Distribución de Procesos de Negocios en Sistemas Móviles de Información Basada en un Algoritmo de Colonia de Hormigas

Business Process Distribution in Mobile Information Systems Based in an Ants Colony Algorithm

Cristhian Figueroa, Andrés Guerrero, Armando Ordóñez, Ing., Mauricio Maca, Lic., Msc. y Juan C. Corrales, Msc.
Universidad del Cauca, Grupo de Ingeniería Telemática.
cfigmart@unicauca.edu.co, anguerrero@unicauca.edu.co, jaordonez@unicauca.edu.co, mmaca@unicauca.edu.co, corral@unicauca.edu.co

Recibido para revisión 26 de Marzo de 2007, aceptado 15 de Junio de 2007, versión final 19 de junio de 2007

***Resumen*-Distribuir los procesos de negocios de una organización entre los dispositivos móviles de los empleados, puede reducir enormemente los tiempos de ejecución. Este artículo 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.**

***Palabras clave* - Sistemas Móviles e Información, Procesos de Negocios, Workflow, Grafos, Distribución, Algoritmo de Hormigas.**

***Abstract* - Distributing business processes of an organization among the employees' mobile devices can particularly reduce the time of execution. This paper 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.**

***Key Words* - Mobile Information Systems, Business Process, Workflow, Graph, Distribution, Ants Algorithm.**

I. INTRODUCCIÓN

CON el fin de conseguir la satisfacción de los clientes hoy en día las empresas se concentran en la gestión eficiente de sus procesos de negocios. En este sentido, dentro de los objetivos de cualquier organización es de vital importancia alcanzar una coordinación entre los empleados, y el tiempo destinado para la realización de las actividades organizacionales, con el fin de evitar retardos en la ejecución de los procesos de negocios. El proyecto que aquí se describe, se interesa por mejorar estos aspectos haciendo uso de los sistemas móviles de información (MobIS) como soporte para las tareas de Workflow, permitiendo optimizar la ejecución de los procesos de negocios empresariales. Anteriormente los trabajadores estaban en su oficina utilizando computadores de escritorio como herramientas para llevar a cabo sus funciones. Hoy en día, los empleados necesitan estar en constante movilidad, de manera que cada trabajador pueda gestionar las tareas de los procesos empresariales que le corresponden, desde su dispositivo móvil, ahorrando de esta manera tiempo y ofreciendo al trabajador la posibilidad de contar con información de sus tareas en cualquier momento y en cualquier lugar, estos dispositivos de los empleados se convierten entonces en sistemas móviles de información.

Un proceso de negocio, según [1], 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. Ahora, para integrar este concepto con su automatización, se introduce la idea de Workflow la cual según [2], es definido como la ejecución de un conjunto de actividades coordinadas por personas y/o software. También se puede entender como una representación de los estados posibles en los que podría hallarse un proceso determinado y las acciones requeridas para que el proceso cambie de un estado a otro. Por otro lado, estos flujos de trabajo, o Workflows, pueden estar representados por máquinas de estado, redes de Petri, autómatas finitos, grafos, entre otros. En este proyecto se ha asumido la teoría de grafos como representación formal de Workflows, ya que ésta facilita el particionamiento de un

proceso de negocio y la posterior entrega de las partes obtenidas a sistemas móviles de información.

Algunas propuestas presentan la inclusión de dispositivos móviles en sistemas basados en Workflow, Por ejemplo en [3] se describe la asignación de responsabilidades de manera tal que cada dispositivo tiene a cargo la ejecución de una única tarea dentro del flujo. Este flujo o plan, es definido como un grafo dirigido en donde los nodos corresponden a las tareas y las aristas imponen el orden de ejecución. Estos sistemas cuentan con algoritmos para la asignación de responsabilidades a los dispositivos móviles, basando su funcionamiento en restricciones (i.e., un trabajador no podrá tener asignadas dos tareas cuyos tiempos de inicio y fin sean iguales, o dos trabajadores no pueden tener asignadas dos tareas consecutivas si el segundo trabajador no conoce los resultados de la primera). En dicho sistema, se requiere de la constante comunicación entre los diferentes trabajadores de la organización. Por su parte, BDMobIS pretende mejorar precisamente este aspecto, por medio de la aplicación de algoritmos de particionamiento de grafos que buscan la disminución de las comunicaciones entre los dispositivos móviles obteniendo como resultado la reducción de costos para las organizaciones. Otros trabajos como [4] [5], proponen un particionamiento de grafos, los cuales son representados por estructuras UML, en donde la validación se hace gracias a la gramática de grafos atribuidos AGG [6]. Nuestra propuesta, incluye un mecanismo de optimización de la comunicación que se hace al aplicar algoritmos de particionamiento de grafos que reducen al máximo las interacciones entre los diferentes trabajadores de la organización.

