Mejorando la Invocación de Aplicaciones desde un Workflow: Un caso de estudio

Marcela Daniele[1], Paola Martellotto[1], Daniel Riesco[2]

[1] Universidad Nacional de Río Cuarto, Facultad de Ciencias Exactas, Físico-Químicas y Naturales, Departamento de Computación, Ruta 36 Km 601 CP X5804BYA, Río Cuarto, Córdoba, Argentina,
{paola, marcela}@dc.exa.unrc.edu.ar
[2] Universidad Nacional de San Luis, Ciencias Exactas, Físicas y Naturales, Departamento de Informática,
Ejército de los Andes 950 - D5700HHW - San Luis – Argentina, driesco@unsl.edu.ar

Resumen. En trabajos anteriores se propuso la optimización de la invocación de aplicaciones desde un motor workflow usando servicios web. Posteriormente se aplicaron reglas de transformación de grafos para lograr una especificación semántica más precisa del servicio web, con el objeto de permitir 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. En este trabajo se propone aplicar la propuesta a un caso de estudio completo denominado OpenUP/Basic, incorporando la atención en las características de calidad de los servicios web para su selección e invocación.

Palabras Clave. Modelo de Referencia de Workflow, Interfaz de Aplicaciones Invocadas, Servicios Web, Atributos de Calidad, Reglas de Transformación de Grafos.

1 Introducción

En la actualidad, es fundamental acceder a la información de un modo ágil y eficaz. Las organizaciones pierden demasiado tiempo localizando información. Los sistemas de gestión de workflow automatizan los procesos y han conseguido importantes beneficios como, ahorro de tiempo, mejora de productividad y del control de procesos, optimización en la atención y servicio al cliente, entre otras.

A partir de la estandarización de la automatización de los procesos de negocio especificada por la WfMC (Workflow Management Coalition) [1], y su definición del modelo de referencia de workflow, es posible contar con un lenguaje común para lograr la interoperabilidad entre diferentes sistemas de este tipo. Los servicios web [2] se convierten en una excelente propuesta para optimizar este tipo de sistemas.

El modelo de referencia de workflow define un conjunto de WAPIs (Workflow Aplication Programming Interfaces) para la gestión de los mismos, y define interfaces para establecer la comunicación del motor del workflow. La interfaz de aplicaciones invocadas [3] está implementada de tal manera que para la invocación de una aplicación distribuida en la red, el motor necesita conocer su dirección exacta y una descripción que la identifique.

Anteriormente, los autores de este trabajo propusieron usar servicios web para optimizar esta invocación de aplicaciones, logrando una especificación que mejora la comunicación de las aplicaciones invocadas con el motor del workflow [4]. Así, el usuario no necesita conocer la ubicación exacta de la aplicación a invocar, y las aplicaciones pueden variar su ubicación en la red sin cambios en su invocación. Luego, se presentó un nuevo inconveniente porque el protocolo UDDI [5], usado para registrar y localizar los servicios web en la red, requiere la descripción sintáctica del servicio que se quiere invocar y si hay más de una aplicación que brinda el mismo servicio, es necesario definir en tiempo de desarrollo cual invocará.

A partir de esta problemática, se elaboró una propuesta [6] con el fin de permitir que el motor del workflow tenga la posibilidad de elegir entre varios servicios web que resuelven el mismo requerimiento. Usando reglas de transformación de grafos [7], es posible especificar de manera formal y precisa la semántica del servicio web, permitiendo establecer una correspondencia entre los requerimientos del usuario y los servicios web disponibles, y que el motor del workflow pueda elegir en el momento de la invocación.

La formalización de la especificación de un servicio web como un grafo con atributos, permite realizar una correspondencia con los servicios web disponibles en la red que cumplen con el morfismo establecido, logrando que el workflow se comporte internamente de forma distribuida. Además, facilita la incorporación de servicios, y la invocación no está limitada a una aplicación específica.

Asimismo, un servicio web está descripto [8] por sus características funcionales y no funcionales. La descripción funcional define el comportamiento del servicio, mientras que la no funcional especifica los atributos de calidad (QoS) que verifica dicho servicio.

