SFDL: definición de vistas dinámicas optimizada para flujos de trabajo

Emilio García, Diego Moreno, Sandra Aguirre, Juan Quemada
Departamento de Ingeniería de Sistemas Telemáticos (DIT)
Universidad Politécnica de Madrid
ETSI de Telecomunicación. Av. Complutense, s/n. 28040 – Madrid
{egarcia, dmoreno, saguirre, jquemada}@dit.upm.es

***Resumen*- La gestión de procesos basada en sistemas de workflow es una tendencia creciente en los entornos colaborativos. Así, cualquier optimización en las tecnologías que facilitan dicha labor, como los flujos de trabajo (workflows), multiplica los beneficios aportados a la colaboración entre individuos. Con el objetivo de mejorar las técnicas de workflow se ha diseñado un nuevo lenguaje de definición de vistas dinámicas denominado SFDL, orientado a la adaptabilidad en distintos entornos, con representaciones en diferentes formatos y pensado para su fácil integración en distintas arquitecturas. Para la validación del diseño expuesto se ha llevado a cabo su implementación en un escenario real, recibiendo realimentación y refinando las especificaciones. El trabajo se ha basado en el uso de estándares ampliamente usados en el ámbito web (XML, YAML, JSON, Atom y REST). Además, en el presente artículo se dan directrices que facilitan la adopción de la solución.**

***Palabras Clave*- Flujo de trabajo (*workflow*), SFDL, trabajo colaborativo, interfaz de usuario (*user interface*)**

I. INTRODUCCIÓN

Los flujos de trabajo (*workflow*) como herramienta para la gestión de grupos de trabajo en un entorno colaborativo ha tomado una papel creciente en el éxito de las empresas, habitualmente involucrando equipos de trabajo a través de distintas organizaciones. La investigación desarrollada en este artículo se ha centrado en la mejora de los sistemas de workflow basados en una interacción web con los usuarios. Se ha diseñado un lenguaje de definición de vistas de usuario de generación dinámica pensado para integrarse fácilmente dentro de un motor de ejecución de workflow. Dicho lenguaje de definición, denominado *Simple Form Definition Language* (SFDL) [1], tiene dos características claves:

- Es representable en tres posibles variantes, XML, YAML y JSON, que le permiten adaptarse y ser óptimo en muchos escenarios.
- Integra la ejecución de funciones, tanto en cliente como servidor, lo que le confiere la capacidad de integrarse en la lógica de un motor de ejecución de workflow.

Además de la especificación formal del lenguaje, en el presente artículo se propone una arquitectura de integración de SFDL en un sistema de ejecución de workflow genérico, en el marco de un entorno bancario. En este entorno, los workflows se tienen que poder desarrollar ágilmente, otorgando a los diseñadores el control de todo el proceso, desde la funcionalidad hasta el diseño de la interfaz para el usuario final.

Toda la propuesta es totalmente compatible con el Modelo de Referencia [2] de la WfMC para su fácil integración en otros sistemas. Además, se basa en estándares como Atom, REST y Wf-XML-R.

El resto del artículo está organizado de la siguiente manera: la Sección II describe el estado del arte y los trabajos previos relacionados con la investigación. En la Sección III se ilustra el entorno de implantación objetivo al que va dirigida la propuesta planteada. La especificación formal de SFDL es introducida en la Sección IV. La Sección V presenta los aspectos a tener en cuenta para la integración de SFDL en un motor genérico de workflow. La validación de toda la propuesta se relata en la Sección VI. Y por último, en la Sección VII se enumeran las conclusiones.

II. ESTADO DEL ARTE

La solución propuesta en este trabajo se centra en la evolución de la interacción con el usuario, incorporando enlaces a la gestión interna de los flujos de trabajo.

Los formularios son, hoy en día, el mecanismo más común y sencillo para permitir el desarrollo de aplicaciones web interactivas. Existen diferentes alternativas para la generación de formularios web: HTML ha sido el lenguaje tradicionalmente usado para la creación de formularios, evolucionando hacia HTML5 [3] para añadir dinamismo; XForms [4] es la recomendación del W3C dentro de la especificación XHTML que permite separar la presentación del contenido, reutilizar esquemas y una reducción de accesos al servidor; MXML [5] es un lenguaje XML usado para los interfaces de usuario en la plataforma Flex de Adobe. Sin embargo, todas las alternativas mencionadas tienen un inconveniente: la dependencia con la sintaxis del lenguaje de script para ejecutar funciones en el lado del servidor.

