Workflow, en el marco del proceso de desarrollo de software OpenUP/Basic.

5.1. La Utilización de Servicios Web en la Invocación de Aplicaciones Externas en OpenUP/Basic

Para ejemplificar el uso de Servicios Web automáticamente seleccionados en la invocación de aplicaciones externas al Workflow, se plantea su aplicación en el marco de las tareas involucradas en el proceso de desarrollo de software denominado *Basic Open Unified Process* (OpenUP/Basic) [1] [8] [7]. OpenUP/Basic es un proceso de desarrollo de software de código abierto diseñado para pequeños equipos bien organizados, interesados en realizar un desarrollo ágil. Si bien OpenUP/Basic mantiene las mismas características del *Racional Unified Process* (RUP) [18], basado a su vez en *El Proceso Unificado de Desarrollo de Software* (*The Unified Software Development Process*) [15], en cuanto al desarrollo iterativo, la definición de casos de uso y escenarios de conducción de desarrollo, la gestión de riesgos, y el enfoque centrado en la arquitectura; define un proceso iterativo mínimo, completo y extensible, es decir, provee la mínima cantidad de procesos para un equipo pequeño, y puede ser usado como está o ser extendido y personalizado para propósitos específicos.
La mayoría de los elementos de este proceso están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances.
En OpenUP/Basic un proyecto se divide en iteraciones, las cuales son planificadas en un intervalo definido de tiempo que no supera las pocas semanas. Se proveen elementos que ayudan a los equipos de trabajo a enfocar los esfuerzos a través del ciclo de vida de cada iteración de tal forma que se

puedan distribuir funcionalidades incrementales de una manera predecible, y obtener una versión totalmente probada y funcional al final de cada iteración.
El OpenUP estructura el ciclo de vida de un proyecto en cuatro fases: concepción, elaboración, construcción y transición. El ciclo de vida del proyecto provee a los interesados un mecanismo de supervisión y dirección para controlar los fundamentos del proyecto, su ámbito, la exposición a los riesgos, el aumento de valor y otros aspectos.
Las disciplinas que proveen organización a las tareas en OpenUP/Basic son: *Análisis y Diseño, Configuración y Administración de cambios, Implementación, Administración del Proyecto, Requerimientos* y *Prueba*. En particular, la Implementación es la disciplina donde se obtiene una solución técnica consistente con el diseño, enmarcada en la arquitectura y que soporta los requerimientos. El objetivo principal es que la abstracción del diseño (clases, componentes, etc.) sea detallada para transformarse en la implementación.
Algunos de los fines de esta disciplina son:

- La reutilización de código: la reutilización de código y las herramientas de generación de código producen código más robusto y son preferibles a escribir código a mano. El código existente a menudo ya ha sido altamente probado, resultando más estable y entendible que el código nuevo. El código fuente creado a partir de una herramienta de generación de código (como una herramienta de modelado visual) automatiza las tareas monótonas de codificación, como por ejemplo la creación de los *getters* y los *setters*.
- La transformación del diseño en implementación: la transformación del diseño en código implementa la estructura del sistema en el lenguaje fuente elegido. También implementa el comportamiento del sistema definido en lo requerimientos funcionales. Implementar el comportamiento del sistema significa escribir el código que permite a las diferentes partes de la aplicación (clases o componentes) colaborar para realizar ese comportamiento del sistema. Existen varias técnicas para transformar el diseño en implementación automáticamente. Una de las técnicas para hacer esto es detallar los modelos y usarlos para generar una implementación. Tanto la estructura (definida en los diagramas de clases y paquetes) como el comportamiento (definido en los diagramas de estados y actividades) puede usarse para generar código ejecutable. Estos prototipos luego pueden ser más refinados, cuando sea necesario.

Una organización dedicada al desarrollo de software que utilice el proceso de desarrollo de software mencionado necesitará generar código automáticamente a partir del diseño. El código obtenido será el esqueleto (prototipo) de las clases a implementar. Para realizar esta actividad el motor workflow podría invocar una aplicación externa que se encargue de, a partir de los diagramas clases obtenidos en el diseño, generar el código correspondiente.
La generación del código se podría realizar con cualquier aplicación disponible en la red, de manera de independizarse de la ubicación específica de dicha aplicación. El motor Workflow, de acuerdo a los requerimientos recibidos, se encarga de realizar la correspondencia a los servicios web disponibles que satisfacen dichos requerimientos.
Para ello, transforma la formulación del requerimiento en grafos con atributos, para luego compararlos con los grafos que describen los servicios web ofrecidos por el proveedor.
Instanciando los grafos de la pre y poscondición del servicio web requerido, se obtienen los grafos resultantes para este ejemplo, como se muestra en la Figura 8.
Igualmente, los grafos de abajo representan instancias (diagrama de objetos) de los diagramas de clases que modelan la pre y poscondición del proveedor.

