Potenciando la Unión entre Workflows y Soluciones EAI

Mario Sánchez
Universidad de los Andes
Bogotá, Colombia
mar-san1@uniandes.edu.co

Eduard Zambrano
Universidad de los Andes
Bogotá, Colombia
e-zambra@uniandes.edu.co

Oscar González
Universidad de los Andes
Bogotá, Colombia
o-gonza1@uniandes.edu.co

Nicolás López
Universidad de los Andes
Bogotá, Colombia
ni-lopez@uniandes.edu.co

ABSTRACT
El cambio que se ha venido presentando de organizaciones basadas en unidades funcionales a organizaciones basadas en procesos de negocio plantea retos tecnológicos importantes para las empresas. Estas organizaciones deben proteger y aprovechar las inversiones realizadas en las aplicaciones que actualmente soportan las actividades productivas, al tiempo que deben aparecer materializados los diferentes procesos de negocio para que puedan ser medidos, controlados, estudiados y mejorados. En este artículo se plantea una posible solución a estos desafíos, la cual está basada en la unión entre un motor de workflow (Cumbia) y una infraestructura de integración de aplicaciones (Eleggua). La unión de estas dos tecnologías permite aprovechar las ventajas que ofrece cada una y lograr coordinar la ejecución de aplicaciones existentes con base en procesos de negocio explícitos y materializados. Dicha solución se ilustra en este artículo usando un proceso de negocio enmarcado en el contexto de procesos de desarrollo de software globalizado.

Palabras Clave
Procesos de negocio, BPM, workflow, integración de aplicaciones.

1. INTRODUCCIÓN
En todo el mundo las empresas están cambiando de una visión de unidades de negocio (orientadas por funciones) a una visión de procesos de negocio transversales a las unidades, y en consecuencia, ha habido cambios en las soluciones informáticas que utilizan [22, 23, 28]. En la orientación a funciones, las aplicaciones fueron desarrolladas con el objetivo de facilitar el procesamiento de la información por los empleados de cada unidad de negocio. Así, la responsabilidad de la integración de los distintos procedimientos quedaba repartida entre las unidades, al igual que el mantenimiento de la coherencia de la información.

Este enfoque llevó a una situación en la que, para que la organización le preste un servicio específico a un cliente final, internamente se deban utilizar varias aplicaciones independientes que dan soporte a diferentes aspectos del proceso de negocio [24]. Por el contrario, ahora se espera que las empresas ofrezcan a sus clientes servicios desde una perspectiva de procesos de negocio: tanto para los clientes como para los empleados, las antiguas funciones deben ser aspectos de un proceso más grande y debe ser completamente transparente el hecho de que más de una aplicación o sistema de almacenamiento den soporte a este proceso [25].

Para poder implementar esta nueva visión de las empresas guiadas por los procesos de negocio es necesario contar con dos cosas. En primer lugar, es necesario contar con los procesos diseñados usando un lenguaje claro y consistente. Tener explícitos los procesos de negocio sirve para que las personas involucradas en ellos los puedan utilizar como guía de trabajo. Sin embargo, el diseño de los procesos de negocio de una empresa no es trivial en la mayoría de los casos. Por ejemplo, en una entidad bancaria el proceso de aprobación de un préstamo debe ser diseñado por una persona (o un equipo de personas) que conozca la información que se debe recolectar, los pasos que se deben seguir, la estructura y el funcionamiento del banco, los riesgos financieros asociados y las implicaciones legales. Debido a estas dificultades los procesos de negocio deben ser diseñados, controlados y supervisados por expertos en el dominio del negocio. Además, en lo posible cada diseño de un proceso de negocio debería ser independiente de las aplicaciones que le dan soporte a las actividades puntuales, puesto que son las herramientas las que deberían adecuarse a la organización y no al contrario.

Después de tener los procesos explícitos, el segundo elemento necesario son las aplicaciones que dan soporte tanto a la ejecución de los procesos mismos como a las diferentes actividades y etapas de cada uno. Es decir, se requiere un conjunto de aplicaciones donde los usuarios puedan realizar las actividades puntuales que hacen parte del proceso (revisar la información de un cliente, reportar la aprobación de un crédito) y un conjunto de aplicaciones que permitan monitorear el estado de los procesos.

Lograr que una empresa basada en funciones se reestructure para que su funcionamiento se base en procesos de negocio no es algo sencillo, debido a dos dificultades principales. En primer lugar los procesos deben ser diseñados para que puedan hacerse explícitos, y esto, como ya dijimos, es una tarea bastante compleja [26]. Sin embargo, existen maneras de reducir la dificultad de esta tarea aprovechando la experiencia de personas que sean expertas en el dominio de negocio o que entiendan perfectamente el funcionamiento de la empresa. La segunda dificultad radica en la ejecución de los procesos una vez que han sido diseñados. Los

mecanismos necesarios para ejecutar dichos procesos no son algo que pueda encontrarse con facilidad y tampoco existen técnicas que puedan usarse en todos los casos. Los únicos elementos comunes a todas las soluciones, son una aplicación que materializa los procesos y mantiene su estado y aplicaciones distribuidas que utilizan los participantes del proceso para controlarlo. Al tener un motor que hace explícita la ejecución de los procesos mejora el control sobre las actividades y se facilita la integración de las aplicaciones desde un punto de vista de proceso: las acciones que realicen los usuarios sobre las distintas aplicaciones tendrán un significado a nivel del proceso global, que deberá verse reflejado en el motor utilizado. Un proceso con una definición conocida y cuya ejecución es explícita es más sencillo de monitorear, controlar, medir, y finalmente mejorar. Por otra parte, al nivel de los procesos de negocio existen con frecuencia conceptos que no hacen parte de ninguna de las aplicaciones independientes. Al materializar la ejecución del proceso tiene sentido que se materialicen también estos conceptos, que podrían entonces consultarse y administrarse.

El problema anterior, hacer explícitos los procesos de negocio y ejecutarlos aprovechando aplicaciones existentes, es el que tratamos de resolver con la propuesta que presentamos en este artículo. Dicha propuesta se basa en la integración de dos familias de soluciones que ya existen y están ampliamente difundidas.

Una parte de las herramientas existentes hoy en día que permiten ejecutar procesos de negocio son llamadas comúnmente motores de workflow [16]. Los procesos que estos motores ejecutan deben modelarse con lenguajes que permiten expresar el orden en el que deben realizarse las diferentes tareas y definir los roles de las personas que participan en el proceso [16]. Para cada uno de estos lenguajes existen editores, algunos de los cuales permiten incluso modelar los datos que hacen parte del dominio de negocio. En este tipo de herramientas el modelamiento de los procesos y de las actividades se hace utilizando conceptos del dominio de negocio. Por ejemplo, si la herramienta utilizada para definir el proceso de aprobación de un préstamo utiliza la notación para procesos llamada BPMN [4], entonces en el diseño del proceso aparecerán los conceptos de "Actividad de entrega de documentos", "Actividad de Revisión Inicial", "Actividad de Validación de Datos", entre otras, y los roles "Solicitante", "Asesor" y "Supervisor", por ejemplo. En el contexto específico de los procesos de desarrollo de software, lo ideal sería que la herramienta utilizara una notación como SPEM [15], para que los conceptos utilizados fueran "Ciclos", "Fases", "Actividades" y "Productos", entre otros. En general las soluciones basadas en workflows ofrecen la ventaja de poder describir los procesos usando conceptos de alto nivel. Además, la variedad de lenguajes existentes permite buscar el más adecuado para el contexto de cada empresa [16].

Otra familia de herramientas son las llamadas tecnologías EAI (Enterprise Application Integration) cuyo objetivo es lograr la integración de varias aplicaciones [6, 8, 17]. Esto incluye tanto la interacción entre aplicaciones a nivel de servicios y requerimientos funcionales, como el manejo de sus datos para mantener la consistencia a nivel global. Los sistemas EAI ofrecen muchas características útiles en una gran cantidad de contextos de integración pero, a diferencia de lo que sucede con los workflow que mencionamos, para poner en funcionamiento una solución EAI es necesario contar con información técnica detallada sobre cada una de las aplicaciones. Sin esta información sería imposible comunicar las aplicaciones y mucho menos mantener la consistencia de los datos.