SFDL se presenta como solución para la integración de la capa de presentación con los sistemas de gestión de workflow, Para ello, se proponen pequeñas modificaciones en las siguientes interfaces del Modelo de Referencia:

- Interfaz 1: el lenguaje de definición de procesos (OpenWFE [6] en nuestro caso, aunque bien podría ser BPEL o XPDL) será extendido para soportar una definición más amplia del workflow, incluyendo definición de vistas e información para el acceso dinámico a los datos.

- Servicio de Ejecución de Workflow: el motor requerirá cambios mínimos para soportar las nuevas características, especialmente aquellas relacionadas con el acceso al modelo de datos.
- Interfaz 2: se ha hecho una propuesta para reusar y extender el protocolo de comunicación basado en Atom de la Interfaz 4, manteniendo la coherencia con la Arquitectura de Referencia.

III. ESCENARIO

Nuestro escenario se desarrolla dentro del proyecto ITECBAN [7], el cual tiene como misión dotar de herramientas de software orientadas a las actividades colaborativas de una organización virtual para la construcción y evolución de los principales sistemas de información de una organización bancaria. ITECBAN debe soportar diferentes actividades colaborativas tales como el proceso de desarrollo de software dentro de un núcleo bancario, videoconferencia, gestión de contenido, etc. El sistema de gestión de workflow debe satisfacer los siguientes requerimientos funcionales:

- Diseño sencillo de formularios que permitan la especificación de tareas, reglas, roles de usuario y tipos de datos de entrada y salida que son necesarios en la ejecución de un workflow.
- Acceso del usuario a través de cualquier navegador web, usando estándares (como REST y Atom).
- Uso de un sistema de gestión de workflow de código abierto.
- Conexión con diferentes bases de datos como MySQL, LDAP –para gestión de roles/usuarios- y CMDB.

En concreto, el trabajo se centra en el diseño de workflows para la gestión de incidencias, cambios, problemas y órdenes de trabajo, con interfaces de usuario sencillos, flexibles, y modificables en cualquier momento.

Teniendo en cuenta los requerimientos descritos, es conveniente realizar una instanciación (Fig. 1) del modelo de referencia WfMC que describe en un alto nivel los principales componentes de un sistema de gestión de workflow, con énfasis en aquellas entidades e interfaces encontradas relevantes para el enfoque presentado en este escenario.

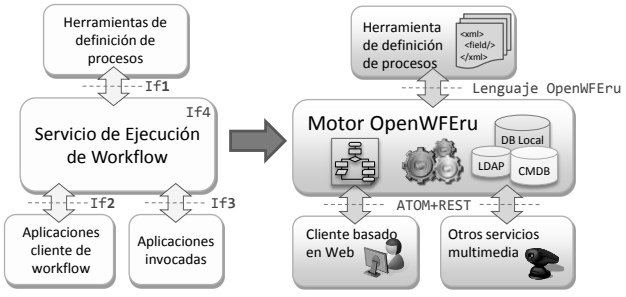

Fig. 1. Instanciación del modelo de referencia WfMC.

IV. SFDL: SIMPLE FORM DEFINITION LANGUAGE

Las extensiones propuestas a la arquitectura de un sistema de workflow están orientadas a la generación de formularios web, para la definición flexible de vistas desde el mismo proceso de diseño; y las operaciones básicas desde el lenguaje para acceder al modelo de datos. De este modo, un diseñador de workflows puede no sólo establecer el patrón de interacción, sino también definir las reglas básicas de la interfaz a través de la cual el usuario final hará uso de toda la funcionalidad subyacente, usando un único lenguaje: SFDL.

