MOBFLOW PLATAFORMA DE DESPLIEGUE DISTRIBUIDO DE PROCESOS DE NEGOCIO EN SISTEMAS MÓVILES DE INFORMACIÓN

Monografía presentada para optar al título de Ingeniero de Sistemas

Gustavo Adolfo Aponzá Villaquirán

Henry William Dorado Gomez

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Sistemas

Línea de investigación en Ingeniería de Sistemas Telemáticos

Popayán, Junio de 2010

MOBFLOW PLATAFORMA DE DESPLIEGUE DISTRIBUIDO DE PROCESOS DE NEGOCIO EN SISTEMAS MÓVILES DE INFORMACIÓN

Monografía presentada para optar al título de Ingeniero de Sistemas

Gustavo Adolfo Aponzá Villaquirán

Henry William Dorado Gomez

Director

Dr. Ing. Juan Carlos Corrales

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Sistemas

Línea de investigación en Ingeniería de Sistemas Telemáticos

Popayán, Junio de 2010

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPITULO I

INTRODUCCIÓN

1.1. CONTEXTO

Las diferentes organizaciones o empresas generalmente operan con uno o muchos procesos empresariales, tales como traspaso de información (formularios, informes, etc.) de un miembro de la organización a otro, creación y uso de una factura, el procesamiento de una orden, contestar un teléfono, para citar solo unos cuantos ejemplos que pueden considerarse como procesos empresariales o de negocio. La ejecución de estos procesos de negocio de manera manual, muchas veces exige demasiado tiempo, resultan costosos y requieren del uso de los diferentes recursos empresariales (humanos, materiales).

A través de los sistemas Workflow se puede realizar la automatización de dichos procesos de negocios, brindando una estructura estable para dar soporte a estos y además permitiendo priorizar las tareas para ejecutarlas tan pronto como sea posible, en su respectivo orden y por la persona encargada. De igual manera, un sistema de Workflow proporciona grandes beneficios como ahorro de tiempo, costos y recursos a las organizaciones que lo utilizan; además permite establecer eficiencia y control en los procesos de la organización, junto con una estandarización de los mismos, logrando asignación de tareas, disponibilidad de recursos y diseño de procesos, entre otros [1], [2].

Pasado el tiempo los procesos de negocio sé tornaron cada vez más complejos, involucrando diferentes entidades y locaciones geográficas, esto dio paso al surgimiento de los sistemas BPM (*Business Process Management - Administración de Procesos de Negocio*) [15] los cuales están cambiando la manera de realizar la ejecución de estos Workflows a un ambiente más distribuido, supliendo las carencias de los sistemas Workflows en el campo de los procesos de negocio complejos. BPM esta soportado sobre tecnología de información, para automatizar las actividades que son parte del ciclo de vida de un proceso de negocio, y dar agilidad a los cambios requeridos por la empresa.

La tecnología que posibilita la implantación y adopción de BPM constituye una categoría nueva de sistemas informáticos, denominada Sistema de Gestión de Procesos de Negocio (Business Process Management System - BPMS) [16]. A diferencia de los sistemas de información tradicionales basados en la gestión de datos, estos sistemas se especializan en la gestión de procesos de negocio. De estos surgen diferentes estándares entre los cuales se destacan: *Electronic Business XML (ebXML)* [30], *Yet Another Workflow Language* (YAWL) [28]**,** *Web Services Choreography Description Language* (WS–CDL) [43], *Business Process Modeling Language* (BPML) [27], *Web Services Flow Language* (*WSFL*) [44] y *Business Process Execution Language* (BPEL) [26]. Los cuales permiten la definición y ejecución de Procesos de negocio.

Por otro lado, tradicionalmente el desarrollo de aplicaciones de Workflow – BPM ha sido orientado a la ejecución en dispositivos de cómputo tradicionales, pero con la popularización de los dispositivos móviles como celulares, teléfonos inteligentes (smartphones), PDA's y tecnologías como Bluetooth, Wi-Fi que prescinden de los cables, se puede extender las capacidades que actualmente brindan los sistemas BPM y aprovechar todas esta nuevas tecnologías, de esta forma, se facilita el acceso a información desde fuera de la oficina, aumentado la disponibilidad de los participantes (en tiempo y lugar), y dando lugar a procesos de negocio más dinámicos y con menores tiempos de respuesta. A los dispositivos móviles que ejecutan estos procesos de negocio se les denomina Sistemas Móviles de Información [8].

