ARQUITECTURA PARA LA DISTRIBUCIÓN DE PROCESOS DE NEGOCIO EMPRESARIALES EN SISTEMAS MÓVILES DE INFORMACIÓN

CRISTHIAN FIGUEROA MARTÍNEZ
ANDRÉS GUERRERO ARCINIEGAS

Director: Ing. JOSÉ ARMANDO ORDÓÑEZ CÓRDOBA

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELEMÁTICA
POPAYÁN
2007

ARQUITECTURA PARA LA DISTRIBUCIÓN DE PROCESOS DE NEGOCIO EMPRESARIALES EN SISTEMAS MÓVILES DE INFORMACIÓN

CRISTHIAN NICOLÁS FIGUEROA MARTÍNEZ
ANDRÉS GUERRERO ARCINIEGAS

Monografía presentada para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Director: Ing. JOSÉ ARMANDO ORDÓÑEZ CÓRDOBA

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELEMÁTICA
POPAYÁN
2007

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

PRESIDENTE DEL JURADO

_____

JURADO

Popayán, 2007

Contenido

Lista de figuras

1. INTRODUCCIÓN

Con el fin de conseguir la satisfacción de los clientes las empresas se concentran en la gestión eficiente de sus procesos de negocios y la consecuente optimización de los recursos. Para lograr este objetivo, es vital que exista una buena coordinación entre los trabajadores y que el tiempo que se destine para cada actividad sea el mínimo posible evitando así retardos en la ejecución de los procesos de la empresa.

El presente proyecto de grado se enfoca en mejorar estos aspectos, haciendo uso de dispositivos móviles, para realizar parte de las tareas de la empresa, y optimizar así los tiempos de ejecución de los procesos. Así, los empleados pueden estar en constante movilidad, de manera que cada trabajador puede gestionar las tareas de los procesos empresariales que le corresponden, desde su dispositivo móvil, ahorrando tiempo y contando con información de sus tareas en cualquier momento y lugar. Visto de este modo los dispositivos móviles de los empleados se convierten en sistemas móviles de información MobIS (Sistemas Móviles de Información).

Para proveer esta solución, se necesitan de mecanismos robustos de sincronización entre las partes y una constante comunicación entre todos los dispositivos y el motor central, lo cual evidentemente implica un alto consumo de ancho de banda, que en muchas ocasiones se hace inviable debido a los costos asociados. Para solucionar estos problemas, el presente proyecto de grado propone a BDMobIS una arquitectura que se basa en la distribución de sub-flujos de trabajo en dispositivos móviles minimizando precisamente las comunicaciones necesarias para la coordinación de los procesos de negocios. Esto se logra por medio de la aplicación de un algoritmo de particionamiento de grafos que reduce las comunicaciones entre los dispositivos móviles disminuyendo considerablemente los costos para las organizaciones. BDMobIS será descrita con detalle en capítulos posteriores.

Este trabajo de grado pretende mostrar la eficiencia del algoritmo de particionamiento multiagente en la reducción de las comunicaciones necesarias para resolver un proceso de negocio. BDMobIS es capaz de reducir el tiempo necesario para conseguir un objetivo empresarial. Además, aprovecha las ventajas de movilidad que tienen hoy en día los empleados de las empresas ya que hace uso de sus dispositivos móviles así como de un popular servicio como lo es el de mensajería corta.

El presente documento se organiza de la siguiente manera: el capítulo siguiente muestra los lenguajes utilizados para la composición de servicios web. El capítulo 3 se concentra en la escogencia de una representación gráfica que permita el particionamiento de grafos basándose en un contenido teórico matemático. El capítulo 4 hace referencia a los trabajos relacionados con BDMobIS. El capítulo 5 describe la arquitectura que define a BDMobIS como un sistema para la distribución de procesos de negocios entre sistemas móviles de información. El capítulo 6 muestra un ejemplo de implementación en el que se muestra una aplicación particular de la arquitectura propuesta. Finalmente, el capítulo 7 concluye lo realizado en este proyecto mostrando también los trabajos futuros para esta arquitectura.

Cabe aclarar que esta monografía es escrita con base en 3 publicaciones obtenidas por este proyecto las cuales apoyan: la arquitectura de BDMobIS propuesta por el capítulo 5, la representación gráfica escogida por este proyecto mostrada en el capítulo 3 y el prototipo desarrollado con base en BDMobIS explicado en el capítulo 6. La primera publicación fue hecha en la Revista de Avances en Sistemas e Informática, una revista clasificada en los índices Latindex y Publindex de COLCIENCIAS en categoría C; la segunda publicación fue aceptada en la Revista Tecnura de la Universidad Distrital de Colombia, otra revista clasificada en el índice Publindex de COLCIENCIAS en categoría C y la tercera fue oportuna para el Primer Congreso Colombiano de las Comunicaciones realizado en la ciudad de Bogotá en donde la IEEE acepto a BDMobIS como ponencia en dicho evento. Estas publicaciones se pueden encontrar en el anexo E.

Capítulo II

2. ESTÁNDARES DE WORKFLOW

Según Assaf Askin en la introducción al lenguaje BPML dada en [AAG01], un proceso de negocio se define como la interacción entre participantes y la ejecución de actividades de acuerdo a un conjunto de reglas definidas, con el fin de alcanzar un objetivo en común. La automatización de estos procesos, es lo que se conoce como Workflow, lo cual según Gabriel Cor en su presentación [GAC06], se define como el conjunto de actividades que coordinan personas y/o software. Estos flujos de trabajo, o Workflows, pueden estar representados por máquinas de estado, redes de Petri, autómatas finitos, grafos, entre otros, para su análisis matemático o visual. Dichos procesos de negocio son muy frecuentes en las empresas los cuales frecuentemente requieren de la integración de diferentes sistemas necesitándose enormes esfuerzos de interoperabilidad. Para este fin, se recurre a los servicios web ya que estos, independientes del lenguaje de implementación, aportan al usuario la funcionalidad que este necesite a través de estándares web ampliamente conocidos; de esta manera se puede llegar a la implantación de los procesos de negocios en las empresas. En la figura 2.1 se muestra la arquitectura de estándares y tecnologías en donde cada nivel depende de los inferiores.

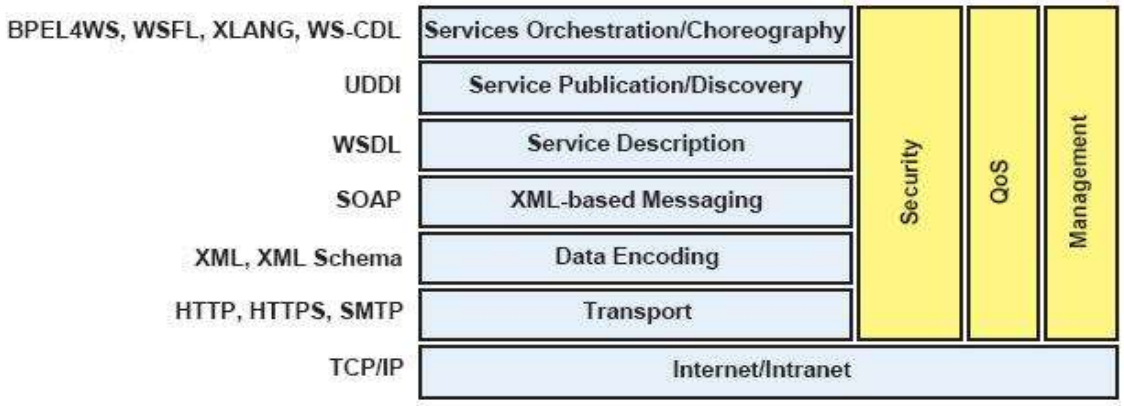

Figura 2.1. Arquitectura de los servicios web. Tomado de [YWL06]

En la figura se puede observar como los estándares SOAP (Protocolo Simple de Acceso a Objetos), WSDL (Lenguaje de Descripción de Servicios Web) y UDDI (Descripción, Descubrimiento e Integración Universal), los cuales están basados en XML (Lenguaje de Marcado Extensible) e Internet, constituyen el núcleo de los servicios web. Según lo dice Cheng Yushi en su cuarta sección acerca de composición de servicios [YWL06], SOAP define un protocolo simple basado en XML para el intercambio de estructuras de información en un entorno descentralizado y distribuido. Un mensaje SOAP es usado para que los usuarios accedan a los servicios web sin importar que plataforma ni que lenguaje estén utilizando. WSDL provee un formato XML para describir los servicios que están disponibles en la red exponiendo de ellos sus interfaces, las cuales poseen las operaciones que son capaces de realizar. UDDI es un conjunto de definiciones para un registro de servicios en donde la información de negocios, organizaciones, servicios Web disponibles e interfaces técnicas para su acceso es almacenada, incluyendo la definición de información para la publicación y descubrimiento de los mismos.

Como se puede observar en la figura 2.1., los servicios web y Workflow se interceptan en la capa de orquestación y coreografía la cual se ubica en el punto más alto de la arquitectura. Por consiguiente, es importante para el entendimiento de este proyecto tener claros algunos conceptos.

2.1. Coreografía

Un modelo de coreografía describe la colaboración entre una colección de servicios con el fin de alcanzar un objetivo en común. Por otro lado, la preocupación más importante de la coreografía de servicios web es la interacción de los servicios con sus usuarios. Tal y como lo dice Randy Howard y Larry Kerschberg en la presentación de un Framework para la semántica de Servicios Web dada en [RHL04], en donde se explica la gestión de servicios web, cualquier usuario automatizado o no automatizado de un servicio web, es un cliente de ese servicio. Estos usuarios podrían a su vez, ser servicios web o seres humanos. Las transacciones entre servicios web y sus clientes deben ser bien definidas en el momento de su ejecución y pueden constituirse por múltiples interacciones separadas cuya composición conforma una transacción completa. Esta composición, sus protocolos para mensajería, interfaces, secuenciamiento y la lógica asociada es lo que se considera como coreografía, definida en los requerimientos necesarios para la coreografía de Servicios Web expuesta por Daniel Austin et al. en el documento [ABP03].

2.2. Interfaz de Comportamiento

Aquí se tratan los aspectos de comportamiento de las interacciones en las cuales un servicio en particular puede encajar para alcanzar un objetivo. Una interfaz de comportamiento captura las dependencias entre las interacciones tales como las dependencias de control de flujo, las dependencias de flujo de datos, restricciones temporales, mensajes de correlación y las dependencias transaccionales, etc. A diferencia de la coreografía, la interfaz de comportamiento (llamada *proceso abstracto* en BPEL) se enfoca en capturar las interacciones, no de todo el proceso, sino de una parte en particular que lo integre. Algo que si tienen en común tanto la coreografía como la interfaz de comportamiento es que ninguna se relaciona con el comportamiento interno de las partes que interactúan en el proceso de negocio, como es el caso de la transformación interna de datos.

2.3. Orquestación

La orquestación describe tanto las acciones de comunicación como las acciones internas en las cuales un servicio encaja. Las acciones internas son las transformaciones de datos y las invocaciones de entes externos al proceso, como por ejemplo, servicios web. Ahora, tal y como se dijo, una orquestación podría también contener acciones de comunicación o dependencias entre acciones de comunicación que no aparecen en la ya nombrada interface de comportamiento. La orquestación es también llamada *proceso ejecutable* debido a que esta es ejecutada por *motores de orquestación*.

A partir de estas definiciones, se puede llegar a otras que complementan lo propuesto inicialmente. Según esto, es de vital importancia que el lector comprenda dos conceptos. El primero de ellos es el de proceso de negocio, el cual ya fue definido al inicio de este capítulo. Para que ello sea más entendible, este proyecto toma un caso de estudio en particular y lo modela como un proceso de negocio. Dicho ejercicio es una secuencia de pasos que, modelados correctamente, representan la solución de un caso crítico de la Universidad del Cauca. Esto se describirá posteriormente en el Capitulo 6. Ejemplos de procesos de negocios pueden ser la compra de bienes, la aprobación de créditos bancarios y hasta la simple traducción de textos, todos enmarcados dentro de Workflow. Debe quedar claro que el prototipo implementado es solo un ejemplo y que BDMobIS es flexible a todo tipo de actividades tales como la Banca, Seguridad Social, Administración

Legal y General y en fin, todo tipo de procesos que sean susceptibles de ser automatizados.

La segunda idea que se debe tener clara es que hay varias formas de representar un Workflow. Actualmente existen varios lenguajes para componer estas estructuras. Estos lenguajes deben procurar cubrir, según lo plantea Alistair Barros en una visión crítica hacia los lenguajes de composición de servicios web dada en [BPT05], las definiciones arriba nombradas: coreografía, interfaz de comportamiento y orquestación.

Adicional a esto, según lo explica Cheng Yushi en la cuarta sección acerca de composición de servicios web [YWL06], es muy importante que además de que el lenguaje escogido por el desarrollador soporte estas tres características, debe poder separar la parte abstracta del proceso, de la parte concreta. Esto con el fin de poder reutilizar todo lo que tenga que ver con el proceso de negocio en sí (participantes y las interacciones que hay entre ellos, es decir, la parte abstracta del proceso). Se explicarán ahora algunos estándares o lenguajes existentes con los cuales se pueden definir flujos de trabajo Workflow y por último se expondrá el lenguaje que este proyecto utilizará.

Algunos lenguajes se inclinan más hacia la coreografía como es el caso de BPEL. Hay otros estándares que se preocupan también por modelar procesos de negocios pero sin utilizar servicios web, como XPDL (Lenguaje XML para la Descripción de Procesos) y ebXML BPSS (Esquema para la Especificación de Procesos de Negocios), los cuales extienden su funcionalidad hacia seres humanos. En este punto hay que hacer una aclaración, ya que BPEL actualmente soporta la interacción con participantes humanos pero la versión utilizada por BDMobIS no alcanza este tipo de socios del proceso aunque si es flexible hacia versiones que lo hagan.

2.4. Lenguajes de Composición

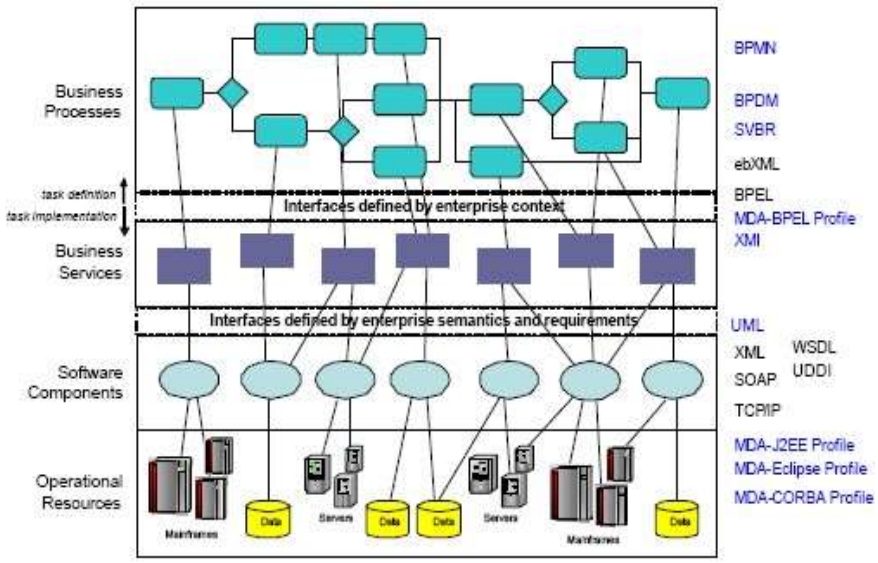

Figura 2.2. Ubicación de los estándares de Workflow en el mundo empresarial.

Se puede ver en la Figura 2.2. la ubicación de los diferentes lenguajes existentes hoy en día para la definición de procesos de negocio. Según esta gráfica, en la parte inferior se pueden apreciar todos los partes referentes a datos y su manipulación. En una capa superior, se pueden apreciar los protocolos SOAP y el lenguaje para la descripción de Servicios Web junto con el servicio para su descubrimiento UDDI. Como ultima capa encontramos los procesos de negocios junto con los lenguajes que se pueden utilizar para su especificación como lo son ebXML o BPEL.

A continuación se tratarán brevemente algunos de los lenguajes utilizados hoy en día para la definición de procesos de negocios. Entre estos esta el utilizado actualmente por BDMobIS, para el cual se hará una descripción de los elementos que principalmente lo componen.

2.4.1. WS – CDL (Lenguaje para la Descripción de la Coreografía de Servicios Web)

Es una especificación XML que apunta hacia la composición interoperable y la colaboración punto a punto entre los servicios web participantes del proceso de negocio. Este lenguaje provee una vista imparcial y global de las interacciones entre dos o más sistemas. Su objetivo es definir contratos o acuerdos entre varias partes, los cuales describen el comportamiento externo observable de los servicios web y sus clientes

(usualmente otros servicios web) mediante la descripción de los mensajes que entre ellos se intercambian. La idea es que las descripciones que ofrece el lenguaje WS – CDL, sean los bosquejos para definir los procesos abstractos en BPEL.

La coreografía descrita por WS – CDL es esencialmente un contenedor el cual almacena una colección de actividades que pueden ser ejecutadas por uno o más de los participantes. Estas actividades están clasificadas en tres tipos: actividades de control de flujo (*control – flow* activities), actividades *WorkUnit* y actividades básicas.

Hay tres actividades en la primera categoría (flow – control activities) llamadas *Sequence*, *Parallel* y *Choice.* Estas actividades son capaces de encerrar un número de sub actividades. Una actividad Sequence, por ejemplo, describe una o más actividades que deben ser ejecutadas en un orden secuencial. Una actividad Parallel describe una o más actividades que pueden ser ejecutadas en cualquier orden pero al mismo tiempo. Una actividad Choice describe la ejecución de una actividad escogida de entre un conjunto de actividades que compiten entre si.

Las actividades *WorkUnit* describen la ejecución condicional y posiblemente repetida, de una actividad. Sintácticamente, una actividad WorkUnit tiene varias partes: una referencia a la actividad que encierra, una guarda, una condición de bloqueo y una condición de repetición, las cuales se son modeladas como expresiones booleanas. Las condiciones de guarda y repetición son usadas para determinar si la actividad encerrada se ejecutará una o más veces. Así la ejecución de una actividad de este tipo comenzará con la evaluación de la condición de guarda. Si la condición de guarda es afirmativa, la actividad encapsulada se ejecutará, de otro modo se saltará. Una vez que la actividad se ejecuta por primera vez, se procede nuevamente a evaluar la condición de repetición. Dependiendo del resultado de dicha evaluación, se procede a ejecutar la actividad una vez más o simplemente se suspende el proceso. De esta manera son implementados los bucles en WS – CDL.

Finalmente en las actividades básicas se encuentran las actividades *Interaction, NoAction, SilentAction, Assign* y *Perform.* Las actividades de NoAction y SilentAction son las que le ordenan a un integrante del proceso no hacer nada o en el caso de que se necesite realizar una acción lo haga en un contexto que no afecte al resto de la

coreografía. La actividad Assign es la actividad que transfiere el valor de una variable a otra con el fin de darle continuidad al proceso. En BPEL, esta actividad tiene el mismo nombre (<*assign*>) y tiene la misma función. La actividad Perform es usada para hacer que otra coreografía se ejecute dentro del mismo contexto que la actual.

Algo muy importante que debe tener todo lenguaje de composición de servicios es la capacidad de poder hacer intercambio de información entre partes. Este tipo de operaciones le son permitidas a WS-CDL gracias a la actividad *Interaction*. Una interacción describe precisamente el intercambio de información concentrándose en el ente que la recibe. Se podría decir entonces que una Interaction tiene 3 modalidades: la primera es hacer peticiones o *request,* la segunda es generar respuestas o *respond* y la tercera es hacer peticiones que requieren de una respuesta o *request–respond.* En una actividad Interaction se pueden identificar tres partes importantes. La primera corresponde a los participantes involucrados en la conversación. La segunda corresponde a la descripción de la información que va a ser intercambiada y la tercera trata acerca del canal que va a ser utilizado para la comunicación. Un ejemplo de una actividad Interaction se encuentra referenciada en la visión crítica de este lenguaje dada por Alistair Barros en [BPT05] y se reproduce a continuación.

```
<interaction name= "QuoteRequest"
    initiate= "true"
    align= "false"
    operation= "ReceiveRFQ"
    channelVariable= "requestQuoteChannel">
    <participate toRole= "supplier"
        fromRole= "customer"
        relationship= "GetQuote"/>
    <exchange name= "e1"
        action= "request"
        informationType= "quoteRequest" >
        <send variable= "rfq"/>
        <receive recordReference="record1"
        variable= "rfq"/>
    </exchange>
```

```
<record name= "record1"
    when= "after" >
<source variable= "rfq"/>
<target variable= "quoteForm"/>
</record>
</interaction>
```

Según lo anterior, la especificación existe puramente para la especificación de procesos abstractos de negocios sin importar la plataforma sobre la cual la implementación funcione y sin tener en cuenta el lenguaje de programación que se use para desarrollar los servicios web que interactúan en el proceso. En este lenguaje intervienen principalmente los mensajes que intercambian las partes, los cuales son modelados a través de variables y banderas (Variables, Tokens) cuyo tipo puede ser especificado en el archivo WSDL o en los esquemas XML.

2.4.2. BPML (Lenguaje para la Gestión de Procesos de Negocios)

Según lo señala Assaf Arkin en la especificación de este lenguaje dada en [AAG01], los procesos de negocios son estructuras de acción adaptativas en las cuales varios participantes juegan una variedad de roles. La gestión en los procesos de negocios hace posible que la colaboración de tales participantes se haga de una manera confiable, segura y escalable gracias al soporte que prestan topologías de procesos dinámicas las cuales permiten separar a los participantes de los procesos.

Cualquier sistema que utilice este lenguaje debe cumplir con las características que se citan a continuación:

a. Permitir el modelamiento de procesos de negocios los cuales deben ser totalmente independientes de los protocolos de B2B (Business to Business).

b. Preocuparse por mantener el proceso de negocio modelado en un estado robusto y fijo cuando intervenciones humanas ocurran durante la ejecución. Esto vendría implementado por la inclusión de manipuladores de fallas las cuales garantizarían el estado consistente del proceso.

c. Ser capaz de adaptarse dinámicamente a los posibles cambios que ocurran en las condiciones del proceso de negocio.

d. Permitir la ejecución de diferentes tareas con el fin de acercarse a la realidad de los procesos de negocios empresariales. Tales actividades pueden ser: obtención de información, actualización de la información, comunicación síncrona y asíncrona, transacciones de ciclos de vida cortos y largos, toma de decisiones, interacción con usuarios y acceder a fuentes de información locales y remotas.

A continuación se nombrarán algunas partes importantes de este lenguaje.

2.4.2.1. Mensajes

Dado que BPML es un lenguaje para la composición de servicios web, es vital para dicha composición el intercambio de mensajes. BMPL emplea un modelo basado en mensajería en el cual todos los participantes del proceso interaccionan a través del intercambio de mensajes en donde el proceso es quien define la forma que debe tener el flujo de mensajes que se intercambian al igual que la manera en que circula la información embebida en cada mensaje.

Cada proceso en BPML incluye la definición de todos los mensajes que se emplean en la comunicación entre el proceso y los participantes. Los mensajes que son específicos al proceso están incluidos en su definición, mientras que los específicos a un participante se importan de la definición del proceso central en la que está contenido dicho participante.

2.4.2.2. Participantes

Al igual que en BPEL, los participantes son las entidades con las cuales el proceso central interactúa. Los participantes en un proceso BPML pueden ser: sistemas IT, aplicaciones, usuarios y otros procesos. La definición del proceso refleja dos tipos de participantes: los estáticos y los dinámicos. Los participantes estáticos son definidos cuando el proceso es modelado de forma que este siempre interactuará con el mismo conjunto de participantes estáticos. Por otro lado, los participantes dinámicos no se conocen al momento de modelar el proceso de negocio. Los participantes dinámicos son muy importantes ya que le ayudan al proceso a tomar ventaja de entornos que pueden cambiar dinámicamente. Esta es indudablemente una ventaja que tiene BPML frente a

otros leguajes que se confían en la definición de participantes estáticos para sus procesos de negocios.

2.4.2.3. Actividades

Los procesos se basan tanto en la ejecución de actividades como en el flujo de información entre ellas y los participantes. BPML tiene disponibles las siguientes actividades para el modelamiento de un proceso de negocio:

a. Las actividades simples, o básicas como las llamarían en BPEL, son usadas para modelar la producción y consumo de mensajes, tareas y datos.

b. Las actividades complejas, o estructuradas como las llamarían en BPEL, son usadas para controlar el flujo de información sin importar que sea serial, paralelo o condicional y además para modelar estados complejos que a su vez constan de múltiples sub actividades.

Dentro de las actividades simples se encuentran:

- La actividad *consume*: representa la aceptación y el consumo de un mensaje. También puede ser usada para modelar como el proceso acepta una tarea, requiere información o simplemente recibe algo.

- La actividad *produce*: representa la producción y entrega de un mensaje. También puede ser usada para modelar como un proceso delega una tarea, almacena datos o simplemente crea algo.

- La actividad *operation*: se utiliza para modelar acciones atómicas tal y como invocar un servicio, creación de cuentas o cambio de los términos de una orden.

Dentro de las actividades complejas se encuentran:

- La actividad *sequence*: modela la ejecución serial de actividades.

- La actividad *all*: modela la ejecución paralela de actividades.

- La actividad *choice:* modela la posibilidad de escoger una rama de entre un conjunto de mutuos flujos exclusivos.

Las actividades de procesamiento son usadas para gestionar los distintos procesos como por ejemplo para dividir y unir procesos, para suspender y completarlos y finalmente para repetir actividades o sentencias como es el caso de los bucles finitos o loops.

BPML soporta el patrón de juntura, permitiéndole a un proceso encajar en el consumo sincronizado o la producción de mensajes, para la coordinación de actividades ejecutadas por múltiples participantes. En resumen, lo que este patrón quiere decir es que BPML le permite a un proceso tener sincronía de mensajes con el fin de realizar una acción u otra dependiendo de las respuestas dadas por mas de un ente dentro del proceso. Esto se puede aclarar siguiendo las páginas 11 y 12 de la especificación de BPML dada por Assaf Arkin y Ashish Agrawal en [AAG01].

2.4.2.4. Reglas

La lógica compleja de negocios demanda que un proceso en unas ocasiones seleccione una actividad de entre varias y en otras discrimine información hasta que algo ocurra. Esto es expresado en BPML bajo la forma de reglas que afectan la selección de actividades (ramificación y repetición) y gobiernan el consumo de mensajes. Esto dado en la especificación dada por Arkin en [AAG01].

La ramificación en un proceso es consecuencia de la toma de decisiones por parte del mismo. Este fenómeno modela la manera en la cual la ejecución del proceso será afectada por la información recolectada y creada durante el ciclo de vida del sistema.

La ramificación de un participante del proceso es consecuencia de alguna decisión tomada por este y es comunicada al proceso bajo la forma de un mensaje y se utiliza para modelar procesos colaborativos en los cuales el proceso reacciona a peticiones o reportes hechos por sus participantes.