Con la incorporación de los atributos de calidad (QoS) a la descripción de los servicios web se hace posible evaluar diferentes características y compararlos, con el fin de obtener el servicio que más se adecua a los requerimientos del solicitante.

Para optimizar la selección del servicio web que satisfaga tanto la funcionalidad como los atributos de calidad requeridos por el solicitante, siempre en el marco de optimizar la invocación de aplicaciones desde un sistema de gestión de workflow, se describe la funcionalidad y los atributos de calidad del servicio web con reglas de transformación de grafos, a los fines de establecer una correspondencia entre los requerimientos del usuario y los servicios web disponibles. El presente trabajo describe esta propuesta a través de un caso de estudio basado en OpenUP/Basic [9].

En la sección 2 se describe brevemente la propuesta que incluye la especificación de la interfaz de las aplicaciones invocadas del modelo de referencia de workflow usando servicios web y reglas de transformación de grafos. La sección 3 describe el contexto del caso de estudio con el proceso de desarrollo OpenUP/Basic. Y la sección 4 muestra la implementación de la propuesta sobre el caso de estudio. Por último, se presentan conclusiones de este trabajo.

2 Invocación de aplicaciones con servicios web y reglas de transformación de grafos

Para describir un servicio web es necesario definir la dirección donde se ubica, las operaciones a invocar, los parámetros de entrada y sus tipos, y los distintos formatos de mensajes. Para ello se utiliza el lenguaje WSDL (Web Service Description Language) [10] [11] que permite describir las funcionalidades del servicio. Se propone incorporar las características de calidad (QoS) a la especificación de un servicio web, utilizando este mismo lenguaje.

Las características de calidad de un servicio web, tales como disponibilidad, tiempo de respuesta o fiabilidad se refieren a la habilidad del mismo para responder satisfactoriamente a las solicitudes del usuario del workflow [8].

La incorporación de las características de calidad a la descripción de los servicios web, es tenida en cuenta al momento de realizar la correspondencia entre los requerimientos del usuario y la selección automática de los servicios web disponibles.

Esta correspondencia se realiza usando reglas de transformación de grafos [7], por ser una técnica gráfica, formal y declarativa muy utilizada en transformación y simulación de modelos y chequeos de consistencia entre modelos.

La propuesta consiste en utilizar la técnica de reglas de transformación de grafos para definir la funcionalidad del servicio web, a través de la especificación de la pre y poscondición del servicio, y la pre y poscondición de los distintos proveedores. Luego, se analiza la correspondencia entre los grafos construidos. Esta especificación, contiene el tipo de aplicación, los parámetros, los requerimientos adicionales y se incorporan los requerimientos no funcionales del solicitante. El solicitante detalla los atributos de calidad requeridos en el documento WSDL que describe el servicio.

Además, el motor del workflow llevará una historia de las ejecuciones o invocaciones previas. Para realizar una invocación tendrá que considerar, por un lado, las características de calidad de los servicios ofrecidos por los proveedores, y obtener una lista de aquellos que satisfacen las funcionalidades y características de calidad requeridas por el solicitante. Por otro lado, analizar cuáles fueron los servicios que en ocasiones anteriores, y bajo las mismas condiciones, se eligieron para invocarse.

Esta propuesta es desarrollada completamente a continuación a partir de un caso de estudio.

3 Open UP/Basic

OpenUP/Basic [9] es un proceso de desarrollo de software para proyectos de código abierto, diseñado para pequeños equipos bien organizados, con necesidad de realizar un desarrollo ágil. Los miembros del equipo se reúnen diariamente para comunicar el estado de sus trabajos, toman decisiones en cuanto a las tareas a realizar, las prioridades, y la manera de abordar los requerimientos.

El trabajo es listado, seguido y asignado a través de la *lista de ítems de trabajo*. Este único repositorio contiene todas las tareas que necesitan ser registradas y seguidas, incluyendo requerimientos de cambio, errores y requerimientos de los

interesados en el proyecto. Los miembros del equipo describen los requerimientos usando *casos de uso*.

Las habilidades del equipo de desarrollo se representan por roles: stakeholder, analista, arquitecto, desarrollador, responsable de pruebas y gestor de proyecto.