1.2. ESCENARIOS DE MOTIVACIÓN

Los distintos actores de los procesos empresariales, muchas veces requieren, por las labores que ejecutan, de una constante movilidad lo que puede generar retrasos en la ejecución de sus procesos empresariales. A continuación se presenta un escenario donde se visualiza esta problemática.

Venta de artículos en un Almacén: Consideremos un almacén que vende determinado tipos de artículo, para realizar la venta los productos participan los siguientes actores.

Vendedor. Para el proceso de ventas, el vendedor se encarga de ofrecer los productos ya sea en el almacén o puerta a puerta. Cuando hace una venta, debe llenar una serie de formularios con la información de la venta, y enviarlos al almacén central para que se realice su respectivo registro.

Cajero. Para el proceso de ventas se encarga de realizar el registro de toda la información de las ventas y recibir el dinero. Además es encargado de otras tareas como consultar las diferentes ventas, muchas veces de varias sucursales y también manejar aspectos financieros del almacén.

Analista de crédito. En el proceso de ventas es el encargado de analizar una venta a crédito, y determina la aprobación o no de una venta. Esta función la realiza consultando los historiales de crédito del cliente, la información de los estados financieros y la información de los fiadores. En algunos casos realiza visitas domiciliarias para confirmar la información de los clientes.

Inventario. Finalmente esta el encargado de la bodegas e inventario quien para este proceso actualiza el inventario según la venta realizada. Además tiene otras labores como el control de compra de productos, distribución de artículos a diferentes bodegas, registro de salida e ingreso de producto de las bodegas.

Para este escenario, los actores del proceso de ventas debido a sus múltiples actividades y a la naturaleza de algunas, requieren movilizarse a diferentes lugares para cumplir con sus tareas, por consiguiente, los participantes en dichos procesos no mantienen todo el tiempo en sus puestos de trabajo. Lo anterior genera retrasos en la ejecución de los procesos de negocio de la organización, ya que los participantes se están movilizando constantemente y no permanecen frente a un equipo de oficina para la ejecutar sus actividades como lo requiere el proceso.

En este caso, se puede observar la necesidad de implementar un sistema móvil de información que permita a los actores del proceso desplazarse y realizar sus tareas desde cualquier lugar.

1.3. PLANTEAMIENTO DEL PROBLEMA

Hoy en día el modelamiento de los procesos de negocio y la automatización de estos, mejor conocida como Workflows, son una iniciativa clave en todas las grandes empresas, debido a permiten a las organizaciones aumentar la competitividad que tanto requiere el mercado actual [2].

Para lograr una mayor competitividad en las organizaciones, ha surgido la necesidad de definir y automatizar procesos de negocio complejos que involucran diversos participantes, manejo de diferentes dependencias y recursos, además de coordinación entre los trabajadores que pueden estar distribuidos geográficamente [1]. Por lo cual dichos trabajadores, actores de los procesos de negocio demandan continuamente movilidad por las labores que desempeñan, generando que los participantes de dichos procesos no permanezcan todo el tiempo en sus puestos de oficina trabajando en sus correspondientes tareas como sucedía anteriormente.

Esto genera retrasos en la ejecución de los procesos de negocio empresariales, debido a que requieren de la presencia del trabajador dentro de la empresa, ya que los sistemas de gestión de los procesos de negocio empresariales funcionan habitualmente dentro de la intranet de la organización [3].

Aprovechando la popularización de los diferentes dispositivos móviles y tecnologías inalámbricas que permiten movilidad de las personas [7], muchas empresas buscan la manera de utilizar estas tecnologías para ejecutar sus procesos de negocio, lo que conlleva a que los procesos de la organización estén distribuidos en un ambiente compuesto por actores nómadas y cambios dinámicos.

El problema a ser direccionado en esta disertación, es como lograr que los participantes de los procesos empresariales puedan ejecutar sus tareas desde el lugar que lo requieran.

A partir de esto se planteo la siguiente pregunta de investigación: ¿Cómo distribuir las actividades de un proceso de negocio en sistemas móviles de información?

1.4. NUESTRA PROPUESTA

Nuestra propuesta consiste en la estructuración e implementación de una plataforma para la distribución, ejecución y monitoreo de procesos de negocio en sistemas móviles de información. Para esto la plataforma utiliza los procesos descritos en BPEL, los cuales se transforman a un modelo de grafos, posteriormente se le aplican algoritmos de particionamiento y sincronización para obtener subprocesos sincronizados, los cuales son adaptados y distribuidos a los diferentes dispositivos móviles de cada actor del proceso para su ejecución y posterior monitoreo.