2.4.2.5. Transacciones

BPML soporta dos modelos para transacciones: coordinados y extendidos. El modelo coordinado para las transacciones provee una garantía de todo o nada, la cual asegura que todos los participantes de una transacción estén de acuerdo ya sea con completarla o con abortarla. Este modelo soporta transacciones aisladas lo cual le da la habilidad de soportar transacciones coordinadas distribuidas. El modelo extendido para las transacciones se despreocupa por los requerimientos de transacciones aisladas mientras conserva la naturaleza de todo o nada.

2.4.2.6. Procesos

Los procesos abstractos definen la interacción entre los procesos y sus participantes. La parte abstracta omite información no pertinente a un conjunto particular de participantes tal así que no puede ser ejecutada. Los procesos abstractos pueden describir un sistema o un socio de negocios y es por ello que los servicios web, gracias a sus WSDL, pueden ser mapeados a dichas definiciones.

Los procesos ejecutables son los que definen completamente un proceso y son precisamente estos los que permiten la ejecución satisfactoria de los mismos. BPML usa tanto los procesos ejecutables como los procesos abstractos para gestionar el ciclo de vida del proceso, su disponibilidad y para establecer restricciones en la manera como estos son utilizados.

2.4.3.WSFL (Lenguaje de Flujo de Servicios Web)

WSFL es un lenguaje utilizado para describir la composición de servicios web, cuya definición puede ser extendida por Frank Leymann en la especificación de este lenguaje dada en [PLF01]. Para describir la composición de servicios web, WSFL considera dos tipos de composiciones:

- El primer tipo de composición es la que arroja como resultado la descripción de un proceso de negocio. En ella se especifican los *patrones de uso* apropiados de una colección de servicios web de forma que el resultado de dicha composición sea la descripción de cómo alcanzar un objetivo de negocio. La composición es creada describiendo como utilizar la funcionalidad proporcionada por la colección especificada de servicios web compuestos. Esto también es conocido como coreografía de servicios web,

término explicado anteriormente. WSFL modela este tipo de composiciones como especificaciones de la ejecución secuencial de la funcionalidad provista por los servicios web compuestos. Por esta razón, este tipo de composición es también llamada por la especificación como *modelo de flujo* el cual puede ser usado tanto para modelar procesos de negocios como Workflows basados en servicios web. Este modelo de flujo define la estructura del proceso de negocio por medio de actividades, las cuales describen paso a paso el proceso, y por medio de datos y enlaces de control los cuales representan las reglas de secuenciamiento y flujos de información entre dichas actividades. Para cada actividad, se identifica un servicio responsable por la ejecución de esa parte del proceso y se define una asociación con las operaciones que este provee, tal y como lo especifica Frank Leymann en la definición de este lenguaje dada en [PLF01].

- El segundo tipo de composición especifica *patrones de interacción* de una colección de servicios web. En este caso el resultado es una descripción global de cómo todos los socios de un proceso de negocio interactúan. Aquí, WSFL no especifica una ejecución secuencial. En lugar de ello, provee una descripción de cómo los Servicios Web compuestos interactúan entre sí. Estas interacciones son modeladas como enlaces o conexiones entre las interfaces de un servicio web con las operaciones que otro tiene disponibles. Debido a la naturaleza descentralizada y distribuida de estas interacciones, la especificación incluye el término de *modelo global* para referirse a este tipo de composiciones.

Otra cualidad importante con que cuenta la especificación es la de la composición recursiva. Dicha característica muestra que WSFL habilita la interconexión de composiciones de servicios web ya sean de tipo *modelos de flujo* o *modelos globales*. De manera que, una composición de servicios se muestra así misma como un nuevo servicio web la cual puede ser utilizada por otra composición teniendo de esta manera una composición de composiciones.

Adicional a esto, WSFL soporta interacciones tanto jerárquicas como de tipo punto a punto. Las interacciones jerárquicas son a menudo estables en relaciones de término prolongado (long - term) entre socios o partners. Las interacciones punto a punto reflejan relaciones que a menudo se establecen dinámicamente bajo el principio de *por –*

instancia. Estos modelos servirán para examinar un poco la forma en que el lenguaje modela un proceso de negocio o un Workflow en general.

2.4.4.BPEL (Lenguaje de Ejecución de Procesos de Negocio)

Lenguaje utilizado por BDMobIS. Según lo explica Frank Leymann en la especificación [CYG03], este lenguaje se basa en el secuenciamiento de servicios web para sacar su máximo provecho y llegar a la consecución de un objetivo, es decir, la ejecución de un proceso de negocio. En sus inicios BPEL era simplemente eso, un lenguaje. Este concepto solamente encerraba ideas como el secuenciamiento de un solo proceso en el cual otros ayudantes eran poco aclamados por el sistema. Los procesos de negocios de las empresas eran, por decirlo así, menos complejos que los que hoy en día se definen y es por tal motivo que algún protocolo o método de organización más complejo era inoficioso. Hoy en día las empresas demandan de herramientas más poderosas en el tratamiento de sus procesos de negocios en donde la intervención de diferentes socios o partners es algo necesario para solucionar un problema crítico interno de una empresa.

En este aspecto hay que aclarar que existe cierta diferencia entre BPEL y BPEL4WS (Lenguaje de Ejecución de Procesos de Negocios para Servicios Web) y sus posteriores versiones. Dado que las organizaciones en sus inicios solo interactuaban internamente, el lenguaje básico necesario para especificar procesos de negocios era simplemente BPEL. Posteriormente, las organizaciones comenzaron a definir actividades más complejas en las cuales más de un participante estaba incluido en el proceso de negocio. Estos entes exponían al entorno web lo que podían aportar haciendo uso de servicios web y es por ello que BPEL migra hacia su versión BPEL para servicios web o como su sigla lo indica, BPEL4WS. Este lenguaje es creado gracias a la unión de esfuerzos hechos por organizaciones como IBM, BEA, Microsoft y Siebel Systems. La ultima versión propone de nuevo el uso de la sigla BPEL pero siguiendo la misma filosofía anteriormente planteada. En ella se realizan modificaciones como la eliminación de las etiquetas <partner> entre otras.

Es precisamente el concepto de Partner lo que hace de BPEL un lenguaje poderoso en donde la invocación asíncrona o síncrona de ellos proporciona la funcionalidad especial de dicho sistema.

Los procesos de negocios son modelados en BPEL principalmente gracias al uso de:

- Actividades. Estas representan el tipo de interacciones que se llevan a cabo internamente en la empresa. Como ejemplo de ellas están la recepción de mensajes (<receive>), invocaciones hechas al proceso (<invoke>), respuestas dadas a los usuarios tras invocaciones realizadas (<reply>), actualizaciones de valores presentes en el proceso mediante asignaciones (<assign>), secuenciamiento de actividades (<sequence>), entre otras. En estas actividades se ven involucrados tanto el proceso central como los diferentes partners o sujetos que interaccionan con el. Estas actividades, si se observan con cuidado en un programa BPEL, modelan paso por paso lo que el proceso de negocio intenta resolver.

- Ejecutantes de las actividades. Estos vienen representados especialmente por 3 grupos. El primero esta representado por el proceso central, el segundo por los usuarios del servicio y el tercero por los servicios web. Estos dos últimos vienen descritos en el proceso por los ya nombrados PartnerLinks los cuales tienen asignados los roles que ejecutan y cuentan con los medios para comunicarse como lo son los PortTypes, de los cuales se hablará posteriormente.

Teniendo ya planteadas las dos principales partes que intervienen en la definición de procesos de negocios, se tratará de explicarlas más a fondo, desglosando los conceptos que intervienen para que se tenga una idea más exacta de su alcance. Se comenzará con la explicación de las actividades BPEL, tanto de las básicas como de las estructuradas, clasificación que puede extenderse remitiéndose a la especificación dada en [CYG03]. Para las actividades básicas, remitirse a dicho documento desde la página 53 hasta la 57 y para las actividades estructuradas revisar dicho documento desde la pagina 58 hasta la 68.

2.4.4.1. Actividades Básicas en BPEL

BPEL maneja dos tipos de actividades, las básicas y las estructuradas. Las básicas se encargan, entre otras cosas, de realizar la invocación de Servicios Web, generar respuestas a los usuarios, manipular expresiones (igualdad, desigualdad, mayor que, menor que, etc.) que modelen la lógica del proceso y de darle al proceso su naturaleza Statefull (es el caso de las asignaciones <*assign*>). Las actividades estructuradas se

encargan de definir más a fondo la estructura del proceso de negocio modelando su comportamiento por medio de la inclusión de otras actividades estructuradas.

Entre las actividades básicas están:

2.4.4.1.1. **La invocación.** Ayuda al cliente a sacar provecho de las operaciones que los Web Services proveen. Dicha Invocación esta representada por el tag *<invoke>.* La redacción para este tipo de actividades viene dada por la siguiente estructura

```
<invoke partnerLink="ncname" portType="qname" operation="ncname"
        inputVariable="ncname"? outputVariable="ncname"?
        standard-attributes>
    standard-elements
    <correlations>?
        <correlation set="ncname" initiate="yes|no"?
                pattern="in|out|out-in"/>+
    </correlations>
    <catch faultName="qname" faultVariable="ncname"?>*
        activity
    </catch>
    <catchAll>?
        activity
    </catchAll>
    <compensationHandler>?
        activity
    </compensationHandler>
</invoke>
```

La invocación es quizás una de las actividades mas frecuentes en BPEL. En ella se pueden identificar varias sub partes entre las que se tiene, principalmente, la captura de excepciones. Estas se pueden presentar cuando dicha invocación resulta en un error. Estos normalmente vienen identificados en BPEL por nombres cualificados los cuales denotan el error, seguido de quien lo produjo.

Otro aspecto muy importante de las invocaciones es que pueden tener dos naturalezas. La primera de ellas es síncrona, en este caso, la actividad requiere tanto de una variable de entrada como de salida para poder cumplir con su funcionalidad. La segunda es asíncrona. Para ello la actividad requiere solo de una variable de entrada ya que en este caso no se espera una respuesta como parte de la operación.

Entre otras características de esta actividad se tiene el uso de juegos de correlación los cuales se preocupan por el enrutamiento de banderas, características particulares de las conversaciones, de las cuales se hablará posteriormente.

2.4.4.1.2. **Recibir y Responder.** Estas actividades están representadas por los tag <*receive*> y <*reply*> respectivamente. Es por medio de estas estructuras que un proceso de negocio *recibe* información del cliente para ser procesada y generar una *respuesta*. La estructura de estas actividades es la siguiente: en el caso de receive

```
<receive partnerLink="ncname" portType="qname" operation="ncname"
            variable="ncname"? createInstance="yes|no"?
  standard-attributes>
      standard-elements
      <correlations>?
         <correlation set="ncname" initiate="yes|no"?>+
      </correlations>
</receive>
```

Y en el caso de reply

```
<reply partnerLink="ncname" portType="qname" operation="ncname"
 variable="ncname"? faultName="qname"?
 standard-attributes>
  standard-elements
  <correlations>?
    <correlation set="ncname" initiate="yes|no"?>+
  </correlations>
</reply>
```

Cuando en un proceso hay una actividad receive, es de esperarse que haya una respuesta por medio de una actividad reply, siendo esta la naturaleza síncrona de un proceso de negocio. Es decir, una actividad reply es usada para enviar una respuesta a una petición que ha sido previamente aceptada a través de una actividad receive, tal y como lo indica la especificación del lenguaje en [CYG03].

Como caso particular para la actividad receive, y como más adelante se tratará en la actividad pick, la recepción de mensajes dispara la instanciación de procesos de negocios. Como se mostró en su estructura, si el atributo createInstance, se encuentra en su estado afirmativo (*yes),* hace posible que, cada vez que llegue un mensaje deseado, se cree una instancia del proceso. Para ello, además de ser afirmativo el atributo createInstance, debe cumplirse que la actividad receive sea una actividad de inicio. Esto se cumple si y solo si las otras actividades básicas son ejecutadas antes o al mismo tiempo de que a la actividad le llegue su turno.

2.4.4.1.3. **Asignación.** Un proceso BPEL tiene la característica de ser Statefull y para que el proceso de negocio conserve esta cualidad, es necesaria la manipulación de variables que conserven su contenido durante todo su ciclo de vida. A lo largo de dicho ciclo hay varias operaciones que pueden ocasionar la pérdida de información y es algo primordial que alguna estructura BPEL trate de solucionar este percance. Las estructuras <*assign*> son las encargadas de cumplir dicha misión y su esquema se describe a continuación:

```
<assign standard-attributes>
    standard-elements
    <copy>+
        from-spec

      to-spec
    </copy>
</assign>
```

La tarea de una asignación es bastante simple, copiar y pegar el contenido de una variable a otra. De esta manera se garantiza que la información se conserve a lo largo del proceso y que de operación en operación se manipule su contenido sin llegar a perderlo. Las asignaciones son muy poderosas ya que no solo copian contenidos entre variables, también lo pueden hacer, además de otras opciones de copia, entre partnerLinks (entes que serán explicados posteriormente). Este tipo de operaciones exigen algunas condiciones pero persiguen el mismo fin, evitar la perdida de datos.

2.4.4.1.4. **Lanzamiento de fallas.** Toda falla en BPEL debe tener un único nombre global. Esto con el fin de que los manipuladores las señalicen apropiadamente. Para cumplir con este objetivo BPEL provee la estructura <*throw*> que se preocupa por la generación de estos identificadores además de proveer información más ampliada acerca de la falla para que los manipuladores puedan compensarlas fácilmente. La estructura de esta actividad básica es la siguiente:

<throw faultName="qname" faultVariable="ncname"? *standard-attributes*>
    standard-elements
</throw>

2.4.4.1.5. **Espera.** BPEL permite la definición de estructuras con el fin de permitirle a un proceso de negocio especificar un tiempo durante el cual se espere, por ejemplo, hasta la invocación de un servicio web específico. Este tipo de estructuras son las llamadas <*wait*>*.* Su estructura es como sigue:

<wait (for="duration-expr" | until="deadline-expr") *standard-attributes*>
    standard-elements
</wait>

2.4.4.1.6. **Hacer Nada.** BPEL permite la definición de estructuras con el fin de permitirle a un proceso de negocio especificar un tiempo durante el cual no se *haga nada* y se espere, por ejemplo, hasta la captura y eliminación de una excepción. Este tipo de estructuras son las llamadas <*empty*>*.* Su estructura es como sigue:

<empty *standard-attributes*>

standard-elements

</empty>

Este tipo de actividades son las utilizadas constantemente en la definición de procesos de negocios. Ellas aportan funcionalidades que son indispensables para un proyecto, manipulando a un bajo nivel la información que se mueve en el. Se utiliza la expresión de bajo nivel debido a que las actividades estructuradas, que se tratarán a continuación, utilizan las actividades básicas, delegándoles el tratamiento básico de los datos; así mismo se denomina actividades de alto nivel a las actividades estructuradas, buscando de esta forma darles un tipo de estratificación.

2.4.4.2. Actividades Estructuradas en BPEL

A continuación se tratará de explicar el funcionamiento de las actividades estructuradas proponiendo su disposición al igual que una pequeña descripción.

Tal y como lo indica la especificación de este lenguaje, dada en [CYG03], las actividades estructuradas describen cómo un proceso de negocio es creado, mediante la composición de actividades básicas, las cuales forman estructuras que expresan patrones de control, flujo de datos, manipulación de fallas y eventos externos, al igual que la coordinación en el intercambio de mensajes entre instancias de procesos. De esta manera, se procede a desglosar cada una de estas acciones o características proponiendo para cada una de ellas la actividad candidata a realizarla.

Cuando se necesite control secuencial entre actividades, se pueden utilizar estructuras como sequence, switch o while. Si se necesita sincronización y concurrencia, se puede hacer uso de la actividad estructurada flow. Finalmente, cuando se necesite que un evento externo excite la elección de un camino definido en el proceso, es recomendable hacer uso de una actividad estructurada llamada pick.

De esta manera se tiene:

2.4.4.2.1. **Secuenciamiento.** Esta actividad es la más presente en la definición de procesos debido a que dentro de ellas generalmente se enmarca todo el sistema a orquestar. La actividad Sequence se preocupa por la ejecución secuencial de las actividades que se encuentren en su interior. El orden en el cual estas actividades serán

ejecutadas viene determinado por la disposición en la cual se encuentran listadas dentro del elemento <*sequence*>.

En su interior, la actividad sequence puede contener cualquier otra actividad estructurada como por ejemplo una actividad flow, un pick o un scope, como se vera a continuación:

<sequence *standard-attributes*>
standard-elements
activity+
</sequence>

Donde activity+ representa a todas las acciones que se pueden llevar a cabo secuencialmente. Un ejemplo más explicativo se muestra a continuación.

<sequence>
<flow>
...
</flow>
<scope>
...
</scope>
<pick>
...
</pick>
</sequence>

El anterior ejemplo sugiere que primero se ejecute la clausula flow, luego el scope y por último la actividad pick.

2.4.4.2.2. **Escogencia.** En un proceso de negocio es muy frecuente encontrarse con situaciones que ameritan la ejecución de actividades siempre y cuando se cumplan ciertas condiciones. Esto es posible en BPEL gracias a las clausulas <*switch*>. La redacción de una de estas actividades es como se explica a continuación.

<switch *standard-attributes*>
standard-elements
<case condition="bool-expr">+
activity
</case>
<otherwise>?
activity
</otherwise>
</switch>

Se podría decir que esta actividad consta de dos partes. La primera expone la condición que debe cumplirse para que ciertas actividades se ejecuten. La segunda parte especifica lo que se debe hacer en caso de que esta condición no se cumpla, es decir en todos los otros casos.

2.4.4.2.3. **Bucles finitos.** Hay ciertas ocasiones en donde se necesita que algo se ejecute *mientras* alguna condición se cumpla dentro del proceso. Este comportamiento es modelado por BPEL gracias a las sentencias *<while>.* Ellas hacen posible que se lleven a cabo otras actividades simples o estructuradas siempre y cuando la condición especificada se cumpla. La redacción propuesta por la especificación para este tipo de sentencias es la siguiente:

<while condition="bool-expr" *standard-attributes*>
standard-elements
activity
</while>

Es claro que la actividad que se debe ejecutar debe estar dentro de las etiquetas <while> y </while>, mientras que la condición que se debe cumplir se encuentra al inicio de estas. Esto se puede observar en el fragmento de código representado por la palabra *activity*.

2.4.4.2.4. **Actividades concurrentes.** BPEL da la posibilidad a los usuarios de modelar la ejecución de actividades concurrentes. Cuando un proceso de negocio es

disparado, es posible que se necesite no solo la ejecución de una tarea sino de varias. Además de ello, es posible también que se necesite de alguna estructura de control arbitraria que gestione dichas actividades para saber el flujo de su ejecución, de hecho es posible que una de ellas necesite los resultados de otra. Es por ello que existen las sentencias <flow>y <link> las cuales permiten tanto la ejecución de actividades en paralelo (<flow>), como el control entre ellas (<link>). La redacción en BPEL de esta actividad es como se muestra a continuación.

<flow *standard-attributes*>
standard-elements
<links>?
<link name="ncname">+
</links>
activity+
</flow>

La cláusula <*flow*> contiene en su interior actividades estructuradas <*sequence*> las cuales representan lo que ordenadamente se debe ejecutar paralelamente. Las sentencias <*link*> a pesar de no ser indispensables para definir flujos en BPEL, representan el control que se debe tener entre las actividades que integran la sentencia <*flow*>, especificando en su definición un origen (<*source*>) y un destino (<*target*>).

El prototipo exploratorio planteado por este proyecto hace uso de la especificación BPEL la cual encierra todo lo explicado anteriormente. En el proceso de negocio escogido para la implementación no intervienen estructuras muy complejas. Simplemente se hace uso de invocaciones de servicios web externos, bucles finitos, asignaciones, ejecución condicional de ramas del proceso, entre otras, con el fin de poder simular el comportamiento real de un proceso de negocio.

2.4.4.3. Partes que interactúan con el proceso y sus características.

A continuación se tratará de explicar otro aspecto importante para modelar un proceso de negocio. Esto corresponde a las partes que interactúan con el proceso central junto con sus características.

Como ya se introdujo inicialmente, un proceso de negocio puede ser desde una simple orden de compra hasta un movimiento bancario. En estas actividades no interviene una parte sino varias, las cuales aportan constructivamente diferentes funcionalidades al proceso de negocio y la forma de incluirlas en él es haciendo uso de Partner Links.

Las empresas hoy en día, tienen que interactuar entre sí para poder resolver un objetivo de negocio y para ello sacan el máximo provecho de lo que cada una ofrece. Los servicios web apuntan a resolver este problema ofreciendo funcionalidades específicas al mundo exterior en donde una aplicación que reside en un equipo remoto puede ser ejecutada por otro que no necesariamente se encuentre cerca de él, tal y como lo introduce la Sun Microsystems cuando refiere a los Web Services como la promesa del ambiente distribuido dado en [SMI04]. Los servicios web, en otras palabras, le dan a las empresas la gran ventaja de poder utilizar funcionalidades y servicios proveídos por otros. Siendo consecuentes con esto, es de esperar que la orquestación de los mismos sea indispensable para un programa BPEL, el cual solo los puede ver como socios o partners. De esta manera, es indispensable entonces mencionar los conceptos que están relacionados con los socios del proceso de negocio central.

2.4.4.3.1. Port Types:

Es un concepto ligado al lenguaje de descripción de servicios web, WSDL [WSP05]. En el se muestran las operaciones que son soportadas y ofrecidas por el servicio web que referencia el documento WSDL. En la definición de un portType figura el nombre del servicio web seguido de las funciones que este puede ofrecer. Para comprender mejor, se puede observar en el siguiente ejemplo:

```
<portType name="schedulingPT">
    <operation name="requestProductionScheduling">
        <input message="pos:POMessage"/>
    </operation>
    <operation name="sendShipingSchedule">
        <input message="pos:scheduleMessage"/>
    </operation>
</portType>
```

2.4.4.3.2. Partner Link Types:

Un Partner Link Type caracteriza la relación conversacional que pueda existir entre dos servicios definiendo los roles que juega cada uno en la conversación. Esto se hace especificando el portType, de un documento WSDL, que debe soportar cada rol. Normalmente los PortTypes especificados por el partnerLinkType pertenecen a espacios de nombre diferentes, sin embargo, cuando se trata de interacciones asíncronas en donde se necesita de invocaciones *callback* los PortTypes pueden provenir de un mismo espacio de nombre.

La sintaxis para definir un partnerLinkType es:

```
<plnk:partnerLinkType name="ncname">
    <plnk:role name="ncname">
        <plnk:portType name="qname"/>
    </plnk:role>
    <plnk:role name="ncname">?
        <plnk:portType name="qname"/>
    </plnk:role>
</plnk:partnerLinkType>
```

Nótese que en algunos casos, la definición de los partnerLinkTypes puede enmarcar solo un rol. En este caso el servicio que soporta ese rol manifiesta su voluntad de poder trabajar con cualquier otro servicio sin ponerle algún tipo de restricción o requerimiento.

2.4.4.3.3. Partner Links:

Los servicios con los cuales un proceso de negocio interactúa son modelados como partner links en BPEL. Estos Partner links son caracterizados por los ya nombrados partnerLinkTypes los cuales pueden enmarcar a más de un socio en el proceso de negocio. Cada estructura de estas tiene su propio nombre el cual se utiliza para todas las interacciones que haya entre el proceso de negocio central y el servicio que se este consumiendo. Refiriéndose más a los PartnerLinkTypes, hay que tener en cuenta que el atributo myRole indica cual de los roles especificados por ese partnerLinkType juega el proceso central y el atributo partnerRole indica cual de los roles especificados por ese

partnerLinkType juega el Partner link. Se podría decir entonces que cada definición de un nuevo Partner dentro del proceso, deberá ir acompañada de su caracterización la cual esta definida en el partnerLinkType. Para entender un poco más lo señalado anteriormente se procede a mostrar un ejemplo de definición de un Partner Link.

```
<partnerLinks>
    <partnerLink name="ncname" partnerLinkType="qname"
            myRole="ncname"? partnerRole="ncname"?>+
    </partnerLink>
</partnerLinks>
```

En el anterior fragmento se tienen dos partes muy importantes. La primera especifica el nombre del Partner Link que se esta definiendo y a continuación cual es el partnerLinkType que lo caracteriza. La segunda parte explica qué papel, dentro de los especificados por el partnerLinkType, juega el actual Partner link.

De esta manera entonces se ha explicado a grandes rasgos lo que el prototipo exploratorio utiliza, actividades estructuradas o simples y Partner links. La idea principal es mostrar un pequeño ambiente real en el cual este tipo de lenguajes pueda ser puesto a prueba para que demuestre cuán robusto es, mostrando la automatización en todo o en parte de un proceso de negocio. En dicho sistema, los Partner links tratarán de acercarse a la realidad teniendo como característica central su sincronía con el proceso.

A continuación se hará una comparación entre los lenguajes tratados anteriormente. Esto con el fin de aclarar sus diferencias y sentar de una manera más clara los conceptos nombrados anteriormente.

CARACTERÍSTICA	BPEL4WS	BPML	WS-CDL
Modelamiento de Colaboración	Fuertemente soportado. El concepto clave de Partner link es usado para modelar los mensajes de colaboración peer-to-peer.	Soportado indirectamente.	Fuertemente soportado. El concepto clave de coreografía es usado para modelar la colaboración peer-to-peer.

CARACTERÍSTICA	BPEL4WS	BPML	WS-CDL
Modelamiento del control de ejecución.	Fuertemente soportado. Dado por el modelo de control híbrido de estructura en bloque[1] y estructura de transición[2].	Fuertemente soportado. Esto esta dado en este lenguaje gracias a las estructuras en bloque.	No soportado.
Representación del rol.	Débilmente soportado. Los atributos 'myRole' y 'partnerRole' son las únicas formas de asignar roles en los partnerLinkType.	No soportado.	Fuertemente soportado. Los Roles y los Participantes son dos mecanismos para modelar roles en este lenguaje con suficiente semántica.
Transacciones y compensación.	Soportadas indirectamente. Las transacciones son realizadas a través de los manipuladores de fallas y compensación.	Fuertemente soportadas. Las transacciones pueden ser demarcadas como Actividades Complejas. La persistencia del proceso es soportada para recuperar el estado del proceso.	Indirectamente soportadas. La compensación se confía a la realizada por el Bloque Finalizador de este lenguaje.
Manipulación de Excepciones.	Fuertemente soportado. El manipulador de fallas es usado para capturar un error. La actividad "Terminate" es usada para terminar todas las actividades que puedan excitar posteriores fallas.	Fuertemente soportado. El proceso de Excepción y fallas son los dos posibles medios para manipular excepciones en este lenguaje.	Soportado. El Bloque de Excepción es usado para capturar fallas. Múltiples errores están pre-definidos.
Seguridad y mensajería confiable.	No soportada. Se recomienda el uso de Seguridad en Servicios Web (WS-Security) para alcanzar este propósito.	No soportada.	No soportada. Se recomienda adoptar Seguridad en Servicios Web (WS-Security), Confiabilidad en Servicios Web (WS-Reliability) y mensajería confiable para alcanzar este propósito.

