Especificación de la Interfaz de Aplicaciones de Cliente del Modelo de Referencia de WorkFlow utilizando Web Services

Paola Martellotto[1] **Marcela Daniele**[1] **Daniel Riesco**[2]
[1] 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
[2] Universidad Nacional de San Luis
Departamento de Informática
Tel: + 54 (0) 2652 – 424027 ext. 251 / Fax: + 54 (0) 2652 – 430059
driesco@unsl.edu.ar

Resumen

La globalización y los cambios de paradigmas empresariales, la evolución de las tecnologías de la información y la política liberal imperante hoy en el mundo, ubican a las organizaciones en el juego de la competitividad internacional. Las organizaciones se orientan a ser más horizontales hacia el enfoque de redes de procesos. Un conocimiento claro y ordenado de los procesos de negocios facilita su optimización y adaptación. Los sistemas de administración de Workflow definen, crean y administran la ejecución de procesos de negocio a través del uso de software que se ejecuta sobre uno o más motores Workflow. La WorkFlow Management Coalititon ha desarrollado un Modelo de Referencia de Workflow identificando las interfaces con las estructuras genéricas de las aplicaciones de Workflow.

Por otro lado, los Web Services 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 del mismo.

En este trabajo se propone una especificación de la Interfaz de las Aplicaciones de Cliente, del Modelo de Referencia de Workflow, utilizando Web Services. Con esta especificación el WorkFlow no debe conocer la ubicación exacta de la aplicación que desea invocar para poder hacerlo, y las aplicaciones pueden cambiar su ubicación en la red sin que esto implique ningún cambio en su invocación.

1. Introducción

Los sistemas de Administración de Workflow se están desarrollando en las empresas como una nueva forma de organizar y administrar la información, ayudando a automatizar los procesos de trabajo. Tales sistemas pueden predefinir un procedimiento de trabajo con la información relevante, el rol de los participantes en cada paso de trabajo, y la aplicación de software requerida para procesar cada paso. Cada nuevo caso se asigna automáticamente a los participantes en una secuencia concreta, la información se entrega a la gente que la necesita y las aplicaciones se ejecutan cuando es necesario.

Para lograr cierto nivel de interoperabilidad entre los diversos productos de Workflow desarrollados en las empresas, es necesario definir un conjunto de interfaces y formatos para el intercambio de datos entre dichas componentes. La WorkFlow Management Coalititon (WFMC) [1] ha desarrollado un Modelo de Referencia de Workflow [3] identificando las interfaces con las estructuras genéricas de las aplicaciones de Workflow, para permitir a los productos comunicarse a distintos niveles. En particular, para permitir la interacción de los usuarios con el motor de WorkFlow utiliza una WorkList, que es manejada por un adminitrador. La Interfaz de las Aplicaciones de Cliente es la encargada de manejar la interacción entre el motor de Workflow y el administrador de la WorkList. La WFMC ha especificado un conjunto de APIs (Aplication Programming Interfaces) para la administración de WorkFlows [4], las cuales están soportadas por los productos WorkFlow y se denominan WorkFlow Aplication Programming Interfaces (WAPIs).

El ***Modelo de Referencia de Workflow*** [3] fue desarrollado desde estructuras genéricas de aplicaciones de Workflow, identificando las interfaces con estas estructuras, para permitir a los productos comunicarse a distintos niveles. Todos los sistemas de Workflow contienen componentes genéricos que interactúan de forma predefinida. Para poder tener cierto nivel de interoperabilidad entre los diversos productos de Workflow, es necesario definir un conjunto de interfaces y formatos

para el intercambio de datos entre dichos componentes. En el modelo adoptado hay una separación entre los procesos y el control de la lógica de las actividades. Esta lógica esta dentro del Workflow Enactment Service y permite la integración de las diversas herramientas con una aplicación particular.

Por otro lado, los Web Services 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 del mismo [2]. Para la descripción de un Web Service se utiliza WSDL (Web Service Description Language), basado en XML. Además, es necesario describir los mensajes entre las aplicaciones y el web service, y la forma en que los mismos serán transportados a través de la web. SOAP (Simple Object Access Protocol), 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 web service, para lo cual se define un directorio de web services 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). El uso de estos protocolos estándares permite lograr la interoperabilidad en ambientes heterogéneos, con independencia del Sistema Operativo, lenguaje de programación, etc.