Este artículo se organiza de la siguiente manera: la sección dos presenta la motivación para nuestra investigación; la tercera sección describe la arquitectura de BDMobIS; la cuarta sección se centra en los módulos de definición de procesos de negocios; la quinta sección detalla el analizador de documentos para su transformación de BPEL a grafos; la sexta muestra el proceso de particionamiento de grafos incorporado en BDMobIS basado en el algoritmo de colonia de hormigas y la séptima explica la ejecución y sincronización de los sub procesos en sistemas móviles de información. Para finalizar se plantean las conclusiones y los trabajos futuros.

II. MOTIVACIÓN

Los procesos de negocios empresariales requieren de la intervención de muchos participantes, algunos de los cuales necesitan gran movilidad. Esta situación, origina enormes retardos ya que los procesos requieren de los participantes una permanente conexión con el motor de ejecución, en busca de una adecuada realización y sincronización las tareas que les corresponden.

Una solución a ésta problemática es la distribución de los procesos entre los dispositivos móviles de los empleados. Sin embargo, esta alternativa involucra la constante comunicación entre estos dispositivos y el motor central de ejecución lo cual implica un consumo amplio de ancho de banda, que en muchas ocasiones hace inviable esta solución por el tema de los costos asociados.

Figura. 1. Arquitectura BDMobIS

III. ARQUITECTURA

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. 1), transforma un documento BPEL a un grafo y viceversa. El módulo *Particionador de Grafos* contiene el algoritmo de particionamiento que se aplica al grafo BPEL obtenido por el modulo anterior. Las partes del grafo BPEL entregadas por el algoritmo particionador son transformadas nuevamente a un archivo BPEL con el objetivo de facilitar su ejecución en los motores BPEL instalados en los dispositivos móviles. Finalmente, el módulo *Distribuidor,* distribuye a los dispositivos móviles las sub partes del proceso BPEL obtenidas por el particionamiento. Por otro lado, el modulo *Distribuidor* realiza el proceso de reconfiguración en caso de que uno de los dispositivos móviles sea cambiado o esté apagado, lo cual se detallará posteriormente.

IV. DEFINICIÓN DE PROCESOS DE NEGOCIOS

Aunque BDMobIS puede ser adaptada a cualquier protocolo de orquestación de servicios Web, para el presente trabajo se selecciono BPEL como lenguaje de 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 [7]. Algunas 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 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 y a estructuras de transición.
- *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 ya nombrados.

- *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 y la parte concreta tiene que ver con todo lo que los motores de orquestación necesitan para su funcionamiento.

V. ANALIZADOR DE DOCUMENTOS PARA CAMBIO DE NOTACIÓN DE BPEL A 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 en [8], el cual toma como entrada un archivo BPEL y genera un modelo BPEL representado por objetos Java. El funcionamiento del proyecto 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*. Este posee una función, similar a la expuesta por el paquete antes nombrado, que permite almacenar en un metamodelo de objetos la estructura resultante. En dicho paquete se encuentra la clase *ProcessGraph* la cual representa un grafo mediante arcos, funciones, conectores y nodos start/end. Según [8], las funciones son los nodos del grafo que representan acciones específicas. Los nodos de start/end representan el nodo inicial y el nodo final del grafo. Los conectores son nodos 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 nodos que se encuentran entre ellos. Por otro lado, es posible agregar a las aristas condiciones de transición mediante la clase Guard.

La parte mas importante del analizador es el paquete *bpm.graph.transform*. 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*. Esta contiene todas las estrategias de transformación comunes a todas las actividades.

En cuanto al formato de salida del analizador, los grafos producidos son del formato GML (Graph Modelling Language) [9].