[1] Estructura en bloque se refiere a las estructuras del lenguaje que están en sí mismas contenidas. Cada estructura define su propia área de definición y comportamiento. La estructura puede estar anidada. Las actividades estructuradas prescriben el orden en el cual un conjunto de actividades toman lugar.

[2] Estructuras de transición se refiere a las estructuras del lenguaje que soportan la representación de un grafo dirigido. Estas prescriben el orden de las actividades en términos del estado de transición.

CARACTERÍSTICA	BPEL4WS	BPML	WS-CDL
Nivel de Abstracción.	Muy alto. Soporta la definición abstracta de PartnerLinkType. La selección dinámica de los servicios ofrecidos por los partner se hace asignando referencias a puntos finales de los diferentes socios del proceso.	Baja	Alta. Soporta la definición abstracta de channelType e informationType, los cuales pueden ser usados para este propósito.
Soporte para semántica.	No soportado.	No soportado.	No soportado.
Soporte para contexto.	Fuertemente soportado. La actividad Scope es usada para brindar esta característica a este lenguaje.	Fuertemente soportado. El contexto es usado para definir el entorno de ejecución para actividades y procesos. Hay un soporte débil para correlación de variables.	Soportado. La coreografía esta relacionada con el contexto de este lenguaje.
Manipulación de eventos.	Fuertemente soportado. El proceso y las actividades pueden ser asociados con el Manipulador de Eventos para procesar mensajes entrantes y alarmas.	Soportado. El proceso puede estar definido para responder mensajes entrantes y señales.	No soportado.

Este cuadro encierra una serie de conceptos tales como persistencia de variables, manipulación de fallas, contextos y otros importantes, los cuales pueden ser extendidos en las especificaciones de cada lenguaje dadas en [AAG01] para BPML, [BPT05] para WS-CDL y [CYG03] para BPEL. La columna del extremo izquierdo muestra las características tenidas en cuenta para realizar la comparación entre estos 3 lenguajes. Esta comparación fue obtenida de la lectura del artículo [YWL06] cuando se hacia un entendimiento de los diferentes lenguajes existentes para la composición de servicios web. Teniendo en cuenta las características arriba nombradas, cabe resaltar una de ellas: la manipulación de eventos y fallas. Esta propone a BPEL como el lenguaje propicio para la definición de procesos hospitalarios médicos, ya que da las herramientas al desarrollador para proponer la ejecución de actividades alternas al normal curso del proceso de negocio.

No es bueno casarse con una tecnología específica, pero BPEL proporciona unas buenas bases para la composición de servicios web, como lo son el manejo de contextos (actividad Scope), la manipulación de fallas, manejo del comportamiento del sistema y el modelamiento de colaboración entre socios (PartnerLinks), todo lo anterior con el fin de poder integrar varias funcionalidades y resolver un objetivo de negocio.

REFERENCIAS CAPÍTULO 2

[AAG01] A. Arkin and A. Agrawal, “Business Process Modeling Language,” *Working Draft 0.4. BPMI.org Intalino, Inc. Status: Working Draft. Draft Unpublished*. 3/8/2001

[GAC06] G. Cor, “10 Oportunidades arquitecturales para Workflow,” Paradigma Software. *Microsoft Regional Director. Uruguay, Paraguay y Bolivia*. Draf Unpublished. 2006

[YWL06] C. Yushi, L. Wah and D. Limbu, “Web Services Composition – An Overview of Standards,” *Singapore Institute of Manufacturing Technology Lee Eng Wah*, Chairman of the Information Exchange Technical Committee.

[RHL04] R. Howard and L. Kerschberg, ”A Framework for Dynamic Semantic Web Services Management,” choward@gmu.edu, kersch@gmu.edu. E-Center for E-Business, http://eceb.gmu.edu/ Department of Information and Software Engineering MSN4A4, *George Mason University, 4400 University Drive, Fairfax, VA, 22030-4444*.

[ABP03] D. Austin, A. Barbir, E. Peters and S. Ross-Talbot, “Web Services Choreography Requirements,” *1.0, W3C*, 2003

[BPT05] A. Barros, M. Dumas and P. Oaks, “A Critical Overview of the Web Services Choreography Description Language,” *BPTrends.* March 2005

[PLF01] F. Leymann, “Web Services Flow Language (WSFL) 1.0.,” *IBM*. IBM Software Group.

[CYG03] F. Curbera, Y. Goland, T. Andrews, H. Dholakia, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic and S. Weerawarana, ”Business Process Execution Language for Web Services Version 1.1.,” *Copyright© 2002, 2003 BEA*

Systems, International Business Machines Corporation, Microsoft Corporation, SAP AG, Siebel Systems. All rights reserved. 5 May 2003

[SMI04] Java 2 Platform, Micro Edition (J2METM) Web Services. A technical White Paper. July 2004. Sun Microsystems.

[WSP05] M. Feldgen and O. Clúa, "Web Services," Fiuba. *Facultad de Ingeniería. Universidad de Buenos Aires*. EGRIET.

Capítulo III

3. GRAFOS Y ALGORITMOS DE PARTICIONAMIENTO

Ya en el capítulo 2 se seleccionó BPEL como lenguaje para la definición del proceso de negocio, ahora se requiere de una forma de representación matemática que facilite la manipulación de dicha definición del proceso para su posterior distribución en dispositivos móviles. Así pues, en este capítulo se definirá esa representación matemática de los procesos de negocios para facilitar la distribución de los procesos de negocio, además se hará una revisión de los principales procedimientos de distribución y se elegirá uno para su posterior implementación en el lenguaje de programación Java.

3.1 Modelos de Representación de Workflows

A continuación se estudian algunas de las formas de representación matemática que se pueden utilizar para representar procesos de negocio.

3.1.1. Redes de Petri

Formalmente, según [MMM06], las redes de Petri son grafos bipartitos dirigidos. En los cuales se pueden identificar 2 tipos de nodos junto con unas banderas. Estos tipos de nodos pueden ser transiciones o lugares. Una transición es habilitada si y solo si todos los lugares entrantes (*input places*) están marcados con banderas y los lugares salientes (*output places*) no. Una transición es habilitada por lugares. Un Cliente Genérico *GC*, es quien marca con su bandera el lugar que habilita una transición.

Las redes de Petri proveen formas poderosas para especificar y verificar la coordinación de actividades en entornos concurrentes. En este campo, tanto las redes de Petri como las máquinas de estado finitas sirven para modelar Workflows. Lo que le da el valor agregado a las redes de Petri es que estas sirven para modelar Workflows concurrentes y se utilizan como representación gráfica de los modelos de flujos de trabajo. Su ventaja radica en la combinación de fundamentos matemáticos con una representación gráfica

comprensiva y la posibilidad de hacer simulaciones y verificaciones. Otro aspecto importante a tener en cuenta es la especificación de flujos de control y de datos que debe tener una descripción de un Workflow, lo cual se realiza en el caso de que existan varias actividades concurrentes que podrían compartir las mismas fuentes de información. En este caso, el flujo de datos representa la comunicación que hay entre dichas actividades concurrentes.

Con el propósito de aclarar más el uso de las Redes de Petri, se relacionará a [WMP06], en donde se dice que las Redes de Petri están ubicadas en la gestión de Workflows, WFMS. El fin último de la gestión de Workflows es ver que la actividad correcta sea ejecutada por la persona correcta en el momento correcto. Cabe aclarar que las redes de Petri convencionales no sirven para modelar procesos reales. Para ello se han propuesto unas extensiones las cuales son:

- Color: para modelar datos.
- Tiempo
- Jerarquía: para estructurar modelos grandes.

Una red de Petri que haya sido extendida con estas características es llamada una red de Petri de alto nivel.

Muchos de los sistemas de gestión de Workflow para soportar su función, separan el modelamiento de los procesos de Workflow (¿Cómo se hace? Process Dimension) del modelamiento de la estructura de la organización (¿Quién lo hace? Resource Dimension). En la dimensión del proceso, se especifica que tareas necesitan ser ejecutadas y en que orden. En este proceso las Redes de Petri son más claras puesto que las tareas son modeladas por las transiciones, las condiciones son modeladas por los lugares y los casos son modelados por las banderas. El término *resource* se utiliza para un *actor* que es capaz de ejecutar tareas específicas. En muchos entornos estos *resources* normalmente son seres humanos.

Para cada tarea es necesario identificar quien puede realizarla. Esta asociación entre actividades y su ejecutante se realiza por medio de *links* lo cual podría traer consecuencias negativas ya que si la persona que ejecuta una tarea no se encuentra, la

ejecución de esta actividad no sería posible. Peor aún, la definición de un proceso de negocio tendría que ser hecha de nuevo cada vez que se asocie un nuevo actor. Una forma de solucionar este problema es no asignando una tarea a un específico *resource* o *actor* sino a grupos, llamados *clases*, en los cuales estos son agrupados siguiendo sus características.

Los análisis que se le pueden hacer a la definición de un proceso Workflow son:

- Validación: probar si el Workflow se comporta como se esperaba.

- Verificación: para ver la exactitud del Workflow.

- Análisis de desempeño: evaluar la habilidad de conocer requerimientos con respecto a los tiempos de rendimiento, niveles de servicio y utilización de los actores.

La validación puede ser hecha mediante simulaciones interactivas. Se introducen casos ficticios y se prueba si este los manipula correctamente. Para análisis de verificación y desempeño se necesitan técnicas más avanzadas como análisis de cubrimiento de grafos, revisión de modelamiento y técnicas de reducción pueden ser usadas para verificar el comportamiento dinámico de una Red de Petri.

La existencia de herramientas para el análisis de Redes de Petri permiten que estas se postulen como buenas herramientas tanto para la definición de procesos de negocios como para su análisis. Contrario a estas ventajas, estas redes no proveen métodos para su verificación, y por lo tanto ejecutan Workflows sólo como agentes operacionales sin probar su eficiencia. Los Workflows que son especificados en términos de Redes de Petri pueden ser verificados gracias a técnicas de análisis del estado del arte.

Todo esto aclara el uso de las Redes de Petri y su posicionamiento en la Gestión de los flujos de trabajo Workflow. Debido a esto, esta forma de representación no es utilizada por BDMobIS ya que estas representaciones son usadas solo con el fin de realizar el particionamiento de procesos de negocios para lo cual se ha comprobado la eficiencia de los Grafos.

3.1.2. Gramática de Grafos Atribuidos (AGG)

Según [MRG99], la gramática de grafos atribuidos es un lenguaje visual basado en reglas el cual soporta un acercamiento algebraico para la transformación de grafos. Esta puede ser usada como un motor de transformación de grafos de propósitos generales para aplicaciones Java de alto nivel, empleando métodos de transformación de grafos. Debido a su carácter basado en reglas, la gramática de Grafos Atribuidos puede ser usada también en la inteligencia artificial. Las características de esta gramática son:

a. Estructuras de datos complejas son modeladas como grafos los cuales pueden ser categorizados por *grafos atribuidos.*

b. El comportamiento del sistema es especificado por reglas de grafos usando una descripción del tipo: si - entonces.

c. La aplicación de una regla transforma la estructura del grafo.

d. La aplicación secuencial de varias reglas muestra el escenario de una aplicación.

e. Los grafos AGG pueden ser vistos como objetos Java y por tipos de datos en Java.

f. Por otro lado, nuevas clases Java pueden ser incluidas.

g. Las reglas de grafos pueden ser atribuidas o pueden ser ayudadas por expresiones Java las cuales son evaluadas durante la aplicación de las reglas.

h. Adicional a esto, las reglas pueden ser condiciones vistas como expresiones Java booleanas.

a. Por otra parte, los programas AGG consisten principalmente de dos partes: Una gramática de grafos atribuida por objetos Java.

b. Los objetos java pueden provenir de clases definidas por el usuario, ese conjunto de clases corresponde a la segunda parte importante en AGG.

Para tal fin las librerías de clases Java como el JDK están disponibles, pero no son consideradas como parte de un programa AGG. La gramática de grafos contiene un grafo inicial y un conjunto de reglas que podrían tener condiciones negativas de aplicación. La forma en como las reglas sean aplicadas materializa directamente el acercamiento hacia la transformación de grafos. En cuanto a la atribución de los nodos y las aristas, esta se hace, como ya se dijo anteriormente, mediante objetos Java y mediante expresiones. La principal diferencia es que en AGG esto se hace mediante objetos Java y mediante expresiones en vez de utilizar especificaciones y términos algebraicos.

El sistema AGG soporta dos nociones diferentes de grafos las cuales están estrechamente relacionadas. La primera tiene que ver con los grafos simples. El segundo acercamiento es hacia los grafos jerárquicos. Un grafo simple consta de dos conjuntos. Uno de nodos y otro de aristas. Estos dos conjuntos están vistos como los objetos del grafo. Cada arco representa una conexión dirigida entre dos nodos, los cuales son llamados fuente y destino de la arista. Con el fin de realizar una clasificación más a fondo de cualquier objeto, a cada uno se le puede asociar una etiqueta la cual se puede escoger de entre un conjunto de etiquetas. Las etiquetas son también llamadas tipos. Un objeto "o" puede tener entonces una etiqueta "l" para ello se dice que "o es de tipo l". Nótese que dentro de la noción de grafos simples se puede tener múltiples aristas del mismo tipo. Los nodos de un grafo son bosquejados como rectángulos y las aristas son flechas dirigidas desde sus nodos origen a su nodos destino. A continuación se observa un ejemplo de grafo simple.

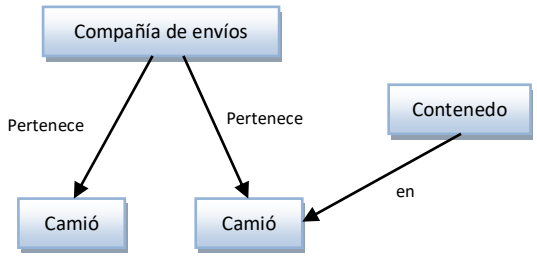

Figura 3.1. Una compañía de envíos modelada como un grafo simple. Imagen tomada de [MRG99].

Como se puede apreciar, cada contenedor tiene su peso y contenido. Estos atributos, sin duda, son información valiosa también pero por simplicidad esta no es incluida en el grafo. Para ello es que se utiliza precisamente la atribución. Esto no se puede hacer

mediante nodos y aristas. Esto se puede hacer mediante enteros y cadenas de caracteres. Un grafo atribuido puede ser fácilmente entendido como Clases con sus atributos. Como se verá a continuación:

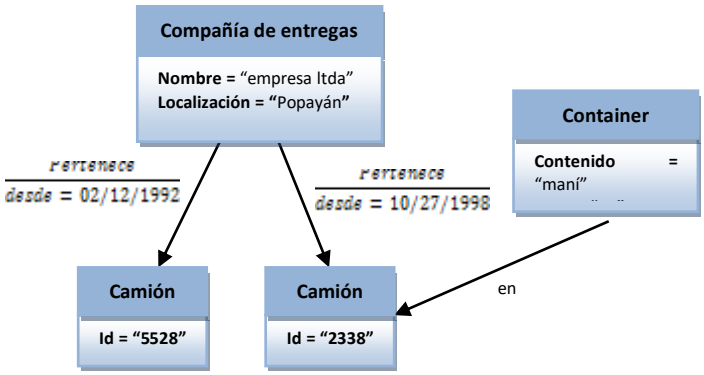

Figura 3.2. Representación atribuida de la compañía de envíos mostrada en la Fig. 3.1 Imagen tomada de [MRG99].

Esto es ya un grafo atribuido lo que se acerca bastante al concepto de orientación a objetos. La pregunta que cabe ahora es, ¿qué tipo de datos pueden usarse para atribuir? La respuesta es cualquier tipo de dato válido para Java.

Como ya se ha visto, se han estado describiendo sistemas que en cierto punto son estáticos. Se tratará entonces de describir situaciones, acciones incluso procesos de negocio que extienden la aplicación de la gramática de grafos atribuidos AGG. Una acción puede ser vista como una transición de estados y por supuesto una transición de estados puede ser especificada dando descripciones de los estados antes y después de la acción es cuestión. Esto apoyado en el uso de dos grafos los cuales describen estas dos situaciones. En el estado de antes lo que se hace es recolectar todas las precondiciones que tienen que haber.

Para el ejemplo de "cargarle un contenedor a un camión" tienen que haber los siguientes requerimientos:

a. Tiene que haber un camión.

b. Tiene que haber una bodega con por lo menos un contenedor adentro.

c. El camión a ser cargado tiene que estar en frente de la bodega.

Una vez se tienen estas precondiciones, hacer el grafo para el estado de "después" es como sigue. En vez de tener una arista que indica una relación de "in" entre el contenedor y la bodega, se tiene que tener una de "on" que relacione el contenedor y el camión. Nótese que esta operación es abstracta ya que no se indica específicamente qué contenedor o qué camión o qué bodega intervienen en esta operación.

Por tanto se ha visto que el lado izquierdo de una regla de grafos establece todas las condiciones necesarias para que la operación especificada pueda llevarse a cabo. Una regla puede aplicarse si todos los requerimientos se cumplen. Obviamente esto corresponde a una clausula de "if-then" sin una rama "else" hablando en términos de un lenguaje de programación. La revisión del cumplimiento de las condiciones finaliza o se concluye tratando de encontrar un grafo que se iguale al grafo que dicta las precondiciones o sea al grafo de la mano izquierda.

3.1.3. Grafos

Los grafos constituyen una herramienta matemática para modelar sistemas. En [HLE95] Hendrickson y Leland afirman que: “Para muchas aplicaciones en el cálculo científico, el problema de descomponer un cálculo entre varios procesadores se puede describir convenientemente en el lenguaje de los grafos. Un vértice en el grafo representa un cálculo, mientras que una arista entre dos vértices indica dependencia de datos”, esto lleva a la conclusión de que los grafos son la herramienta matemática idónea para representar Workflows y posteriormente particionarlos para su distribución.

Dentro de las matemáticas y la ciencia de la computación el estudio de los grafos es realizado por la teoría de grafos, la cual identifica grafos no dirigidos y grafos dirigidos o dígrafos. En [CLR90] se encuentran definiciones importantes acerca de los grafos desde el punto de vista de la ingeniería. A continuación, se hace un breve resumen de algunas de esas definiciones importantes alrededor de los grafos.

Un digrafo *G* es un par $(V$, donde *V* es un conjunto finito y A es una relación binaria sobre *V*. El conjunto *V* se denomina conjunto de vértices de *G*, y sus elementos son denominados vértices. El conjunto *A* se denomina conjunto de aristas de *G* y sus elementos son pares ordenados $(u$, donde $u, v \in V \; y \; u$.

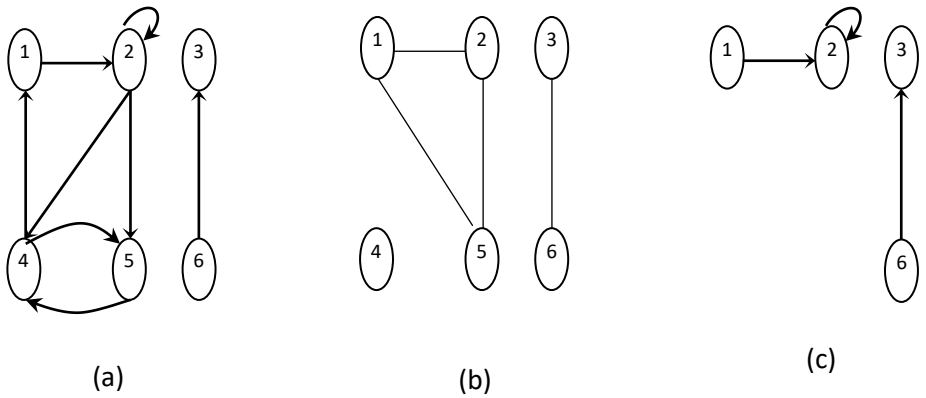

Figura 3.3. Grafos: (a) Grafo dirigido con auto ciclos, (b) Grafo no dirigido, (c) Un subgrafo del grafo presentado en (a). Imagen tomada de [CLR90].

En la figura 3.3 (a) se puede observar un grafo dirigido con el conjunto de vértices $\{1,2,3,4,$. Los vértices son representados por círculos y las aristas por flechas. Nótese que también existe la posibilidad de autociclos desde un vértice hacia el mismo.

En un grafo no dirigido $G = (V,$ el conjunto de aristas *A* consiste de pares no ordenados de vértices en lugar de pares ordenados. Es decir una arista se denota por $\{u,$ donde $u, v \in V \; y \; u$. La figura 3.3 (b) es la representación gráfica de un grafo no dirigido cuyo conjunto de vértices es $\{1,2,3,4,$.

A continuación se presentan algunas propiedades de los grafos:

Si $(u$ es una arista en un grafo dirigido $G = (V,$ se dice que $(u$ incide desde o sale de un vértice "*u"* y es *incidente* a o entra al vértice "*v"*. Por ejemplo, en la figura 3.3(a) las aristas salientes del vértice 2 son (2,2), (2,4) y (2,5). Las aristas entrantes al vértice 2 son (1,2) y (2,2).

Si $(u$ es una arista en un grafo $G = (V$ se dice que el vértice *"v"* es *adyacente* al vértice *"u".* Cuando un grafo es no dirigido, la relación de adyacencia es simétrica. Para el caso de un grafo dirigido esa relación de adyacencia no necesariamente es simétrica, así pues, si *"v"* es adyacente a *"u"* algunas veces se escribe $u \cdot$.

El *grado* de un vértice en un grafo no dirigido es el número de aristas incidentes en él. Por ejemplo el vértice 2 en la figura 3.3 b) tiene grado 2. En un grafo dirigido el grado de

salida de un vértice es el número de aristas que salen de él y el grado de entrada de un vértice es el número de aristas que entran a él. El grado de un vértice en un grafo dirigido es igual al grado de entrada más el grafo de salida. El vértice 2 en la figura 3.3 (a) tiene un grado de entrada igual a 2, un grado de salida igual a 3 y por lo tanto, su grado es igual a 5.

Se dice que un grafo $G' = (V'$, es un *subgrafo* de $G = (V$ si V' y A'. Dado un conjunto V', el subgrafo de *G* inducido por *V'* es el grafo $G' = (V'$, donde $A' = \{(u, v) \in E: u, v \in$. El subgrafo inducido por el conjunto de vértices $\{1,2,$ en la figura 3.3 (a) aparece en la figura 3.3 (c) y tiene un conjunto de aristas $\{(1,2), (2,2), (6.$

En BDMobIS se tienen actividades BPEL que pueden verse como vértices en un grafo, donde las aristas entre vértices determinan el flujo de control entre ellas [BPA06]. Así pues, una vez se tiene el Workflow con sus respectivas actividades (vértices) y comunicaciones (aristas) se debe distribuir en varios sub-Workflows (subgrafos) para lo cual existe un procedimiento dentro de la teoría de grafos denominado "Particionamiento de grafos" del cual se hablará en la siguiente sección.

3.2. Algoritmos de particionamiento de grafos

El particionamiento de grafos consiste en distribuir los vértices del grafo en diferentes subconjuntos o subgrafos de tamaños específicos de tal manera que el costo de comunicación (números de aristas) entre los subgrafos sea minimizado [HLE95]. Una definición formal de algoritmo de particionamiento se puede encontrar en [KLI70] donde se define de la siguiente manera:

Sean:

- *G* un grafo de *"n"* vértices, de pesos $w_i > 0, i = 1, ..$
- *"p"* un número positivo, tal que $0 < w_i$ para todo .
- $C = (c_{ij}), \text{donde } i, j = 1, .$ una matriz de conectividad ponderada (es decir con pesos) descrita por las aristas de *G.*
- *k* un número entero positivo.

Una partición $v_1, ..$ de *G* es un conjunto no vacío de subconjuntos que forman pares disyuntos de *G,* tal que $\bigcup_{i=1}^{k} v_i$:. Una partición es admisible si:

$$|v_i| \leq p \text{ para todo } i$$

Donde el símbolo indica el tamaño (peso) de un conjunto *"v"* que es igual a la suma de los tamaños de todos los elementos de *"v"*. El costo de una partición es la suma de sobre todos los , tal que estén en diferentes subconjuntos. El costo total es entonces la suma de todos los costos externos en la partición. Por lo tanto el problema de particionamiento consiste en encontrar una partición admisible de *G* y con un costo mínimo.

Este proceso se conoce también como "k corte", el cual, en otras palabras, consiste en realizar una partición de un grafo en k partes balanceadas tal que el número de *aristas de corte* (aristas que interconectan un subgrafo con otro) sea minimizado [CSA06]. Este problema es difícil de resolver pues se ubica dentro de la clase de problemas denominados NP-completos [HER00].

Los problemas NP- Completos son los más complejos dentro de los problemas de decisión y se encuentran enmarcados dentro de la teoría de la complejidad de la que se hablará a continuación.

Cuando aparece un problema existen varios algoritmos que se pueden aplicar para resolverlo. Se dice que un problema tiene el mismo orden de complejidad que el del algoritmo que se conozca para resolverlo. Si, por ejemplo, para resolver un problema existe un algoritmo, con entradas de tamaño *"n"*, que en su peor caso se ejecuta en un tiempo $O($ para alguna constante *"k"* dicho algoritmo se conoce como algoritmo de tiempo polinomial.

Así pues la teoría de la complejidad clasifica a los problemas de acuerdo al tiempo que tome un algoritmo en resolverlos. Es natural preguntarse si todos los problemas pueden

ser resueltos en tiempo polinomial y la respuesta es no. Por ejemplo, existen problemas muy renombrados como el problema de Turing conocido también como "Problema de la parada"[3] (halting problem) que no puede ser resuelto por ningún computador, sin importar cuanto tiempo le dedique, estos problemas se conocen como problemas *irresolubles o impredecibles,* a este tipo de problemas no se les puede asignar una respuesta ni afirmativa ni negativa, es decir no hay algoritmo alguno que permita determinar si tienen solución. Existen, además, problemas que se pueden resolver pero no en un tiempo $O($ para alguna constante k. Generalmente, se piensa que los problemas que se pueden resolver con algoritmos de tiempo polinomial son *tratables* y los problemas que se resuelven en tiempo superpolinomial son *intratables*.

Se cree que los problemas NP-completos son intratables, la razón es que si un solo problema NP-completo se puede resolver en tiempo polinomial, entonces todos los problemas NP-completos tienen un algoritmo de tiempo polinomial. Dado que hasta la fecha se ha estudiado un amplio rango de problemas NP-completo sin ningún progreso hacia una solución en tiempo polinomial, sería ciertamente asombroso si todos ellos pudiesen ser resueltos en ese tiempo polinomial.

Un buen diseñador de algoritmos debe entender las nociones básicas de la teoría de problemas NP-completo. Si se establece que un problema es NP-completo se debe proveer una buena evidencia de su intratabilidad. Por eso como ingenieros es mejor desarrollar un algoritmo de aproximación en lugar de buscar un algoritmo rápido que soluciones el problema de manera exacta [CLR90].

Para más información acerca de la teoría de la complejidad por favor remitirse al Anexo A del presente documento.

Como ya se dijo y teniendo en cuenta que el particionamiento de grafos es un problema de tipo NP completo es de esperarse que para alcanzar una solución óptima el tiempo de cálculo requerido por una máquina crezca de manera exponencial con el número de vértices que tenga el grafo. Por lo tanto, la solución óptima es muy difícil de alcanzar y una búsqueda exhaustiva resultaría intratable.