Se puede ver entonces que tanto los motores de workflow como los sistemas EAI son alternativas diferentes para solucionar el problema del soporte a los procesos de negocio, cada uno con sus ventajas y desventajas. Sin embargo, utilizar únicamente uno de los sistemas disponibles, que ofrezca las ventajas de sólo una de las familias, probablemente no sea suficiente para solucionar el problema completo: si se usaran únicamente motores de workflow sería difícil implementar una solución completamente integrada al problema planteado puesto que al diseñar un proceso de negocio no se contaría con información suficiente para integrar las aplicaciones que lo soportan [8, 27]. En algunos sistemas esta información podría incluirse, pero entonces se perdería la abstracción a nivel de procesos de negocio. Por el contrario, si se usara únicamente una solución EAI, habría conceptos importantes que nunca se materializarían, empezando por el del proceso mismo [5].

La propuesta presentada en este artículo plantea una solución al problema del soporte a los procesos de negocio que aprovecha tanto las ventajas de un motor de workflow específico (Cumbia) como las de un sistema EAI (Eleggua). El motor Cumbia, que presentaremos más en detalle en la sección 3, es un motor que permite la ejecución de procesos definidos en diferentes lenguajes: esto permite que los procesos de negocio sean definidos usando el lenguaje más adecuado a cada contexto. Eleggua, cuyos detalles serán presentados en la sección 4, es una infraestructura para la integración de aplicaciones basada en eventos. Definiendo reglas de integración sobre Eleggua es posible lograr la interacción entre las aplicaciones y el mantenimiento de la consistencia de los datos globales. La relación entre Cumbia y Eleggua se basa en eventos y en hacer que las actividades que se definen a nivel de los procesos de negocio se relacionen con acciones que ejecuten los usuarios dentro del contexto de las aplicaciones ya existentes. Al unir a Cumbia y Eleggua en una sola solución, se obtiene una plataforma que permite materializar procesos de negocio y soportar su ejecución sobre aplicaciones que anteriormente eran independientes.

Para ilustrar la propuesta dentro de este artículo, se usará un caso de estudio basado en un proceso de negocio definido dentro del contexto de los procesos para desarrollo de software. Dicho proceso se presenta en la sección 2 de este documento. Las siguientes secciones (3 y 4) presentan los detalles principales de Cumbia y Eleggua, y se enfocan en resaltar las ventajas que cada uno ofrece para solucionar el problema presentado. Luego, en la sección 5, es presentado en forma detallada la propuesta, utilizando como ejemplo el caso de estudio y los resultados que se obtuvieron en la implementación realizada. En la sección 6 discutimos trabajos relacionados que hemos estudiado y que nos permiten comparar nuestro sistema. Cerramos el artículo presentando una breve descripción del trabajo futuro que esperamos realizar dentro del contexto del proyecto, y con conclusiones generales.

2. CASO DE ESTUDIO: UN PROCESO DE PRODUCCIÓN DE UN NUEVO RELEASE DE UNA APLICACIÓN

Para ilustrar el proyecto, este artículo utilizará una versión simplificada de un proceso real de Administración de la Configuración, que hace parte del conjunto de procesos definidos para el grupo de desarrollo Qualdev [18]. El objetivo del proceso es definir claramente la creación de un nuevo release de una aplicación a partir de la integración de un conjunto de solicitudes de cambio sobre el release anterior. El proceso seleccionado es muy apropiado para ilustrar este artículo por varios motivos: en primer lugar se trata de un proceso de negocio relativamente sencillo pero que requiere de la interacción de varias aplicaciones independientes y de varios miembros de un equipo; por otra parte hay información que hace parte del proceso de negocio pero que no aparece en ninguna de las aplicaciones. Al hacer explícito el proceso, esta información debería materializarse de alguna forma.

Las Solicitudes de Cambio (SC) son el elemento principal en el proceso de producción de un nuevo release. Estas solicitudes son propuestas por los diferentes *stakeholders* de un proyecto (usuarios, desarrolladores, responsables de pruebas, etc.). Una solicitud de cambio puede expresar, por ejemplo, un nuevo requerimiento que un cliente quiere que se incluya en la aplicación o puede también utilizarse para reportar un defecto o error que requiera una solución en un próximo release. Cada solicitud de cambio debe tener la información necesaria para que se pueda implementar (una descripción detallada) y se pueda incluir dentro del proceso de desarrollo: prioridad, tiempo estimado, etc. Dentro del proceso de producción de un nuevo release, el Líder del Proyecto, de acuerdo con las prioridades y los recursos, debe seleccionar el conjunto de solicitudes de cambio que se quisiera incluir para la siguiente versión de la aplicación. También es necesario seleccionar un equipo de desarrolladores que serán los que evaluarán e implementarán cada una de las solicitudes de cambio.

Después de seleccionar las solicitudes de cambio, el proceso establece que éstas deben ser estudiadas para establecer la viabilidad de que se lleve a cabo lo que cada una indica, dentro del contexto del release. Es posible que, después de esta revisión, haya varias solicitudes que se descarten y no se incluyan dentro del release. Para las solicitudes restantes se debe establecer un tiempo estimado de implementación y se debe asignar a los diferentes participantes del proyecto. En la siguiente etapa del proceso, cada uno de los participantes debería implementar las solicitudes de cambio que le fueron asignadas y verificar su correcta implementación. Si un participante encuentra algún problema inesperado que impide definitivamente la realización de una solicitud de cambio, entonces puede terminarla en un estado que indica que fue imposible completarla. Cuando todas las solicitudes de cambio han sido terminadas (exitosamente o no), uno de los participantes del proyecto debe realizar una tarea de fusión de todos los cambios, con la que se obtiene una versión completa del producto desarrollado. Sobre esta versión completa se deberían realizar pruebas y después de esto se podrá liberar el nuevo release del producto. Adicionalmente, por cada actividad relacionada con la producción del release, se debe realizar previamente una planeación en la que se establece un tiempo estimado que tomará la tarea; al terminarla, se debe registrar el tiempo real que tomó.

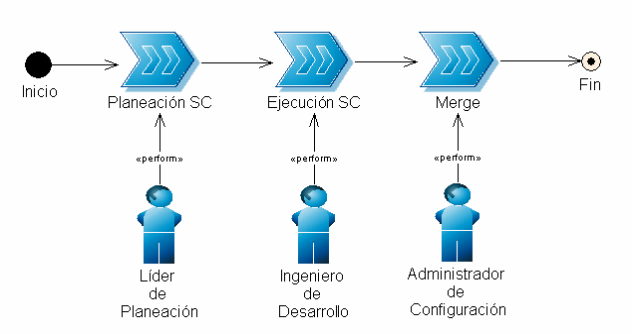

Figura 1. Diseño del proceso.

Para simplificar las explicaciones, el resto del artículo utiliza una versión resumida de este proceso en la cual se han dejado solamente las actividades más importantes. El proceso reducido se ilustra en la figura 1, en la cual hemos utilizado como lenguaje de descripción SPEM [15]. En dicha figura pueden apreciarse las tareas que hacen parte del proceso (Planeación de las Solicitudes de Cambio, Ejecución y Merge) y los roles involucrados (Líder del Planeación, Ingeniero de Desarrollo y Administrador de Configuración).

Según la definición del proceso, los participantes en el desarrollo deben utilizar al menos dos aplicaciones: Changeset y DotProject. Changeset es un sistema de administración de configuraciones de alto nivel que permite versionar productos, agruparlos en líneas de base y realizar un seguimiento a su desarrollo y estado mediante solicitudes de cambio [11]. Varios de los conceptos que hacen parte del proceso definido existen dentro de Changeset, tales como solicitudes de cambio, proyecto y rol. Sin embargo, el concepto fundamental para este proceso, el "Release", no existe dentro de la herramienta. Si en un momento dado se quisiera conocer el estado de un release o las solicitudes que han sido seleccionadas para hacer parte de un release no se podría encontrar esta información en Changeset, por lo cual surge la necesidad de materializar este concepto en algún otro lado.

Por otra parte, DotProject es la herramienta utilizada para realizar la planeación y el seguimiento a las tareas que deben hacerse dentro del contexto de un proyecto [9]. En DotProject pueden crearse equipos de trabajo, asignar tareas, estimar su tiempo de realización y registrar el tiempo total requerido. DotProject es la herramienta donde cada miembro del equipo de desarrollo puede encontrar siempre actualizada la lista de tareas que debe realizar. Un detalle importante es que Changeset está construido en Java y todos sus datos persisten en una base de datos relacional, mientras que DotProject está construido en PHP y sus datos persisten en una base de datos relacional diferente a la de Changeset.