VI. PARTICIONAMIENTO DE GRAFOS

Para la arquitectura propuesta es necesario utilizar un algoritmo que realice la división de un workflow centralizado en varios sub-workflows distribuidos entre los dispositivos asociados al sistema (ver Fig. 1). Para muchas aplicaciones de cálculo científico, el problema de descomponer un cómputo entre varios procesadores se puede describir convenientemente usando la teoría de grafos [10], un vértice en el grafo representa un cálculo, mientras que una arista entre dos vértices indica dependencia de datos, con lo cual se puede concluir que los grafos son la herramienta idónea para representar workflows y posteriormente particionarlos para su distribución. Para el caso particular del presente proyecto los nodos representan actividades BPEL y las aristas comunicaciones entre ellas.

El particionamiento de grafos consiste en distribuir los nodos de un grafo en diferentes subconjuntos o sub-grafos de tamaños específicos de tal manera que el costo de comunicación (número de aristas) entre los subgrafos sea minimizado [10]. Este proceso se conoce también como k-corte y es un problema de tipo NP-completo [11] los cuales tienen la característica de que hasta el momento no se conocen algoritmos eficientes tiempo polinomial que los resuelvan y lo más probable es que no existan tales algoritmos, es por esto que una máquina no puede resolverlos en un tiempo razonable [12]. Entonces es de esperarse que para obtener una solución óptima el tiempo de cálculo requerido por una máquina crezca de manera exponencial con el número de nodos que tenga el grafo. La mejor forma de resolver este tipo de problemas es a través de algoritmos basados en métodos heurísticos, geométricos y evolutivos [13]. Entre los principales algoritmos de particionamiento [14] [15] [16], los que gozan de mayor aceptación en la actualidad son los algoritmos bio-inspirados en especial los algoritmos inspirados en colonias de hormigas [17]. Estos algoritmos fueron inicialmente propuestos por Marco Dorigo [18] [19] [20] y se basan en el comportamiento de hormigas reales en su proceso natural

para conseguir alimento. Una hormiga puede encontrar el camino entre su hormiguero y la fuente de comida sin necesidad de indicaciones visuales. Estos algoritmos han demostrado ser más eficaces que métodos clásicos como simulated annealing y algoritmos genéticos [21].

```
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 vértice adyacente
   En caso contrario
      Mover aleatoriamente a cualquier vértice adyacente
   Fin Si
   Si(random<Pc)
      Cambie el color del vértice al mejor color
   En caso contrario
      Cambie a un color elegido aleatoriamente
   Fin Si
   Mantener el balanceo
   Para los vértices elegidos y todos sus 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
```

Figura. 2. Pseudocódigo del algoritmo multiagente de particionamiento de grafos basado en colonias de hormigas. Tomado de [13]

Para el caso de particionamiento de grafos a cada nodo se le asigna un color que represente el subdominio o sub-grafo al cual pertenece. Así entonces se distribuye un cierto número de hormigas a través de los nodos de un grafo coloreándolos (moviéndolos de un sub-grafo a otro) mediante un criterio de optimización local [21].

El particionamiento implementado en BDMobIS, se basa en el algoritmo multiagente de particionamiento de grafos presentado por Comellas en [13] (ver Fig. 2), el cual, a diferencia de otros algoritmos de optimización de colonia de hormigas no utiliza el concepto de feromonas y por esta razón se simplifica su implementación y su ejecución.

En el anterior pseudocódigo n es el número de hormigas el cual 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 valores 0.9 y 0.85 respectivamente, aunque son parámetros ajustables y permiten que el algoritmo escape del mínimo local y alcance un mínimo absoluto.

Figura. 3. Movimiento de las hormigas al nodo adyacente con menor costo local. Imagen tomada de [13].

Tal como lo describen sus autores, inicialmente se colorea el grafo de manera aleatoria conservando balanceado el número de vértices para cada color. Un número de hormigas (generalmente 3 para grafos pequeños) se coloca aleatoriamente sobre los vértices y posteriormente se mueven a través del grafo hacia los nodos adyacentes remplazando su color con un nuevo color que disminuya una función de costo local (la cual es la relación entre el número de vecinos con diferente color y el número de vecinos totales) (ver Fig. 3).