[3] Busca determinar si una máquina es capaz de reconocer si un programa entrará o no en un bucle infinito.

Por esta razón, muchos avances matemáticos buscan soluciones muy cercanas a la óptima mediante algoritmos basados en métodos heurísticos, geométricos y evolutivos [CSA06].

A continuación se hace una revisión de los principales esfuerzos por encontrar una solución para el problema de particionamiento de grafos.

3.2.1. Soluciones Exactas

Se puede encontrar una solución óptima a un problema de particionamiento de grafos mediante soluciones exactas, las cuales son estrictamente procedimientos exhaustivos. Para mirar esto se toma la representación matemática realizada por Kernighan y Lin en [KLI70]: Sea *G* un grafo con *"n"* vértices de tamaño 1 el cual se va a particionar en *"k" subconjuntos* de cardinalidad *"p"*, donde kp. Entonces hay formas de elegir el primer subconjunto, $\binom{n-}{p}$ de elegir el segundo y así sucesivamente, puesto que el orden de los subconjuntos es irrelevante, entonces el número de casos sería:

$$\frac{1}{k!}\binom{n}{p}\binom{n-p}{n}\dots\binom{2p}{n}\binom{p}{n}$$

Como se puede observar para muchos valores de *"n"*, "*k"* y "*p"* esta expresión arroja un número muy grande, por ejemplo para un grafo con 30 vértices y que se quiera dividir en 4 subconjuntos de cardinalidad 10 cada uno, se tiene $n =$, $p =$ y k : el número de formas de elegir los subconjuntos sería mayor que 10^{20}. Normalmente este tipo de soluciones exactas se solucionan a través de la Programación Lineal Entera (ILP) [BBC05], el cual tendría un gran número de restricciones (ecuaciones) para poder expresar el problema de particionamiento de grafos y cuya solución óptima requerirá una exagerada carga de computación.

Es por esta razón que se ha investigado en métodos heurísticos que permitan producir buenas soluciones (iguales o muy cercanas a la óptima) de manera relativamente rápida.

A continuación se realiza un breve recorrido por muchos de estos acercamientos heurísticos en la búsqueda de soluciones buenas y con una baja carga en tiempo de cómputo.

3.2.2. Soluciones Aleatorias

Estos métodos buscan generar soluciones aleatorias, es decir, ubicando de manera aleatoria vértices entre los diferentes sub-grafos, y almacenando cada vez la mejor solución hasta que se cumpla determinado tiempo o valor. Este método es rápido pero no es satisfactorio para problemas con gran número de vértices, puesto que las soluciones cercanas a la óptima aparecen con muy baja probabilidad. La probabilidad de éxito sobre cualquier prueba es menor a 10^{-7} [KLI70].

3.2.3. Algoritmo de Kernighan y Lin

Este algoritmo que fue publicado por W. Kernighan y S. Lin [KLI70] en septiembre de 1969, se clasifica dentro de los algoritmos de migración de grupos que consisten de métodos deterministas e iterativos. Es una técnica de optimización que asigna a las divisiones del grafo una *función de beneficio Q* e intenta optimizarla. Dicha función de beneficio consiste del número de aristas que hay dentro de dos grupos menos el número que hay entre ellos [MLO05]. Para esto, se realizan movimientos de los vértices entre los grupos en búsqueda de la minimización de las aristas de corte [HEL95].

Actualmente, el algoritmo de Kernighan-Lin se utiliza de manera frecuente como algoritmo de refinamiento para otras técnicas de particionamiento más avanzadas como el caso de los algoritmos multinivel, de los que se hablará posteriormente. El algoritmo de Kernighan-Lin produce muy buenos resultados siempre y cuando reciba muy buenas particiones iniciales, es decir, antes de realizar los movimientos de vértices de un grupo a otro [GRI07].

En la actualidad hay muchos algoritmos que implementan mejoras a la heurística Kernighan Lin [GRE03] [HEL00] [BSM05].

3.2.4. Simulated Annealing

La heurística Simulated Annealing fue introducida por Kirkpatrick [KGV83] y es una de las herramientas más exitosas para resolver problemas de optimización combinatoria [KDI03].

Los algoritmos basados en la heurística Simulated Annealing buscan encontrar una buena aproximación al óptimo global de una función en un espacio de búsqueda suficientemente grande. Su nombre viene del proceso de templado en metalurgia, en el cual se pierden los átomos en un metal cuando se calienta y luego se enfría lentamente [CRO05].

Algunos algoritmos de particionamiento de grafos de la actualidad utilizan Simulated Annealing para resolver problemas como dividir los vértices de una red en varios subdominio balanceados minimizando los caminos de interconexión entre diferentes subdominios [GBM06], o particionar sistemas Software o Hardware [BDU04] como en el caso de posicionamiento VLSI (Very Large Scale of Integration) [SMA91].

3.2.5. Algoritmos espectrales

Los algoritmos basados en técnicas espectrales [ALO04] se basan en los valores propios y vectores propios de las matrices de adyacencia o de la matriz Laplaciana. La matriz de adyacencia de un grafo $G = (V$ es la matriz $A = (a_{u,v})_u$ en la cual $a_{u,v}$ si uv , de lo contrario $a_{u,v}$. La matriz Laplaciana de *G* es $Q = D$, donde $D = (d_{u,v})_u$ es la matriz diagonal en la cual (es el grado d de en G y $d_{u,v}$ para todo u . Las dos matrices anteriores son simétricas y por lo tanto, tienen valores propios reales y una base ortonormal de vectores propios. El hecho de que todos los valores propios y vectores propios se puedan computar eficientemente permite su utilidad en el diseño de algoritmos eficientes [ALO98].

Un método espectral de particionamiento hace biseccciones recursivas sobre un grafo considerando el vector propio de la matriz asociada para ganar un entendimiento global de las propiedades del grafo. Ofrece un buen balanceo, calidad y eficiencia [HLE93], aunque se obtienen buenos resultados se requiere un alto costo computacional [PIC06].

En la actualidad existen muchos algoritmos de particionamiento de grafos basados en técnicas espectrales como los presentados en [AKA94], [HLE92] y [HLE93].

3.2.6. Algoritmos multinivel

Los métodos multinivel fueron inicialmente introducidos por Bui y Jones [BJO93] y posteriormente mejorados por Hendrickson - Leland [HLE95] y por Karypis – Kumar [KKU95]. Estos métodos son rápidos y efectivos pero son más recomendables en el caso de tener grandes grafos (con millones de vértices) pues reducen su tamaño a pocos miles o cientos de vértices a través del proceso de engrosamiento (coarsening) del que se habla más abajo y entregan soluciones cercanas a la óptima. Para grafos pequeños es preferible el uso de otras técnicas, por ejemplo, técnicas exactas como ILP [BBC05] ya que la respuesta encontrada es la óptima aunque el costo de cálculo se incrementa con el número de vértices del grafo. Los algoritmos multinivel constan típicamente de cuatro fases: engrosamiento, Partición inicial, des engrosamiento y refinamiento [BBC05].

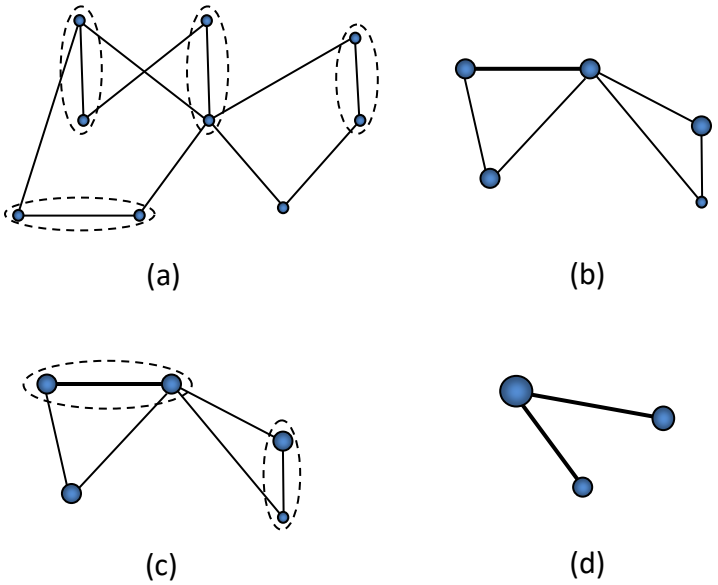

Figura 3.4: Proceso de engrosamiento: (a) Grafo original y fusión de vértices, (b) grafo después del proceso de engrosamiento, (c) fusión de vértices del grafo ya engrosado, (d) grafo después de dos procesos de engrosamiento. Imagen tomada de [GUP96]

En la primera fase se realiza un engrosamiento mediante repetidas fusiones de vértices (proceso conocido como *matching*), con esta etapa se busca reducir el tamaño del grafo preservando sus propiedades esenciales, ver figura 3.4.

En la segunda fase se hace una partición inicial sobre el grafo ya engrosado, se entiende que particionar el grafo engrosado es similar a particionar el grafo original. El método de partición utilizado es indiferente y su elección realmente no es muy importante, por ejemplo en [HLE95] sugieren un método espectral. En la tercera fase el proceso de des engrosamiento busca deshacer el proceso de engrosamiento, desuniendo los vértices a diferentes niveles. Finalmente la cuarta fase de refinamiento podría requerir del uso recursivo de algoritmos para lograr un mejoramiento local de cada partición, para esto se puede utilizar algoritmos como Kernighan-Lin [KLI70].

3.2.7. Algoritmos paralelos

Los algoritmos paralelos se basan en la idea de dividir un problema dado en sub-problemas que se puedan resolver de manera concurrente sobre diferentes procesadores de un computador paralelo [RAN00], es decir, que se busca distribuir datos y trabajo entre diferentes procesadores de tal manera que se reduzca el costo de comunicación y se alcance la máxima ejecución posible [DBH06].

En los procesos de distribución de trabajo y balanceo de carga se utilizan algoritmos de particionamiento de grafos muy eficientes como por ejemplo los basados en técnicas multinivel, pero para grandes grafos generalmente se exceden los límites de memoria de una sola CPU, en este caso se debe utilizar algoritmos de particionamiento paralelos [DRA99]. Actualmente hay varios esfuerzos en esta área algunos de ellos se encuentran en los estudios [KKU95], [DBH06] y [KZA04].

3.2.8. Algoritmos Genéticos

Los algoritmos genéticos fueron inicialmente investigados por Jhon Holland [HOL92] quien propuso una simulación de la evolución biológica, en la cual la población experimenta adaptación, bajo estrategias de evolución controlada [BCP99].

En la selección natural, de una población sobreviven los individuos mejor dotados y sus características se heredarán en posteriores generaciones. De la misma manera, en los algoritmos genéticos las soluciones de un problema específico “compiten” para ver cual constituye la mejor solución. El ambiente constituido por las otras soluciones, ejercerá presión selectiva sobre la población, de manera que solo las mejores soluciones (las que

mejor resuelvan el problema) sobrevivan y leguen su “material genético” a las siguientes generaciones [MER97].

Para el problema de particionamiento de grafos se han comparado los algoritmos genéticos con otras heurísticas como Simulated Annealing. Como resultado de esta comparación se obtiene que el tiempo de ejecución de los algoritmos genéticos es menor y que aún así sus resultados son similares o mejores que los obtenidos por otros métodos [BCP99] [KKN03] [BMO96].

En la actualidad existen algunas investigaciones alrededor del tema de particionamiento de grafos utilizando algoritmos genéticos como por ejemplo, los publicados en los estudios de: [LAS91], [MMM94] y [CHP06].

3.2.9. Algoritmos basados en colonias de hormigas

Los algoritmos de colonias de hormigas también conocidos como optimización de colonia de hormigas (ACO por sus siglas en inglés) pertenecen al igual que el anterior tipo a los algoritmos bioinspirados que como su nombre lo indica están inspirados en algún comportamiento de los seres vivos.

Los algoritmos de colonias de hormigas fueron propuestos inicialmente por Marco Dorigo [CDM91] [DOR92] [DST04], cuya inspiración proviene de la investigación sobre el comportamiento de hormigas reales en el proceso natural que utilizan para conseguir alimento. Una hormiga puede encontrar el camino entre su hormiguero y la fuente de comida sin necesidad de indicaciones visuales. Al caminar las hormigas depositan una substancia denominada feromona y gracias a un fenómeno que se produce por la presencia de muchas hormigas que cada vez se mueven depositado feromonas se forma una trayectoria química que posteriormente será reforzada y seguida, con alta probabilidad, por otras hormigas [HJS03]. Ese mecanismo de búsqueda de cada hormiga se basa en la información almacenada en la memoria colectiva del sistema grabada químicamente en el soporte físico del suelo. A esta comunicación a través de feromonas se le denomina *stigmetry*[4] [KRZ05] y consiste en la sinergia que se realiza entre los miembros de la colonia a través del ambiente en que se mueven [MER97].

[4] Corresponde a un tipo de comunicación indirecta a través del entorno, en este caso representada por las feromonas.

Los algoritmos de hormigas han demostrado ser más eficaces que métodos clásicos de optimización combinatoria incluyendo Simulated Annealing y algoritmos genéticos [COC99]. Para el caso de particionamiento de grafos se asigna un color a cada sub-grafo, es decir que a cada vértice se le asigna un color que represente el subdominio o sub-grafo al cual pertenece. Así entonces y teniendo en cuenta que el algoritmo de hormigas es un algoritmo de asignación, basado en la idea de búsqueda paralela, se distribuye un cierto número de hormigas a través de los vértices de un grafo para colorearlos (moverlos de un sub-grafo a otro) mediante un criterio de optimización local [COC99] relacionado con la minimización de las aristas de corte.

En la actualidad este tipo de algoritmos gozan de gran reputación y son utilizados en la solución de muchos problemas como por ejemplo enrutamiento de redes con QoS [SZE00], distribución de actividades industriales de área diferente [HJS03], coloreo de grafos [SHZ01], enrutamiento de vehículos [BMC04], asignación de frecuencias en redes inalámbricas [COZ95] y como en el caso del presente proyecto como algoritmo de particionamiento de grafos [CSA06].

3.3. Selección del algoritmo de particionamiento

Como se describió anteriormente el presente proyecto requiere de un algoritmo de particionamiento de grafos que permita realizar la distribución de los vértices, correspondientes a actividades BPEL de la definición del proceso de negocio, entre los diferentes dispositivos asociados al sistema. A continuación se hará un repaso a través de los algoritmos de particionamiento de grafos que se investigaron antes de llegar a una solución aceptable.

Inicialmente se pensó en utilizar una técnica de particionamiento de grafos que arrojara una respuesta óptima sin importar el tiempo que tardase la ejecución, dado que los grafos que se manejan en el proyecto de grado tienen muy pocos vértices (10 - 30 vértices aproximadamente). Para esto se tomó como referencia el trabajo realizado en [BBC05] en el cual se afirma que un sistema software se puede describir convenientemente en el lenguaje de los grafos, específicamente a través de un *call graph.* Un call graph es un grafo dirigido donde los vértices representan programas, clases o unidades de programas

y los arcos llamadas a un programa y se representan como *arc(a, b)*, que indica que un programa b llama a un programa a. A continuación formulan el problema de dos maneras diferentes, primero como un problema de particionamiento de grafos y segundo como un problema de programación lineal entera (ILP por sus siglas en inglés) con valores en {0,1}. En este caso es interesante plantear el problema de particionar grafos como un problema ILP. Un problema ILP es un problema de optimización con una función objetivo y restricciones lineales donde las variables deben tomar valores enteros. La dificultad encontrada para implementar un algoritmo basado en ILP es que la cantidad de restricciones del problema ILP crece de acuerdo al número de vértices del grafo.

Posteriormente se analizó el algoritmo de particionamiento basado en técnicas multinivel presentado en [HLE95], pero se encontró que este procedimiento es más eficiente para el caso de hipergrafos (es decir grafos con miles de vértices), lo que implicaba sobredimensionar el problema ya que los grafos que se maneja en este proyecto son en su mayoría pequeños.

Finalmente se seleccionó un algoritmo de particionamiento de grafos basado en algoritmos de optimización de colonias de hormigas, los cuales son muy populares especialmente en los últimos años dado que entregan respuestas muy cercanas a la óptima en tiempos relativamente bajos y está demostrado que son algoritmos más eficaces que otros algoritmos de optimización combinatoria [COC99]. Para la implementación de BDMobIS se hizo algunas modificaciones al algoritmo presentado por F. Comellas en [CSA06] el cual, a diferencia de otros algoritmos de optimización de colonia de hormigas no utiliza el concepto de feromonas y por esta razón facilita su implementación. En el capítulo 5 se describirá con detalles este algoritmo con las respectivas adaptaciones para el caso particular del presente proyecto de grado.

Hasta este punto ya se ha considerado a BPEL como lenguaje de definición de proceso de negocio y a los grafos como un lenguaje matemático conveniente para representarlo y posteriormente distribuirlo. Además ya se definió como algoritmo de particionamiento el descrito en [CSA06] con algunas adaptaciones para los propósitos de este proyecto.

REFERENCIAS CAPÍTULO 3

[MMM06] M. Merz, K. Moldt and W. Müller, “Workflow Modeling and Execution with Coloured Petri Nets in COSM,” *University of Hamburg. Hamburg, Germany*. Departament of Computer Science. Vogt-Kölln-Str. 30 - D-22527.

[WMP06] W. van der Aalst, “The application of Petri Nets to Workflow Management,” Department of Mathematics and Computing Science, Eindhoven *University of Technology. Eindhoven, Holanda* 2006.

[MRG99] M. Rudolf and G. Taentzer, “Introduction to the Language Concepts of AGG,” *TU Berlin* November 10, 1999.

[HLE95] B. Hendrickson and R. Leland, “A multilevel algorithm for partitioning graphs*,”* en *Proceeding of Supercomputing '95*, ACM, Diciembre 1995.

[CLR90] T. Cormen, C. Leiserson and R. Rivest, “Introduction to Algorithms,” *MIT Press*. ISBN 0-262-03141-8. 1990.

[BPA06] F. Bellas and C. Pan, *“*Orquestación de servicios Web*,”* Análisis y diseño orientado a objetos (ADOO). *Universidade da Coruña*. En Internet: h*ttp://www.tic.udc.es/~fbellas/teaching/adoo/Tema6Apartado6.2.pdf* 2006.

[KLI70] B. W. Kernighan and S. Lin, “An efficient heuristic procedure for partitioning graphs,” *Bell Systems Tech*. J., 49 (1970), pp. 291-308.

[CSA06] F. Comellas and E. Sapena, “A multiagent algorithm for graph partitioning,” Lecture Notes in Comput. Sci. vol. 3907, pp. 279--285 ISSN: 0302-9743. 2006.

[HER00] G. Hernández, “Complejidad y Grafos,” II Seminario de matemática discreta y codificación de la Informática. *Universidad Politécnica de Madrid* Abril. 2000

[BBC05] R. H. Bisseling, J. Byrka, S. Cerav-Erbas, N. Gvozdenovic, M. Lorenz, R. Pendavingh, C. Reeves, M. Röger and Arie Verhoeven, “Partitioning a Call Graph,*”* Study Group Mathematics with Industry, *Amsterdam* 2005.

[MLO05] M. Martínez and L. López, “CONAN: A generic Complex Networks Analysis library,” *Universidad Rey Juan Carlos*. *España*. 2005.

[HEL95] B. Hendrickson and R. Leland, “The Chaco User's Guide,” Version 2.0, Technical Report SAND 94-2692, *Sandia National Laboratories, Allbuquerque*, NM, 1995.

[GRI07] S. Griffin, “Two-Way Partitioning with Stable Performance,” *ECE*. Enero 2007.

[GRE03] W. Greene, “A Kernighan-Lin Local Improvement Heuristic that Softens Several Hard Problems in Genetic Algorithms,” CEC’03. The 2003 Congress on Evolutionari Computation. ISBN: 0-7803-7804-0. 2003.

[HEL00] K. Helsgaun, “An effective implementation of the lin-kernighan traveling salesman heuristic*,” European Journal of Operational Research*, 126:106--130, 2000.

[BSM05] R. Brglez, X. Li, M. Stallmann and B. Militzer, “Evolutionary and Alternative Algorithms: Reliable Cost Predictions for Finding Optimal Solutions to the LABS Problem,” Information Sciences**,** in press. 2005.

[KGV83] S. Kirkpatrick, CD. Gellat, M.P. Vecchi, “Optimization by simulated annealing,” Science, 220:671-680, 1983.

[KDI03] K. Dowsland and A. Díaz, “Diseño de Heurísticas y Fundamentos del Recocido Simulado,” Inteligencia Artificial, *Revista Iberoamericana de Inteligencia Artificial*. No 19. 2003.

[CRO05] C. Roger, “Simulated Annealing,” MathWorld--*A Wolfram*. En internet: http://mathworld.wolfram.com/SimulatedAnnealing.html. Febrero de 2005.

[GBM06] C. Gil, R Baños, M. Montoya and J. Gómez, “Performance of Simulated Annealing, Tabu Search, and Evolutionary Algorthms for Multi-objective Network Partitioning,” Algorithmic Operations Research Vol.1 (2006) 55–64

[BDU04] S. Banerjee and N. Dutt, “Very Fast Simulated Annealing for HW-SW partitioning,” *University of California*. Irvine, USA. Junio 2004.

[SMA91] K. Shahookar and P. Mazumder, “VLSI cell placement techniques,” *ACM Computing Surveys (CSUR)* Volume 23, Issue 2 P. 143 – 220. ISSN: 0360-0300. June 1991.

[ALO04] N. Alon, “Spectral Techniques in Graph Algorithms,” LATIN’98. Theoretical Informatics. P.206. *Springer. Berlin*. ISBN: 3-540-64275-7. Febrero 2004.

[ALO98] N. Alon, “Spectral Techniques in Graph Algorithms, Proceedings of the Third Latin American Symposium on Theoretical Informatics”, p.206-215, April 20-24, 1998.

[HLE93] B. Hendrickson and R. Leland, “Multidimensional spectral load balancing,” Tech. Rep. SAND93-0074, *Sandia National Laboratories, Albuquerque*, NM, Enero 1993.

[PIC06] J. Pichel, ”Técnicas de optimización de la localidad para códigos irregulares sobre arquitecturas multiprocesador y multithreading,” *Santiago de Compostela*, Junio 2006.

[AKA94] Alon and N. Kahale, “A spectral technique for coloring random 3-colorable graphs,” Proc. of the 26th ACM STOC, ACM Press (1994), 346-355. Also; SIAM J. Comput., in press.

[HLE92] B. Hendrickson and R. Leland, “An improved graph partitioning algorithm for mapping parallel computations,” *Sandia National Laboratories, Albuquerque*, NM 87185, 1992.

[BJO93] T. Bui and C. Jones, "A Heuristic for Reducing Fill in Sparse Matrix Factorization," Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, 1993, pp. 445-452.

[KKU95] G. Karypis and V. Kumar, “Parallel multilevel graph partitioning,” Technical Report TR 95-036, Department of Computer Science, *University of Minnesota,* 1995.

[GUP96] A. Gupta, “Fast and effective algorithms for graph partitioning and sparse-matrix ordering,” IBM Journal of Research and Development, 41 (1996), pp. 171-183.

[RAN00] R. Rangaswami, “Parallel Algorithm Design,” Topics in Computer Science. *The University of Edinburgh*. En Internet: http://www.dl.ac.uk/TCSC/Staff/Hu_Y_F/PROJECT/pdcp_siam/node19.html Marzo 2000.

[DBH06] K. Devine, E. Boman, R. Heaphy, R. Bisseling and U. Catalyurek, **“**Parallel Hypergraph Partitioning for Scientific Computing,” *SIAM Parallel Processing for Scientific Computing*, February 2006.

[DRA99] N. Drakos and R. Moore, ”Load Balancing for Unstructured Mesh Application,” Marzo 1999.

[KZA04] Z. Kasheff and Zardosht, “Partial Parallelization of Graph Partitioning Algorithm METIS,” *Massachusetts Institute of Technology*. Dept. of Electrical Engineering and Computer Science. 2004.

[HOL92] J. Holland, “Adaptation in Natural & Artificial Systems,” MIT press, 1975 (2a Ed. 1992).

[BCP99] Z. Baruch, O. Creţ and K. Pusztai, ”Genetic Algorithm for Circuit Partitioning,” *Analele Universităţii din Oradea, Fascicola Electrotehnică*, p. 19-23, 27–29 mai, Băile Felix, *ROMÂNIA*, 1999

[MER97] J. Merelo, ”Informatica evolutiva. Algoritmos genéticos,” Escuela Técnica Superior de Ingeniería Informática. *Granada España*: En internet: http://geneura.ugr.es/~jmerelo/ie/ags.htm Mayo 1997

[KKN03] K. Kohmoto, K.Katayama and H.Narihisa, “Performance of a Genetic Algorithm for the Graph Partitioning Problem,'' Mathematical and Computer Modelling, vol. 38, no. 11--13, pp. 1325--1332, 2003.

[BMO96] T.Bui and B.Moon, “Genetic algorithm and graph partitioning,” IEEE Transactions on Computers, 45 (1996), pp. 841-855

[LAS91] G. Laszewski, "Intelligent structural operators for the k-way graph partitioning problem,'' Proc. of the 4th ICGA, 1991, pp. 45-52.

[MMM94] H. Maini, K. Mehrotra, C. Mohan, and S. Ranka, "Genetic Algorithms for Graph Partitioning and Incremental Graph Partitioning,'' Proc. Supercomputing '94. Nov. 1994.

[CHP06] C. Chevalier and F. Pellegrini, “Improvement of the Efficiency of Genetic Algorithms for Scalable Parallel Graph Partitioning in a Multi-level Framework,” Euro-Par 2006: 243-252

[CDM91] A. Colorni, M. Dorigo, and V. Maniezzo, “Positive Feedback as a Search Strategy,” Tech. Rept. 91-16 *Politecnico di Milano - Department of Electronics*, Milano - Italy, November, 1991.

[DOR92] M. Dorigo, “Optimization, Learning and Natural Algorithms,” PhD thesis, Politecnico di Milano, Italy. 1992.

[DST04] M. Dorigo and T. Stützle, “Ant Colony Optimization,” MIT Press. ISBN 0-262-04219-3. 2004.

[HJS03] A. Hospitaler, P. Jaén, C. Santamarina and J. Montalvá, “Algoritmo Híbrido basado en colonias de hormigas para la solución del problema de la distribución en planta en el marco del S.L.P”. 2003.

[KRZ05] W. Krzysztof, “Ant algorithm for flow assignment in connection-oriented networks,” International Journal of Applied Mathematics and Computer Science, No. 2, Vol. 15, 2005, pp. 205-220

[COC99] F. Comellas, J. Ozón, A .Cortés, J. Abril and M. Vaquer, “Sistemas multiagente para la asignación de frecuencias en redes celulares,” IX Jornadas de I+D en Telecomunicaciones, *UPC, Barcelona*. ISBN: 84-7653-730-1. 17-18 Noviembre 1999.