Actualmente la definición del proceso sirve como guía documental para que quienes intervienen sepan qué tareas realizar y cuándo. Cada participante sabe utilizar y se apoya tanto en DotProject como Changeset. Esta forma de poner en práctica el proceso tiene varios inconvenientes. En primer lugar, al no existir una versión del proceso que esté materializada dentro de una herramienta que lo conozca y ejecute, es difícil conocer el estado de su ejecución. Así mismo, es difícil obtener métricas sobre la ejecución del proceso, por ejemplo para controlar el tiempo invertido en las distintas actividades y fases del proceso. Por este motivo, la productividad del grupo de desarrollo para el proceso

específico del nuevo release no se puede conocer. Otro inconveniente es que existe información del proceso que no es conocida ni controlada por las aplicaciones individuales. Por ejemplo, la asociación de las solicitudes de cambio al nuevo release, sólo existe en la documentación del proceso que se está ejecutando.

Además de los problemas anteriores, hay una sobrecarga para el equipo de desarrollo con respecto a la utilización de las aplicaciones: desde el punto de vista del control y de la ejecución del proceso, las acciones que se realicen sobre una de las aplicaciones no tienen ningún efecto sobre la otra. Por ejemplo, el hecho de asignarle una solicitud de cambio a alguien dentro de Changeset no implica que para esa persona se cree la tarea respectiva dentro de DotProject: las tareas deben ser creadas en DotProject una por una para que el tiempo que se invierte en la resolución de una solicitud de cambio pueda ser registrado.

A partir de los problemas mencionados en el caso de estudio, a continuación se enumeran los requerimientos de la solución buscada.

1. El proceso debe poder ser definido a un alto nivel de abstracción en términos de los conceptos del negocio, sin involucrar detalles técnicos.
2. El proceso debe poder ser ejecutado y su ejecución debe guiar a los participantes en la realización de sus actividades. Esto es, debe informarles cada paso a seguir y proveerles la información que se necesite para realizarlo. Si se necesita información adicional para realizar la ejecución, esta deberá ser especificada con relación a elementos del proceso de alto nivel.
3. En cualquier momento se debe poder monitorear el estado del proceso en ejecución.
4. El proceso en ejecución debe guardar información que eventualmente no está materializada en ninguna de las aplicaciones involucradas.
5. La ejecución del proceso también debe orquestar la utilización de las aplicaciones.
6. Se deben poder automatizar las acciones que no requieren intervención humana y que se ejecutan sobre una aplicación como consecuencia de una acción realizada por un usuario sobre otra de las aplicaciones.

La solución que proponemos consta de tres elementos: un sistema de workflow (Cumbia) que ofrece una base para los primeros tres requerimientos; una infraestructura de integración de aplicaciones (EAI) (Eleggua) que permite automatizar las acciones y reacciones entre aplicaciones (requerimiento 6); y un mecanismo para lograr que Cumbia y Eleggua se integren de manera armónica y permita que se logren los objetivos restantes (4 y 5). En las siguientes secciones se presentan los elementos individuales que conforman esta solución y la forma en la que se unen para dar una solución completa al problema.

3. UNA MÁQUINA DE WORKFLOW: CUMBIA

CUMBIA [20] es un motor experimental de workflow, de propósito general, cuyo objetivo principal es servir como núcleo para construir sobre él motores de workflow de propósito específico, utilizando para esto los mecanismos de extensión y adaptación que provee. CUMBIA se basa en un conjunto minimal de elementos, cuya semántica ha sido expresada de manera explícita y externa por medio de autómatas de estados finitos. Mediante la sincronización y composición de instancias de dichos autómatas, el motor es capaz de materializar y ejecutar procesos en distintos dominios. El hecho de tener expresada la semántica del motor de esta manera, nos permite extenderla y adaptarla de manera sencilla, manejando un fino nivel de granularidad. El motor es además completamente reflexivo, lo que implica que es posible manipular y modificar en ejecución los elementos que componen los procesos.

CUMBIA basa su estructura en 8 conceptos: (1) proceso, que representa una tarea compuesta por actividades simples o por otros procesos, (2) actividad, que representa una tarea atómica, (3) actividad múltiple, que representa una tarea atómica que se puede instanciar múltiples veces en la ejecución, (4) dataflow, que indica el flujo de control y de datos entre actividades, (5) puerto, que representa un punto de inicio o terminación de una actividad, y el cual se encuentra conectado con otra actividad por medio de un dataflow, (6) workspace, que representa el punto de la actividad en la cual el participante hace el trabajo, (7) dato, unidad estructurada de información que fluye por los dataflows entre las actividades y (8) participante, que representa al recurso que debe ejecutar la tarea. Cada uno de estos elementos cuenta con una memoria en la cual puede mantener información relevante para la instancia. En la figura 2 se ilustran los principales elementos del modelo.

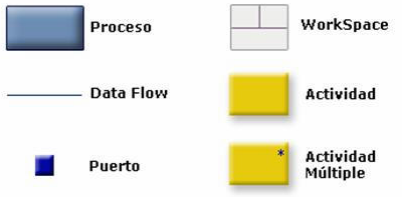

Figura 2. Principales elementos del modelo.

Utilizar un motor de workflows como CUMBIA tiene varias ventajas: por una parte se cuenta con herramientas que permiten supervisar la ejecución de los procesos; por otra parte, y es lo que diferencia a CUMBIA de los demás motores de workflow, es posible definir los procesos usando diferentes lenguajes de alto nivel que luego son traducidos al modelo propio de CUMBIA para su ejecución. Esto quiere decir que los procesos de negocio no necesitan ser expresados en términos de los 8 elementos anteriormente presentados. En nuestro caso el proceso fue diseñado usando SPEM y fue convertido a un modelo CUMBIA: este modelo y una explicación de cómo se ejecuta es presentado a continuación.

3.1 Definición y Ejecución del Proceso

El proceso CUMBIA para el caso de estudio que presentamos en la sección 2 se muestra en la figura 3. Este proceso está compuesto por tres actividades que se deben ejecutar secuencialmente: cuando la actividad “Planeación SC” termine su ejecución, deberá ejecutarse la actividad “Ejecución SC”. Luego, cuando se hayan terminado de ejecutar todas las solicitudes de

cambio, se ejecutará la actividad "Merge". Cada una de las tres actividades, así como el proceso, tiene un puerto de entrada y un puerto de salida que están conectados entre sí a través de dataflows.

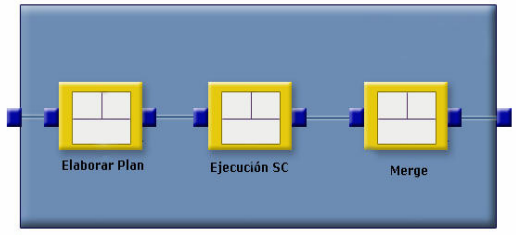

Figura 3. Proceso Cumbia.

Para iniciar la ejecución de una instancia del proceso se deben dejar los datos iniciales en el puerto de entrada. A través del dataflow estos datos llegarán al puerto de entrada de la primera actividad, que iniciará su ejecución. Cuando esta actividad termine (después de que se realice la planeación del release), dejará un dato en su puerto de salida y este dato llegará al puerto de entrada de la segunda actividad, que en ese momento podrá iniciar su ejecución. Cuando la segunda actividad termine su ejecución (se hayan terminado de ejecutar todas las solicitudes de cambio seleccionadas para el release) entonces quedará un dato en su puerto de salida y este llegará al puerto de entrada de la actividad de Merge, que se ejecutará en ese momento. Finalmente, cuando termine dicha actividad, el dato que quede en su puerto de salida llegará al puerto de salida del proceso y con esto terminará la ejecución de la instancia.

3.2 El Espacio de Trabajo (Workspace)

La descripción anterior únicamente muestra la secuencia de actividades que deben realizarse a nivel del proceso, sin entrar en detalles sobre qué debe hacerse en cada uno de estos pasos y cómo se debe manejar la información generada y consumida por cada una de las actividades. Como parte de la definición de Cumbia no existen componentes especializados que sepan realizar las tareas especializadas que requiere este proceso, pero si existen mecanismos de extensión y puntos de flexibilidad que permiten agregar el comportamiento particular que se requiere. En la sección 5, al presentar la solución completa, se entenderá completamente cómo se aprovechan dichos mecanismos de extensión para integrar a Cumbia con Eleggua y así controlar la ejecución del proceso desde las mismas aplicaciones participantes. Por ahora es importante decir que el punto de flexibilidad utilizado en este caso son los Workspaces: un desarrollador puede implementar nuevos Workspaces para que realicen acciones específicas en el momento en el que se ejecuten las actividades que los contienen. En este caso las actividades especializadas tendrán responsabilidades relacionadas con la integración con el resto de aplicaciones usando Eleggua.