Las disciplinas que organizan las tareas son: *Análisis y Diseño*, para crear un diseño de los requisitos, *Configuración y Gestión de Cambios*, que describe el control de cambios en los artefactos, asegurando la evolución sincronizada de los productos de trabajo que componen el sistema, *Implementación*, para desarrollar una solución técnica conforme al diseño y a la arquitectura, *Gestión del Proyecto*, para mantener al equipo focalizado en la entrega continua de productos de software, ayudar a priorizar la secuencia de trabajo y crear un ambiente de trabajo eficiente para maximizar la productividad del equipo, informar sobre el progreso del proyecto, y proveer un marco para la gestión de riesgos del proyecto, *Requerimientos*, que define las tareas mínimas para especificar, validar y gestionar los requerimientos del sistema y *Prueba*.

La figura 1 grafica el ciclo de vida de OpenUP/Basic, como un proceso iterativo distribuido. Posee cuatro fases: Inicio, Elaboración, Construcción y Transición. Cada fase consiste de una o más iteraciones, donde se trabaja en versiones estables del software que son desarrolladas y liberadas. El ciclo de vida del proyecto provee 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.

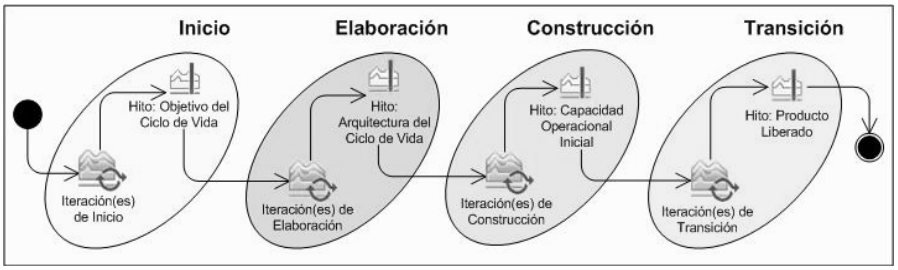

Fig 1. Fases en el ciclo de vida en OpenUP/Basic.

4 Los Servicios Web en la Invocación de aplicaciones externas en OpenUP/Basic

Se plantea el uso de servicios web seleccionados automáticamente en la invocación de aplicaciones externas al workflow, a través de su aplicación en el marco de las tareas involucradas en el proceso de desarrollo de software OpenUP/Basic.

En OpenUP/Basic, la disciplina *Gestión del Proyecto* plantea entre sus tareas principales la planificación del proyecto y la planificación de las iteraciones, dando lugar a los productos de trabajo *Plan de Proyecto* y *Plan de Iteración*, respectivamente.

El *plan de proyecto* reúne toda la información necesaria para gestionar el proyecto, y su propósito es que cualquier miembro del equipo pueda encontrar en este documento la información sobre cómo se administrará el proyecto. Define los parámetros para el seguimiento del avance y los objetivos de alto nivel de las

iteraciones y sus hitos. Además, define cómo esta organizado el proyecto y qué funciones se asignan a cada miembro del equipo. El director del proyecto es el responsable de desarrollar el plan del proyecto, trabajando muy estrechamente con el resto del equipo. El plan del proyecto permite a los miembros del equipo y demás interesados tener un panorama general del proyecto y saber cuándo se espera alcanzar un nivel de funcionalidad.

El nivel de detalle y formalidad de este documento dependerá de las características y necesidades específicas del proyecto de que se trate. Un documento más complejo podría basarse en diagramas de Gantt o Pert utilizados para realizar la planificación.

El objetivo de la planificación del proyecto de software es proporcionar un marco de trabajo que permita al director realizar estimaciones razonables de recursos, costos, y planificación temporal. Estas estimaciones se hacen en un tiempo limitado al comienzo del proyecto, y deben actualizarse regularmente a medida que progresa el mismo.

El modelo de estimación COCOMO utiliza fórmulas derivadas empíricamente para predecir esfuerzo, costos y duración como una función de las líneas estimadas de código y los puntos de función (medida indirecta orientada a la funcionalidad del sofware) [12]. La jerarquía de modelos de Boehm está constituida por tres modelos: básico, intermedio y avanzado. Estos modelos se definen para tres tipos de proyectos: orgánico (proyectos pequeños), semiacoplado (proyectos intermedios en tamaño y complejidad) y empotrado (proyectos muy complejos).