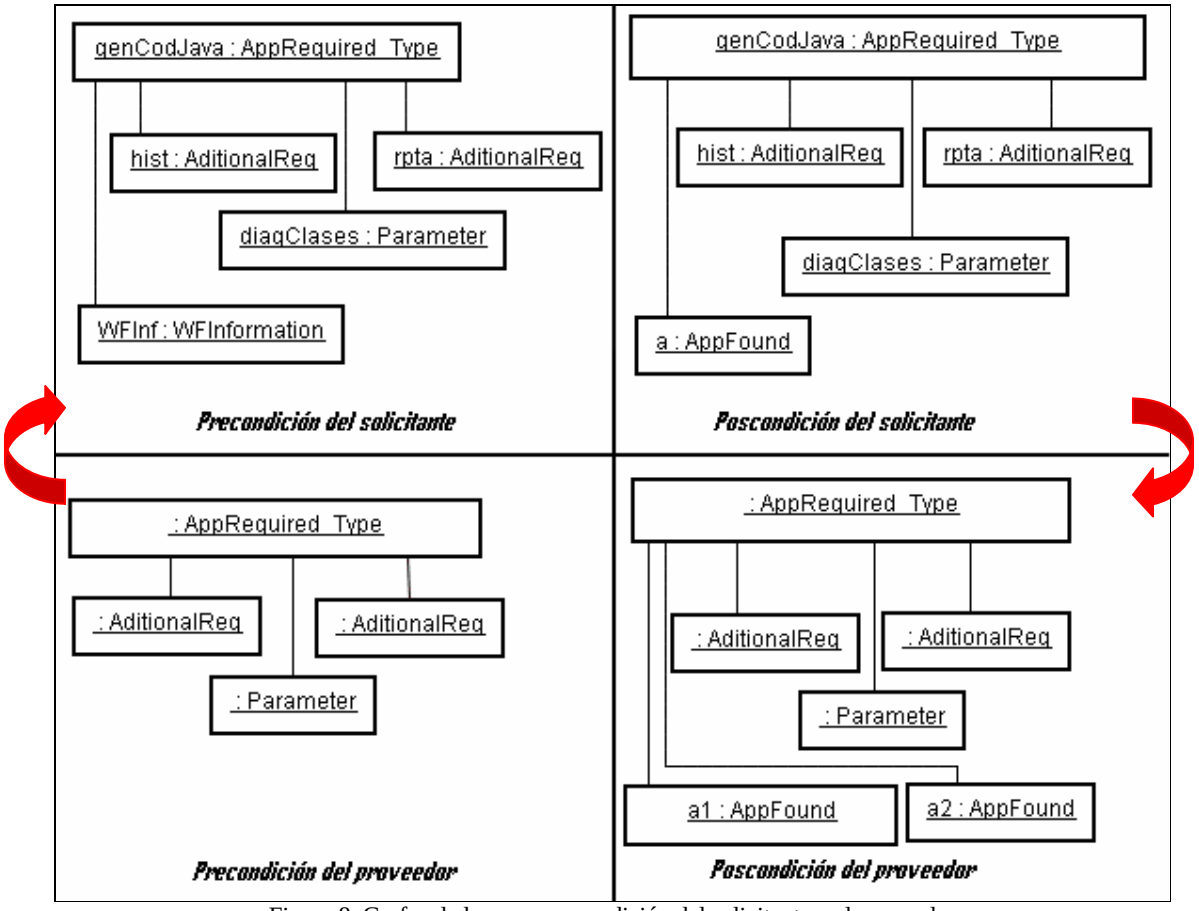

Figura 8: Grafos de la pre y poscondición del solicitante y el proveedor

Una vez realizada la correspondencia entre el requerimiento (una aplicación para generar código Java a partir de un modelo de diseño representado con un diagrama de clases UML) y los servicios web disponibles, el motor Workflow obtiene dos servicios web (las instancias a1 y a2). Para que el motor Workflow finalmente determine cuál es la aplicación que invocará existen dos opciones: requerir la interacción con el usuario que realizó la solicitud para que éste decida, o definir un criterio de selección adicional (por ejemplo, la primera de la lista, la que requiera menos recursos de hardware, la que se invoque más rápido, etc.) y devolver la aplicación que lo cumpla. Esto otorga mayor flexibilidad al motor Workflow, a la hora de invocar aplicaciones externas al sistema de administración de workflow de la organización, y favorece la distribución de las aplicaciones en la red.