4. UNA INFRAESTRUCTURA PARA LA INTEGRACIÓN DE APLICACIONES: ELEGGUA

Eleggua es una infraestructura para integración de aplicaciones basada en eventos asincrónicos y que permite la automatización de acciones sobre las aplicaciones de acuerdo con la definición de reglas de cooperación. Esta sección presenta algunos conceptos de Eleggua que son básicos para entender la solución global. Información adicional sobre la infraestructura y su implementación pueden ser encontrados en [5, 6].

4.1 Servicio de Notificación de Eventos

Un nodo Eleggua está formado por un conjunto de aplicaciones que intercambian eventos y ejecutan acciones como consecuencia de la recepción de esos mismos eventos. Un evento representa una acción ocurrida en una aplicación como consecuencia del llamado de un servicio de la aplicación por un usuario externo o por otra aplicación. Por ejemplo, se puede definir que cada vez que se cree una solicitud de cambio en la herramienta Changeset se genere un evento informando este hecho. Así, las aplicaciones publican los eventos que pueden generar y se suscriben a los eventos para los que desean recibir una notificación. De esta forma se logra una comunicación de bajo acoplamiento entre las aplicaciones, ya que ellas no saben quién o quiénes producirán los eventos que les interesan ni quiénes recibirán los eventos que ellas producen. Además del bajo acoplamiento, se espera que en una solución de integración las aplicaciones no sean modificadas. Por esta razón, se cuenta con el concepto de Proxy de Cooperación. Cada aplicación tiene su propio proxy que la representa y que se encarga de la emisión de eventos, de la recepción de eventos y de la ejecución de acciones. Generalmente las aplicaciones integradas son heterogéneas, por lo cual el proxy tiene la responsabilidad de hacer invisibles esas diferencias. En el caso de estudio, la aplicación DotProject está desarrollada en PHP, mientras que tanto Changeset como Cumbia están desarrollados en Java.

La figura 4 ilustra el nodo Eleggua para el caso de estudio. En este nodo se encuentran las tres aplicaciones, DotProject, Changeset y Cumbia, cada una con su respectivo proxy de cooperación. El proxy de cooperación de DotProject publica los eventos relacionados con el cierre de las tareas. El de Changeset publica los eventos relacionados con la asignación, cierre y rechazo de solicitudes de cambio. Ambos proxys tienen la responsabilidad de ejecutar acciones sobre su respectiva aplicación cuando se reciban eventos según se haya definido en las reglas de cooperación. En el capítulo 5 se presenta en más detalle cómo Cumbia participa en este nodo Eleggua.

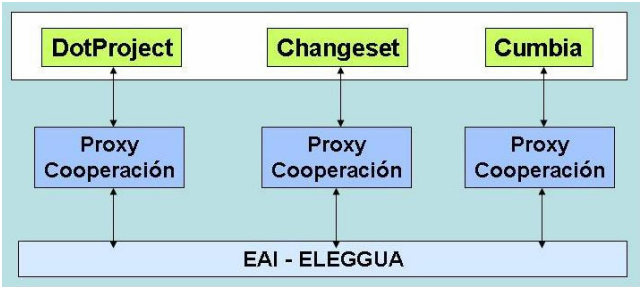

Figura 4. Nodo Eleggua para el caso de estudio.

4.2 Reglas de Cooperación

Eleggua cuenta con mecanismos para especificar, a partir de un diagrama de actividades UML, qué aplicación genera cada evento, a qué aplicaciones les interesa ese evento y qué acciones deben tomar una vez lo reciban. Este mecanismo permite definir

las reglas de cooperación a un alto nivel de abstracción por parte de los desarrolladores, y facilita el proceso de validación por parte de los usuarios. Esta característica, sumada con el hecho de contar con la herramienta EAI-Rules [5] para automatizar la transformación de la definición de la regla de cooperación en código ejecutable por la infraestructura, se convierte en una gran ventaja para Eleggua sobre otras alternativas. En estos diagramas de actividades, se puede diseñar la interacción entre las aplicaciones usando tres conceptos básicos: observaciones, eventos y reglas ECA (Evento-Condición-Acción):

- Observación: Una observación representa una actividad de interés en una aplicación que genera un evento lógico. Una actividad de interés es la interacción de un usuario con una aplicación a través de un servicio cuya ejecución es de interés para otras aplicaciones en el proceso. Una observación se encarga de interceptar le ejecución del servicio y generar un evento lógico.

- Evento lógico: los eventos lógicos son el concepto principal para el intercambio de información entre las aplicaciones. Es definido por un tipo de evento y un conjunto de parámetros. El tipo representa un concepto de dominio que es común o de interés para varias aplicaciones.

- Regla ECA: una regla ECA representa el conjunto de acciones que una aplicación debe ejecutar como respuesta a un evento lógico. Una regla está definida por el tipo de evento que inicia su ejecución, un conjunto de condiciones evaluadas para cada evento y un conjunto de acciones que son ejecutadas cada vez que un evento lógico cumple las condiciones. Las acciones usualmente invocan servicios en la aplicación interesada en el evento usando su API o Web Services, o producen nuevos eventos lógicos.

Una regla de cooperación es el conjunto de observaciones, eventos lógicos y reglas ECA asociados a la automatización de condiciones de negocio. Durante la ejecución de las aplicaciones el sistema de ejecución basado en eventos intercepta las observaciones, produce los eventos y ejecuta las reglas ECA.

La figura 5 muestra un diagrama de actividades definido para una de las reglas de integración que se ejecutan en relación con la actividad de planeación del proceso. Esta regla observa la acción de creación de instancias de procesos en el motor Cumbia. Cada vez que se crea una de estas instancias se debe crear un proyecto y una tarea de planeación inicial dentro de la herramienta DotProject. El proyecto agrupará todas las tareas necesarias para generar un nuevo release y la tarea de planeación le indicará al líder de planeación que debe evaluar la factibilidad de implementar las solicitudes de cambios seleccionadas. El diagrama de actividades especifica una observación sobre la creación de instancias de proceso en Cumbia (primera actividad del diagrama). La actividad (ó servicio) de interés en el motor Cumbia es el método "activate". Esta observación produce un evento lógico con la información del release (releaseInfo), que incluye los identificadores de las solicitudes de cambio seleccionadas y los roles que asume cada integrante del proyecto (desarrollador, líder de planeación, administrador de configuraciones). Tanto la actividad de observación como la generación del evento lógico ocurren en Cumbia. El evento pasa al sistema manejador de eventos y es reenviado a aquellas aplicaciones que manifestaron interés en él, para este caso, DotProject. Cuando se realiza la notificación de evento recibido

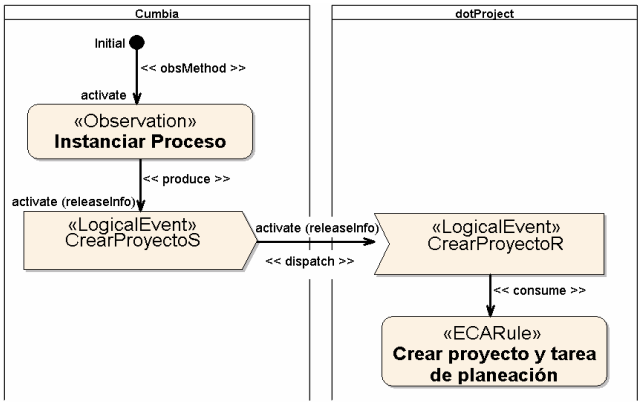

Figura 5. Regla de integración "Crear proyecto".

en DotProject, debe ejecutarse la acción que contiene la lógica de creación de un proyecto y de una tarea de planeación que está encapsulada en una regla ECA.

Además de contar con una arquitectura distribuida, débilmente acoplada, que permite integrar las aplicaciones con un mínimo de modificaciones, Eleggua incluye facilidades para monitoreo, control y escalabilidad [7], lo cual también constituye una ventaja sobre otras plataformas de integración.

5. INTEGRACIÓN DE CUMBIA Y ELEGGUA

Cuando se presentó el caso de estudio en la sección 2 de este artículo, se identificaron seis requerimientos que se han ido solucionando. Cumbia ofrece las herramientas necesarias para solucionar los tres primeros (diseño, ejecución y monitoreo de los procesos) mientras que Eleggua soluciona el sexto (integración de las aplicaciones). En esta sección presentamos la forma en la que integramos a Cumbia y Eleggua, con lo cual conseguimos solucionar los dos requerimientos restantes (orquestación de las aplicaciones con base en el proceso y materialización de conceptos del proceso).