A. Definición del lenguaje SFDL

El Lenguaje Simple de Definición de Formularios (*Simple Form Definition Language*, SFDL) es un lenguaje de propósito especial definido teniendo en cuenta todos los requisitos funcionales previamente detallados. En particular:

- Soporta multitud de elementos de formularios web: selectores, tablas, elecciones, campos de entrada/salida…
- Es autocontenido: toda la información necesaria –componentes y estilo– está en un único fichero.
- Múltiples pantallas por vista: una única actividad en un workflow puede estar compuesta por varias pantallas en el cliente, que deben ser completadas antes de enviar la información de vuelta al servidor.
- Puede ser expresado en distintos lenguajes de marcado estándares, lo cual reduce la complejidad en el lado servidor.
- Soporta la ejecución de funciones en el servidor, para gestionar el modelo de datos desde/hacia las vistas.

Frente a otras alternativas de lenguajes de presentación, SFDL tiene ventajas:

- XForms comparte la orientación a diseño de formularios de SFDL, pero carece de un modo sencillo de ejecución de funciones. Además, su complejidad es elevada (usa CSS para definir los estilos) y su soporte en los clientes web actuales es prácticamente inexistente.
- Los formularios HTML5 gozan de un creciente soporte en los navegadores, pero no pueden ser fácilmente serializados para su proceso en JavaScript. Además, HTML no incluye llamadas anidadas a funciones.

Los ficheros en SFDL pueden estar definidos en una de tres posibles variantes: SFDL-X, -J o -Y (en XML, JSON o YAML), los cuales, aun siendo completamente equivalentes en funcionalidad, tienen sus propias particularidades que convierten a cada uno en adecuado para un entorno diferente.

B. Definición de vistas en SFDL

Para definir vistas dentro de las actividades de un workflow se usa un esquema simple basado en etiquetas para indicar la posición, tipo y valor de cada elemento, junto con algunos parámetros que definan con más precisión su estilo o funcionalidad. La Tabla 1 resume todos los campos.

Etiqueta	Descripción	Valores
type	Funcionalidad del elemento	label, input_text, text_area, text_block, selector, table, dynamic_table, link, attach, checkbox
params	Estilo del elemento	halign, width, height, hint…
value	Valor del elemento	Numérico, alfanumérico, función
result	Resultado de la interacción del usuario con el elemento	

Tabla 1. Campos de una vista.

Cada elemento está precedido de un identificador numérico que define la posición que tendrá en la pantalla del usuario. Por ejemplo, la Fig. 2 muestra la definición de un campo de tipo etiqueta en SFDL-Y, estando su valor definido como el resultado de la ejecución de la función *user-data*. El elemento se posicionará en las coordenadas (04, 30).

```
- id: 0430
  type: label
  value:
    function-name: user-data
    attribute-name: telephone
  params:
    halign: left
    width: "60"
```

Fig. 2. Definición de un campo en SFDL-Y.

Es importante destacar en este punto el hecho de que todos los formatos SFDL-* son equivalentes. La Fig. 3 muestra la definición del campo antes mencionado, pero ahora en SFDL-X (XML), con la particularidad de que este formato, una vez haya sido procesada la función, será idéntico al enviado al cliente a través de la Interfaz 2.

```
<field>
  <_>
   <id> 0430 </id>
   <type> label </type>
   <value>
    <_>
     <function-name> user-data </function-name>
     <attribute-name> telephone </attribute-name>
    </_>
   </value>
   <params>
    <halign> left </halign>
     <width> 60 </width>
   </params>
  </_>
</field>
```

Fig. 3. Definición de un campo en SFDL-X.

C. Funciones de acceso al modelo de datos

Uno de los requisitos más importantes de SFDL es ofrecer acceso al modelo de datos desde la definición de los formularios, empleando funciones. En este punto es necesario clarificar los dos modos posibles de ejecución de funciones:

- En tiempo de ejecución de workflows, cuando el motor de procesos ejecuta la definición.
- En tiempo de presentación, cuando se presenta el formulario al usuario final.

Para soportar ambos comportamientos se ha definido un mecanismo para llamar a las funciones tanto desde el lenguaje OpenWFE como desde las definiciones en SFDL, siguiendo un modelo funcional.

1) Funciones en tiempo de workflow

Las llamadas a función desde el lenguaje de definiciones han sido implementadas como referencias a un participante especial, siendo éste uno de los métodos más directos en OpenWFE. No obstante, la generalidad de esta aproximación queda garantizada en el sentido de que todo lenguaje tiene algún modo de añadir funciones externas. Manteniendo la línea del ejemplo de la Fig. 2, la llamada a *functions* para obtener el número de teléfono de un usuario sería la mostrada en la Fig. 4:

```
<participant ref="functions"
             function-name="user-data"
             attribute-name="telephone"
             out-field="phone"/>
```

Fig. 4. Llamada a función definida en un workflow.