6. CONCLUSIONES

La automatización de procesos de negocio o workflows ha producido en las organizaciones una importante reducción de costos, incremento de eficiencia, menores oportunidades de errar, mejor control y calidad en beneficio de todos. Para lograr la interoperabilidad entre diferentes sistemas de administración de workflows y otras aplicaciones involucradas, la WfMC creó un modelo genérico, llamado modelo de referencia de workflow. Sus Interfaces 2 y 3 definen un marco genérico para la invocación de aplicaciones de cliente y aplicaciones externas, así como también el seguimiento del estado de las mismas.

Con el fin de permitir que el motor workflow utilice una aplicación sin necesidad de conocer su ubicación, se planta una solución usando servicios web. Especificando la operación que invoca las

aplicaciones externas al workflow como un servicio web, se logra optimizar la comunicación del workflow con las aplicaciones externas, ya que permite usar un servicio web disponible independizándose de su ubicación.
Actualmente, los servicios web son registrados y localizados usando el protocolo UDDI (Universal Description, Discovery and Integration), el cual requiere una descripción específica de la aplicación o servicio concreto que se quiere invocar que intenta describir de alguna forma la semántica del mismo. Pero los resultados de una consulta estén limitados por las palabras clave de la búsqueda, y no son de todo fiables.
En este trabajo se aplican reglas de transformación de grafos con el fin de lograr una especificación semántica precisa del servicio web, lo que permite establecer una correspondencia entre los requerimientos del usuario y los servicios web disponibles, y que el Workflow pueda elegir en el momento de la invocación. Se modela con un Diagrama de Clases UML la interfaz del servicio web *InvokeApplication.* La formalización de la especificación como un grafo con atributos, permite realizar una correspondencia con los servicios web disponibles en la red que cumplen con el morfismo establecido. Se presenta como caso de estudio el proceso de desarrollo de software OpenUP/Basic, donde por ejemplo, la disciplina de implementación propone transformar el diseño en implementación tomando los modelos detallados de diseño y usándolos para generar código automáticamente.
En un Sistema de Administración de Workflow que implemente este proceso de desarrollo, el motor workflow puede invocar una aplicación externa que se encargue de realizar esta transformación, sin conocer la aplicación específica ni su ubicación exacta en la red. En tiempo de ejecución, el motor workflow obtiene una lista de los servicios web que se corresponden con este requerimiento, y elige uno de ellos. Con esta especificación de la Interfaz de Aplicaciones Invocadas con servicios web y la optimización de la búsqueda y selección de los servicios en la red, se logra que el Workflow se comporte internamente de forma distribuida. Además, se facilita la incorporación de servicios nuevos, y la invocación no está limitada a una aplicación específica, definida en tiempo de desarrollo.

Referencias Bibliográficas

[1] Balduino R. Introduction to OpenUP (Open Unified Process). http://www.eclipse.org/epf/general/OpenUP.pdf. 2007. Último acceso en Agosto de 2008.

[2] Baresi L., Heckel R. Tutorial Introduction to Graph Transformation: A Software Engineering Perspective. First International Conference on Graph Transformation. Spain. v. 2505, pp. 402-429. (ICGT 2002). 2002.

[3] Booch G., Rumbaugh J., Jacobson I., The Unified Modeling Language. Addison Wesley, Second Edition. 2005.

[4] Cardozo J., Shelt A. Introduction to Semantic Web Services and Web Process Composition. Publication of LSDIS. Large Scale Distributed Information Systems. University of Georgia. Computer Science Department.

[5] Cherchago A., Heckel R. Specification Matching of Web Services Using Conditional Graph Transformation Rules, In G. Engels, H. Ehrig, F. Parisi-Presicce, and G. Rozenberg (Editors): Proc. 2nd International Conference on Graph Transformation (ICGT 04), Roma, Italy, Volume 3256 of Lecture Notes in Computer Science. Springer-Verlag, 2004.