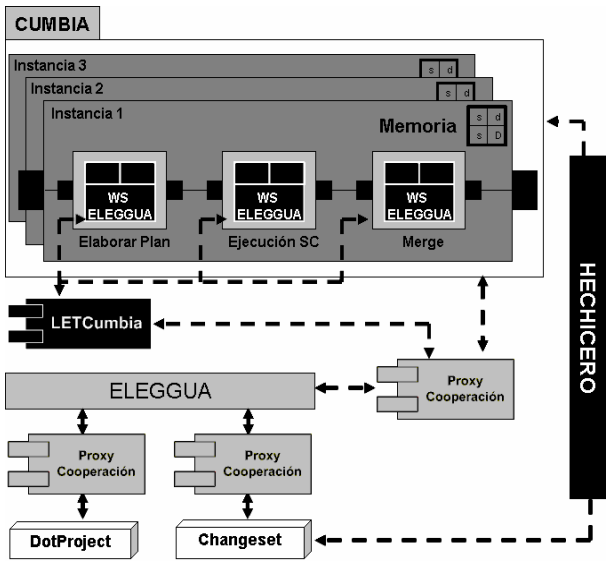

Figura 6. Arquitectura de la solución.

La arquitectura general de la solución propuesta se puede observar en la figura 6. El elemento principal de nuestra solución es un workspace especializado para este proyecto, que llamamos WS-Eleggua. Este elemento ofrece la posibilidad tanto de generar eventos según lo que pase en el proceso, como de recibir eventos provenientes de otras aplicaciones y de reaccionar según se requiera. Como se explicará más adelante, este workspace debe ser especializado en cada proceso para concretar los detalles de la ejecución que no pueden ser descritos.

La responsabilidad de manejar la información del proceso se encuentra repartida entre dos componentes. Por una parte el workspace WS-Eleggua se encarga de mantener y de administrar la información que pertenece sólo al proceso y no a las aplicaciones. Por otra parte, otro componente llamado El Hechicero obtiene la información necesaria para inicializar los procesos interactuando con los usuarios responsables de crear las instancias.

Existe un tercer elemento en la solución llamado LETCumbia. Gracias a este componente es que se pueden tener varias instancias del mismo proceso ejecutándose al mismo tiempo, sin tener que generar más eventos de los necesarios.

A continuación se presenta en detalle cada uno de los tres elementos.

5.1 El Hechicero

La mayoría de procesos necesitan datos iniciales para comenzar su ejecución. Estos datos generalmente son administrados por otras aplicaciones y varían en cada instancia de proceso que se ejecuta. El Hechicero es una aplicación que permite consultar información de una o varias aplicaciones y crear instancias de proceso inicializándolas con estos datos. La interacción del Hechicero con los usuarios está inspirada en los "Wizards" que son usados en una gran cantidad de aplicaciones y que se componen de una serie de pasos en los que un usuario debe ir indicando progresivamente la información que se requiere para lograr un objetivo específico. Para cada instancia, Cumbia mantiene una memoria independiente y El Hechicero usa esta memoria mediante mecanismos ofrecidos por el motor de Cumbia. Para crear instancias de proceso y colocar datos iniciales en sus memorias, el Hechicero ofrece al usuario una interfaz gráfica desde la cual puede seleccionar la información que incluirá en cada proceso que ejecute. El hechicero permite ver además el estado de las instancias ya creadas, dando una vista global del avance del proceso.

Para iniciar una instancia del proceso de producción de releases, El Hechicero consulta en Changeset los proyectos disponibles para que el usuario seleccione uno de ellos. Luego, el usuario escoge las solicitudes de cambio que incluirá en su release, que hacen parte del proyecto seleccionado y que son presentadas por el Hechicero después de obtenerlas de Changeset. Por último, el usuario asigna roles a los usuarios en Changeset asociados con el proyecto seleccionado. Con esta información, El Hechicero se comunica con el motor de Cumbia, crea una instancia del proceso de producción de releases y almacena en su memoria todos los datos obtenidos en la interacción con el usuario.

5.2 WS-Eleggua

El workspace especializado que llamamos WS-Eleggua tiene varias responsabilidades que presentamos a continuación.

- Controlar la ejecución del proceso. Dependiendo de las acciones que se realicen sobre cada una de las aplicaciones el estado del proceso deberá cambiar.

- Administrar la información de la instancia del proceso a la que pertenece el workspace. Esta información estará almacenada en la memoria de la instancia y será modificada a medida que se vayan recibiendo eventos.

- Permitir que las modificaciones que se hagan a la información del proceso lleguen a las aplicaciones donde sea necesario. Esto quiere decir que entre las responsabilidades del workspace está mantener la consistencia entre el proceso y las aplicaciones participantes.

Es importante recalcar que en cada proceso será necesario especializar este workspace para que sea capaz tanto de manejar la información adecuada para cada proceso como de solicitar acciones especializadas sobre otras actividades. En este momento la cantidad de código que debe ser escrita en cada caso es comparativamente poca, pero parte de lo que se está investigando dentro del contexto del proyecto son estrategias para reducirla al mínimo.

5.2.1 Información del Proceso

Como se mencionó anteriormente, al incluir una aplicación dentro de un nodo Eleggua se logra que la interacción de los usuarios con dicha aplicación genere eventos. Algunos de estos eventos no tienen ningún impacto sobre las otras aplicaciones, por lo cual no tienen ninguna reacción asociada. Otros, para los cuales existen reglas ECA asociadas, generan la ejecución de acciones sobre otras aplicaciones que podrían no tener impacto sobre el proceso mismo que se está ejecutando, por ejemplo cuando simplemente se actualiza un dato en otra aplicación para garantizar la consistencia de los datos. Para los casos en los cuales los eventos generados sí implican reacciones en la ejecución del proceso materializado, se requiere que el workspace correspondiente se haya suscrito indicando su interés en estos eventos. Esto permite que Eleggua le informe a Cumbia sobre las acciones que realicen los usuarios sobre las aplicaciones y que podrían tener un impacto sobre el proceso.

Los eventos enviados por las aplicaciones participantes hacia el proceso pueden ser de dos tipos: aquellos que alteran la memoria del proceso o aquellos que afectan el estado general de la instancia de proceso, es decir, que solamente controlan su ejecución. Este segundo tipo de eventos será tratado más adelante, cuando se explique cómo se controla el proceso (sección 5.2.3).

Todo evento recibido por el Workspace puede ser diferenciado de acuerdo con su tipo y con los parámetros que contiene. Para representar esto, es posible definir una función que relaciona un evento recibido con la forma como debe ser alterada la memoria de la instancia de proceso. Tal función debe cumplir con el siguiente formato:

$$F(\textit{te}, \textit{evento}) = \text{método\_te}(\textit{evento})$$

donde *te* es el tipo de evento recibido por el Workspace, *evento* es el evento mismo y *método_te* es el método o servicio que debe ser

ejecutado como respuesta a la recepción del evento. En este método deberá escribirse el código encargado de modificar la memoria de la instancia del proceso utilizando la información contenida dentro del evento. Es importante decir que se espera que una acción particular de un usuario en una de las aplicaciones participantes, produzca siempre el mismo efecto sobre la memoria del proceso.

Cuando se modifica la memoria es posible que sea necesario informar a una o varias aplicaciones participantes sobre el cambio o que incluso se tengan que ejecutar acciones adicionales sobre estas aplicaciones. Para lograr esto, el WS-Eleggua tiene la capacidad de producir eventos que pueden ser usados para transmitir datos o para que, según lo indicado por una regla ECA, se realicen acciones.

La producción de eventos debería ocurrir cuando la memoria de la instancia de un proceso se encuentre en estados particulares: será responsabilidad del Workspace evaluar si debe o no producirse alguno de estos eventos cada vez que se modifique la memoria. Dadas estas condiciones, se pueden definir ahora funciones que relacionan un estado de la memoria de la instancia de proceso con la generación de eventos particulares. Dichas funciones deben cumplir con el siguiente formato:

F (*condición*) = *método_pe*

Donde *condición* expresa una condición sobre los datos de la memoria de la instancia de proceso y *método_pe* es un método o servicio que será ejecutado cuando la condición establecida se cumpla. Este método tiene la responsabilidad de producir un evento con el tipo y parámetros adecuados a las reglas de cooperación. Toda la lógica para construir el evento se encuentra encapsulada dentro de ese método, que es específico para cada caso, pero el trabajo futuro del proyecto incluye una nueva definición de las funciones en la cual se describa de forma declarativa el tipo de evento generado y la forma de obtener y organizar sus parámetros.