Una organización dedicada al desarrollo de software que utilice OpenUP/Basic como proceso de desarrollo necesitará realizar la planificación de sus proyectos. Para ello deberá estimar la duración total del proyecto, los costos y los esfuerzos. La organización podría realizar estas estimaciones en base al modelo COCOMO antes descripto, para lo cual existen muchas herramientas libres disponibles en la web que implementan este modelo.

Para realizar esta actividad el motor workflow podría invocar una aplicación externa que se encargue de estimar, a partir de la información provista por la organización sobre el proyecto en cuestión. La estimación 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 con los servicios web disponibles que satisfacen dichos requerimientos. Para ello, deberá transformar la formulación del requerimiento en grafos con atributos, para luego compararlos con los grafos que describen los servicios web ofrecidos por el proveedor.

4.1 Grafos de los servicios web del solicitante y los proveedores

Los grafos de las Figuras 2 y 3 representan instancias (diagrama de objetos) de los diagramas de clases que modelan la pre y poscondición del servicio web que representa la aplicación requerida por la solicitante. En la Figura 2, los objetos p1, p2, p3 y p4 de tipo Parameter, almacenan los parámetros necesarios para realizar la estimación, es decir, la información necesaria para poder invocar la aplicación.

El objeto p1 almacena la cantidad estimada de líneas de código del software. El objeto p2 almacena el monto establecido por desarrollador. Los objetos p3 y p4 se

refieren al modelo de COCOMO elegido y el tipo de proyecto, en este caso el básico y orgánico. El objeto rpta:QoSReq especifica que el solicitante de la aplicación requiere que el tiempo de respuesta esté entre los valores 0,5 y 0,7 y el objeto SO:AditionalReq indica que el sistema operativo requerido es Windows XP.

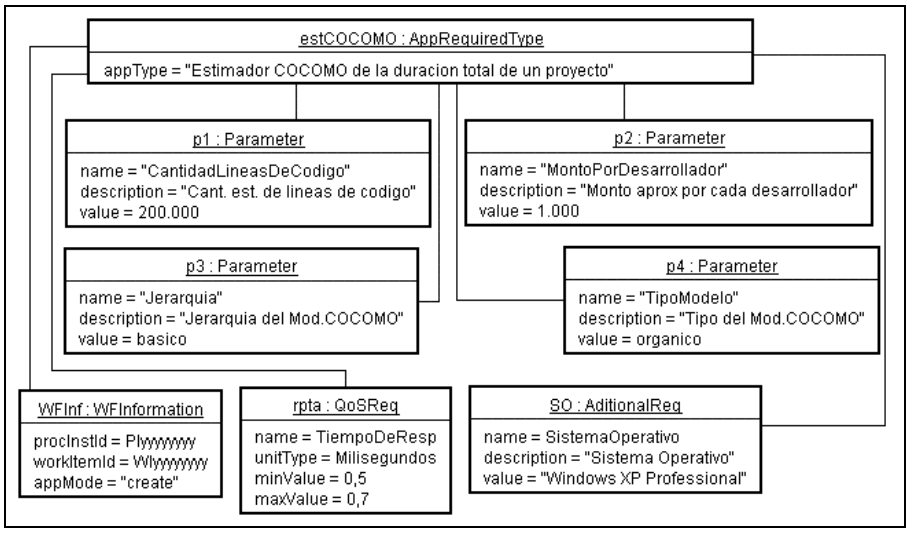

Fig 2: Grafo de la precondición del solicitante

En la Fig. 3, el objeto a:AppFound, no especifica valor para sus atributos, ya que el solicitante al inicio no conoce los servicios que cumplirán sus requerimientos.