1.5. CONTRIBUCIONES

Las contribuciones realizadas por el presente trabajo de grado son:

- **Arquitectura para el despliegue distribuido de procesos de negocio en sistemas móviles de información**

  En la arquitectura propuesta en el presente trabajo de grado se definen diferentes módulos para llevar a cabo el particionamiento, sincronización, distribución, ejecución y monitoreo de procesos de negocio.

  Estos aspectos fueron tratados en proyecto como [11], [32] y [33] los cuales se orientan básicamente en la manera de ejecutar los procesos de negocio en los dispositivos móviles, además en trabajos de investigación como [3], [4], [10] y [8] se enfocan principalmente en el particionamiento de procesos negocio de manera eficiente. Estas investigaciones manejan estos aspectos de forma individual, a diferencia del presente trabajo de grado donde se define una arquitectura que trata todos estos aspectos de manera integral.

  Por otro lado, la arquitectura presentada en este proyecto propone diferentes módulos para la funcionalidad Web y Móvil, exponiendo la manera que interactúan estos dos componentes en el despliegue distribuido de procesos de negocio, aspecto que no ha sido tratado en otros trabajos.

- **Algoritmos de particionamiento y sincronización de documentos BPEL, basados en las reglas propuestas en [8].**

  En los trabajos [3], [4], [8] y [10] se proponen una serie de reglas formales para el particionamiento de procesos de negocio en sistemas móviles de información, estudios que plantean como trabajo futuro un motor de particionamiento automático de documentos BPEL para generar subprocesos sincronizados.

  En el presente trabajo de grado se define e implementa una serie de mecanismos y algoritmos, necesarios para realizar el particionamiento y sincronización de procesos de negocio descritos en BPEL, los cuales están basados en las reglas propuestas en [8]. Logrando así, el desarrollo de una herramienta de particionamiento automático, para la generación de subprocesos sincronizados, a partir de un proceso de negocio BPEL.

- **Desarrollo de mecanismos para la distribución de procesos de negocio en dispositivos móviles.**

  El presente proyecto de grado define mecanismos para la distribución de procesos de negocio, los cuales permiten asociar los actores con cada subproceso de acuerdo al rol que desempeñan, para que estos sean distribuidos y descargados en los dispositivos móviles de cada actor participante en la ejecución del proceso de negocio. Aspecto que no ha sido tratado de esta manera por ninguna investigación.

- **Definición de algoritmos y mecanismos para ejecución sincronizada de subprocesos de negocio en sistemas móviles de información.**

  Como ya se mencionó en los trabajos [3], [4], [8] y [10] se propone una serie de reglas para el particionamiento de procesos de negocio en subprocesos BPEL sincronizados, a través de una serie de reglas, pero no tienen en cuenta un mecanismo de sincronización de la ejecución, después de que estos han sido particionados. Aporte que hace el presente proyecto de grado al definir e implementar algoritmos y mecanismos para la ejecución sincronizada, que permiten enviar la información necesaria entre dispositivos de forma que se ejecuten de manera sincronizada y similar al proceso global.

- **Definición de Algoritmos y Mecanismos para el monitoreo de procesos de negocio en dispositivos móviles.**

  Es este proyecto se definen algoritmos y mecanismos para el monitoreo de la ejecución distribuida de los procesos de negocio en los sistemas móviles de información, estableciendo los mecanismos necesarios para registrar las actividades ejecutadas en los dispositivos móviles y el despliegue de esta información. Además se desarrollaron los algoritmos para lograr la visualización del monitoreo en base a su ejecución.

- **Desarrollo de una aplicación móvil que integra un motor BPEL Genérico para la ejecución distribuida de procesos de negocio.**

  En los trabajos [3], [4] y [5] proponen como trabajo futuro incluir un motor BPEL genérico para la ejecución de procesos de negocio en los dispositivos móviles, punto que tiene en cuenta el presente proyecto al desarrollar una aplicación móvil que incluye el motor BPEL propuesto en [11], la cual junto con este motor de ejecución permite lograr la descarga de procesos, ejecución sincronizada y envió de información para el registro de actividades ejecutadas.

1.6. CONTENIDO

El presente trabajo de grado está compuesto por 6 capítulos:

Capítulo II.

Es el producto de la exploración del estado del arte relacionado con la distribución de procesos de negocio en sistemas móviles de información. En este capítulo se describen conceptos teóricos fundamentales, además de los trabajos relacionados a la presente propuesta.

Capítulo III.

Dentro de este capítulo se presenta la metodología utilizada para el desarrollo del proyecto y la arquitectura que da soporte a la plataforma propuesta, junto con los diferentes artefactos obtenidos en dicho proceso de desarrollo.