[SZE00] Z. Subing and L.Zemin, “A QoS routing algorithm based on ant algorithm,” En Proceedings of the 25th Annual IEEE Conference on Local Computer Networks. LCN. 2000. ISBN: 0- 7695-0912-6

[SHZ01] J. Shawe and J.Zerovnik, “Ants and graph coloring,” in *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms,* (2001) *ICANNGA'01*, pp. 276-279. Springer-Verlag.

[BMC04] J. E. Bell and P. R. McMullen, “Ant colony optimization techniques for the vehicle routing problem,” *Advanced Engineering Information*, vol. 18, pp. 41–48, 2004

[COZ95] F. Comellas and J. Ozón, “Graph coloring algorithms for assignment problems in radio networks,” *Applications of Neural Networks to Telecommunications 2.* J. Alspector, R. Goodman and T.X. Brown (Eds.), Lawrence Erlbaum Ass., Inc., Publis., Hillsdale, NJ (1995), pp. 49-56; ISBN 0-8058-2084-1.

Capitulo IV
4. TRABAJOS RELACIONADOS

Distribuir un proceso en un grupo de tareas aliadas que se ayudan para cumplir un mismo objetivo permite dividir los esfuerzos entre los diferentes participantes de tal manera que se aprovechan mejor los recursos que cada uno de estos ofrece. El presente proyecto se orienta justamente a realizar la distribución de procesos de negocios entre diferentes dispositivos móviles buscando explotar las ventajas de la movilidad que estos ofrecen.

Actualmente existen muchos trabajos de investigación relacionados con la distribución de tareas entre diferentes máquinas, pero la investigación sobre distribución en dispositivos móviles es escasa. A continuación se pretende describir algunos de los trabajos más relevantes para este proyecto de grado en especial los relacionados con sistemas móviles de información.

Una de las investigaciones, quizá más significativas, alrededor de la ejecución descentralizada de procesos de negocios en sistemas móviles de información, fue descrita en [BMM04] y [MAM05] por Baresi et.al, en ella proponen un método de dos fases para controlar la distribución de un proceso de negocio mediante el particionamiento del Workflow relacionado en sub-Workflows, cada uno de los cuales tiene asignado un controlador diferente. Dicho en otras palabras, pretenden dividir la ejecución de flujos de trabajo BPEL en pequeños flujos BPEL a ejecutarse en dispositivos móviles. Para lograr esto proponen reglas formales de particionamiento cuyo objetivo es transformar un único Workflow en juegos de Workflows aliados.

Esto constituye una nueva metodología, para descentralizar el control de flujo de un proceso de negocio, que provee las reglas necesarias para descomponer cada actividad

estructurada o básica y construir una vista local de cada actor que facilite la ejecución de Workflows autónomos mientras se mantienen las propiedades del Workflow original.

El particionamiento de Workflows hace posible la transformación de una orquestación centralizada a una orquestación descentralizada de Workflows aliados que pueden ser ejecutados por diferentes motores. Para esto, en [BMM04] los autores inicialmente plantean una trasformación de la descripción XML de BPEL a un grafo a través de diagramas de actividad UML. Un proceso BPEL se define en términos de sus interacciones con sus partners, así pues representan cada actividad básica de UML como un vértice, los enlaces entre actividades con aristas de tipo follow y las actividades estructuradas con dos vértices de propósito especial denominados *Activity*.

Para implementar las reglas de particionamiento utilizan AGG como herramienta de soporte y validación de las reglas de particionamiento. (Para más información acerca de AGG por favor remitirse al capítulo 3).

Una vez definido el proceso utilizando BPEL y UML se introducen tareas de sincronización entre los diferentes controladores, las cuales permiten que la ejecución de todos los Workflows locales logre el mismo resultado que el Workflow original conservando correcto el flujo de ejecución del proceso de negocio. Finalmente, se crean para cada actor una vista de procesos locales que representa la vista del proceso completo para el punto de vista de un actor involucrado en el control de la ejecución del proceso. Así, entonces la vista local de un proceso para un actor A incluye solamente la parte del proceso que tiene que ser controlada por A, de esta manera se remueven todas las actividades cuya ejecución no sea controlada por A.

Para asegurar la correcta conclusión del Workflow en un entorno distribuido este trabajo define reglas de particionamiento, las cuales tienen un comportamiento funcional, son confluentes y terminales. Estas reglas descomponen cada actividad BPEL para construir una vista local de actores y se pueden aplicar en tiempo de diseño y en tiempo de ejecución ya que soportan las modificaciones dinámicas de los Workflows de acuerdo al contexto. A continuación se hace una breve descripción de cada una de ellas:

- Cada vez que una actividad controlada por un actor A_1 sea seguida por otra actividad controlada por un actor A_2, se introducen nuevas tareas para el control del proceso de delegación. Las nuevas tareas de sincronización insertadas deben proveer las capacidades para entregar el control de ejecución de un actor a otro, además del paso de variables y mensajes, ya que deben mantener correcto el flujo de ejecución.

- Es necesario establecer un controlador para cada actividad estructurada (con excepción de *sequence*), de tal manera, que cada una tenga un *StartController* y un *EndController*. En el caso de *While* los controladores deben ser ejecutados por el mismo actor. Lo que se busca con esto es distribuir cada tarea estructurada de manera completa a un actor en específico, así pues el inicio y fin de una tarea estructurada debe ser ejecutada por un solo actor.

- El vértice *Start* de las actividades estructuradas, Pick, Switch y While, debe estar a cargo de la evaluación de la condición de dichos bucles. El vértice start debe controlar hacia que rama del Workflow debe seguir la ejecución.

- La ejecución del flujo de control de un conjunto específico de tareas se puede asignar solamente a un dispositivo. Esto en el caso de tareas que estén fuertemente ligadas, como en el caso de tareas dentro de un *sequence.*

- En un entorno distribuido no son posibles las variables globales, por lo tanto, en los procesos de negocio, no deben existir y todas las variables se deben pasar como parámetros entre los diferentes actores, es decir, que las variables globales se transforman en parámetros de mensajes.

El uso de variables globales dentro de los procesos BPEL ofrece grandes ventajas, pero no es posible en un entorno distribuido donde los trabajadores actúan de manera independiente. Para resolver este problema se tiene en cuenta que BPEL permite el uso de mensajes para la comunicación entre diferentes actores y así transformar variables globales en parte de los mensajes intercambiados. El comportamiento de cada actor será modelado de acuerdo a esta especificación.

Otro aporte importante de Baresi et.al en [BMM04] y [MAM05] son las reglas de construcción de una vista local para un proceso**.** La vista local de un proceso para un actor A incluye la parte del proceso que tiene que ser controlada por A. La construcción de estas vistas locales es la segunda y última fase del método y se obtiene gracias a las siguientes reglas:

- Todas las actividades cuya ejecución no esté controlada por A se deben remover.

- Todas las actividades estructuradas, con la excepción de *sequence* (que no incluya actividades) deben removerse.

- Dentro de un *Pick* o un *Switch*, todas las ramas que no incluyan actividades se deben remover.

En resumen los aportes de [BMM04] y [MAM05] para el presente proyecto de grado son muy grandes, por una parte ofrecen la idea principal de distribuir procesos de negocio en sistemas móviles de información y por otra plantean tareas de sincronización para lograr este objetivo. El presente proyecto de grado a diferencia de estas investigaciones, profundiza en mejorar la distribución reduciendo al mínimo el número de comunicaciones entre los diferentes actores del sistema utilizando heurísticas muy novedosas de particionamiento de grafos bioinspiradas como el algoritmo multiagente de hormigas presentado por [CSA06] .

Otro trabajo importante se puede encontrar en [MSV04], el cual presenta una técnica de distribución de servicios Web compuestos, descritos por un programa BPEL, en un conjunto equivalente de procesos descentralizados. Esto lo realizan mediante la acción de un algoritmo de particionamiento de grafos buscando minimizar los costos de comunicación y maximizar el rendimiento de múltiples instancias concurrentes de programas de entrada. Además aportan un modelo de estimación de costo para apreciar el rendimiento de una partición de código dada.

Las actividades *receive, reply* e *invoke* del archivo BPEL se colocan como vértices fijos de un grafo, las demás actividades se destinan como vértices portables. Para esto utilizan un grafo tipo PDG (Program Dependence Graph) que consiste de aristas de dependencia

de control y dependencia de datos superpuestas en el mismo conjunto de vértices. El algoritmo de particionamiento que utilizan realiza fusiones de vértices de acuerdo a las aristas de dependencia del PDG, para minimizar el costo de comunicación, e insertan aristas extra para permitir la distribución.

Dicha inserción extra de aristas en el PDG se realiza en los siguientes casos:

- Dependencias con sentencias fuera de una actividad *flow*

- Dependencias entre secciones paralelas, que son las que capturan órdenes basadas en sincronización entre dichas secciones.

Además hay que tener en cuenta que una partición no es válida cuando se genera un ciclo de dependencia.

Otro aporte de esa investigación es el modelo de costo donde toman R como el número de peticiones enviadas a un servidor por unidad de tiempo y calculan el rendimiento global del servicio como una función *T(R)* que representa el número promedio de peticiones procesadas por unidad de tiempo. Típicamente *T(R)* crece con incrementos de R hasta alcanzar un punto de estado estático cuando uno o más recursos sean utilizados completamente, eventualmente se alcanza una fase de falla cuando se acumula trabajo y el rendimiento del sistema declina dramáticamente.

En el sistema propuesto en el artículo [MSV04], el modelo de ejecución descentralizado es un sistema que consiste de un conjunto de vértices servidor comunicados $S= \{S_1,..., S_k\}$, cada uno de los cuales implementa una porción del servicio global como lo dictan las tareas particionadas. El rendimiento desarrollado por cada vértice servidor individual contribuye a subir el límite del rendimiento global así:

$$T(S) \leq min(T(S_1), \dots, T(S_k))$$

El costo de un *send* o un *receive* es típicamente menor que el de un *invoke*. El costo de cada tarea depende de varios factores incluyendo el tamaño de datos que se manipulan y

la complejidad de los mismos. El costo se calcula mediante una técnica denominada micro-benchmarking, aunque solo lo hacen para cuatro actividades *receive*, *reply*, *assign* e *invoke* y de acuerdo a la siguiente fórmula.

$$peak\ rate(tarea) = \frac{capacidad\ de\ la\ tarea}{costeos\ de\ las\ actividades\ involucradas}$$

Dónde:

$$T(S_i) \leq \frac{capacidad}{costo}$$

$$T(S_i) \leq \frac{Número\ de\ hilos}{Camino\ crítico}$$

El camino crítico es el tiempo que ocupa un hilo a través de un camino crítico.

$$T(S_i) \leq min\left(\frac{capacidad}{costo}, \frac{Número\ de\ hilos}{camino\ crítico}\right)$$

Para finalizar se concluye que los aportes más importantes de este artículo para el presente proyecto de grado son el modelo de merging para particionar grafos y las técnicas de evaluación de costos de tareas básicas (micro bench-marking). A diferencia de BDMobIS, en dicho trabajo no utilizan la idea de reconfiguración que ofrece al sistema la capacidad de adaptarse a los cambios a los que está expuesto un sistema móvil, no aplican sincronización para mantener el flujo de ejecución, no utilizan algoritmo de hormigas ni trabaja directamente con móviles aunque lo proponen como campo de acción.

En [HHC06] Hackman et.al. presentan un motor BPEL que soporta una gran variedad de dispositivos, desde teléfonos móviles hasta computadores de escritorio, denominado SLIVER, el cual desarrollaron teniendo en cuenta las restricciones que tienen los dispositivos móviles para ofrecer un tiempo de ejecución razonable. Sliver provee una clara separación entre las comunicaciones y el procesamiento. Para observar más de cerca este pequeño motor BPEL a continuación se describe su arquitectura.

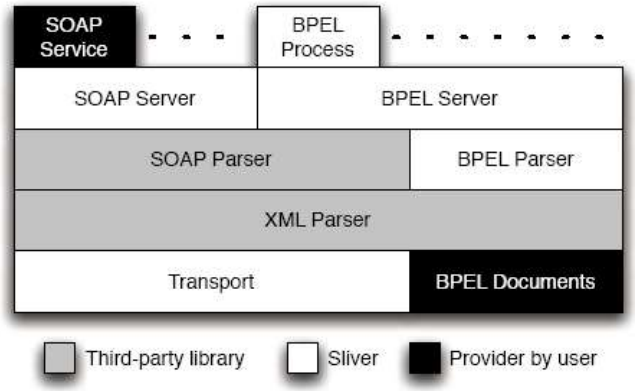

Figura 4.1: Arquitectura de Sliver. Tomada de [HHC06]

En el nivel más bajo de la arquitectura (ver fig. 4.1) está la capa de transporte la cual involucra varios protocolos y medios de red con una interfaz consistente. La capa de transporte intercambia mensajes de objetos en forma de cadenas XML serializadas, esas cadenas se convierten hacia y desde objetos Java por medio de las capas parser de XML y SOAP.

La capa SOAP server involucra servicios Java proveídos por el usuario con una interfaz de servicios Web. Cuando llegan los mensajes sin serialización desde las capas XML y SOAP, el servidor SOAP los direcciona al servicio correspondiente. La respuesta del servicio se serializa a través de las capas XML y SOAP y luego se envía sobre la red a través de la capa de SOAP.

Muchas de esas capas se reutilizan en el servidor BPEL de Sliver. La capa parser XML en conjunto con la capa parser BPEL, convierten documentos BPEL a procesos concretos ejecutables. La capa servidor BPEL mantiene los procesos que esas capas producen. Tal como el servidor SOAP, el servidor BPEL recibe las peticiones que llegan desde la capa de transporte y las enruta al proceso apropiado.

La capa de Transporte es la responsable de la transmisión de los mensajes producidos en las capas superiores. Cuando los protocolos tradicionales como http no sean factibles, Sliver puede utilizar cualquier protocolo disponible como SMTP o SMS.

Todos los mensajes intercambiados sobre la capa de transporte se codifican en forma XML (Parser XML/SOAP). Para esto se utiliza kXML, el cual está pensado para móviles,

es decir, que ocupa un tamaño muy pequeño de tan solo 11KB. kXML implementa un Pull Parser (XPP) el cual parte el XML en flujos de tokens que representan cada tag.

Sliver utiliza kSOAP para codificar y decodificar los mensajes. kSOAP se sitúa en la parte superior del API XPP de kXML y convierte objetos Java desde y hacia documentos XML codificados como SOAP. kSOAP tiene un tamaño cercano a los 47KB cuando se combina con kXML.

Ahora se explicarán algunas partes constitutivas de Sliver.

El protocolo SOAP provee un estándar para llamadas a procedimientos remotos (RPC). La clase SOAPServer de Sliver implementa la porción RPC del protocolo SOAP.

Desafortunadamente, el API MIDP carece de las características necesarias para dirigir peticiones a los métodos correctos automáticamente. Sobre esos dispositivos, cada manipulador de servicios debe implementar un método *invoke* además de sus servicios. Cuando llega una petición, Sliver pasará el nombre del servicio y los parámetros al correspondiente método *invoke* de la clase; el método *invoke* debe dirigir la petición al método apropiado.

Los Workflows BPEL se tratan como servicios SOAP altamente especializados; se invocan utilizando exactamente los mismos protocolos que otros servicios. Sliver representa cada tag en la especificación BPEL con una clase Java correspondiente; por ejemplo, el tag <assign> se manipula a través de la clase *Assign*. Cada clase tiene un constructor el cual participa en un objeto XmlPullParser proveído por la librería kXML.

El Servidor BPEL está representado por la clase BPELServer, la cual, almacena los procesos BPEL que el parser BPEL genera. En el estado actual Sliver soporta un grupo de características del núcleo de BPEL y tiene un tamaño total de 114KB incluyendo todas las dependencias (Excluyendo la librería opcional HTTP).

Para finalizar, se puede afirmar que Sliver es un buen complemento para el presente proyecto de grado, dado que ofrece la facilidad de tener un motor BPEL en cada

dispositivo agregado al sistema y las ventajas de su pequeño tamaño y bajo consumo de memoria.

En [STS04] y [WBV03] WebV2 inc presenta una solución denominada WebV2 la cual permite la participación asíncrona de usuarios humanos y aplicaciones en los procesos de negocio. Es muy liviana y basada en java por lo tanto se puede ejecutar en plataformas heterogéneas incluyendo dispositivos móviles. Esto permite la fácil movilización e inclusión de dichos dispositivos en la empresa y sus procesos de negocio.

WebV2 posee un producto denominado ProcessMobilizer el cual está construido bajo componentes livianos de Java denominados ProcessCouplers que se instalan sobre un dispositivo que puede ser móvil. Cada ProcessCoupler posee un motor BPEL interno que coordina la ejecución de un proceso de negocio de larga duración con otros ProcessCouplers en la red. Cada ProcessCoupler representa un participante específico (sistema o usuario) en un proceso de negocio BPEL. Los ProcessCouplers se pueden instalar donde sea, embebidos en un dispositivo móvil, envolviendo una aplicación legada o ejecutándose dentro de un servidor de aplicaciones. La solución WebV2 promete gran productividad para el trabajador móvil y ejecución mejorada de proceso empresariales siendo capaz de invocar más inteligentemente usuarios móviles en la ejecución de procesos empresariales.

Este aporte al igual que SLIVER se presenta como otra opción para ejecutar procesos de negocio en dispositivos móviles y además muestra la descentralización de actividades y la ejecución de procesos de negocio no solamente con servicios Web sino que también incluye actores humanos.

En [MHS00] . Meng et. al. presentan la arquitectura e implementación de un sistema simple de Workflow ad-hoc basado en agentes móviles. Los Workflow ad-hoc son actividades y tareas internas de las organizaciones que no tienen que ver con procesos ni procedimientos ya establecidos. Los autores argumentan que los Workflow ad-hoc y los agentes móviles pueden permitir la creación de procesos de negocios electrónicos (e-business) sobre servicios Web poco acoplados. Gracias a los agentes móviles los servicios Web pueden integrarse y diseñarse como servicios cooperativos distribuidos capaces de comunicarse con motores Workflow remotos, además los agentes móviles

permiten la comunicación asíncrona en este entorno. Un agente móvil es un proceso que puede migrar de un computador a otro, clonarse, fusionarse y coordinar sus cálculos. Los agentes móviles son autónomos en el sentido de que controlan su comportamiento de traslado en búsqueda de la meta con la cual están atareados. Así entonces el motor de reglas de negocio y el lado del servicio, juntos con el agente móvil Workflow, constituyen el motor Workflow para un Workflow ad-hoc.

En resumen, la propuesta busca el uso de agentes móviles como estrategia para descentralizar la ejecución de actividades Workflow, sin embargo, no utilizan BPEL, como en el presente proyecto de grado, el cual está completamente relacionado con Servicios Web.

En [BPP00] Balasooriya et. al. describen el diseño, arquitectura y ejecución de BondFlow, un sistema que permite la configuración y ejecución de Workflows utilizando enlaces Web sobre objetos Web heterogéneos. Éste sistema permite desarrollar y desplegar aplicaciones colaborativas y Workflows generando automáticamente objetos de un *proxy wrapper* liviano que habilita servicios Web encapsulados para interconectarlos a través de enlaces Web. Los *wrappers* son muy pequeños (menores a 10KB) lo que los hace aptos para residir sobre dispositivos portátiles con soporte de Java. Para esto utilizan el concepto de enlaces de coordinación Web, los cuales permiten a las aplicaciones crear datos y flujos de control entre entidades, forzando interdependencias y restricciones y llevando a cabo transacciones atómicas sobre un grupo de entidades o procesos Web.

Como se puede observar esta propuesta constituye otro enfoque para la descentralización de actividades en dispositivos móviles, donde se utilizan objetos de un proxy wrapper para permitir interconectar servicios Web, a diferencia de BDMobIS no utilizan un motor de orquestación de procesos de negocio para orquestar los servicios Web simplemente se basan en interconectar los servicios Web a través de enlaces Web.

En [WMT03] se habla acerca de la comunicación y la sincronización entre Workflows de diferentes organizaciones, atacando el problema de reorganizar objetos de bases de datos soportados en diferentes motores Workflow. Describen el diseño del modelo así como la arquitectura para proveer soporte de transacciones avanzadas de Workflows

distribuidos orientados al comercio electrónico interempresas. Hacen, además, estudios de modelos de seguridad para dichos Workflows.

Para esto se requiere realizar una administración de operaciones de Workflow en un entorno distribuido, por lo tanto, proponen las siguientes funciones:

- Un mecanismo para controlar la ejecución de Workflows diseñados sobre otros sistemas de administración de Workflows.

- Un mecanismo para monitorear el estado de los Workflows ejecutándose en otros sistemas de administración de Workflow.

Proponen además un conjunto mínimo de información que se requiere para comunicación y sincronización entre tareas en Workflows autónomos cooperativos:

- Un protocolo para establecer la localización e invocación de métodos de tareas.

- Un protocolo para coordinar la recepción de respuestas.

En resumen, este artículo aporta el diseño de un modelo de comunicación interworkflows y es importante para BDMobIS, puesto que éste último, divide el proceso de negocio en subprocesos, es decir, se tienen pequeños Workflows intercomunicados.

En [MUR04] Murthy describe un paradigma Workflow basado en objetos para soportar transacciones distribuidas en un entorno de comercio electrónico móvil. Divide las transacciones en dos tipos, internas y externas, además, describe un protocolo denominado *intention - action* para llevar a cabo tales transacciones y especificar como implementarlas por un diseño contractual utilizando una herramienta llamada *iContract* de Java y UML.

Una transacción desde un host móvil se denomina transacción móvil, la cual, es una transacción distribuida que se puede ejecutar en parte desde un host móvil como una transacción interna y en parte en otros host fijos, como transacción externa. Cada host fijo tiene un coordinador que recibe operaciones de transacciones externas desde hosts

móviles y monitorea su ejecución en los servidores de bases de datos dentro de los hosts fijos. Similarmente cada host móvil tiene un coordinador. Las transacciones convencionales necesitan satisfacer las propiedades ACID[5], las cuales son: atomicidad, consistencia, aislamiento y durabilidad.

El aporte de este artículo para BDMobIS, es similar al del artículo anterior, sólo que en este caso se basa en transacciones entre los diferentes dispositivos y la búsqueda de satisfacción de propiedades ACID de las transacciones.

En [VEV06] Baousis et. al realizan la integración de servicios Web con agentes móviles. Explotan las capacidades que tienen los dispositivos móviles para consultar e invocar Servicios Web semánticamente enriquecidos, sin la necesidad de que el servicio solicitante esté presente simultáneamente en línea. Tales servicios son ideales para la computación móvil donde los terminales de usuario no necesariamente tienen que estar conectados durante su sesión completa. Estudian además las ventajas del framework de la Web semántica para mejorar las capacidades de los registros de servicios Web reunidos en una arquitectura de referencia de servicios Web.

Este artículo brinda la idea de ofrecer una calidad de servicio en los servicios Web con agentes móviles al hacer uso de la Web semántica. Esto se podría implementar en BDMobIS mediante la orquestación de servicios Web semánticamente enriquecidos y el uso de agentes móviles.

En [TSA05] Umedu et. al. proponen la descomposición de programas que exceden los límites de recursos de los dispositivos móviles. Los programas se descomponen en dos grupos de módulos uno de esos grupos se asigna a un Terminal móvil y el resto se ejecutan en un servidor Proxy. El Terminal móvil invoca los módulos del Proxy a través de RMI. Se busca reducir la carga total de comunicaciones, el tiempo de retardo y el consumo de potencia entre los móviles y el servidor Proxy. Para esto utilizan una técnica de la evolución de la ejecución. En esta técnica insertan trozos de código que almacenan datos estadísticos de la ejecución para así dividir aplicaciones dadas mediante simulación, es decir, ejecutando las clases junto al código insertado.

[5] Acrónimo de las cuatro propiedades garantizadas por las transacciones: atomicity, consistency, isolation, and durability

Para evaluar la optimización tienen en cuenta, la cantidad de comunicaciones y el retardo de comunicación y para evaluar el consumo de potencia se asume que: el retardo de comunicación se aproxima con el número de invocaciones al método, el consumo de potencia está en proporción a la cantidad de códigos en los dispositivos portátiles y los módulos que consumen mucho más tiempo se deben ejecutar del lado del servidor. Además hacen unas estimaciones de costo, retardo y tiempo de comunicación gastado por cada módulo.

Los aportes de este artículo para BDMobIS son los mecanismos de medición de ejecución de programas Java en dispositivos móviles. Esto puede ser muy útil al evaluar el consumo de memoria y de potencia que se introduce en los dispositivos móviles al ejecutar los archivos BPEL de los subprocesos.

REFERENCIAS CAPÍTULO 4

[BMM04] L. Baresi, A. Maurino and S. Modafferi, “Workflow Partitioning in Mobile Information Systems,” Politecnico di Milano Dipartimento di Elettronica e Informazione. P. zza L. Da Vinci 32 – 20133 – *Milano Italy*, Draft Unpublished. pp. 3 – 12. 2004.

[MAM05] A. Maurino and S. Modafferi, “Partitioning rules for orchestrating mobile Information systems,” Politecnico di Milano Dipartimento di Elettronica e Informazione. Piazza L. Da Vinci 32 – 20133 – Milano Italy. 2005. Pers Ubiquit Comput (20005) 9: 291 – 300. *DOI 10.1007/s00779-004-0333-4*. Received: 30 July 2004/ Accepted: 10 November 2004/Published online: 17 August 2005. Springer-Verlag London Limited. 2005

[CSA06] F. Comellas and E. Sapena, “A multiagent algorithm for graph partitioning,” Lecture Notes in Comput. Sci. vol. 3907, pp. 279--285 (2006). ISSN: 0302-9743.

[MSV04] M. Gowri, S. Chandra and V. Sarkar, “Decentralizing Execution of Composite Web Services,” IBM India Research Laboratory and IBM T.J. Watson Research Center. Caterories and Subject Descriptors D.3.4 [Programming Languages]: Processors – optimization. OOPSLA ’04, Oct. 24-28, 2004, *Vancouver, British Columbia, Canada*. Copyright 2004 ACM 1-58113-831-8/04/0010

[HHC06] G. Hackmann, M. Haitjema, G. Christopher and R. Gruia – Catalin, “Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices,” *WUCSE – 2006 – 37.* Department of Computer Science & Engineering. University in St. Louis School of Engineering and Applied Science, *in Press*, pp. 5 – 14. 2006

[STS04] S. Stephansen, “Service-Oriented Mobile Business Process Execution,” BPTrends. October., 2004. www.bptrends.com Copyright (c), WebV2 Inc. All Rights Reserved 2004.

[WBV03] Web V2. “Dynamic and Mobile Federated Business Process Execution”. A WebV2 Whitepaper. WebV2, Inc. 510 Logue Ave. Mountain View, CA 94043. www.webv2.com sales@webv2.com. Copyright (c) 2002, 2003 WebV2, Inc. All rights reserved.