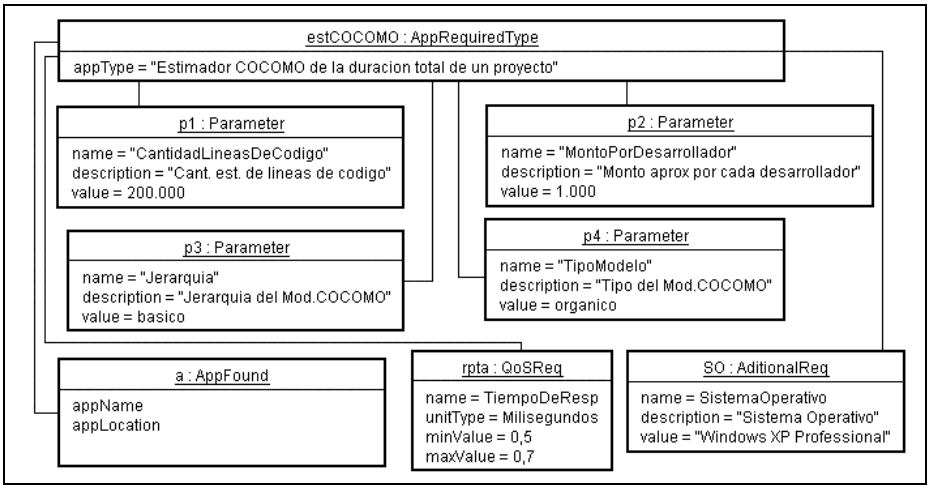

Fig 3: Grafo de la poscondición del solicitante

Los grafos de las figuras 4, y 5 instancian los diagramas de clases que modelan la pre y poscondición del proveedor que ofrece un servicio, que permite invocar una aplicación de estimación según el modelo COCOMO. En la Fig. 4, los objetos

:QoSReq y :AditionalReq almacenan la información del tiempo de respuesta y el sistema operativo del servicio web ofrecido por el proveedor.

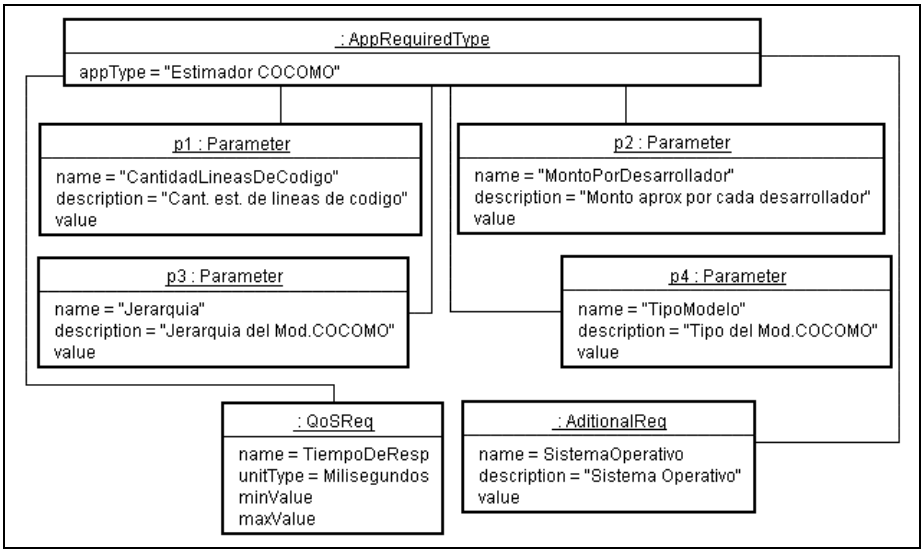

Fig 4: Grafo de la precondición del proveedor

Por su parte, en la Figura 5, los objetos :QoSReq y :AditionalReq almacenan la información del proveedor sobre el tiempo de respuesta y el sistema operativo de su servicio web ofrecido. El objeto :QoSReq indica que el tiempo de respuesta para invocar la aplicación ProyectoCocomoApplication es de 0,7 milisegundos.