Para este trabajo se ha implementado en el motor un conjunto de funciones que cubren las operaciones básicas de entrada/salida desde/hacia el modelo y las bases de datos empleadas en la arquitectura (LDAP, CMDB): *read-attribute*, *write-attribute*, *cmdb-out*, *user-data*… Sin embargo, el mecanismo de definición de funciones, permite fácilmente la extensión del conjunto inicial.

2) Funciones en tiempo de presentación

Las funciones que se ejecutan cuando el usuario abre una vista específica están definidas en el mismo lenguaje que dicha vista: SFDL, en cualquiera de sus variantes (-X, -J o -Y). Algunos ejemplos ya se han visto en la Fig. 2 y la Fig. 3. Esta aproximación tiene dos aspectos positivos de gran utilidad:

- Las funciones pueden ser anidadas y, siendo un lenguaje funcional, pueden ser usadas en cualquier punto de una definición de vista SFDL en lugar de un valor concreto.
- Hay una sola biblioteca de funciones en el sistema, con un único conjunto de nombres y parámetros (es decir, una sola interfaz de llamada), de modo que las llamadas desde SFDL son idénticas a aquellas realizadas desde OpenWFE, dando consistencia a la solución.

V. Implantación en un motor de workflow genérico

Una de las principales premisas en el diseño de SFDL ha sido la independencia con el motor de workflow. Para conseguir esta independencia, se ha reducido el acoplamiento con el motor, limitando los puntos de integración. De esta forma, se consigue la adopción de SFDL en cualquier motor con los mínimos cambios.

El único requisito para la inclusión de SFDL en un motor de workflow es la capacidad de llamar a funciones externas al propio motor en dos momentos distintos: en primer lugar, durante la ejecución del flujo de trabajo y, en segundo lugar, en el momento de presentar información al usuario. El objetivo es poder ejecutar las mismas funciones independientemente del momento. Así, todas las funciones disponibles estarán empaquetadas en una biblioteca común.

Para un motor de workflow con SFDL, una posibilidad muy recomendable es que su comunicación con los clientes se realice a través de una interfaz REST basada en Wf-XML-R [8]. La representación de las variables necesarias para la generación dinámica de formularios va más allá del ámbito de Atom y Wf-XML-R. De esta forma, en el ámbito de esta investigación se ha creado una extensión, con su espacio de nombres propio, para la adopción de SFDL dentro de Atom.

VI. Validación

Nuestro trabajo ha sido validado en el escenario bancario ya presentado. Para los flujos de trabajo definidos dentro de ITECBAN se realizó el proceso de identificar las variables y operaciones a tener en cuenta en la ejecución de los flujos, así como las interfaces de usuario requeridas. Las primeras

fueron codificadas con el lenguaje OpenWFE y ejecutadas con el motor correspondiente. Las interfaces de usuario se especificaron a través del lenguaje SFDL (la Fig. 5 recoge un extracto del código para la primera vista).

```
screens:
  - id: default
    title: "%title.incident.register%"
    fields:
      - id: "0204"
        value:
          function-name: read-attribute
          attribute-name: launcher
        type: label
        params:
          halign: left
      - id: "1001"
        type: result_table
        params:
          data_source: cmdb
          dynamic_columns: true
          height: "110"
      - id: "2405"
        value:
          1:
            id: urgency.high
            name: "%urgency.high%"
          2:
            id: urgency.low
            name: "%urgency.low%"
        type: selector
        params:
          default: urgency.high
[…]
      - id: "2611"
        value: submit
        type: button
        params:
          button_text: "%incident.open%"
          button_result: "open"
          compulsory_fields:
            - "1001"
```

Fig. 5. Definición en SFDL-Y

La Fig. 6 representa la vista o interfaz de usuario para registrar o crear una nueva incidencia generada en el flujo de gestión de incidencias. Esta interfaz corresponde al primer paso del workflow ("Registrar Incidencia").