[MHS00] J. Meng, S. Helal and S. Su, “An Ad-Hoc Workflow System Architecture Based on Mobile Agents and Rule-Based Processing,” Computer and Information Science and Engineering. *University of Florida, Gainesville*, FL 32611. {jmeng,helal,sul}@cise.ofl.edu

[BPP00] J. Balasooriya, M. Padhye, S. Prasad and S. Navathe, “BondFlow: A System for Distributed Coordination of Workflows over Web Services,” *Georgia State University, Atlanta. Georgia Institute of Technology, Atlanta*. sprasad@gsu.edu, sham@cc.gatech.edu

[WMT03] V. Wietrzyk and M. Takizawa, “Distributed Workflows: A Framework for Electronic Commerce,” School of Computing & Information Technology University of *Western Sydney*. Penrith South DC NSW 1797, Australia. Department of Computers and Systems Engineering Tokyo Denki University Saitama, 3500394, *Japan*. Journal of Information Science an Engineering 19, 15-38 (2003).

[MUR04] V.K. Murthy, “Contract-Based Workflow Paradigm For Mobile E-Commerce,” School of IT & EE, UNSW@ADFA *University of New South Wales Canberra* ACT 2600 *Australia*. murthy@cs.adfa.edu.au. DASD ’04 156555-277-6.

[VEV06] V. Baousis, E. Zavitsanos, V. Spiliopoulos, S. Hadjiefthymiades, L. Merakos and G. Veronis, “Wireless Web Services using Mobile Agents and Ontologies,” University of Athens, Department of Informatics & Telecommunications, Communication Networks Laboratory, *Panepistimioupolis, Ilisia, 157 84 Athens, Greece*. E-mail: {bbaous,std00025,std00108,shadj,merakos,std00024}@di.uoa.gr

[TSA05] T. Umedu, S. Urata, A. Nakata and T. Higashino, “Automatic Decomposition of Java Program for Implementation on Mobile Terminals,” Graduate School of Info. Sci. and Tech., Osaka Univ., Japan. Yamadaoka, Suita, Osaka 565-0871, *Japan*. {umedu,s-urata,nakata,higashino}@ist.osaka-u.ac.jp Proceedings of the 19th International

Conference on Advanced Information Networking and Applications (AINA ‘05) 1550-0445X/05 © 2005 IEEE.

Capítulo V
5. ARQUITECTURA

Actualmente, existen sistemas Workflow los cuales son utilizados para automatizar un proceso de negocio. La persona que maneja dichos sistemas debe conocer el problema que se intenta solucionar, de manera que se puedan identificar los principales puntos a resolver, es decir, las principales funcionalidades que debe tener el proceso de negocio junto con su comportamiento interno. Se debe conocer también, el flujo de la información para así poder establecer que ente la manipula y de que manera. Una vez se tiene un conocimiento exacto de que se quiere resolver y cómo, se proceden a elaborar todos los elementos que contribuirán a la solución del objetivo de negocio. Entre estos están:

- Bases de Datos. Son las encargadas de almacenar persistentemente toda la información. En estas se hacen todas las consultas y se almacenan todos los datos relevantes al proceso de negocio.
- Servicios Web. Son los encargados de ir a las bases de datos y realizar todas las consultas que el usuario desee. Son la parte más importante de la arquitectura, ya que el lenguaje utilizado en BDMobIS hace uso principalmente de servicios web. En ellos se encuentran principalmente el protocolo de conexión utilizado, que en este caso es JDBC (por sus siglas en Inglés, Java Database Connectivity), junto con el modelo de acceso a datos DAO (por sus siglas en Inglés, Data Access Object). Estos prestan al sistema, funcionalidades específicas tales como la validación de usuarios, almacenamiento de pagos, consulta de estados de cuentas bancarias, etc.

Estos elementos conforman la parte mas baja de la arquitectura propuesta por BDMobIS. Una vez se tiene lista esta base, se procede a escribir el proceso de negocio utilizando los ya nombrados lenguajes de definición de flujos de trabajo Workflow. En el caso de BDMobIS, se utiliza BPEL. Hasta este punto, cualquier sistema existente para

flujos de trabajo puede llegar, ya que de ahora en adelante solo resta la ejecución y cumplimiento del objetivo de negocio. BDMobIS lleva este proceso a un punto mas alto, haciendo de un flujo de trabajo estático algo portable y ejecutable por dispositivos móviles. Según lo anterior, podríamos ver a BDMobIS como la siguiente figura:

Figura 5.1. Posicionamiento de las diferentes partes que interactúan en BDMobIS.

De esta manera se introduce al particionamiento multiagente como la opción de dividir el proceso de negocio en partes mas pequeñas para su ejecución. Esta distribución sugiere la inclusión de otros componentes, que se aclararán posteriormente, con el fin de realizar el particionamiento, dejar los sub-flujos obtenidos entendibles a los dispositivos móviles y mantener el flujo de ejecución de todo el proceso de negocio. Este capítulo contiene una descripción detallada de cada uno de los módulos que componen a BDMobIS, explicando la funcionalidad de cada uno de ellos, para posteriormente implementarlos en un prototipo que valide la presente arquitectura, lo cual se encuentra en el capítulo 6. El contenido aquí relacionado apoyó la redacción del artículo “Distribución de Procesos de Negocios en Sistemas Móviles de Información Basada en un Algoritmo de Colonia de Hormigas” el cual fue revisado y aprobado por el Comité Evaluador de la Revista Avances en Sistemas e Informática la cual esta reconocida por Colciencias en Categoría C Latindex y Publindex.

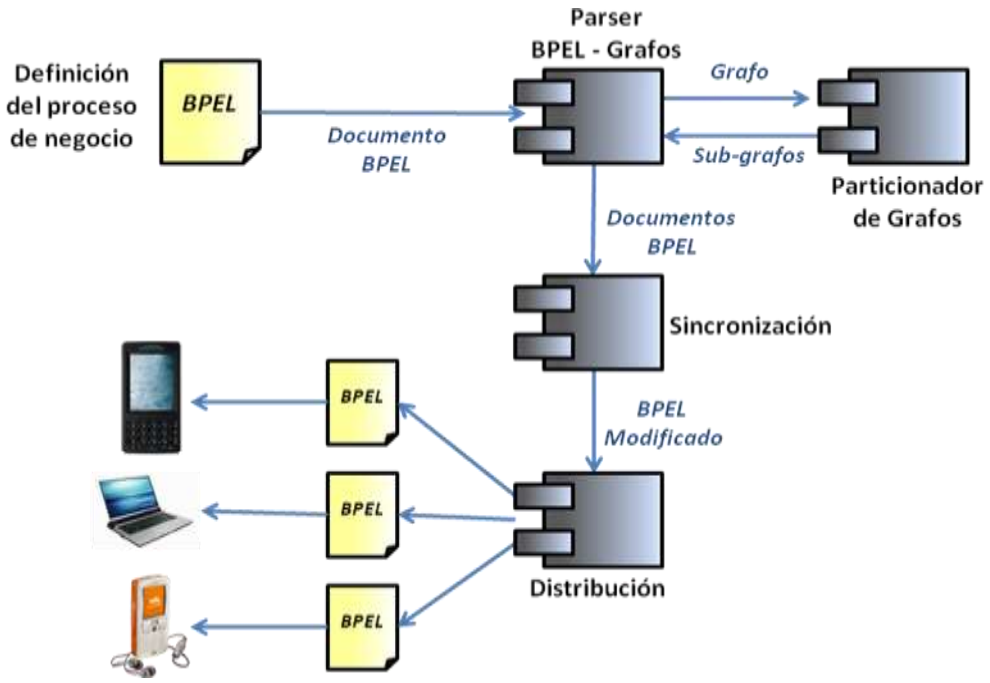

Figura 5.2: Arquitectura BDMobIS

En la figura 5.2 se puede observar la arquitectura BDMobIS planteada en el presente proyecto de grado. A continuación se describe brevemente y posteriormente se detalla cada uno de los módulos.

Con el objetivo de reducir el problema de partición de procesos de negocios a un problema de particionamiento de grafos, el módulo *Parser BPEL-Grafos* (ver Fig. 5.2), transforma un documento BPEL a un grafo y viceversa. El módulo *Particionador de Grafos* contiene el algoritmo de particionamiento que se aplica al grafo obtenido por el modulo anterior, para dividirlo en múltiples subgrafos. Las partes (subgrafos) del grafo entregadas por el algoritmo particionador se transforman nuevamente a un archivo BPEL con el objetivo de facilitar su ejecución en los motores BPEL instalados en los dispositivos móviles. El módulo *Sincronizador,* se encarga de insertar tareas de sincronización al proceso particionado para mantener correcto el flujo de ejecución. Finalmente, el módulo de *Distribución,* distribuye a los dispositivos móviles las sub partes del proceso BPEL obtenidas por el particionamiento, además realiza el proceso de reconfiguración en caso de que uno de los dispositivos móviles sea cambiado o esté apagado, lo cual se detallará posteriormente.

5.1. Definición del Proceso de Negocio.

Aunque BDMobIS puede ser adaptada a cualquier lenguaje de orquestación de servicios Web, para el presente trabajo se selecciono BPEL para la definición de los procesos de negocio de la organización. Esta selección esta basada en la flexibilidad de BPEL para modelar un proceso de negocio, en comparación con otros lenguajes como lo son BPML y WS-CDL [YWL04]. Algunos de los criterios que se tuvieron en cuenta para su selección son:

- *Modelado de colaboración*: con relación a BPML, BPEL tiene a los Partner Links como conceptos clave y los usa para modelar mensajes de colaboración Peer – to – Peer. BPML tiene esta característica, pero como algo indirecto.

- *Modelado de control de ejecución*: con relación a WS-CDL, el cual no cuenta con esta facilidad, BPEL posee un fuerte soporte gracias a un modelo de control hibrido el cual aplica a estructuras en Bloque[6] y a estructuras de transición[7].

- *Manipulación de Excepciones*: BPEL hace uso de Fault Handlers para capturar errores, de igual manera, se hacen uso de actividades *Terminate* para terminar de manera anormal una instancia de un proceso de negocio que reciba entradas no esperadas. A diferencia de esto, tanto BPML como WS-CDL hacen uso de mecanismos para la manipulación de errores pero no tan robustos como los de BPEL.

- *Nivel de abstracción*: A diferencia de BPML, BPEL tiene un muy alto nivel de abstracción ya que soporta la definición abstracta de *PartnerLinkType*. Este lenguaje soporta tanto la definición de procesos de manera abstracta como de manera concreta. La abstracción define todo lo referente a los protocolos de negocio que están envueltos en el proceso. Además, posee una parte concreta que tiene que ver con todo lo que los motores de orquestación necesitan para su funcionamiento.

[6] Según [YWL04], las estructuras en bloque se refieren a estructuras del lenguaje que están contenidas en sí mismas. Cada estructura define su propia área de trabajo para definiciones, declaraciones y comportamientos. Las actividades estructuradas prescriben el orden en el cual un conjunto de actividades toman lugar.

[7] Según [YWL04], las estructuras de transición se refieren a estructuras que pueden ser representadas por grafos dirigidos. Estos prescriben el orden de las actividades en términos de estados de transición.

5.2. Parser BPEL - Grafos

Una vez elaborado el modelo del cual se crearán las múltiples instancias del proceso, se procede a transformar su notación con el fin de realizar su particionamiento (*Parser BPEL-Grafos*). Dicha transformación se lleva a cabo utilizando la propuesta especificada por Baloche et.al. publicada en [BDR06], en la cual toman como entrada un archivo BPEL y generan un modelo BPEL representado por objetos Java. Este analizador es un proyecto desarrollado en Java el cual sigue el siguiente diagrama de clases para realizar la transformación de un documento BPEL a grafos.

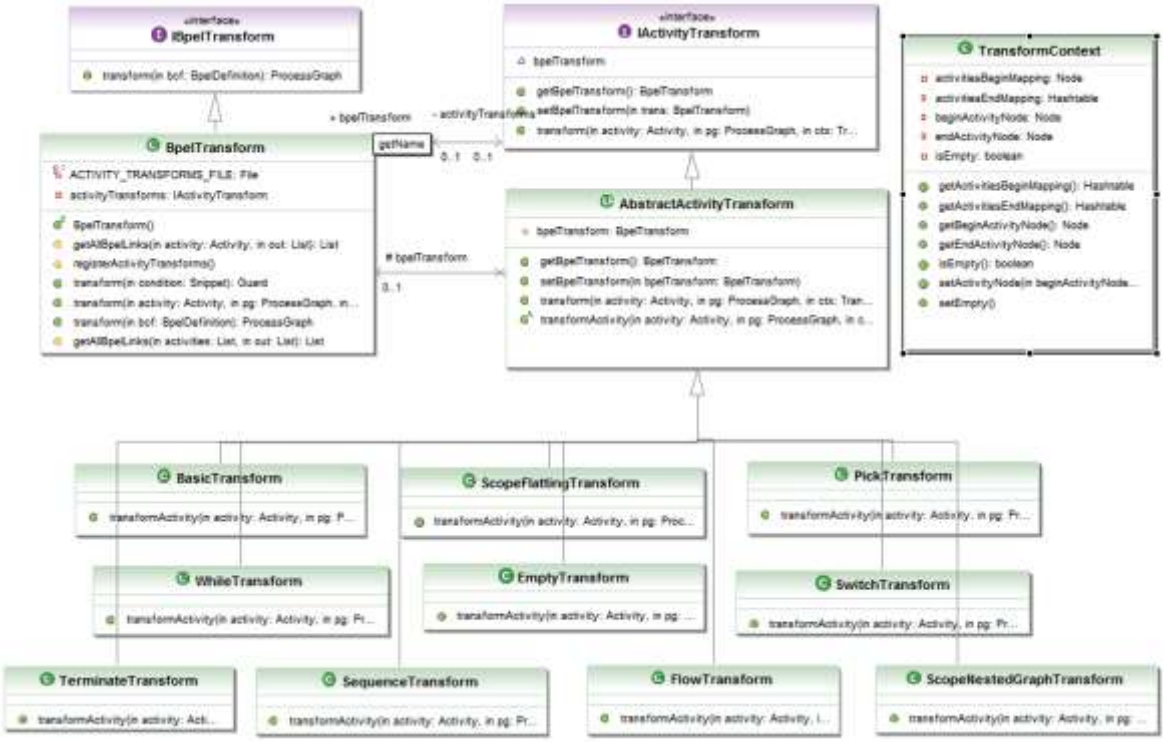

Figura 5.3. Diagrama de Clases para la transformación de un documento BPEL a grafos.

La figura 5.3 solamente esquematiza el fin último de este proyecto, pero internamente cuenta con otros procesos como el de la lectura de documentos BPEL o la generación de un metamodelo de grafos. Para complementar esto, su funcionamiento general se describe a continuación.

Inicialmente la clase *BpelReader*, perteneciente al paquete *bpm.bpel.input*, carga el archivo (.bpel) y escoge un lector especializado para cada actividad. Estos lectores no son

más que clases, las cuales implementan la interfaz *IActivityReader*. Esta interfaz se configura manipulando un archivo XML el cual especifica la lectura de cada actividad gracias a las etiquetas <*activityReader*> las cuales tienen dos atributos, el primero corresponde al nombre de la actividad que va a ser procesada y el segundo a la clase Java que realiza la lectura. Seguidamente el paquete *bpm.bpel.model* almacena todo lo concerniente a una representación en objetos del documento BPEL recién leído, para cada parte del archivo de entrada se extraen sus componentes y se crean objetos que representan fielmente al proceso de negocio. Otro paquete que integra al *Parser BPEL-Grafos* es el *bpm.graph.model*, éste posee una función, similar a la expuesta por el paquete antes nombrado, que permite almacenar la estructura resultante en un metamodelo de objetos. En dicho paquete se encuentra la clase *ProcessGraph* la cual representa un grafo mediante arcos, funciones, conectores y vértices start/end. Según Baloche et.al. en [BDR06], las funciones son los vértices del grafo que representan acciones específicas. Los vértices de start/end representan el vértice inicial y el vértice final del grafo. Los conectores son vértices que hacen posible las conexiones entre elementos al igual que pueden llevar a cabo diversas conexiones en el proceso. En la clase *ConectorType* se especifican los distintos tipos de conectores. Finalmente los arcos (aristas) hacen posible atar los vértices que se encuentran entre ellos. Por otro lado, es posible agregar a las aristas condiciones de transición mediante la clase Guard.

La parte más importante del analizador es el paquete *bpm.graph.transform* apreciable en la figura 5.3. Este es el responsable de transformar el metamodelo de BPEL en un objeto de tipo *ProcessGraph*, clase anteriormente explicada. La clase *BpelTransform*, incluida en el paquete, es el punto de entrada o el punto de inicio para la transformación de un proceso BPEL en un Grafo. Al igual que en el paquete relacionado con bpel, es decir, *bpm.bpel.*,* este paquete posee también una interfaz, llamada *IActivityTransform*, la cual debe ser implementada por cualquier clase que quiera transformar una actividad en particular. En conclusión, la clase *BpelTransform*, esta tiene las estrategias de transformación de cada actividad BPEL.

Otra clase importante del paquete en cuestión y en cuanto a estrategias se refiera, es la *AbstractActivityTransform*, la cual contiene todas las estrategias de transformación comunes a todas las actividades.

Cabe aclarar que la arquitectura propuesta es flexible a cualquier lenguaje nombrado en este capítulo. El único cambio que tendría que hacerse es la modificación del analizador de procesos de negocios, el cual será descrito en el capítulo 5, es decir, en vez de que el analizador tenga como entrada un documento BPEL, este podría analizar documentos WSFL, WS-CDL o BPML según sea el caso.

En cuanto al formato de salida del analizador, los grafos producidos son del formato GML (Graph Modelling Language) [HIM96]. Hasta este punto ya se tiene una representación de proceso de negocio como un grafo en un modelo Java que lo representa.

5.3. Particionador de Grafos

Una vez obtenida la representación del Workflow como grafo, y dicho grafo representado como objetos Java, se procede a realizar un proceso conocido como particionamiento de grafos. Particionar un grafo consiste en dividir un grafo en sub grafos o sub conjuntos de vértices y aristas de tal manera que las aristas entre los subgrafos (que representarán las comunicaciones entre los dispositivos móviles) conocidas como aristas de corte, sean las mínimas posibles, de esta manera se reduce el número de comunicaciones entre los procesos involucrados en cada subgrafo, para más información acerca del particionamiento de grafos por favor remitirse al capítulo 3. Para este módulo se empleó un algoritmo bioinspirado basado en colonia de hormigas descrito por Comellas y Sapena en [COS06] el cual se desarrolla en dos etapas que se describen a continuación.

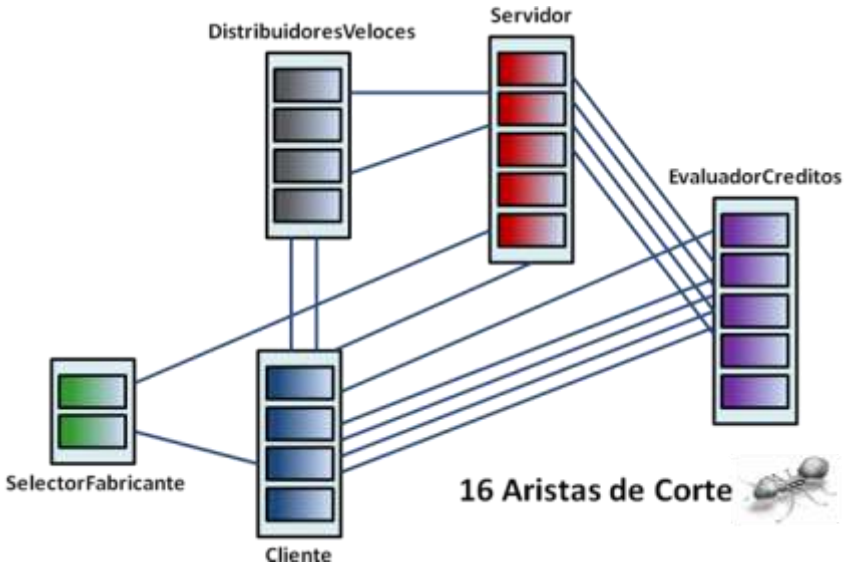

Fig. 5.4. Proceso de particionamiento inicial. Obsérvese que hay 16 aristas de corte.

La primera etapa, tal y como lo describen sus autores, inicialmente colorea el grafo de manera aleatoria conservando balanceado el número de vértices para cada color. En BDMobis este particionamiento inicial también posee esa característica aleatoria, pero a diferencia del propuesto por Comellas y Sapena en [COS06], se tienen restricciones iniciales para la distribución, ya que toma los *partnerlinks* del archivo BPEL y de acuerdo a ellos genera un número de módulos o subgrafos, por ejemplo, el grafo de la figura 5.4, muestra un proceso de negocio particionado en cinco módulos entre los que se aprecian DistribuidoresVeloces, SelectorFabricante y EvaluadorCréditos los cuales corresponden a los partnerlinks del proceso y más el cliente y el servidor. El número de módulos será igual al número de partners más uno que va a representar al módulo del servidor.

Este particionamiento inicial puede realizarse de diferente manera, como por ejemplo, sin tener en cuenta los *partnerlink* para crear los módulos y en lugar de eso permitir el uso de actores (que pueden ser empleados de la empresa), es decir, que por cada actor que utilice un dispositivo asociado al sistema, se tendrá un módulo con las tareas que debe realizar dicho actor quien además es el propietario de ese módulo. Luego a los módulos se les asigna una cardinalidad inicial, dependiendo de las capacidades de cada dispositivo asociado a cada módulo. El número de vértices por cada módulo se obtienen mediante la detección de las capacidades de los dispositivos involucrados y de acuerdo a ellas se asigna una cardinalidad máxima, es decir, el mayor número de vértices que

puede tener un módulo. El algoritmo de partición inicial se encarga de distribuir los vértices de manera aleatoria entre los diferentes módulos, respetando las cardinalidades de éstos y además respetando los vértices propietarios de cada módulo (entiéndase por vértice propietario a un vértice asociado a un actor). Para el caso de utilizar *partnerlinks*, los vértices propietarios son los que corresponden a funciones básicas de BPEL (las que contienen en sus atributos al *partnerlink*) y en el caso de utilizar *actores* los vértices propietarios son los que corresponden a tareas asignadas a ese actor o trabajador. Dichos vértices propietarios no se pueden mover de un módulo a otro, ya que están fijos en el módulo propietario.

Una vez realizado el particionamiento inicial se procede con la segunda etapa de particionamiento, en la cual se toma como punto de partida los subgrafos obtenidos del proceso de particionamiento inicial. A continuación se fijan de manera aleatoria, sobre los vértices, un número de hormigas (generalmente 3 para grafos pequeños) y posteriormente se mueven a través del grafo hacia los vértices adyacentes remplazando su color con un nuevo color (que es lo mismo que mover vértices de un módulo a otro) de manera conveniente tal que disminuya una *función de costo local.* Esta función de costo local corresponde a la relación entre el número de vecinos con diferente color y el número de vecinos totales del vértice actual sobre el que está la hormiga. A este proceso se le conoce como criterio local de optimización ver Figura 5.4.

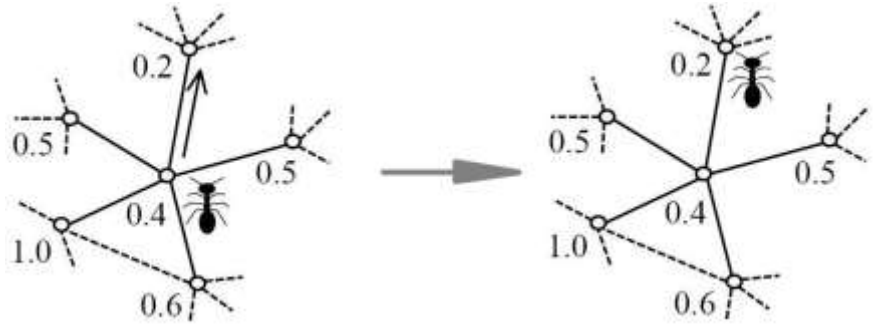

Figura 5.5. Movimiento de las hormigas al vértice adyacente con menor costo local. Imagen tomada de [COS06]

En cada iteración cada hormiga se mueve desde el vértice actual a un vértice adyacente con el mayor número de vecinos con diferente color de acuerdo a una probabilidad p_m (de lo contrario se mueve a un vértice aleatorio) y asigna el mejor color (este mejor color será el nuevo color del vértice y corresponde al color que disminuye la función de costo local),

bajo la probabilidad p_c (de lo contrario se le asigna un color aleatorio). Hay que tener en cuenta que se debe mantener el balanceo entre los vértices de cada color (sub-grafo), es decir, se toma aleatoriamente un vértice del grafo con el nuevo color y se cambia al anterior color. Con la disminución de la función de costo local se busca conseguir una reducción en una función de costo global que simplemente cuenta el número de aristas que unen vértices de diferentes colores, así después de cada cambio de color que realice una hormiga en un vértice determinado se deben actualizar tanto la función de costo global del grafo completo como la función de costo local para dicho vértice y sus vértices adyacentes. Este proceso se puede ejecutar tantos ciclos como se considere necesario para que el algoritmo converja a un valor mínimo o por lo menos cercano al mínimo en el número de aristas de corte. Una vez encontrado dicho valor, se toma como restricción y el algoritmo se vuelve a ejecutar hasta encontrar una solución que corresponde a un conjunto de subgrafos que satisfaga esa restricción.

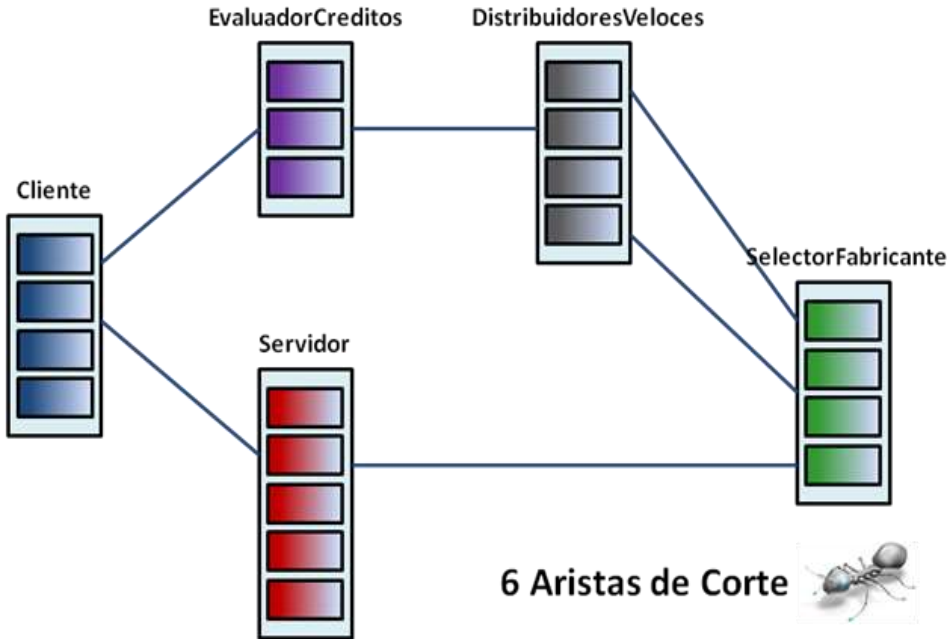

Fig. 5.6. Grafo después del proceso de particionamiento realizado con el algoritmo de colonia de hormigas. Solo queda con 6 aristas de corte de las 16 iniciales vistas en la Fig. 5.3.