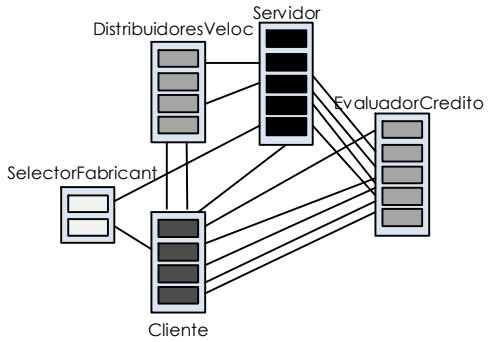

Figura. 4. Proceso de particionamiento inicial. Obsérvese que hay 16 aristas de corte.

Hay que tener en cuenta que se debe mantener el balanceo entre los vértices de cada color (sub-grafo), es decir se toma un vértice del grafo con el nuevo color y se cambia al anterior color. De la misma manera se tomarán los grafos obtenidos del proceso de negocio inicialmente definido mediante BPEL y se procederá a realizar la distribución a los diferentes dispositivos móviles.

Una vez se tiene el grafo, representado como objetos Java, se lo somete a un proceso de particionamiento multiagente [13], el cual se divide en dos etapas. La primera etapa consiste en realizar un particionamiento inicial (ver Fig. 4), el cual 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 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 además 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 no tener en cuenta los *partnerlink* para crear los módulos, sino, 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 nodos 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 nodos que puede tener un módulo. El algoritmo de partición inicial se encarga de distribuir los nodos de

manera aleatoria entre los diferentes módulos, respetando las cardinalidades de éstos y además respetando los nodos propietarios de cada módulo (entiéndase por nodo propietario a un nodo asociado a un actor). Para el caso de utilizar *partnerlinks*, los nodos 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 nodos propietarios son los que corresponden a tareas asignadas a ese actor o trabajador. Dichos nodos 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 a realizar la segunda etapa de particionamiento multiagente mediante el algoritmo de hormigas tomado de [13] (ver Fig. 2) con algunas adaptaciones al proceso de particionamiento del proceso BPEL representado mediante grafos en el modelo de objetos Java.

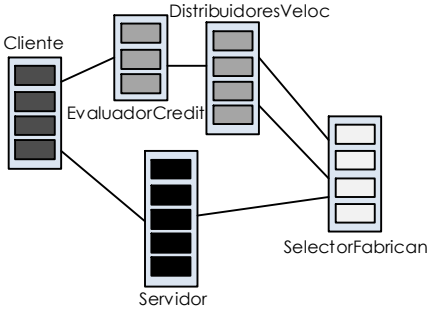

Figura. 5. Grafo después del proceso de particionamiento realizado con el algoritmo de colonia de hormigas. Solo queda con 6 aristas de corte de las 16 iniciales vistas en la Fig. 4.

En esta segunda etapa, el proceso de particionamiento toma como punto de partida el particionamiento inicial, luego, para utilizar el algoritmo de hormigas se colorean los nodos de los módulos de tal manera que cada módulo tenga un color diferente. A continuación se crean k "hormigas", también conocidas como agentes para que recorran el grafo y cambien el color de los nodos (que es lo mismo que mover nodos de un módulo a otro) de manera conveniente, para que se reduzca al mínimo una función de costo global, la cual simplemente cuenta el número de aristas que unen nodos de diferente color (diferente módulo) conocidas también como aristas de corte. El código se puede ejecutar tantos ciclos como se considere necesario y al final de la ejecución se arroja el valor mínimo encontrado durante ese número de ciclos (muy cercano o igual al óptimo). A continuación el código toma ese valor mínimo de costo global y vuelve a ejecutar el algoritmo de hormigas, hasta que se encuentre una solución (partición del grafo) que tenga esa función de costo global mínima encontrada en el paso anterior. Ésta solución representa un grafo particionado en k módulos con cardinalidades distintas, nodos propietarios y con un número mínimo de aristas de corte (ver Fig. 5), esto se puede observar teniendo en cuenta que la figura 4 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. La solución encontrada será mejor entre más ciclos se utilice en la ejecución del algoritmo.