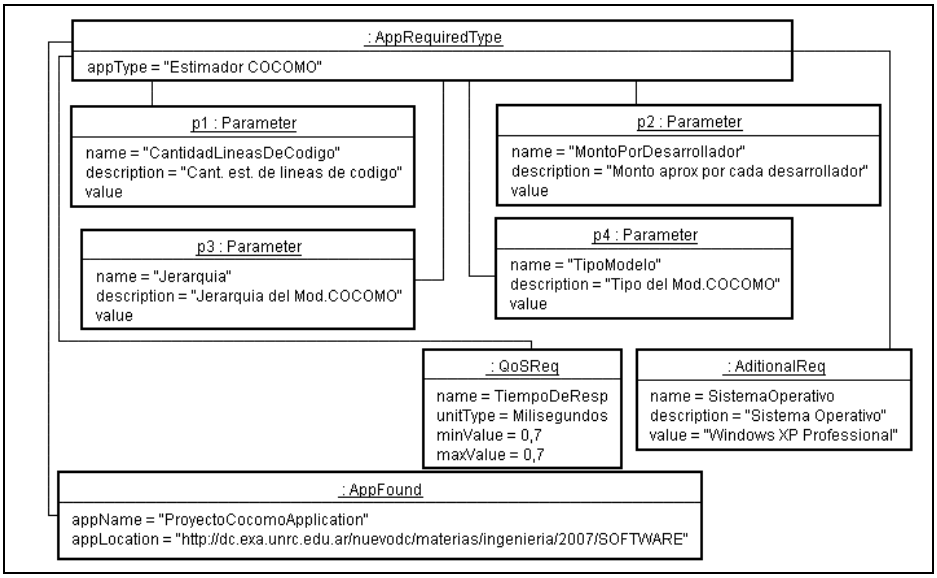

Fig 5: Grafo de la poscondición del proveedor

4.2 Correspondencia entre los grafos

Para decidir automáticamente si el servicio web ofrecido por el proveedor satisface la demanda del solicitante, es necesario comparar los grafos generados por ambos. Se formaliza 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.

En los dos grafos superiores de la Figura 6 se observan las precondiciones del solicitante y del proveedor. La del proveedor contiene siete objetos, :AppRequiredType, p1, p2, p3 y p4: Parameter, :AditionalReq y :QoSReq. Por su parte, la precondición del solicitante, posee estos mismos siete objetos, más el objeto :WFInformation, el cual almacena información que sólo es importante para el motor workflow. Claramente, la precondición del proveedor contiene todos los objetos de la precondición del solicitante, por lo tanto, es un subgrafo.

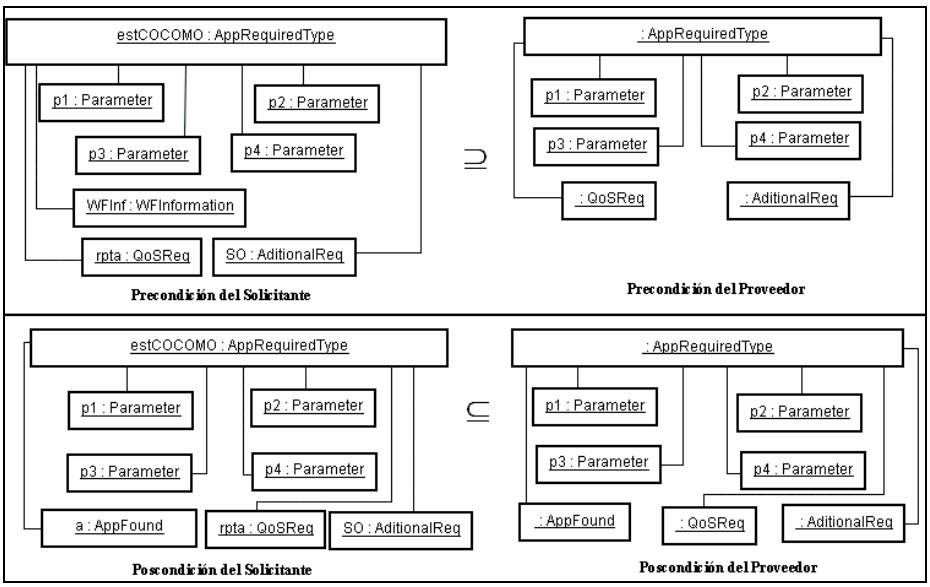

Figura 6: Grafos de la pre y poscondición del solicitante y el proveedor

Por otro lado, 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 podrían ser los errores posibles.

En los dos grafos inferiores de la Figura 6, se observa que la poscondición del solicitante contiene todos los objetos especificados en la poscondición del proveedor (AppRequiredType, Parameter, AditionalReq, QoSReq, AppFound).