Dicha solución representa un grafo particionado en k módulos con cardinalidades distintas, vértices propietarios y con un número mínimo de aristas de corte (ver Fig. 5.6), lo cual se puede observar teniendo en cuenta que la figura 5.3 corresponde al proceso de

negocio particionado inicialmente con 16 aristas de corte, y que después del proceso de particionamiento multiagente se ven reducidas a tan solo 6 como se aprecia en la figura 5.6. La solución encontrada será mejor entre más ciclos se utilice en la ejecución del algoritmo.

Es importante destacar que la naturaleza probabilística de este algoritmo le permite escapar de un mínimo local y obtener particiones con k cortes más cercanas al mínimo absoluto.

A continuación se reproduce el pseudocódigo presentado en [COS06], el cual se utilizó como base para la implementación del algoritmo de particionamiento del presente proyecto de grado.

Inicializar variables: n (numero de hormigas), k (numero de módulos o subgrafos), Pm=0,9 Pc=0,85 y la Cardinalidad de cada subconjunto

Poner cada hormiga en un vértice elegido aleatoriamente.

Colorear cada vértice del grafo de manera aleatoria formando k subconjuntos con cardinalidad predefinida.

Para todos los vértices

inicializar funcion_de_costo_local

Fin del Para

Inicializar funcion_de_costo_global

mejor_costo := funcion_de_costo_glogal

***Mientras** (mejor_costo >0) hacer*

Para todas las hormigas

***Si**(random<Pm)*

Mover la hormiga al peor vertice adyacente (El vertice con mayor número de vecinos de diferentes colores)

En caso contrario

Mover aleatoriamente a cualquier vértice adyacente

Fin Si

***Si**(random<Pc)*

Cambie el color del vértice al mejor color posible (El color que disminuya la función de costo local)

En caso contrario

Cambie a un color elegido aleatoriamente

Fin Si

mantener el balanceo(Cambiar aleatoriamente un vétice que tenga el nuevo color al viejo color)

Para los vétices elegidos y todos su vértices adyacentes

Actualizar funcion_de_costo_local

Actualizar funcion_de_costo_global

Fin Para

Si*(funcion_de_costo_global < mejor_costo)*

mejor_costo = funcion_de_costo_global

Fin Si

Fin Para

Fin Mientras

En el anterior pseudocódigo *"n"* es un número entero y corresponde al número de hormigas, éste varía de 3 a 9 dependiendo del tamaño del grafo. *"k"* es el número de subgrafos (módulos) y las probabilidades P_m y P_c toman los valores de 0.9 y 0.85 respectivamente, aunque son parámetros ajustables y permiten que el algoritmo escape del mínimo local y alcance un mínimo absoluto.

Para observar algunas diferencias entre el algoritmo de particionamiento inicial ya descrito y el algoritmo de particionamiento multiagente basado en colonia de hormigas se realizó una prueba de rendimiento que los compara. Los resultados se pueden observar en la figura 5.7.

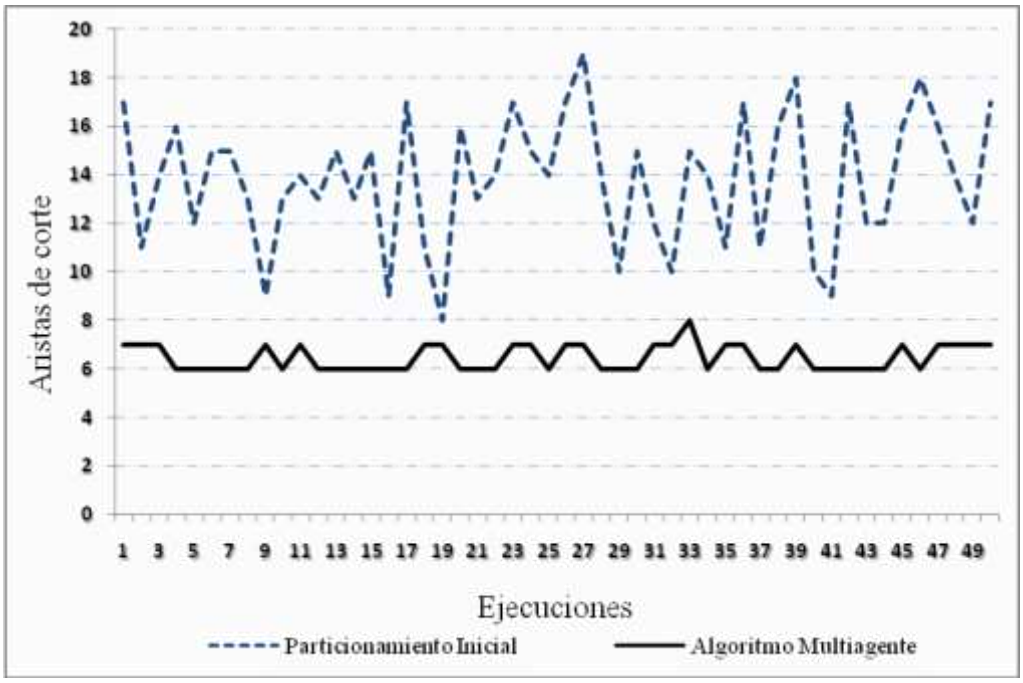

Figura 5.7. Rendimiento del Algoritmo Multiagente o de Hormigas respecto al Algoritmo de Particionamiento Inicial

Para esto se realizaron 50 ejecuciones del algoritmo de particionamiento inicial y del algoritmo de particionamiento multiagente aplicados sobre un mismo grafo que representa a un proceso de negocio, el cual cuenta con 20 nodos, 21 aristas y 4 partner links. Se puede observar que existe una notable reducción del número de aristas de corte presentadas después de la aplicación del algoritmo de particionamiento multiagente, con valores que oscilan entre 6 y 8 (gráfica inferior), respecto al algoritmo de particionamiento inicial que expone su comportamiento aleatorio con valores que oscilan entre 8 y 19 aristas de corte (gráfica superior).

Otro resultado importante de esta gráfica es que parece ser que 6 aristas de corte representan el valor más cercano o quizá sean el valor mínimo de aristas de corte en este proceso de negocio y que precisamente el algoritmo de particionamiento multiagente es quien entrega esos valores muy cercanos a este mínimo, a diferencia del algoritmo de particionamiento inicial el cual se aleja mucho de este valor. Así pues se puede observar claramente la ventaja del algoritmo de hormigas frente a un algoritmo aleatorio como el algoritmo de particionamiento inicial, por lo que se puede concluir que el algoritmo de particionamiento multiagente es una buena opción para realizar particionamiento de grafos y en este caso específico, del proceso de negocio.

La implementación del algoritmo se realizó en BDMobIS mediante el lenguaje de programación Java y consta de cuatro clases que se pueden observar en la figura 5.8: *Ant, InitialPartition, Module* y *MultiagentPartition*. La clase *Ant* corresponde a cada hormiga que se mueve en el grafo, contiene como atributos el nodo actual sobre el que la hormiga se encuentra y una memoria que indica cual fue el último nodo visitado, la clase *InitialPartition* se encarga de realizar el proceso de particionamiento inicial. La clase *Module* corresponde a cada módulo o subgrafo y por lo tanto contiene el color que le corresponde de acuerdo al particionamiento inicial y una cardinalidad máxima que indica el número máximo de nodos que puede soportar. Finalmente, la clase *MultiagentPartition* implementa el algoritmo multiagente de particionamiento de grafos con las respectivas adaptaciones al presente proyecto, como se puede observar en la figura 5.7 posee como atributos el costo global de la función, y vectores que corresponden a los nodos, arcos, hormigas y módulos.

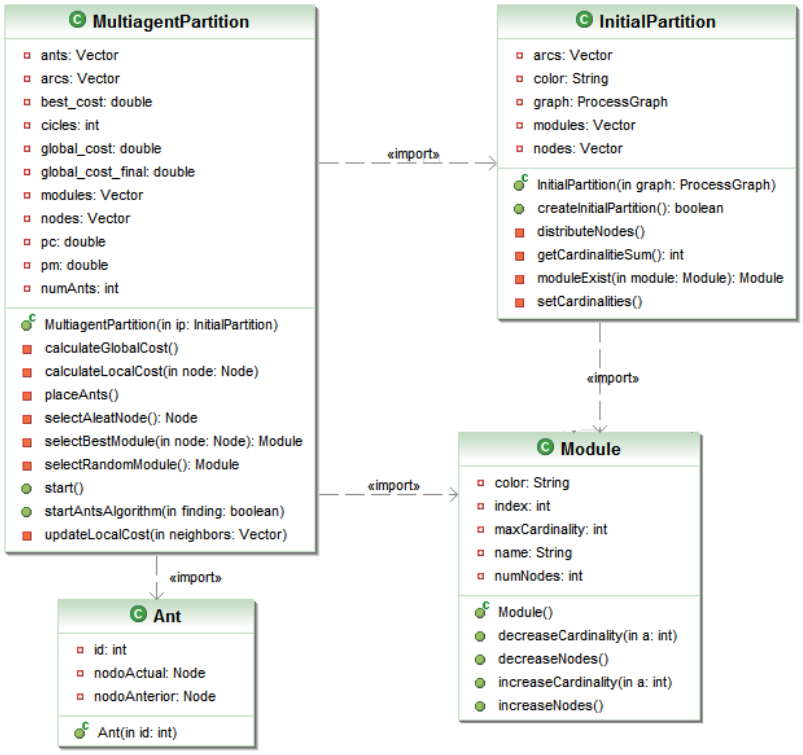

Figura 5.8. Diagrama de dependencia de clases para el módulo de particionamiento de grafos.

5.4. Sincronizador

La sincronización en BDMobIS es uno de los puntos más críticos del sistema. Ella es la responsable de que el flujo normal del proceso no se pierda aún después del

particionamiento. Cuando el proceso de negocio se define por primera vez, las actividades básicas y estructuradas se encargan de mantener su flujo, pero, una vez aplicado el particionamiento multiagente, el proceso pierde su estructura normal y es aquí en donde se debe sugerir una idea de sincronización que conserve de manera consistente el estado del proceso junto con sus variables siguiendo un método de comunicación generalizado. Según lo anterior, el proceso se divide y se envían sus partes a los dispositivos móviles para su ejecución, si entre ellos no existe una comunicación eficiente, el sistema pierde su secuencia y el objetivo de negocio se verá imposible de cumplir. Por esta razón, los participantes deben entenderse utilizando medios eficientes y ampliamente utilizados por la mayoría de los dispositivos móviles.

Para el proceso de sincronización se utilizó el mecanismo descrito en [MMO04], donde se argumenta que la inclusión de actividades de sincronización permite mantener la consistencia del proceso de negocio distribuido.

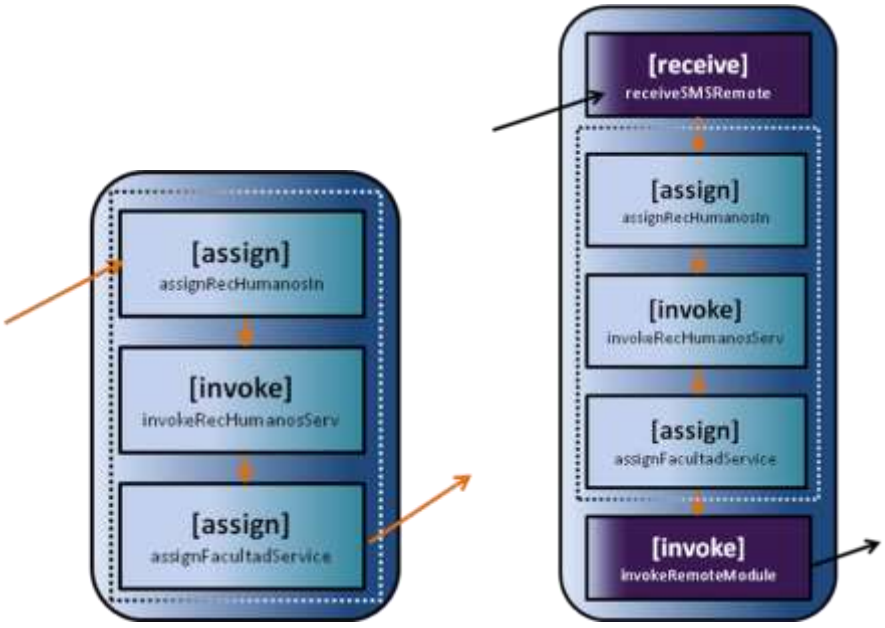

Figura 5.9. Tareas de sincronización. a) Módulo de particionamiento sin actividades de sincronización. b) Inserción de tareas de sincronización, siendo estas la primera (receiveSmsRemote) y la última del módulo (invokeRemoteModule).

En la figura 5.9 se puede apreciar la inclusión de tareas de sincronización. La gráfica de la izquierda posee tres vértices correspondientes al proceso de negocio original al cual se le ha aplicado el algoritmo multiagente de particionamiento. La gráfica de la derecha posee 5 vértices de los cuales 3 de ellos corresponden al proceso original y los 2

restantes corresponden a actividades de sincronización insertadas al proceso. La primera actividad de sincronización es una actividad receive (receiveSmsRemote), la cual está encargada de recibir los resultados arrojados por los socios precedentes a este módulo de particionamiento. La segunda es una actividad invoke (invokeRemoteModule), esta es insertada con el objetivo de notificarle al proceso en ejecución, que otro actor debe ser llamado y que algún parámetro debe ser incluido en la conversación que se mantenga con él. Las invocaciones llaman a otras aplicaciones, llamadas socios del proceso de negocio, las cuales aportan una funcionalidad única. Una actividad invoke junto con una receive conforman una interacción asíncrona muy útil al sistema que permite que el proceso de negocio pueda ejecutar otras actividades mientras socios asíncronos resuelven lo que se les ha encomendado [ANC03].

5.5. Distribuidor

Con BDMobIS cada empleado ejecuta desde su terminal un sub-proceso de negocio cuyos resultados se pasan de un trabajador a otro hasta llegar a la consecución satisfactoria del proceso global. Esto se hace a partir de los sub-procesos generados en la etapa de sincronización que deben ser ejecutados de manera descentralizada por sus respectivos responsables. Para cada socio del proceso de negocio se genera un documento con las tareas que debe ejecutar dicho participante. Estos documentos son almacenados en los dispositivos móviles para reducir las comunicaciones necesarias durante la ejecución del proceso. De esta manera los dispositivos móviles de los empleados poseen toda la información necesaria para ejecutar las tareas sin depender de los servicios remotos. Este módulo, aparte de repartir las diferentes responsabilidades a los trabajadores de la empresa, contará con una característica adicional. Esta es la reconfiguración de la distribución. Esta consistirá en reconocer si un dispositivo móvil está habilitado o no para ejecutar tareas dentro del proceso de negocio. Un dispositivo móvil estará habilitado si y solo si cumple con ciertas condiciones como por ejemplo, estar dentro del área de cobertura o simplemente estar encendido. Si alguno de estos requisitos no se cumple, el sistema propuesto debe ser capaz de reconfigurar su distribución y asignar tareas a trabajadores que si estén en condiciones de aceptar responsabilidades dentro del proceso de negocio.

REFERENCIAS CAPÍTULO 5

[YWL04] C. Yushi, L. Wah and D. Limbu, “Web Services Composition –An Overview of Standards,” Singapore Institute of Manufacturing Technology. Section Four, pp 10. Draft Unpublished. 2004.

[BDR06] I. Baloche and E. Draperi, “Projet Annuel ISTY 3 Transformation BPEL en Graphe,” Draft Unpublished. pp. 5-32. 2006.

[HIM96] M. Himsolt, “GML: Graph modeling language,” Draft Unpublished. 1996.

[COS06] F. Comellas and E. Sapena, “A multiagent algorithm for graph partitioning,” Lecture Notes in Comput. Sci. vol. 3907, ISSN: 0302-9743. In Press. pp. 279—285. 2006.

[MMO04] A. Maurino and S. Modafferi, “Partitioning rules for orchestrating mobile Information systems”, Politecnico di Milano Dipartimento di Elettronica e Informazione. Piazza L. Da Vinci 32 – 20133 – Milano Italy. Pers Ubiquit Comput (20005) 9: 291 – 300. DOI 10.1007/s00779-004-0333-4. Received: 30 July 2004/ Accepted: 10 November 2004.

[ANC03] T. Andrews, F. Curbera, “Business Process Execution Language for Web Services 1.1.,” Copyright© 2002, 2003 BEA Systems, International Business Machines Corporation, Microsoft Corporation, SAP AG, Siebel Systems. All rights reserved.

Capítulo VI
6. PROTOTIPO EXPLORATORIO

Con el fin de mostrar una implementación particular de BDMobIS, se ha desarrollado su arquitectura en un caso de estudio real. El contenido aquí relacionado apoyó la redacción del artículo “Sincronización y Distribución de Tareas para Workflows Distribuidos en Sistemas Móviles de Información” el cual fue revisado y aprobado por el Comité Evaluador de la Revista Tecnura de la Universidad Distrital de Colombia, la cual esta reconocida por Colciencias en Categoría C.

6.1. Identificación del Proceso a Automatizar.

Esta implementación se basa en un proceso de negocio el cual se repite al interior de la Universidad del Cauca más de 300 veces por semestre. Este proceso es el de la concesión de descuentos en la matrícula financiera a los conyugues o hijos de docentes, administrativos o pensionados.

Este proceso viene descrito de la siguiente manera. El estudiante paga en cualquiera de las dos Cajas de Recaudos de la Universidad del Cauca la suma de 8300 pesos, las cuales generan para el estudiante un recibo de pago el cual consta dicha diligencia ante cualquier ente al interior de la Alma Máter. El estudiante lleva este recibo, junto con la identificación del funcionario, ante la División de Recursos Humanos de la Universidad, con el fin de que se expida una constancia laboral del padre, pensionado o conyugue del interesado en el descuento. La División de Recursos Humanos tiene para dicho trámite 3 días hábiles. Al finalizar este período la constancia ha sido expedida por dicho ente si y solo si el funcionario sigue vinculado efectivamente para la universidad. Una vez el estudiante obtiene la constancia laboral de su padre o conyugue, procede a llevar dicho documento ante su facultad, último ente que participa en este proceso de negocio. El estudiante deja en su facultad su código estudiantil más el documento expedido por la

División de Recursos Humanos. Con estos datos, la Facultad a la que el estudiante pertenece, evalúa ahora el rendimiento del interesado en el descuento. Si el estudiante se encuentra en estado de Bajo Rendimiento Académico el descuento es denegado, de lo contrario el descuento es concedido. Una vez estos trámites han sido realizados, el estudiante los puede verificar cuando su recibo sea impreso. Si todo el proceso se ha llevado a cabo de manera satisfactoria, los servicios básicos son descontados al 100 %, lo cual reduce sustancialmente el valor de la matrícula del estudiante. Si el estudiante no aprecia en su recibo de matrícula financiera dicha rebaja, algún procedimiento anteriormente nombrado no ha sido realizado. Según esto, hay dos aspectos vitales para el proceso de negocio.

- Vinculación vigente del funcionario. Esta la verifica la División de Recursos Humanos de la Universidad del Cauca. El estado de esta afecta sustancialmente la concesión del descuento para el estudiante ya que si el funcionario, que puede ser padre (vivo o muerto) o conyugue del estudiante, no se encuentra actualmente vinculado con la Universidad del Cauca, el descuento es denegado. El documento que expide dicha división, hace constar, con nombre e identificación, que el funcionario se encuentra actualmente laborando. Es por ello que dicho documento hace parte vital del proceso de negocio, ya que sin el, la Facultad no tiene como proseguir con la otorgación de dicho descuento.

- Rendimiento Académico del Estudiante. Este lo verifica la Facultad a la cual pertenece el estudiante. Una vez que este lleva la constancia de que es efectivamente hijo o conyugue de un funcionario de la universidad, la Facultad procede a verificar el rendimiento del estudiante. Si el estudiante se encuentra en Bajo Rendimiento Académico, el beneficio del descuento se pierde, de lo contrario se conserva. Esta verificación la realiza la Facultad gracias a que cuando el estudiante deja la constancia laboral de su padre o conyugue, anota al pie del documento su código estudiantil. Con este dato la Facultad procede a verificar en el sistema las notas del estudiante. Cabe anotar que si el estudiante ingresa a la Universidad por primera vez, dicha verificación no se lleva a cabo y automáticamente se le adjudica al estudiante su descuento.

Si alguna de las dos condiciones no se verifica a favor del estudiante, el proceso de negocio arroja como resultado la denegación del descuento para el estudiante.

6.2. Entes participantes en el proceso.

Atendiendo a lo anterior se han identificado los siguientes entes participantes del proceso.

6.2.1. *Estudiante Universitario*. Persona que manifiesta el deseo de obtener el descuento en su matrícula financiera. Este deseo se presenta ya que dicho estudiante es o hijo de un administrativo, o hijo de un docente, o hijo de un pensionado o de un conyugue vinculado actualmente con la Universidad del Cauca. Este ente es implementado en BDMobIS con un cliente móvil J2ME (Edición Micro de Java Versión 2) (ver Fig. 6.1) mediante el cual el estudiante proporciona datos como la identificación del trámite (01 cuando se es Hijo de algún Administrativo, 02 para Hijo de Docente, 03 para Hijo de un Pensionado y 04 cuando el conyugue está vinculado a la Universidad), el documento del Funcionario de la Universidad y el código estudiantil.

Figura 6.1. Interfaz gráfica del Cliente. Con este interfaz el cliente inserta los datos para optar por el descuento en su matrícula financiera.

6.2.2.*Caja de Recaudos de la Universidad del Cauca*. Es el ente que genera la impresión de un recibo el pago para la obtención de la constancia laboral del padre o conyugue del estudiante ante la División de Recursos Humanos de la Universidad del Cauca. Cabe aclarar que no solo las Cajas de Recaudos de la Universidad pueden jugar este papel. También están habilitados para ello los bancos Banco Popular y Bancafé-Concasa. Este ente esta implementado en BDMobIS por un cliente móvil. Esto atendiendo

a la naturaleza móvil que sugiere esta arquitectura aunque el proceso de negocio real no lo sugiera así. El ideal es que dicho ente conserve su naturaleza inmóvil pero con fines de emulación se conservará la movilidad desarrollada. Su interfaz gráfica es muy similar a la mostrada por la figura 1. Este ente se apoya en un servicio web el cual guarda en una base de datos la información suministrada por el estudiante. Esto con el fin de que se dé una constancia del trámite realizado. Dicha aplicación, mediante el modelo de acceso a datos Data Acces Object DAO accede a una base de datos creada en PostgreSQL en donde se guardan datos como: código del estudiante, monto pagado, fecha y concepto de pago.

6.2.3.*Recursos Humanos*. Ente encargado de verificar la vinculación actual del funcionario por el cual el descuento puede ser otorgado. Esta función se apoya en un servicio web el cual consulta en una base de datos la vinculación actual del funcionario. Este ente es implementado también en una consola móvil. Atendiendo a las facilidades de movilidad que brinda BDMobIS cualquier funcionario de esta división que este autorizado para realizar esta comprobación, puede hacerla en cualquier momento del día y en cualquier lugar. De esta manera, la verificación de la vinculación de un funcionario en la universidad tomará solo unos pocos segundos quedando así solucionada de manera síncrona una parte del proceso de negocio. Los datos con los que este módulo trabaja hacen parte del formulario que el usuario llena al inicio del proceso de negocio, tales como cédula y nombre del funcionario.

6.2.4.*Facultad a la cual pertenece el estudiante*. Ente encargado de verificar el rendimiento académico del interesado en el descuento en su matrícula financiera. Este es apoyado en un servicio web que consulta en una base de datos la información académica del estudiante. BDMobIS implementa este ente con un cliente móvil que presenta al usuario una interfaz gráfica agradable con los datos pertinentes para dicha consulta. Estos datos son el nombre del estudiante junto con su código principalmente. Un ejemplo de esta interfaz se puede apreciar en la figura 6.2.

Figura 6.2 Interfaz gráfica de usuario para la Facultad. Por medio de esta se verifica si el estudiante esta en bajo rendimiento o no.

Aparte de los entes anteriormente nombrados, existe uno adicional. Este es el encargado de notificar finalmente al usuario si el descuento se le ha otorgado o se le ha denegado. No se lo ha incluido como ente de proceso de negocio ya que, como tal, no figura como participante de la Universidad del Cauca para el proceso de negocio. Este es un módulo agregado por BDMobIS al sistema con el fin de alivianar la carga en procesamiento que puedan tener los dispositivos móviles. Según esto, la implementación de BDMobIS cuenta con 5 módulos los cuales corresponden a los 4 entes anteriormente enumerados más el módulo del servidor. Este módulo se ejecuta en un sistema de escritorio y es el encargado de realizar las comunicaciones que sean necesarias para notificarle al usuario el estado final del descuento.

La implementación de BDMobIS comienza con la definición del proceso de negocio siguiendo paso a paso lo que se debe hacer para conceder descuentos a los estudiantes. Dicha descripción la proporcionó la Jefatura de la División de Sistemas de la Universidad del Cauca debido al basto conocimiento que tiene esta dependencia de dicho proceso. Todos los pasos son tenidos en cuenta y son pasados finalmente al lenguaje BPEL de BDMobIS. Dicho lenguaje modela el proceso en forma de actividades, las cuales, una a una se comunican entre si para cumplir con el objetivo de negocio. La unión sincronizada de estas actividades conforma el documento BPEL el cual finalmente se pasa al algoritmo particionador multiagente. Este divide el proceso de negocio y lo entrega a los dispositivos móviles para su ejecución. Después de particionar el documento BPEL definido previamente, se insertan tareas las cuales sincronizan las comunicaciones entre los diferentes módulos. Generalmente estas tareas son de invocación de otros módulos

(<invoke> dentro de la sintaxis BPEL) y de recepción de información proveniente de otros módulos del proceso (<receive> dentro de la sintaxis BPEL). Las tareas de recepción de información son insertadas al inicio de los módulos particionados y las de invocación son insertadas al final con el fin de que cada módulo, una vez termine lo que se la encomendado, llame a un homólogo y entregue sus resultados. Un ejemplo de esto se puede apreciar en la figura 6.3. En ella se puede apreciar el Módulo de Caja de Recaudos sin actividades de sincronización (ver figura 6.3 literal a)) y con actividades de sincronización (ver figura 6.3 literal b)). Esta sincronización se puede apreciar en color púrpura.