Un sistema desarrollado bajo la arquitectura BDMobIS debe ser reconfigurable, es decir que si uno de los dispositivos asociados al sistema es reemplazado, se debe asignar una nueva cardinalidad de acuerdo a las capacidades del nuevo dispositivo y realizar de nuevo el proceso de particionamiento. En caso de que un dispositivo se apague, el sistema evaluará si hay actividades que no dependen del propietario del dispositivo (en el que se ejecute un sub grafo) de manera directa o indirecta, a través, de otras actividades. El algoritmo se implementó utilizando el lenguaje de programación Java y consta de cuatro clases: *Ant, InitialPartition, Module* y *MultiagentPartition*. La clase *Ant* corresponde a cada hormiga que se mueve en el grafo, la clase *InitialPartition* se encarga de realizar el proceso de particionamiento inicial, la clase *Module* corresponde a cada módulo o subgrafo y la clase *MultiagentPartition* implementa el algoritmo multiagente de particionamiento de grafos con las respectivas adaptaciones al presente proyecto.

VII. EJECUCIÓN DE LOS SUB PROCESOS EN SISTEMAS MÓVILES DE INFORMACION Y SINCRONIZACION

Una vez el proceso de negocio ha sido particionado, las partes obtenidas son enviadas a los sistemas móviles de información para su ejecución (ver Fig. 1). Para este módulo de la arquitectura se están estudiando dos propuestas. La primera de ellas es una simple implementación del *XMLPull API* llamada *kXML2*, definida en [22]. Este conjunto de clases hacen posible la lectura de documentos XML en dispositivos móviles que cuentan con capacidades de procesamiento reducidas. Esta opción sería considerada solo si el texto que se envía a los dispositivos móviles esta en el formato XML.

La segunda propuesta estudiada es *SLIVER* descrita en [23], que propone un motor de orquestación que ejecuta documentos BPEL. Dicho motor es un sistema que ha sido diseñado para ejecutarse tanto en dispositivos de reducidas capacidades, al igual que *kXML2*, como en computadoras de escritorio; por este motivo es una herramienta más general que la anterior en cuanto a la gama de dispositivos en los cuales esta pueda ser instalada.

En la arquitectura propuesta por *SLIVER*, la capa de transporte es capaz de intercambiar mensajes de objetos en forma de cadenas XML serializadas. Esas cadenas se convierten hacia y desde objetos Java por medio de capas analizadoras de XML y SOAP. La capa SOAP server envuelve servicios Java, proveídos por el usuario, con una interfaz de servicios Web. Cuando llegan mensajes de serializados desde las capas XML y SOAP, el servidor SOAP los direcciona al servicio solicitado por el proceso de negocio. La respuesta generada se serializa y se envía a través de la red gracias a las capas XML y SOAP.

VIII. CONCLUSIONES

En este artículo se ha presentado BDMobIS, una arquitectura para la distribución de procesos de negocios entre sistemas móviles de información. Nuestro trabajo 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. Nuestra propuesta parte de una representación BPEL de un proceso de negocio y lo transforma a un modelo de objetos Java, posteriormente 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.

Hasta el momento se han implementado los módulos parser de BPEL a grafos y el módulo Particionador (ver Fig. 1). Como trabajos futuros, se pretende desarrollar un mecanismo robusto para la distribución de los sub flujos de trabajo en los dispositivos móviles, de manera que se puedan seleccionar diversos lenguajes como BPEL o BPML, dependiendo de la complejidad de las tareas y las capacidades de los dispositivos móviles. Así mismo se busca desarrollar un protocolo de sincronización para BDMobIS, que permita coordinar los flujos de trabajo utilizando el menor número de conexiones entre el sistema móvil de información y el motor central de ejecución del proceso. Finalmente, se pretende construir una interfaz grafica que suministre una actualización personalizada del grado de cumplimiento de las tareas realizadas por los trabajadores que intervienen en el proceso.

REFERENCIAS

[1] A. Arkin y A. Agrawal, "Business Process Modeling Language". Working Draft 0.4. 3/8/2001 BPMI.org Intalino, Inc. Status: Working Draft. Draft Unpublished. 2001

[2] G. Cor, "10 Oportunidades arquitecturales para Workflow". Paradigma Software. Microsoft Regional Director. Uruguay, Paraguay y Bolivia. Draf Unpublished. 2006

[3] G. Hackmann, R. Sen, H. Mart, R. Gruia-Catalin, G. Christopher, "MobiWork: Mobile Workflow for MANETs", Washington University in St. Louis School ofEngineering & Applied Science. Department of Computer Science & Engineering. WUCSE-2006-18. In Press. April 14, pp. 5-12. 2006.