5.2.2 Aplicación al Caso de Estudio

Para ilustrar lo explicado en la sección anterior se usará la primera actividad del proceso de producción de releases, Planeación SC. El objetivo de esta actividad es revisar las solicitudes de cambio seleccionadas cuando se creó la instancia de proceso y decidir cuáles de ellas efectivamente sí se incluirán dentro del release. Una solicitud de cambio puede considerarse aprobada cuando el Líder de Planeación del proceso le asigne un responsable.

En la figura 7 se puede observar cómo, para cada actividad del proceso, se asocian diferentes reglas de cooperación para mantener la consistencia entre las aplicaciones y del proceso mismo.

En primer lugar, para que el Líder de Planeación se entere que debe realizar esta actividad, debe crearse una tarea en DotProject, para lo que se requiere de un proyecto al cual pueda ser asociada. Por estos motivos debe crearse un proyecto y una tarea en DotProject tan pronto inicie la actividad de planeación del proceso (ver Figura 5). Para lograr lo anterior se define la función:

F (*ev_dp_enviado no existe*) = *crearProyectoTareaDP*()

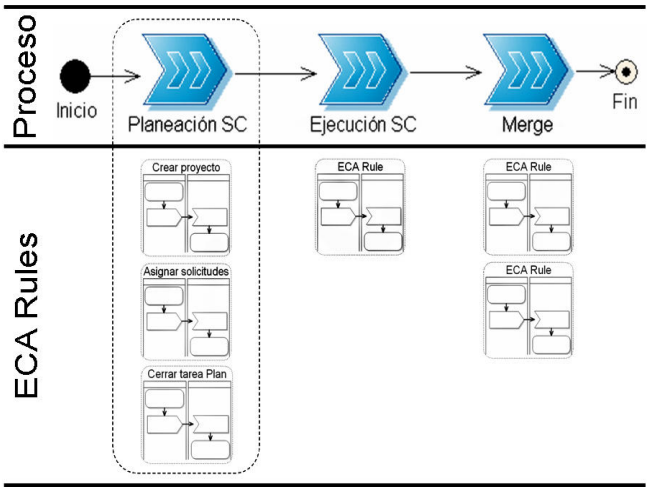

Figura 7. Reglas de cooperación de cada actividad.

En este caso *ev_dp_enviado* (evento DotProject enviado) es un valor almacenado en la memoria de la instancia que sirve para indicar si ya se envió un evento para crear tanto el proyecto como la tarea de planeación en DotProject. Cuando se inicia una instancia del proceso el valor *ev_dp_enviado* no existe, lo cual hace verdadera la condición definida. Cuando la tarea de planeación se inicie detectará que la condición es válida y ejecutará el método definido en la función anterior, *crearProyectoTareaDP*. Este método construye un evento con los parámetros y el tipo de evento necesario para que, a través de Eleggua, el evento llegue hasta el proxy de cooperación de DotProject, desde donde se crearán tanto el proyecto como la tarea de planeación mencionados. El método asociado a la condición también modifica la memoria del proceso, creando el valor *ev_dp_enviado*, de forma que la condición solamente sea válida una vez por cada instancia del proceso.

Cuando el líder de planeación vea en su lista de tareas de DotProject la planeación de un nuevo release, deberá evaluar las solicitudes de cambio seleccionadas cuando se inicializó el proceso y asignar responsables a aquellas que realmente si deban incluirse en el nuevo release. Esta acción de asignar responsable debe informarle a Cumbia para que, dentro de la información asociada al proceso, se sepa qué solicitudes harán parte del release. Este punto es muy importante porque, como se dijo ya, el concepto de release no existe en ninguna de las aplicaciones y sin este paso sería imposible saber cuales son las solicitudes de cambio que hacen parte de un release en particular. La forma en la que se notifique al proceso la asignación de una solicitud de cambio será mediante un evento generado por Changeset y consumido por Cumbia a través del WS-Eleggua de la actividad de planeación. Cuando el líder de planeación concluya la asignación de responsables a las solicitudes de cambio cierra la tarea de planeación que fue creada en DotProject. Esta acción también debe ser informada al WS- Eleggua para que el proceso sepa que ya fueron seleccionadas todas las solicitudes de cambio que van a hacer parte del release. Las funciones utilizadas para manejar los eventos descritos son:

F(“Asignación Responsable SC”, e) = notificarSCAsignada(e)

F(“Cierre Tarea DotProject”, e) = notificarCierreTareaPlaneación(e)

El método "notificarSCAsignada" extrae del evento el identificador de la solicitud de cambio asignada y lo almacena en la memoria, asociado al release, en una colección de identificadores llamada "SC_Seleccionadas". El método "notificarCierreTareaPlaneación" agrega un valor a la memoria, tareaPlaneacionTerminada, que indica la terminación de la tarea realizada por el Líder de Planeación.

En la tabla 1, se resume el comportamiento que tiene la actividad de planeación en el proceso, a medida que se van ejecutando las reglas de cooperación entre las aplicaciones.

Tabla 1. Comportamiento del proceso

Regla ECA	Método Invocado	Comporatmiento
Crear Proyecto	crearProyecto TareaDP	Se crea el valor ev_dp_enviado en la memoria
Asignar solicitudes	notificarSC Asignada	Se almacena en memoria el identificador de una de las solicitudes de cambio asociadas al release
Cerrar tarea planeación	notificarCierreT areaPlaneación	Se crea el valor tareaPlaneacionTerminada en la memoria

5.2.3 Control de la Ejecución del Proceso

En lo que explicado hasta el momento las acciones realizadas por los usuarios solamente producen cambios sobre la memoria de la instancia del proceso. En esta sección se verá cómo el WS-Eleggua maneja acciones que los usuarios realizan sobre las aplicaciones participantes y que producen un cambio en el estado de ejecución del proceso, es decir hacen que una actividad termine. En general, el fin de la ejecución de una actividad dependerá de una condición sobre el estado de la memoria de un proceso que el WS-Eleggua deberá evaluar y detectar cada vez que sea necesario. Cuando una de estas condiciones se cumpla, el Workspace deberá informar a la máquina de ejecución de Cumbia para que termine la ejecución de una actividad y continúe la ejecución del proceso según se haya especificado.

Para el caso de estudio, la condición de terminación de la actividad de planeación evalúa si la memoria tiene un valor bajo el nombre de "tareaPlaneacionTerminada". Este valor fue puesto cuando se recibió un evento de cierre de tarea de planeación en DotProject, indicando que el líder de planeación terminó de asignar responsables a las solicitudes de cambio incluidas en el release. Es posible tener condiciones de terminación mucho más complejas. Por ejemplo, la actividad de planeación podría terminar después de que se asignaran las primeras 10 solicitudes de cambio.

5.3 Componente LETCumbia

Lo que hace que en este proyecto el motor de workflows Cumbia, no sea para Eleggua exactamente igual a cualquier otra aplicación es el hecho de que al mismo tiempo pueden estar en ejecución varias instancias del mismo proceso. Desde el punto de vista de Eleggua esto equivaldría a tener no una sino varias aplicaciones y haría necesario la modificación, durante la ejecución, de las reglas de cooperación y de las aplicaciones conectadas. Si, por el contrario, Eleggua no realizara ninguna distinción entre las instancias y le entregara todos los eventos dirigidos a Cumbia a todos los workspaces suscritos a por lo menos un evento, entonces podría haber problemas de rendimiento. Además, en este caso cada evento tendría que tener información adicional para permitirle saber a cada workspace si debe procesar o no el evento.

Para evitar este problema, la solución propuesta incluye un elemento llamado LETCumbia (Logical Event To Cumbia Mapping), que es un componente en el cual cada WS-Eleggua puede registrar los eventos de su interés indicando tanto el tipo de evento como valores esperados para sus parámetros. Con este sistema es posible enviar cada evento únicamente a los workspaces interesados, minimizando así la cantidad de información enviada. En la figura 8 se puede ver cómo la estructura del sistema de suscripciones permite la realización de suscripciones y la publicación de eventos dirigidos a instancias específicas.

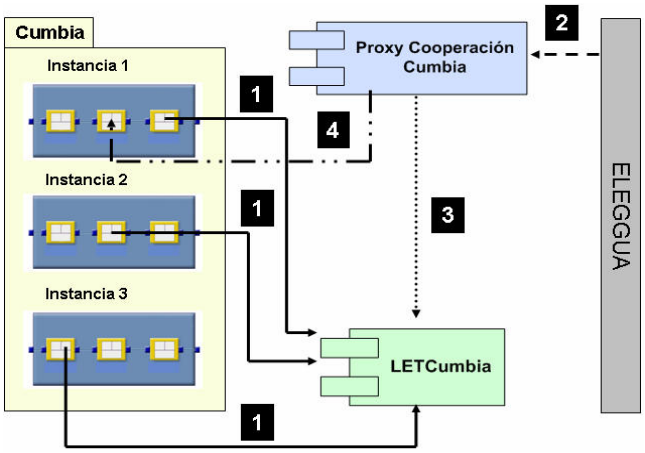

Figura 8. Suscripción a eventos.