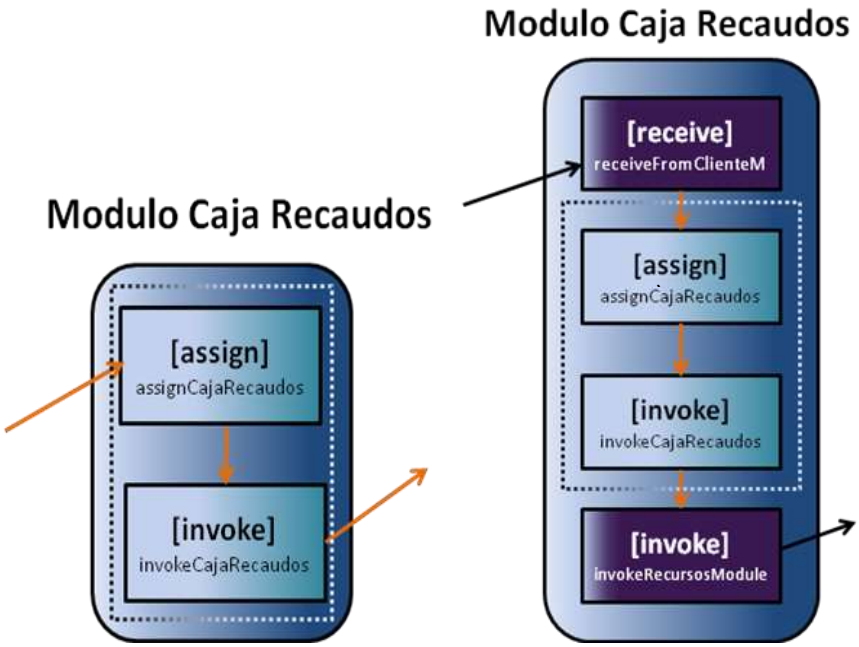

Figura 6.3. Actividades de Sincronización. a) Módulo de Caja de Recaudos sin actividades de sincronización b) Con actividades de sincronización

Gracias a lo anterior el proceso de negocio conserva su secuenciamiento y el estado de las variables globales del proceso.

Una vez se sepa como van a quedar los sub-módulos del proceso de negocio, estos deben escribirse en lenguaje BPEL. Las anteriores apreciaciones fueron hechas a nivel de grafos, con el fin de visualizar más claramente donde deben insertarse las tareas de sincronización. Ahora esto debe pasarse a lenguaje XML BPEL, ya que los dispositivos móviles no están en capacidades de interpretar grafos. Este traspaso de lenguaje se hace

utilizando un entorno integrado para el desarrollo, que en nuestro caso es el JDeveloper BPEL Designer de Oracle. Tomando paralelamente los Grafos resultantes del proceso de particionamiento, más las tareas de sincronización, se hacen uno a uno los sub-procesos de negocio, tanto los módulos que son ejecutados por los dispositivos móviles como el que es ejecutado por el servidor. Una vez se tienen los documentos XML listos, estos son copiados dentro de las aplicaciones móviles de los diferentes trabajadores del proceso.

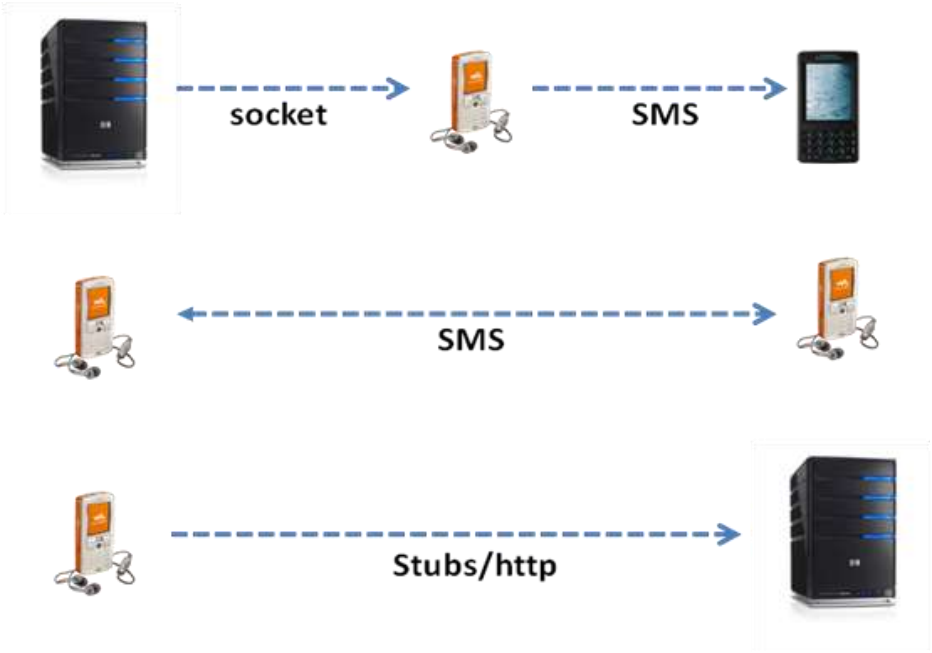

Figura 6.4. Grafico de Distribución. Se especifican también los protocolos existentes en BDMobIS.

En BDMobIS se han identificado tres tipos de comunicación (figura 6.4). La primera es la que se ocupa de comunicar un dispositivo móvil con un sistema de escritorio. Esta comunicación se implementa sobre el protocolo HTTP gracias a las facilidades de navegación que tengan los dispositivos móviles de los trabajadores de la organización. Dicha conexión se lleva a cabo gracias a la generación de stubs los cuales presentan a los dispositivos móviles toda la funcionalidad de las aplicaciones a invocar. En el caso de BDMobIS, Servicios Web. El segundo tipo de comunicación es la que tiene lugar entre dos dispositivos móviles. Para esta comunicación BDMobIS utiliza el servicio de mensajería corta SMS, por ser el protocolo más conocido, aceptado, y compatible con la mayoría de equipos móviles existentes hoy en día. El tercer y último tipo de comunicación identificado es el existente entre una aplicación de escritorio (Servicio Web) y un dispositivo móvil. Para ello BDMobIS utiliza también el envío de SMS. Con fines de

desarrollo del prototipo, BDMobIS utiliza una pasarela implementada con un Servicio Web alojado en el servidor y un dispositivo móvil pasarela (figura 6.4). Entre el proceso BPEL y el dispositivo móvil pasarela se utiliza como protocolo de comunicación la implementación de Sockets y entre el dispositivo móvil pasarela y el dispositivo móvil destino se utiliza el envío simple de SMS. En dicha pasarela se optó el uso de Sockets gracias a que estos establecen comunicaciones no orientadas a la conexión lo cual repercute en el bajo consumo de medios de conexión. Se sabe que en la práctica se suelen utilizar las estaciones SMSC (Centro de Servicio de Mensajería Corta) y como trabajo futuro, para este tipo de comunicación, se probará a BDMobIS bajo este entorno.

Los diferentes tipos de comunicación arriba nombrados se enfocan en el envío de mensajes cortos de texto. Estos transportan el valor actual de variables globales al proceso de negocio y provocan la lectura de los módulos BPEL residentes en los dispositivos móviles. Mediante una interfaz gráfica amigable el dispositivo móvil mostrará al usuario sus actividades pendientes dentro del proceso de negocio y una vez todas estas estén concluidas, el sistema procede a comunicar los resultados producidos por el trabajador al resto de los participantes.

6.3. Pruebas.

Una vez implementado el algoritmo de particionamiento multiagente, se realizaron pruebas para verificar su rendimiento. Estas se basan en:

- Tiempo destinado al particionamiento. Es el tiempo destinado por dicho algoritmo para encontrar la mejor solución, la cual fue tomada para la implementación del prototipo exploratorio. Por ejemplo, si se tenía al principio un grafo con 17 aristas de corte y se obtuvo finalmente un grafo con 8 aristas de corte, se midió el tiempo empleado por el algoritmo para obtener dicha solución. Cabe aclarar que se procuró tener máxima reducción de aristas de corte y mínimo tiempo destinado para ello. El tiempo promedio medido fue de 1400 milisegundos medidos para 50000 ejecuciones del algoritmo. El soporte para esta prueba figura en el Anexo F en su numeral 1.2.

- Reducción de aristas. Esta prueba se basó en ejecutar el algoritmo, cuantas veces fuera posible, con el fin de obtener la mínima cantidad de aristas de comunicación en el

grafo implementado por el prototipo. El desempeño fue muy bueno ya que la reducción de aristas fue del 50% o 55%. Los soportes para esta prueba pueden encontrarse en el numeral 1.1 del anexo F.

Estas pruebas son quizás las más importantes ya que demuestran uno de los principales fines de esta arquitectura que es la reducción de costos para las empresas ya que se disminuyen las comunicaciones entre los diferentes empleados que estén involucrados en un proceso de negocio.

Ahora, implementada la totalidad de la arquitectura de BDMobIS, se pudieron evaluar diferentes aspectos los cuales presentan a esta arquitectura como una buena alternativa empresarial. Entre ellos tenemos:

- Uso de dispositivos móviles. BDMobIS hace un uso eficiente de los recursos que hoy en día poseen los diferentes trabajadores de las empresas como lo son sus teléfonos celulares. Como se puede apreciar en el numeral 2 del Anexo F, BDMobIS es probado en 3 dispositivos móviles que corresponden a los utilizados por los 3 entes del proceso de negocio escogido, que son el Estudiante Universitario, la División de Recursos Humanos de la Universidad del Cauca y la Facultad a la que pertenece el usuario del sistema. Estos están en continuo movimiento y que mejor que su dispositivo móvil celular para estar al tanto de los deberes de la empresa. Por otra parte, el mercado de estos dispositivos es cada vez más amplio y es de esperarse que el soporte Java para ellos sea mayor, dándole así a este trabajo de grado luz verde para automatizar los procesos de negocios de las empresas.

- Reducción del tiempo necesario para concluir un proceso de negocio. Como se dijo anteriormente, el caso de estudio escogido para la implementación del prototipo fue la petición del descuento en la matrícula financiera por parte de los estudiantes hijos de docentes o administrativos de la Universidad del Cauca. Este proceso actualmente se demora entre 3 y 5 días hábiles, dato corroborado por el anexo G. BDMobIS, según las pruebas realizadas repetidamente, lo puede concluir al término de 7 o 22 minutos. Cabe aclarar que esto depende del desempeño de los operadores nacionales de telefonía celular, ya que sus SMSC (Centros de Mensajería Corta) son los responsables del exitoso envío de mensajes de texto entre los trabajadores de la empresa. Un soporte para

demostrar esta reducción del tiempo necesario para concluir un proceso de negocio se puede encontrar en el numeral 2 del Anexo F de esta monografía.

- Reducción de comunicaciones. Esta es una prueba relacionada con el desempeño del algoritmo, ya que, desde el momento de la implementación, el grafo ya viene reducido en sus aristas de corte, característica que se conserva hasta conseguir el objetivo de negocio. Según esto, solo se tienen estrictamente las comunicaciones necesarias.

- Eficiencia del algoritmo. BDMobIS fue probado implementando los entes que participan en el proceso de negocio sin ejecutar el algoritmo de particionamiento. Esta interesante prueba resulto en un prototipo cuyo intercambio de mensajes de texto parecía no tener fin. Esto debido a la constante comunicación innecesaria que había entre los diferentes partes del proceso. Aún así, el proceso de negocio fue concluido exitosamente pero arrojando un tiempo de respuesta de 51 minutos, donde cada ente (3 participantes) utilizó un tiempo promedio de 17 minutos. Esto repercute necesariamente en 3 puntos a tener en cuenta que son:

a. Complejidad en el desarrollo. La arquitectura puede implementarse pero esta labor crece exponencialmente en complejidad siguiendo los principios expuestos en el capítulo 3 de esta monografía.

b. Costos. El costo de la solución del proceso de negocio se vio duplicado debido a que ahora se tendría casi el doble de comunicaciones innecesarias entre los trabajadores de la empresa.

c. Inestabilidad del sistema. Las aplicaciones instaladas en los dispositivos móviles no tenían un buen desempeño ya que se veían constantemente interrumpidas por mensajes de texto innecesarios los cuales entorpecían el actual procesamiento de la información. Esto debido a la naturaleza asíncrona de BDMobIS.

Así mismo, las pruebas referentes a la eficiencia del algoritmo pueden referenciarse también a la figura 5.7. del capítulo 5 de esta monografía. En esta figura se puede corroborar también el desempeño del algoritmo multiagente con relación al simple algoritmo de particionamiento inicial utilizado por BDMobIS.

Por otra parte, BDMobIS puede trabajar perfectamente sin el algoritmo de particionamiento multiagente, pero esto demandaría de un tedioso trabajo por parte del desarrollador para eliminar una a una las aristas innecesarias de comunicación. Es quiere decir que para cada proceso de negocio se tendría que tener un análisis de interconexión presente entre los trabajadores de la empresa. Por tal motivo, el algoritmo multiagente es parte principal de la arquitectura ya que es la herramienta que permite acomodar a BDMobIS en cualquier proceso de negocio.

7. CONCLUSIONES

7.1. Conclusiones.

Se ha presentado BDMobIS, una arquitectura para la distribución de procesos de negocios entre sistemas móviles de información. Aquí se argumenta que la distribución de sub procesos de negocios en los dispositivos móviles de los trabajadores de una empresa, puede disminuir enormemente los tiempos de ejecución, ofreciendo enormes ventajas para las organizaciones. BDMobIS parte de una representación BPEL de un proceso de negocio y lo transforma a un modelo de objetos Java. Es prudente aclarar que BDMobIS representa los procesos de negocios mediante grafos. Esto con el fin de realizar su particionamiento. Otras representaciones, como las Redes de Petri, son usadas también para representar flujos de trabajo pero se preocupan por otros aspectos como su gestión la cual se ocupa de saber quién realiza una función y como la realiza. BDMobIS utiliza los grafos solo con fines de distribución, no para realizar seguimientos o gestiones de sus flujos de trabajo.

Una vez obtenido el modelo de objetos Java del proceso de negocio, se aplica un algoritmo de particionamiento bioinspirado, basado en el comportamiento de las hormigas, para realizar un proceso dinámico de distribución de los procesos de negocios que disminuye al máximo las conexiones entre los dispositivos móviles y el motor de ejecución de procesos central. Finalmente, estos sub procesos son enviados a los dispositivos móviles de los trabajadores para su ejecución. Esta ejecución compartida del proceso de negocio necesita de un entendimiento entre los diferentes integrantes del sistema. Para dicho entendimiento, BDMobIS plantea una sincronización. Esta sincronización es uno de los puntos más críticos del sistema. Ella es la responsable de que el flujo normal del proceso no se pierda aún después del particionamiento. Esta se basa en la inclusión de tareas en cada sub-proceso, lo cual permite coordinar los flujos de trabajo distribuidos, y

guardar una secuencia ordenada de todas las actividades que deben ser ejecutadas en el proceso de negocio. De esta manera, el proceso de negocio particionado es igual al proceso de negocio original definido al inicio de la arquitectura de BDMobIS.

Gracias a la inclusión de la implementación del XMLPull API Kxml2 en BDMobIS, es posible ejecutar flujos de trabajo en dispositivos móviles, así, esta arquitectura saca provecho de la movilidad de los trabajadores de una empresa. Dicha movilidad aventaja a las organizaciones que utilicen lo aquí propuesto ya que cada trabajador, sin importar su ubicación, podrá responder por lo que su empresa le ha encomendado. Esto irremediablemente reduce los tiempos de ejecución de un proceso de negocio lo cual impacta también en el desempeño de una empresa mejorando así su productividad.

Por otra parte se presentaron los tipos de comunicación en un entorno distribuido de ejecución de Workflow y las alternativas seleccionadas para BDMobIS. Es claro que BDMobIS se preocupa por el ahorro de recursos económicos para las empresas. Partiendo desde la aplicación del algoritmo de particionamiento hasta llegar a la elección de las modalidades de conexión entre los diferentes módulos en BDMobIS. Las alternativas de comunicación de BDMobIS hacen uso del protocolo de comunicación más utilizado hoy en día entre dispositivos móviles como lo es SMS así como de sockets, protocolo que, por ser no orientado a la conexión, reduce también costos para las empresas que utilicen esta arquitectura.

Como resultado del presente proyecto de grado se generó documentación que permitirá a estudiantes interesados en dispositivos móviles, servicios web y orquestación, estudiar de manera más profunda y sencilla dichos temas la cual se puede observar en los anexos B, C y D.

BDMobIS constituye un vínculo importante entre el mundo empresarial y la investigación universitaria. Los sistemas distribuidos dibujan un panorama muy prometedor para las empresas, llevando de la mano el ambiente móvil. Para este proyecto no solo se vieron involucrados esfuerzos de la facultad de ingeniería electrónica y telecomunicaciones sino también del departamento de matemáticas. De esta manera el presente proyecto mejorará el ambiente empresarial aplicando el sistema propuesto por este proyecto de grado. Haciendo caso a esta sugerencia se generó un emprendimiento en Parquesoft,

que se constituye en una empresa dedicada al análisis y automatización de procesos de negocio.

7.2. Trabajo Futuro.

Como trabajo futuro se tienen pensados los siguientes aspectos. La inclusión de un motor de ejecución un poco más genérico, como lo es SLIVER, ya nombrado en el capítulo 4. Este sistema es aplicable tanto a aplicaciones de escritorio como para entornos móviles. Sliver sería una muy buena opción para reemplazar el API kXML2 utilizado actualmente en BDMobIS. Otra futura investigación para BDMobIS es la reconfiguración automática del sistema de distribución de tareas a los dispositivos móviles. Para el sistema es de vital importancia que todos los dispositivos de los trabajadores estén disponibles el mayor tiempo posible. Si un dispositivo móvil sale del área de cobertura o simplemente se encuentra apagado, el sistema debe ser capaz de identificar esta de situación y proponer dispositivos candidatos para la ejecución de las tareas pendientes para el proceso. Esto sería una aproximación hacia las redundancias de dispositivos móviles en BDMobIS.

Generalmente, las tecnologías existentes para flujos de trabajo Workflow, trabajan bajo entornos distribuidos los cuales reposan sobre sistemas de escritorio, sistemas inmóviles. BDMobIS por su parte, propone extender este ambiente incluyendo dispositivos móviles para mejorar la ejecución de los procesos de negocio parte de las empresas ayudándoles así a convertirse en empresas más competitivas y eficientes ya que hacen un uso óptimo de sus trabajadores, reducen el tiempo destinado para cada proceso de negocio así como los recursos necesarios para ello, tales como papelería, documentos impresos, etc.

7.3. Recomendaciones.

Para entender los alcances de este proyecto se recomienda extender este proyecto al uso de otros lenguajes como WS-CDL o WSFL. De esta manera la arquitectura aquí propuesta se puede amoldar a más entornos empresariales existentes. Por otra parte, de esta manera se probará la flexibilidad del sistema aquí mostrado. Adicional a ello, se puede incluir de manera más cercana a participantes humanos. Es claro que el lenguaje

empleado por este proyecto no incluye este tipo de socios en los procesos de negocios y sería una propuesta interesante su inclusión en este tipo de trabajos.

8. Glosario

Actividad tipo proceso: es un paso lógico o descripción de una pieza de trabajo que contribuye al desarrollo de un proceso. Una Actividad tipo proceso puede incluir una actividad manual y/o una actividad Workflow automatizada.

Actividad Workflow: es la automatización computarizada de un paso lógico que contribuye a completar un Workflow.

Actividad manual: son los pasos manuales que contribuyen a completar un proceso.

Instancia de actividades tipo proceso: una instancia de una Actividad tipo proceso se define como parte de una instancia de un proceso. Una instancia de este tipo puede incluir una instancia de actividad manual y/o instancia de actividad Workflow.

Instancia de Actividades Workflow: una instancia de una actividad Workflow se define como parte de una instancia de un Workflow.

Unión – AND: cuando dos o más actividades paralelas se ejecutan convergen en un solo hilo de control común.

División- AND: es cuando un solo hilo de control se divide en dos o más actividades paralelas.

Aplicación Workflow: es un programa o grupo de programas que soportan el proceso de un ítem de trabajo en su totalidad o parcialmente con el fin de cumplir el objetivo de la instancia de actividades Workflow.

Datos de aplicación: son datos de aplicación específica y no son accesibles por el sistema de administración Workflow.

Seguimiento: es un registro histórico de las transiciones de estado de una instancia Workflow desde que inicia hasta que se complete o finalice.

Procesos de Negocios: es una clase de procesos en el dominio los negocios teniendo en cuenta su organización, estructura y políticas con el propósito de lograr objetivos de negocios.

BPR - Reingeniería de procesos de negocio: consiste de los procesos de (re-) evaluación, análisis, modelamiento, definición y la subsiguiente implementación operacional del núcleo de los procesos de negocios de una organización u otras entidades de negocio.

Iteración: es un ciclo de actividad Workflow que involucra la ejecución repetitiva de actividades Workflow hasta que una condición sea cumplida.

Unión – OR: es cuando dos o más ramas de actividades Workflow re-convergen in un solo hilo de control sin sincronización

División-OR: es cuando un solo hilo de control toma una decisión sobre el camino a elegir cuando se encuentra con múltiples ramas Workflow.

Procesos: es un grupo de actividades coordinadas tipo proceso (Paralelas o seriales) que se conectan buscando una meta global. Tales actividades pueden consistir de actividades manuales y de actividades Workflow.

Definición de procesos: es la representación computarizada de un proceso que incluye la definición manual y la definición Workflow.

Definición Workflow: es esa parte de la definición de procesos que incluye únicamente aspectos de automatización en contraparte a los manuales.

Definición Manual: es esa parte de la definición de procesos que incluye solamente los aspectos manuales frente a los automáticos (Workflow)

Instancia de procesos: representa una instancia de una definición de procesos que incluye los aspectos manuales y automáticos (Workflow)

Instancia de Workflow: representa una instancia de una definición Workflow que incluye solamente los aspectos automáticos de una instancia de procesos.

Instancia Manual: representa una instancia de una definición manual que incluye todos los aspectos manuales de una instancia de procesos.

Modo de definición de procesos: es el periodo de tiempo cuando las descripciones manuales y/o automáticas (Workflow) de un proceso se definen y/o modifican electrónicamente.

Ejecución de procesos: es la duración en tiempo cuando las ejecuciones manuales y automáticas (Workflow) toman lugar en el soporte de un proceso.

Ejecución de Workflows: es la duración en tiempo cuando una instancia Workflow se crea y se administra mediante un Sistema de administración de workflows basado en una definición de Workflow.

Datos relevantes de Workflows: son datos que son utilizados por un sistema de administración de workflows para determinar el estado de la transición de una instancia de workflows. Se puede escribir (La maquina puede entender) o no escribir (La maquina no puede entender).

Rol organizacional: es una colección de participantes basados en un grupo de atributos, calificaciones y/o habilidades.

Rol de procesos: es un mecanismo que asocia participantes con una colección de actividades Workflow.

Enrutamiento Secuencial: es un segmento de una instancia de procesos donde las actividades se ejecutan en secuencia.

Definición de Subprocesos: son procesos implementados o llamados desde otros procesos o subprocesos que incluyen las partes manual y automática (Workflow).

Herramientas: una herramienta es una aplicación Workflow invocada por el Sistema de Administración de Workflows.

Condición de Transición: es el criterio para moverse, o cambiar de estado, desde la actividad actual a la siguiente actividad o actividades en una instancia de procesos que puede ser manual o automática (Workflow).

WAPI – Interfaz de programación de aplicaciones Workflow: comprende el API para aplicaciones Workflow y las herramientas con el fin de intercomunicarse con el Sistema de Instanciación Workflow (Workflow Enactment System). WAPI es el acrónimo de **W**orkflow **A**pplication **P**rogramming **I**nterface.

WFM – Administrador Workflow: sistema de administración de workflow

Workflow: es la facilitación computarizada o automatización de un proceso en su totalidad o en parte.

Monitoreo Workflow: es la capacidad para rastrear eventos Workflow durante la ejecución de un Workflow.

Motor Workflow: un servicio software o "motor" que provee el entorno en tiempo de ejecución (run-time) para una instancia Workflow (individualmente o en conjunto con otros motores Workflow).

Interoperabilidad Workflow: es la capacidad de comunicación y trabajo en conjunto de dos o más Motores Workflow para realizar trabajo coordinado.

Participante Workflow: es un recurso que ejecuta parcial o completamente el trabajo representado mediante una instancia Workflow.

Sistema de Administración Workflow: es un sistema que define completamente, administra y ejecuta workflows mediante la ejecución de software cuyo orden de ejecución es dirigido por una representación de computador de la lógica del Workflow. Tal sistema traza al Modelo de referencia de la Unión Workflow.

Servicio de Representación o Instanciación Workflow: es un servicio software que puede consistir de uno o más motores Workflow con el fin de crear, administrar y ejecutar instancias Workflow. Las aplicaciones pueden utilizar este servicio mediante la API Workflow (WAPI).

Datos de Control Workflow: son datos administrados por el Sistema de Administración y/o un motor Workflow.

Granularidad: En computación paralela la granularidad significa la cantidad de cómputo con relación a la comunicación, es decir, la relación del cómputo a la cantidad de comunicación.

Granularidad fina: significa que las tareas individuales son relativamente pequeñas en términos del tamaño del código y de tiempo de ejecución. Los datos se transfieren entre los procesadores con poca frecuencia.

Granularidad gruesa: los datos se comunican frecuentemente en cargas de una o pocas palabras en memoria. Cuanto más pequeña es la granularidad el potencial del paralelismo es mayor y por lo tanto se incrementa la velocidad pero incrementan las sobre cargas de sincronización y comunicación.

Stigmetria: En un tipo de comunicación indirecta a través del entrono. La información se almacena en cambios del entrono físico, por ejemplo en el caso de las hormigas por las feromonas puestas en el piso.

Matching: un matching en un grafo G es un conjunto de aristas de G tal que ningún par de aristas comparta vértices en común.

Coarsening: proceso matemático que consiste en fusionar nodos de un grafo de tal manera que el grafo nuevo tenga menos nodos, pero que cualquier operación matemática sobre él afecte de igual manera al grafo inicial.

9. Lista de Acrónimos

ACRÓNIMO	SIGNIFICADO
MobIS	Sistemas Móviles de Información
WSDL	Lenguaje para la Descripción de Servicios Web
SOAP	Protocolo Simple de Acceso a Objetos
UDDI	Descripción, Descubrimiento e Integración Universal
XML	Lenguaje de Marcado Extensible
XPDL	Lenguaje XML para la Descripción de Procesos
BPSS	Esquema para la Especificación de Procesos de Negocios
WS-CDL	Lenguaje para la Descripción de la Coreografía de Servicios Web
BPML	Lenguaje para la Gestión de Procesos de Negocios
B2B	Business to Business
WSFL	Lenguaje de Flujo de Servicios Web
BPEL	Lenguaje de Ejecución de Procesos de Negocio
BPEL4WS	Lenguaje de Ejecución de Procesos de Negocio para Servicios Web

Resumen

Distribuir los procesos de negocios de una organización entre los dispositivos móviles de los empleados, puede reducir enormemente los tiempos de ejecución. Este proyecto presenta a BDMobIS, una arquitectura para la distribución de procesos de negocios en sistemas móviles de información. Los procesos descritos mediante BPEL (Business Process Execution Language), se transforman a un modelo de grafos. Posteriormente se aplica un algoritmo de particionamiento de grafos bioinspirado, basado en colonias de hormigas, que divide el proceso inicial en sub procesos, los cuales son adaptados y enviados a los diferentes dispositivos móviles para su ejecución.

Abstract

Distributing business processes of an organization among the employees' mobile devices can particularly reduce the time of execution. This project presents BDMobIS, an architecture for business process distribution in mobile information systems. The processes described through BPEL (Business Process Execution Language), are transformed into a graph model. Afterward a bio-inspired algorithm of graph partitioning based on ants colony is applied to divide the initial processes in sub-processes, which are adapted and sent to the different mobile devices for their execution.