[4] L. Baresi, A. Maurino y 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.

[5] L. Baresi, A. Maurino y 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.

[6] M. Beyer, "AGG1.0 – Tutorial", Technical University of Berlin, Department of Computer Science. Unpublished. 1992.

[7] C. Yushi, L. Wah, y D. Limbu, "Web Services Composition –An Overview of Standards". Singapore Institute of Manufacturing Technology. Section Four, pp 10. 2004

[8] J. Corrales, D. Grigori, and M. Bouzeghoub. Bpel processes matchmaking for service discovery. In Proc. of 14th International Conference on Cooperative Information Systems (CoopIS), pages 237–254, 2006.

[9] M. Himsolt. "GML: Graph modelling language". Draft Unpublished, Diciembre 1996.

[10] B. Hendrickson y R. Leland, "A multilevel algorithm for partitioning graphs, en Proceeding of Supercomputing". In S. Karin, editor, Proc. Supercomputing . ACM Press, 1995. pp. 1-13

[11] G. Hernández, "Complejidad y Grafos". II Seminario de matemática discreta y codificación de la Informática. Universidad Politécnica de Madrid. Draf Unpublished. Abril 2000.

[12] A. Cortéz. "Teoría de la complejidad computacional y teoría de la computabilidad". Universidad Nacional Mayor de San Marcos. Lima, Perú. 2004. Draf Unpublished.

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

[14] S. Kirkpatrick, CD. Gellat y M.P. Vecchi, "Optimization by simulated annealing". Science, 220:671-680. In Press. 1983.

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

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

[17] Z. Subing y L. Zemin, "A QoS routing algorithm based on ant algorithm". In Press, in Proceedings of the 25th Annual IEEE Conference on Local Computer Networks. LCN. ISBN: 0- 7695-0912-6. 2000.

[18] A. Colorni, M. Dorigo y V. Maniezzo, "Positive Feedback as a Search Strategy". In Press, Tech. Rept. 91-16 Politecnico di Milano - Department of Electronics, Milano – Italy. November, 1991.

[19] M. Dorigo, "Optimization, Learning and Natural Algorithms", PhD thesis, Politecnico di Milano, Italy. Elsevier Publishing. 1992.

[20] M. Dorigo y T. Stützle, "Ant Colony Optimization", MIT Press. ISBN 0-262-04219-3. 2004.

[21] F. Comellas, J. Ozón, A. Cortés, J. Abril y 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. In Press. 17-18 Noviembre 1999.

[22] A. Froufe y P. Jorge, "J2ME Java 2 Micro Edition Manual de usuario y tutorial", In Press. ISBN 84-7897-597-7, edición original publicada por RA-MA Editorial, MADRID, España. Derechos reservados RA-MA Editorial, pp. 381-397. 2004.

[23] G. Hackmann, M. Haitjema, G. Christopher, 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

J. Armando Ordóñez C. Cargo actual: Docente e Investigador grupo de Ingeniería Telemática de la Universidad del Cauca. Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca, Estudiante de Maestría, Área Telemática Universidad del Cauca. Áreas de trabajo: Composición de Servicios Web, Tecnologías de Integración de Servicios de Información

Mauricio Maca Ch. Cargo actual: Docente Departamento de Matemáticas Universidad del Cauca. Magíster en Ciencias Matemáticas, Áreas de Trabajo: Matemática computacional: Teoría de la Complejidad

Juan C. Corrales M. Cargo actual: Docente e Investigador grupo de Ingeniería Telemática de la Universidad del Cauca. Magíster en Telemática Universidad del Cauca y estudiante de Doctorado en la Université de Versailles Saint-Quentin-En-Yvelines. Laboratoire D'Informatique Prism, Áreas de Trabajo: Sistemas de Información Geográfica, Tecnologías de Integración de Servicios de Información.

Cristhian Figueroa M. Cargo actual: Estudiante realizando proyecto de grado para obtener título de Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca. Áreas de Trabajo: Desarrollo de aplicaciones móviles e inalámbricas, integración de procesos de negocio, orquestación de servicios Web.

Andrés Guerrero A. Cargo actual: Estudiante, realizando proyecto de grado para obtener título de Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca. Áreas de Trabajo: Desarrollo de aplicaciones móviles e inalámbricas, integración de procesos de negocio, orquestación de servicios Web.