Las implementaciones conocidas de la Interfaz de las Aplicaciones de Cliente [4], definen las funciones de las interfaces y su especificación, como llamadas a APIs en el lenguaje "C", y para la invocación de una aplicación se requiere conocer específicamente su ubicación. En este trabajo se propone una especificación de dicha Interfaz utilizando Web Services, con el objetivo de que el usuario del WorkFlow no necesite conocer la ubicación de la aplicación que desea invocar, y que cualquier aplicación pueda cambiar su ubicación en la red sin que esto implique ningún cambio en su invocación.

2. Estado del Arte

En [5] 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. Los primeros trabajos de la WFMC (sobre WAPIs) se concentran en la definición de las funciones de las interfaces y su especificación como llamadas a APIs en el lenguaje "C". La especificación de la interoperabilidad se desarrolla subsecuentemente usando IDL para la especificación abstracta y bindings concretos basados en MIME, para usar via Internet. En esta propuesta el manejo de las consultas WAPI se reemplaza por el uso de una colección de objetos OLE y se define un filtro para reemplazar el filtro WAPI. Se define una especificación jFLOW usando la notación UML. En [6] se presenta la especificación JointFlow, adoptada por 19 compañías en respuesta a OMG's Workflow Management Facility RFP. Esta especificación se basa en el trabajo de la Workflow Management Coalition y define las interfaces que soportan la interacción en tiempo de ejecución entre los componentes de WorkFlow, permite la interoperabilidad de los componentes de negocio a través de los dominios de negocio y permite el monitoreo de los procesos. La especificación JointFlow focaliza en definir un conjunto de interfaces que permiten la interoperabilidad de los componentes de WorkFlow, soporta el monitoreo y la asignación de recursos. En [7] los autores presentan una especificación utilizando Grid Computing. En los WFMS tradicionales los procesos son diseñados por herramientas de definición de procesos y ejecutados por el motor WorkFlow, en el mismo WFMS. Las tareas son desarrolladas por los usuarios finales o las aplicaciones. Los procesos solamente pueden ser ejecutados por motores que son accedidos por usuarios específicos y aplicaciones específicas. Así, utilizar WFMS es restrictivo de las localizaciones. En esta propuesta las tareas son desarrolladas por los servicios Grid (los cuales se basan en un conjunto de interfaces estándar). Los servicios pueden ser accedidos por cualquier aplicación de acuerdo a estos estándares. En [8] se introduce un agente de composición de WorkFlow, que es capaz de componer WorkFlow de Web Services, usar descripciones semánticas de los Web Services y encontrar Web

Services para un WorkFlow. Se muestra cómo un servicio web se puede componer utilizando ontologías de Web Services semánticos, que se refiere a definir la semántica de un servicio web, es decir, su significado, más que sus parámetros de entrada y salida. En [9] se propone encapsular la funcionalidad de las organizaciones en interfaces apropiadas y publicarlas como Web Services. Se espera que los Web Services puedan integrarse como parte de los Procesos Web. Pero esta integración es dificultosa dada la alta heterogeneidad, autonomía y distribución de la web. Una solución es el uso de ontologías. También es esencial para los Web Services soportar todas las fases del ciclo de vida de un proceso web. Se describe cómo el aplicar semántica a cada paso del ciclo de vida de un Proceso Web Semántico puede ayudar al reuso, la integración y escalabilidad. En [10] se presenta un lenguaje de WORKFLOW que usa los Web Services como componentes, una arquitectura para un ambiente en tiempo de ejecución para este lenguaje, y aprovecha las ventajas de la utilización de esta clase de tecnología. El lenguaje propuesto AELCWS no soporta la interacción directa con las personas durante la ejecución de un proceso, de modo que la Interfaz de Aplicaciones de Cliente no esta implementada. La Interfaz de Invocación de las Aplicaciones sí es soportada por este lenguaje, a través de la invocación de los servicios que conforman la definición del proceso.

3. Utilización de Web Services para especificar la Interfaz de las Aplicaciones de Clientes

Como ya se mencionó, la Interfaz de las Aplicaciones de Clientes permite la interacción entre las aplicaciones clientes y el motor de Workflow. Para sostener dicha interacción se utiliza una Worklist, que almacena la información de las aplicaciones que se deben invocar y posee un manejador de la misma. La Worklist puede contener ítems relacionados con diferentes instancias de un proceso o ítems de diferentes procesos. El manejador puede interactuar con diferentes motores.