[6] Debnath N., Martellotto P., Daniele M., Riesco D., Montejano G. Using Web Services To Optimize Communication Among The Workflow Engine And Its Client Application Interface. Second International Conference on Information Systems, Technology and Management. ICISTM 2008. Institute of Management Technology. Ghaziabad. Dubai. March 2008.

[7] Dramis, L., Britos, P., Rossi, B., García Martínez, R. Verificación de Bases de Conocimiento Basada en Algebra de Grafos. Revista Del Instituto Tecnológico de Buenos Aires. v.23, p.74 - 86, 2000. http://www.centros.itba.edu.ar/capis/webcapis/planma-esp.html. R-ITBA-23-

verificacionalgebradegrafos.pdf.
[8] Eclipse Process Framework (EPF) Community. Introducción a OpenUP/Basic. Versión 2. http://epf.eclipse.org/wikis/openupsp/. Último acceso en Agosto de 2008.
[9] Hausmann J.H., Heckel R., Lohmann M. Towards Automatic Selection of Web Services Using Graph Transformation Rules. Berliner XML Tage. XML-Clearinghouse, 2003 – 286-291.
[10] Hausmann J.H., Heckel R., Lohmann M. Model-based Discovery of Web Services, Proceedings of the IEEE International Conference on Web Services, 2004.
[11] Hausmann J.H., Heckel R., Lohmann M. Model-based development of Web services descriptions enabling a precise matching concept. International Journal of Web Service Research. Volume. 2. Issue: 2. p. 67 – 84. 2005.
[12] Heckel R., Cherchago A. Application of Graph Transformation for Automating Web Service Discovery, Proc. Dagstuhl Seminar 04101 Language Engineering for Model-Driven Software Development, Dagstuhl, Germany, 2004.
[13] Heckel R., Mariani L. Automatic Conformance Testing of Web Services. Fundamental approaches to software engineering. International conference No8, Edinburgh , ROYAUME-UNI (04/04/2005) 1973, vol. 3442, pp. 34-48, [Note(s) : XIII, 371 p.,] (24 ref.) ISBN 3-540-25420-X.
[14] Hiane da S. Maciel L. A., Toshiro Yano E. Uma Linguagem de WF Para Composicao de Web Services. XIX Simpósio Brasileiro de Engenharia de Software. Uberlandia, MG, Brasil. 2005.
[15] Jacobson I., Booch G., Rumbaugh J., The Unified Software Development Process. Addison Wesley, 1999.
[16] Laukkanen M., Helin H. Composing Workflows of Semantic Web Services. Workshop on Servicios Web and Agent-based Engineering. AAMAS'2003. Melbourne, Australia. 2003.
[17] OASIS. UDDI Version 3.0.2. http://uddi.org/pubs/uddi_v3.htm. Último acceso Mayo 2007.
[18] Rational Unified Process. http://www-306.ibm.com/software/rational/.
[19] Schmidt M.T., Building Workflow Business Objects. IBM Software Group OOPSLA'98 Business Object Workshop IV.
[20] World Wide Web Consortium. Web Service Architecture. http://www.w3.org/TR/ws-arch/. Último acceso Mayo 2007.
[21] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Second Edition). http://www.w3.org/TR/xml. Último acceso Mayo 2007.
[22] World Wide Web Consortium. SOAP Version 1.2 Part 1: Messaging Framework. http://www.w3.org/TR/soap12-part1/. Último acceso Mayo 2007.
[23] World Wide Web Consortium. Web Services Description Language (WSDL) Version 2.0 Part 0: Primer. http://www.w3.org/TR/wsdl20-primer. Último acceso Mayo 2007.
[24] World Wide Web Consortium. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. http://www.w3.org/TR/wsdl20. Último acceso Mayo 2007.
[25] Workflow Management Coalition. The Workflow Reference Model. WfMC-TC00-1003. http://www.wfmc.org/standards/referencemodel.htm. Último acceso Mayo 2007.
[26] Workflow Management Coalition. Programming Interface 2&3 Specification. WfMC-TC-1009. V2.0. http://www.wfmc.org/standards/publicdocuments.htm. Último acceso Mayo 2007.
[27] Workflow Management Coalition. A Common Object Model Discussion Paper. WfMC-TC10-22. http://www.wfmc.org/standards/docs/TC-1022_commom_Object%20Model_Paper.pdf. Último acceso Mayo 2007.
[28] Yang M., Liang H., Xu B., S-WFMS. A service-based WFMS in Grid Environment. Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05). IEEE. 2005.