1. Suscripción a eventos de interés: Cada uno de los WS-Eleggua, debe registrar ante el componente LETCumbia su interés por un evento en particular. Esta suscripción indica el tipo de evento y los valores esperados para cada uno de los parámetros. Por ejemplo, la actividad de planeación debe suscribirse a los eventos de tipo "Asignación Responsable SC" y cuyo parámetro identificador de la solicitud de cambio sea uno de los que inicialmente se seleccionaron al crear la instancia del proceso. La suscripción de cada WS-Eleggua se realiza en el momento que inicia la ejecución de la actividad a la que pertenece. Cada uno de estos workspaces cuenta con el método registerInterests, que se encarga de realizar las suscripciones a los eventos que sean de interés para ese workspace en particular. Las suscripciones del workspace deben mantenerse sólo mientras que este esté activo: cuando termina su ejecución las suscripciones son eliminadas.

2. Notificación de eventos: Eleggua, notificará al motor de Cumbia todos los eventos que sean de interés para el motor de ejecución de procesos, sin importar para qué actividad o en qué instancia de proceso se encuentra el workspace interesado.

3. Búsqueda de receptores Cumbia para el evento: En este paso, el proxy de cooperación de Cumbia, en lugar de informar a todos los WS-Eleggua en todas las instancias del proceso sobre la llegada

del evento, pide al componente LETCumbia que le indique qué Workspaces están interesados y sólo a esos les remitirá el evento.

4. Notificación: el proxy de cooperación de Cumbia informa sobre la llegada del evento sólo a aquellos workspaces que están interesados.

6. TRABAJOS RELACIONADOS

Además de los trabajos ya mencionados, hay otros proyectos que enfrentan, al menos parcialmente, los mismos problemas que nosotros. En este proyecto hay involucradas un gran número de tecnologías diferentes y para cada una hay alternativas que podríamos haber utilizado. Sin embargo, la escogencia de Eleggua y Cumbia se debió a que nos ofrecían ventajas importantes para llevar a cabo este proyecto.

Los sistemas basados en notificación de eventos son un estilo arquitectural ampliamente utilizado para sistemas empresariales de bajo acoplamiento [19]. Estos sistemas tienen gran relevancia en dominios tales como integración y reconocimiento de aplicaciones, monitoreo de aplicaciones, movilidad entre otras. Además de Eleggua, existen otros proyectos como Hermes [17], una plataforma de integración de eventos que da soporte estándar a servicios de observación y notificación de eventos, o JEDI [8], una infraestructura para desarrollar un sistema distribuido de administración de workflows. La principal desventaja de Hermes respecto a Eleggua es la poca escalabilidad hacia redes de área Global, mientras que la desventaja de JEDI es que está estrechamente acoplado a OPSS (ORCHESTRA Process Support System), un motor de workflow que no tiene la suficiente versatilidad como para reemplazar a Cumbia en nuestro proyecto. Además JEDI utiliza una infraestructura centralizada, lo cual puede generar problemas de escalabilidad.

Como hemos visto, cada vez es más importante para las empresas poder modelar y ejecutar sus procesos de negocio. Para atacar diferentes aspectos de estos problemas existe una gama de soluciones que abarcan desde el problema del modelado hasta el problema de la ejecución. BPMN es una iniciativa respaldada por la OMG para definir un modelo que permita modelar procesos de alto nivel de una forma estándar. BPMN es muy utilizado gracias a que existen varias herramientas que facilitan la edición gráfica de los procesos, pero los resultados obtenidos con estas herramientas suelen tener solamente la información para servir de documentación para el proceso y no para ejecutarlo. Además, hay varias características de BPMN que son adecuadas para la descripción de los procesos, pero que hacen que la construcción de un motor de BPMN sea bastante complicada. Según lo que hemos dicho ya en este artículo, si se usa a Cumbia como motor de ejecución de procesos, realizando las transformaciones adecuadas sería posible ejecutar procesos definidos usando BPMN.

Una alternativa que permite tener un mayor control sobre la ejecución es XPDL (XML Process Definition Language) un lenguaje estándar de definición de procesos propuesto por la WFMC [21]. En XPDL se puede especificar la interacción con aplicaciones, que debe realizarse de manera sincrónica y consumiendo servicios. Para esto primero se definen en forma abstracta el conjunto de aplicaciones que interactúan con el proceso y luego el motor se encarga de mapearlos a servicios concretos. Una posible desventaja de XPDL es que una representación gráfica no hace parte del estándar, por lo cual cada herramienta define su propia sintaxis y las extensiones que considere necesarias.

Un sistema manejador de workflows que cada vez es más popular es JBPM (Java Business Process Mangement) [14]. En este sistema, desarrollado por JBoss Inc. los procesos son expresados como grafos dirigidos cuya ejecución es controlada por eventos. Para JBPM el lenguaje estándar de definición de procesos es JPDL, pero una parte importante de la definición de un proceso debe realizarse en Java. Este detalle es muy importante, puesto que le da libertad en cuanto a lo que puede realizarse como actividad dentro de un proceso. JBPM no cuenta en este momento con herramientas de administración de procesos y JPDL no tiene una sintaxis gráfica definida.

Investigando sobre propuestas de integración entre EAI y workflow, encontramos [10], en la cual se presenta una infraestructura basada en eventos para soportar la administración de procesos. La integración entre máquinas de workflow se logra mediante un servicio de notificación de eventos. Una ventaja que presenta Eleggua sobre esta propuesta esta dada por las características adicionales que posee: automatización en la generación de reglas de cooperación, monitoreo y control de la infraestructura.

Si se piensa que un proceso podría ser ejecutado por una sola persona que interactúe con una sola aplicación, entonces tenemos que mencionar varias herramientas que son usadas para diseñar la interacción de un usuario con una aplicación Web. Entre estas herramientas se encuentran WebFlow [13], una herramienta basada en Spring que permite diseñar la interacción de un usuario con una herramienta con base en un proceso, y Seam [12], una herramienta similar a WebFlow en la que los procesos son ejecutados en JBPM. Por otra parte, un proyecto del Politécnico de Milán extiende a WebML, una notación para modelamiento Web, con los elementos básicos necesarios para modelar procesos: de esta manera lo que ellos obtienen es un sistema en el cual se puede especificar cómo se ejecuta (en términos de interacción Web) un proceso definido usando BPMN [3].

Finalmente quisiéramos también mencionar algunos aspectos relacionados con BPEL (Business Process Execution Language) [1, 2, 4, 16], por tratarse de una herramienta muy difundida y que podría solucionar algunos de los problemas descritos anteriormente. BPEL es un lenguaje que permite la definición de procesos y su posterior ejecución con base en la interacción con aplicaciones a través de Web-Services. Aunque su sintaxis es textual, basada en XML, existen varias herramientas que permiten modelar procesos fácilmente usando una sintaxis gráfica. Este detalle podría hacer de BPEL una herramienta adecuada para ser utilizada por personas con limitados conocimientos técnicos. El problema en BPEL es que, al igual que con otras soluciones, para definir completamente un proceso con aplicaciones integradas es necesario incluir una gran cantidad de detalles técnicos. Esto significa que en un sólo punto es necesario incluir tanto información del nivel del negocio (la estructura del proceso) como información del nivel de soporte del proceso (cómo se realiza la comunicación con cada aplicación y cómo se deben transformar y utilizar los datos obtenidos). Por último es importante mencionar que en BPEL el acceso a todas las aplicaciones debe realizarse a través de Web-Services, y esto puede representar una limitación importante en muchos casos.

7. TRABAJO FUTURO

El proyecto, tal como ha sido presentado, se encuentra apenas en una fase inicial. Para continuar con su desarrollo hemos identificado ya varios temas en los que esperamos seguir investigando.

En primer lugar, hemos venido realizando pruebas para validar nuestra aproximación utilizando procesos diferentes. Aunque en este momento no tenemos resultados específicos que reportar, las pruebas hechas hasta el momento nos han permitido afinar algunos detalles de nuestra propuesta original.

Uno de los temas más interesantes, sobre el cual estamos ya trabajando, es la posibilidad de generar la mayor cantidad de código posible a partir de la definición del proceso de alto nivel y de las reglas de cooperación entre aplicaciones. El principal objetivo de este trabajo es facilitar aún más la automatización de los procesos, permitiendo que cada vez se requieran menos conocimientos técnicos para poner en funcionamiento una solución. Este trabajo incluirá, como se menciona dentro del artículo, la revisión de la definición de las funciones, para disminuir la cantidad de código requerido.