En [4] se pueden consultar las APIs asociadas a esta interfaz, la cuales proveen un nivel básico de funcionalidad para soportar la invocación de aplicaciones. Las funciones de la WorkList proveen información a los participantes del WorkFlow sobre los trabajos que ellos tienen asignados. Como lo describe el Modelo de Referencia de la WFMC, un proceso consiste de un conjunto de actividades conectadas que permiten controlar el secuenciamiento de la invocación a aplicaciones. Los participantes pueden tener asignadas una o más piezas de trabajo al mismo tiempo, denominadas “work item”. La “WorkList” es la colección de todos los “work item” asignados.

En particular, la función *WMOpenWorkList* especifica y ejecuta una consulta para producir la WorkList que cumple con el criterio de filtro de la consulta. El comando provee la capacidad de retornar la lista de “work items” asignados a una participante de un WorkFlow particular o a un grupo. El solicitante puede consultar qué trabajos le han sido asignados o esperar que la WorkList se los comunique. La especificación WAPI de esta función se puede consultar en [4].

Como se introdujo anteriormente, WSDL [2] es el lenguaje utilizado para describir un Web Service. WSDL describe un Web Service como un conjunto de puntos finales de comunicación (métodos) capaces de intercambiar mensajes. Es un archivo XML que describe el conjunto de métodos expuestos por un Web Service. Todo documento WSDL está compuesto por un elemento raíz llamado definitions, que a su vez está compuesto por los siguientes elementos:

- **types**: define el tipo de esquema a ser utilizado, usualmente XML.
- **message**: aquí se definen los mensajes de entrada y salida en forma abstracta entre el servidor y el cliente.
- **portType**: define los tipos de mensajes a intercambiar entre el cliente y el servidor.
- **binding**: establece el protocolo concreto para las operaciones y mensajes definidos en un portType particular.
- **service**: informa el punto de acceso a los servicios para cada uno de los protocolos a través de un elemento address.

WSDL se basa en el lenguaje ***XML*** (Extensible Markup Language) [2]. Este es el lenguaje utilizado para definir el formato de documentos o mensajes. XML comprende el uso de etiquetas denominadas *tags* que identifican los contenidos de un documento, y al hacerlo, los describen. Una etiqueta XML identifica información dentro de un documento, como así también la estructura de dicha información. Los documentos XML poseen una estructura bien-formada y generalmente están asociados con un esquema (scheme) que especifica qué etiquetas están permitidas dentro de un documento, la estructura de esas etiquetas, y otras reglas relacionadas, tales como el tipo de dato que se espera dentro de una etiqueta.

Por otro lado, ***SOAP*** [2] es utilizado en los Web Services para el transporte de los mensajes. Los mensajes SOAP están compuestos por un tag principal llamado *Envelope*, que está dividido en una cabecera o *Header* y un cuerpo o *Body*. Dentro del elemento *Body* estarán los elementos correspondientes al Web *Method* y además puede haber o no un elemento en común llamado *fault*, que indica que ha ocurrido un error y la razón de este. Por lo tanto, la definición de un Web Service implica, entonces, construir un documento WSDL que contenga los elementos mencionados anteriormente.

3.1. Definición de la función WMOpenWorkList con Web Service

Para el ejemplo de la función WMOpenWorkList, uno de los elementos a definir son los tipos de mensajes que usará el servicio. Los tipos que son más complejos, se declaran con esquema XML. Para la función WMOpenWorkList, se define un tipo para especificar el criterio de filtro de la consulta para un requerimiento específico.

```
<types>
   <xs:schema
      …
      <xs:element name="pworklist_filter" type="WMTPFilter"/>
         <xs:sequence>
            <xs:element name="sqlString"          type="xs: String"/>
            <xs:element name="attributeName"      type="xs: String"/>
            <xs:element name="comparison"         type="xs: int"/>
            <xs:element name="attributeValue"     type="xs: boolean"/>
         </xs:sequence>
      </xs:complexType>
  </xs:schema>
</types>
```