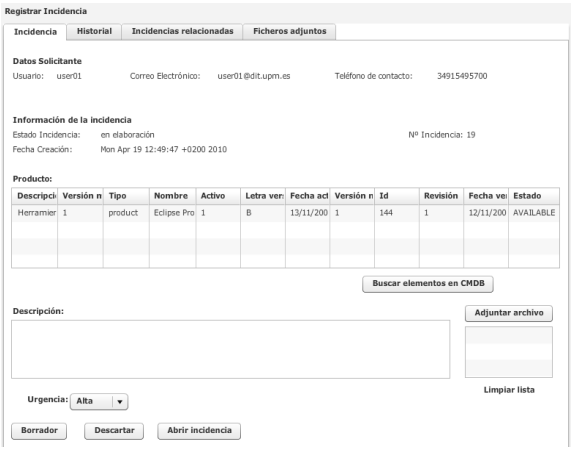

Fig. 6. Pantalla generada en SFDL: Registrar Incidencia.

La pantalla generada a partir de código SFDL tiene un alto grado de funcionalidad, sin sacrificar la usabilidad.

Nuestro enfoque ha permitido a los gestores de workflows del escenario presentado, reducir sus esfuerzos en la creación y ejecución de workflows, permitiéndoles ganar productividad.

VII. Conclusiones

El resultado de esta investigación ha sido la especificación de un nuevo lenguaje de definición de vistas dinámicas, SFDL, optimizado para motores de workflows. Por un lado, a través de la definición de flujos, un motor de workflow puede generar formularios dinámicos con usabilidad mejorada con la posibilidad de interactuar con servicios web y el acceso al modelo de la base de datos. Por otra parte, un usuario con un navegador web puede interactuar con el sistema de workflow a través de estos formularios de workflow dinámicos. Nuestro trabajo se ha basado en el uso de estándares ampliamente aceptados y propuestas abiertas, siendo posible la definición del lenguaje SFDL en XML, YAML y JSON.

La arquitectura propuesta ha recibido bastante realimentación ya que ha sido validada dentro de un escenario bancario real; ha permitido ofrecer a los diseñadores de workflow los mecanismos para incluir especificación de vistas dentro de la definición de procesos. Asimismo se ha permitido establecer una serie de métodos de acceso al completo modelo de base de datos dentro del lenguaje para asegurar que el workflow sea dinámico. Adicionalmente, la arquitectura diseñada es simple, y mantiene la compatibilidad con el Modelo de Referencia (por ejemplo, reutilizando protocolos como el Wf-XML-R), sin renunciar a la portabilidad (evitando introducir profundas modificaciones específicas de una plataforma determinada) y manteniendo la flexibilidad y extensibilidad, las cuales son esenciales en este tipo de proyectos. De esta manera, se recomienda el uso de estos formatos validados y arquitecturas en escenarios con similares requerimientos.

VIII. Agradecimientos

Este trabajo ha sido soportado por el proyecto ITECBAN, el cual ha sido financiado por el CDTI (Centro para el Desarrollo Tecnológico e Industrial) y el Ministerio de Industria, Turismo y Comercio. Asimismo, los autores agradecen a INDRA Sistemas, S.A. (http://www.indra.es/) su valiosa contribución a este trabajo.

Referencias

[1] The SFDL definition: a Simple Form Definition Language. http://sfdl.dit.upm.es/

[2] D. Hollingsworth, "The Workflow Reference Model Version 1.1". Winchester, UK: Workflow Management Coalition, WFMC-TC-1003, Jan. 1995.

[3] W3C HTML 5. Available at http://dev.w3.org/html5/spec/. Mar. 2009.

[4] W3C XForms 1.0 (Third Edition). Available at http://www.w3.org/TR/xforms/. Mar. 2009.

[5] C. Coenraets, "An overview of MXML: The Flex markup language", Adobe Systems. March 2004. Available at http://www.adobe.com/devnet/flex/articles/paradigm.html. Mar. 2009.

[6] OpenWFEru – open source ruby workflow engine. http://openwferu.rubyforge.org/. Last accessed March 2010.

[7] ITECBAN. http://caching.indra.es/sites/default/files/Itecban_baja_0.pdf

[8] M. Zukowski, P. Cappelaere, K. Swenson, "A RESTFul Protocol for Run-Time Integration of Process Engines". Draft 5. Apr. 2008.