Otro campo, en el cual esperamos capitalizar resultados de otros trabajos desarrollados dentro de nuestro grupo de investigación, es en la recolección y el análisis de métricas a partir de la ejecución de los procesos. Estas métricas permitirían, por ejemplo, estudiar y comparar el rendimiento de varias versiones de un proceso o hacerle seguimiento al desempeño de un usuario transversalmente a un conjunto de procesos. Este trabajo de recolección de métricas debe incluir la integración con un sistema de monitores y de tableros de control empresariales.

Un tema del que no nos hemos preocupado hasta el momento es el de la distribución de los procesos. En este momento los procesos materializados se encuentran siempre centralizados en el nodo donde se encuentre el motor de Cumbia. Nosotros esperamos que, aprovechando las facilidades que ofrece Eleggua para el manejo de diferentes contextos de distribución, podamos extender nuestra solución para que incluya la distribución de la ejecución de los procesos, esto principalmente para mejorar la eficiencia de las comunicaciones.

8. CONCLUSIONES

La unión entre Cumbia y Eleggua nos proporciona una plataforma que permite materializar procesos de negocio y soportar su ejecución sobre aplicaciones que anteriormente eran independientes. Estos procesos pueden ser diseñados desde una vista empresarial, sin tener en cuenta aspectos técnicos presentes en cualquier implementación. Por otra parte, la integración de diferentes aplicaciones con la ejecución del proceso guía a los usuarios finales de estos sistemas en la realización de sus actividades, informándoles sobre los pasos que deben efectuar y otorgándoles la información necesaria para realizarlos.

Se puede definir la información que inicializará y administrará cada nueva instancia de un proceso, datos que generalmente provienen de diferentes aplicaciones. Todas las acciones de los usuarios -en las aplicaciones participantes- pueden ser transmitidas automáticamente hacia el proceso, resultando en modificaciones a la información con que inicialmente se instanció o cambios en el estado de ejecución del mismo. El proceso, por su parte, también puede informar a aplicaciones externas, asegurando la consistencia de los datos comunes o de aquellos datos que sólo se encuentran en el proceso, pero que son de interés para otro(s) sistema(s) involucrado(s).

Nuestro trabajo futuro facilitará la implementación de procesos en la plataforma Cumbia-Eleggua y, gracias a la visión global del proceso que permitirá monitorear su ejecución, proporcionaremos una valiosa herramienta empresarial que facilitará la toma de decisiones en tiempo real.

9. AGRADECIMIENTOS

Queremos agradecer a los profesores Rubby Casallas y Jorge Villalobos, del grupo de Construcción de Software de la Universidad de los Andes, tanto por su participación dentro del proyecto como por los valiosos aportes que realizaron a este artículo. Este trabajo está apoyado por el proyecto CARAMELOS que es financiado por el VLIR [29].

10. REFERENCIAS

[1] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, e I. Trickovic. “Business process execution language for web services, version 1.1”. Technical report, BEA, IBM, Microsoft, SAP, Siebel Systems, May 2003.

[2] M. Blow, Y. Goland, M. Kloppmann, F. Leymann, G. Pfau, D. Roller, y M. Rowley. BPELJ: BPEL for Java. BEA and IBM, March 2004. white paper, http://www-106.ibm.com/developerworks/java/library/j-diag0925/

[3] M. Brambilla, S. Ceri, P. Fraternali, e I. Manolescu. “Process modeling in web applications”. ACM Transactions on Software Engineering and Methodology, 15(4):360–409, October 2006.

[4] White, S. A.: Business Process Modeling Notation Specification. BPMI.org. 2004 .

[5] R. Casallas, C. Acero, y N. López. From high level business rules to an implementation on an event-based platform to integrate applications. In Proceedings of the International EDOC Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE 2005), Enschede, The Netherlands, September 2005.

[6] Casallas, R., Lopez, N., Correal, D.: Eleggua: An Event Infrastructure for Application Cooperation. Lecture Notes in Informatics, Vol. P-70. Springer-Verlag, Berlin Heidelberg Erfurt (2005) 109-123.

[7] O. González: Monitoreo, Control y Escalabilidad de una Plataforma de Integración Basada en Eventos. Tesis de Maestría. Universidad de los Andes, 2006

[8] G. Cugola, E. Di Nitto, y A. Fuggeta. “The Jedi event-based infrastructure and its application to the development of the OPSS wfms”. IEEE Transactions on Software Engineering, 27(9), 2001.

[9] DotProject, The Open Source Project Management tool. Documentación disponible en http://www.dotproject.net/

[10] J. Elder y E. Penagos. “Towards distributed workflow process management”. Technical report, AT&T Research Labs, 2006.

[11] Grupo QualDev. Proyecto Changeset. Disponible en http://qualdev.uniandes.edu.co/.

[12] JBoss (2006): SEAM - Contextual Components A Framework for Java EE 5 Version: 1.0.CR2. Recuperado 15.05.2006:http://docs.jboss.com/seam/reference/en/pdf/seam_reference.pdf.

[13] K. Donald, E. Vervaet. "Spring Web Flow, ReferenceDocumentation", version 1.0 EA, Marzo 2006. Disponible en: http://static.springframework.org/spring-webflow/docs/1.0-ea/reference/index.html

[14] J. Koenig. JBoss jBPM white paper. http://www.jboss.com/pdf/jbpm whitepaper.pdf, 2004.

[15] Object Management Group: Software Process Engineering Metamodel Specification, version 1.1, January 2005. Recuperado 05.01.2006: http://www.omg.org/docs/formal/05-01-06.pdf

[16] C. Peltz. "Web services orchestration - a review of emerging technologies, tools, and standards". Technical report, HP, January 2003.

[17] P. Pietzuch. "Hermes: A Scalable Event-Based Middleware". PhD thesis, Queens College, University of Cambridge, 2004.

[18] D. Rivera y N. López. "Administración y planeación fundamentada en datos históricos de las solicitudes de cambio, 2006." Universidad de los Andes, Bogotá, Colombia.

[19] R. Silva, C. De Souza, y D. Redmiles. "The design of a configurable, extensible and dynamic notification service". Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS'03), June 2003.

[20] M. Sánchez, P. Barvo: Construcción de una línea de producción de motores de workflow basada en modelos ejecutables. Tesis de Maestría, Universidad de los Andes, 2006.

[21] WfMC, Workflow Standard, Process Definition Interface - XML Process Definition Language, Technical Report Document Number WFMC-TC-1025, Workflow Management Coalition, 2005.

[22] J. Meng, S. Su, H. Lam, A. Helal, J. Xian, X. Liu, S. Yang.: *"DynaFlow: A Dynamic Inter-Organizational Workflow Management System"*. Int. Journal of Business Process Integration and Management (IJBPIM) 1, 2 (2005).

[23] J. Bae, H. Bae, S. Kang, and Y. Kim.: *"Automatic control of workflow processes using ECA rules,"* IEEE Trans. Knowl. Data Eng., vol.16, no.8, pp.1010–1023, Aug. 2004.

[24] O. Michiko, N. Komoda.: *Multiple Type Workflow Model for Enterprise Application Integration.* Proceedings of the 34th Annual Hawaii International Conference on System Sciences, 2001, (2001) pp. 2554 -2561, ISBN: 0-7695-0981-9/01.

[25] K. Myungjae, D. Han, J. Shim.: *"A Framework Supporting Dynamic Workflow Interoperation and Enterprise Application Integration"*. Proceedings of the 35th Hawaii International Conference on System Sciences, (2002) pp. 3798-3807, ISBN: 0-7695-1435-9/02.

[26] N. Craven, D. Mahling.: *"Goals and processes: a task basis for projects and workflows"*. In Proceedings of Conference on Organizational Computing Systems (Milpitas, California, United States, August 13 - 16, 1995). N. Comstock and C. Ellis, Eds. COCS '95. ACM Press, New York, NY, 237-248. DOI= http://doi.acm.org/10.1145/224019.224045

[27] L. Aversano, A. Cimitile, A. De Lucia.: *"A Communication Protocol for Distributed Process Management"*. Workshop on Global software development, collocated to the International Conference on Software Engineering, ICSE 2003, pp 16-20.

[28] M. Hammer and J. Champy. Reengineering the corporation: a manifesto for business revolution. Harper, 1993.

[29] CARAMELOS Project. http://ssel.vub.ac.be/caramelos/