Entonces, también se cumple que la poscondición del solicitante contiene todos los objetos de la poscondición del proveedor, por lo tanto, es un subgrafo.

Como se cumple con el morfismo establecido, entonces es posible asegurar que el proveedor cumple los requerimientos del solicitante, tanto los funcionales como los

no funcionales. Así, la aplicación ofrecida por el proveedor es una aplicación adecuada para estimar la duración total del proyecto, costos y esfuerzos, y puede utilizarse durante la planificación del mismo.

Otros elementos que tiene el motor workflow para determinar cuál es la aplicación que invocará, en el caso de haber más de una que satisface los requerimientos, es la historia de las ejecuciones previas y las características de calidad del propio motor workflow. La información adicional especificada por los proveedores, como por ejemplo otras características de calidad que en principio no son requeridas por el solicitante, se confronta con las características de calidad del motor workflow y sirven como otro elemento para decidir cuál servicio web es el más adecuado.

Si finalmente siguen quedando las dos o más aplicaciones como candidatas 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, etc.) y devolver la aplicación que lo cumpla.

5 CONCLUSIONES

En sistemas de gestión de workflow, a menudo, el usuario requiere la invocación de aplicaciones externas para desarrollar sus tareas. Los servicios web se convierten en una excelente propuesta para optimizar este tipo de sistemas.

La técnica de reglas de transformación de grafos permite lograr una especificación semántica precisa de un servicio web y analizar la correspondencia entre la aplicación requerida por el usuario del workflow y los servicios ofrecidos por los distintos proveedores, especificados con servicios web.

Un importante aporte de esta propuesta es la posibilidad de incorporar las características de calidad en la misma especificación de las funcionalidades del servicio web, con la ventaja de que no se requiere modificar el modelo de servicio web actual definido por el protocolo UDDI.

En este trabajo se ha presentado un caso de estudio que ejemplifica esta propuesta. Este ejemplo demuestra en una situación concreta el comportamiento del motor workflow cuando necesita invocar una aplicación externa, de la cual desconoce su ubicación.

Con esta propuesta, es posible establecer un morfismo entre los grafos de los requerimientos del usuario y los servicios web disponibles, con el objeto de que el motor workflow en el momento de la invocación de un servicio, pueda elegir el aquel que más se adecua al requerimiento del solicitante. Esto otorga mayor flexibilidad al motor workflow, a la hora de invocar aplicaciones externas al sistema de gestión de workflow de la organización, y favorece la distribución de las aplicaciones en la red.

Referencias

1. Workflow Management Coalition. The Workflow Reference Model. WfMC-TC00-1003. http://www.wfmc.org/standards/referencemodel.htm. Último acceso Mayo 2008.

2. World Wide Web Consortium. Web Service Architecture. http://www.w3.org/TR/ws-arch/. Último acceso Mayo 2008.
3. Workflow Management Coalition. Programming Interface 2&3 Specification. WfMC-TC-1009.V2.0. http://www.wfmc.org/standards/publicdocuments.htm. Últ. acceso Mayo 2008.
4. Debnath Narayan, Martellotto Paola, Daniele Marcela, Riesco Daniel, Montejano Germán. 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.
5. OASIS. UDDI Version 3.0.2. http://uddi.org/pubs/uddi_v3.htm. Últ. acceso Mayo 2008.
6. Daniele Marcela, Martellotto Paola, Riesco Daniel. Implementación de la comunicación entre el Workflow y las Aplicaciones Externas con Servicios Web y Reglas de Transformación de Grafos. Workshop de Ingeniería de Software y Base de Datos, CACIC 2008. Universidad Nacional de Chilecito. Chilecito, La Rioja, Argentina. 6-10/oct/2008.
7. 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.
8. Papazoglou M., Web Services: Principles and Technology. Ed. Pearson Education Limited, Prentice Hall, First Edition. 2008.
9. Eclipse Process Framework (EPF) Community. Introducción a OpenUP/Basic. Versión 2. http://epf.eclipse.org/wikis/openupsp/. Último acceso en Agosto de 2008.
10. 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 2008.
11. 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 2008.
12. Pressman, Roger. Software Engineering: A Practitioner's Approach. Fifth edition. ISBN: 0071181822. McGraw Hill. 2001.