WSDL permite separar la descripción de la funcionalidad abstracta de un servicio Web de los detalles concretos de cómo y dónde la funcionalidad es ofrecida. Una interfaz WSDL define la interfaz abstracta de un servicio Web como un conjunto de operaciones abstractas.

```
<interface name = "WMOpenWorkListInterface" >
        <fault name = "WMOpenWorkListFault_InvalidFilter"
              element = "wfms: WMInvalidFilter"/>

        <operation name="opWMOpenWorkList"
            pattern=http://www.w3.org/2004/03/wsdl/in-out>
            <input messageLabel="In"      element="wfms: psession_handle" />
            <input messageLabel="In"      element="wfms: pworklist_filter" />
            <input messageLabel="In"      element="wfms: count_flag" />
            <output messageLabel="Out"    element="wfms: pquery_handle" />
            <output messageLabel="Out"    element="wfms: pcount" />
            <outfault ref="tns: WMOpenWorkListFault_InvalidSessionHandle" messageLabel="Out"/>
            <outfault ref="tns: WMOpenWorkListFault_InvalidFilter" messageLabel="Out"/>
   </operation>
</interface>
```

Otro elemento a especificar es el binding, que permite detallar cómo los mensajes pueden ser

intercambiados. Especifica detalles del formato concreto de mensajes y del protocolo de transmisión para una interfase, y provee tales detalles para cualquier operación y falla en la interfaz.

```
<binding name="WMOpenWorkListSOAPBinding"
    interface="tns: WMOpenWorkListInterface"
    …/>

   <operation ref="tns: opWMOpenWorkList"
    wsoap:mep=…/>

    <fault ref="tns: WMOpenWorkListFault_InvalidFilter"
    wsoap:code="soap:Sender"/>
</binding>
```

Finalmente, resta definir el servicio WMOpenWorkList. Esta definición implica especificar dónde el servicio puede ser accedido, mediante el uso del elemento service. Un servicio WSDL especifica una interfaz simple que soportará el servicio, y una lista de ubicación de puntos extremos (endpoints) donde ese servicio puede ser accedido.

```
<service name="WMOpenWorkListService"
    interface="tns: WMOpenWorkListInterface">
    <endpoint name="WMOpenWorkListEndpoint"
          Binding = "tns: WMOpenWorkListSOAPBinding"
          address = …/>
</service>
</description>
```

4. Conclusiones

Un proceso de negocio es un conjunto de tareas relacionadas lógicamente que se ejecuta con la intención de obtener un resultado de negocio particular, el cual incluye recursos humanos y recursos materiales. El modelado de procesos de negocio a través de los sistemas de administración de Workflow permite definir, crear y administrar la ejecución de procesos de negocio a través del uso de software que se ejecuta sobre uno o más motores Workflow. El Modelo de Referencia de WorkFlow de la WFMC identifica las interfaces con las estructuras genéricas de las aplicaciones de Workflow y define APIs para implementar dichas interfaces. En este trabajo se propone aprovechar los beneficios de los Web Services, utilizándolos para especificar las funciones de la Interfaz de las Aplicaciones de Cliente, facilitando la comunicación del WorkFlow con las aplicaciones. El WorkFlow se comporta internamente de forma distribuida, es decir, no necesita conocer dónde están las aplicaciones para poder invocarlas. Simplemente, requiere servicios y hay aplicaciones que le proveen dichos servicios. La implementación de las interfaces del WorkFlow se vuelve independiente del lenguaje de programación, y de la plataforma subyacente.

5. Referencias Bibliográficas

[1] Workflow Management Coalition. http://www.w3.org/
[2] World Wide Web Consortium. http://www.w3.org/
[3] Workflow Management Coalition, The WorkFlow Reference Model. WfMC-TC00-1003. 1995.
[4] Workflow Management Coalition, Programming Interface 2&3 Specification. WfMC-TC-1009. V2.0. 1998.
[5] Workflow Management Coalition, A Common Object Model Discussion Paper. WfMC-TC10-22. 1998.
[6] Schmidt M.T., Building Workflow Business Objects. IBM Soft. Group OOPSLA'98 Business Object Workshop IV.
[7] 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.
[8] Laukkanen M., Helin H.: Composing Workflows of Semantic Web Services. Workshop on Web Services and Agent-based Engineering. AAMAS'2003. Melbourne, Australia. 14/15 de Julio de 2003.
[9] 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.
[10] 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.