Capítulo IV.

Describe como se realizan los aspectos más críticos e importantes de la plataforma que son particionamiento y sincronización de procesos de negocio, distribución de subprocesos, ejecución sincronizada y monitoreo de la ejecución, aspectos que a la vez representan los objetivos del presente trabajo de grado.

Capítulo V.

En este capítulo se presenta la descripción del caso de estudio y el prototipo de validación implementado para la plataforma, además se describen los criterios y la metodología de evaluación del prototipo, junto con los resultados obtenidos de las diferentes pruebas realizadas a la plataforma.

Capítulo VI.

Presenta las conclusiones y propone algunos trabajos futuros alrededor de la distribución de procesos de negocio en sistemas móviles de información.

CAPITULO II

ESTADO ACTUAL DEL CONOCIMIENTO

En este capítulo se presentan las bases teóricas, estándares y especificaciones más relevantes, que brindaron las bases conceptuales para el planteamiento de la ***plataforma de despliegue distribuido de procesos de negocio en sistemas móviles de información.***

Igualmente, se exponen los trabajos relacionados que sirvieron para plantear y desarrollar la plataforma. A continuación se presentan los conceptos y descripciones de Arquitectura Orientada a Servicios, Workflows y posteriormente se presentan iniciativas relacionadas con este proyecto.

2.1 MARCO TEÓRICO

2.1.1 Procesos de Negocio Web y Workflows

Existen actualmente diferentes organizaciones dedicadas a la investigación de automatización de procesos de negocio. Entre estas, la principal organización es la Coalición para la Gestión de Workflow (Workflow Management Coalition - WfMC) [13], la cual define a un ***proceso de negocio*** como "*un conjunto de uno o más procedimientos o actividades directamente ligadas, que colectivamente realizan un objetivo del negocio, normalmente dentro del contexto de una estructura organizacional que define roles funcionales y relaciones entre los mismos*".

Un ejemplo de un proceso puede ser el conjunto de pasos que se requieren para la creación y el respectivo uso de una factura, que incluye además información de interés para los diferentes miembros de la organización, entre los cuales pueden estar un operario, una secretaria o un gerente, debido a que el proceso de negocio inmiscuye todos los aspectos de la empresa.

Para integrar el concepto de proceso de negocio con su automatización, se introduce la idea de ***Flujos de Trabajo (Workflows)***, la cual según la WfMC se define como: *"La automatización de un proceso de negocio, total o parcialmente, durante el cual los documentos, información y tareas son pasadas de un participante a otro, acorde a un conjunto de reglas establecidas".*

Adicional a esto, es importante entender el concepto de particionamiento de Workflow, que es el proceso por el cual un Workflow o proceso de negocio se puede dividir en subprocesos, que corresponden a una parte del proceso de negocio original.

Por otro lado, para que este concepto funcione es necesario especificar el proceso de negocio, donde se establece un inicio de actividades, unas reglas, excepciones, entre otras. A lo cual se le denomina *definición del proceso*, que expresa la forma de representar un proceso de negocio mediante documentos

2.1.1.1 Sistema de gestión de Workflow (Workflow Management System) y Modelo de Referencia

Como se menciono anteriormente, en un Workflow las tareas, información y documentos pasan de un participante a otro, para que se realicen una serie de acciones de acuerdo a un conjunto de reglas de negocio. Los sistemas que dan soporte a la definición del flujo de trabajo y a su posterior ejecución se denominan *Sistema de gestión de Workflow (*Workflow Managment System - WMS), el cual es definido como "*un sistema que define, crea y gestiona la ejecución de flujos de trabajo (Workflow) mediante el uso de software, interpreta la definición del proceso, interactúa con los participantes y, siempre que se requiera, invoca el uso de herramientas y aplicaciones*" [13][14]. En la figura 1 se muestran las características de un WMS.

Estos sistemas de gestión de Workflow, pueden ser implementados con cualquier tecnología y comúnmente se caracterizan por tener tres áreas de funcionalidad, la primera de ellas corresponde a las *funciones de tiempo de Construcción (Build-time functions)* que son usadas para la definición y modelado de un proceso junto con todas sus actividades concernientes. También están las *funciones de control en tiempo de ejecución (Run-time control functions)*, estas funciones se usan para controlar el

proceso en el ambiente de ejecución. Finalmente, se tienen las *funciones de interacción en tiempo de ejecución (Run-time interaction),* utilizadas para interactuar con los usuarios o aplicaciones externas, con el fin de que los participantes del proceso puedan llevar a cabo sus tareas.

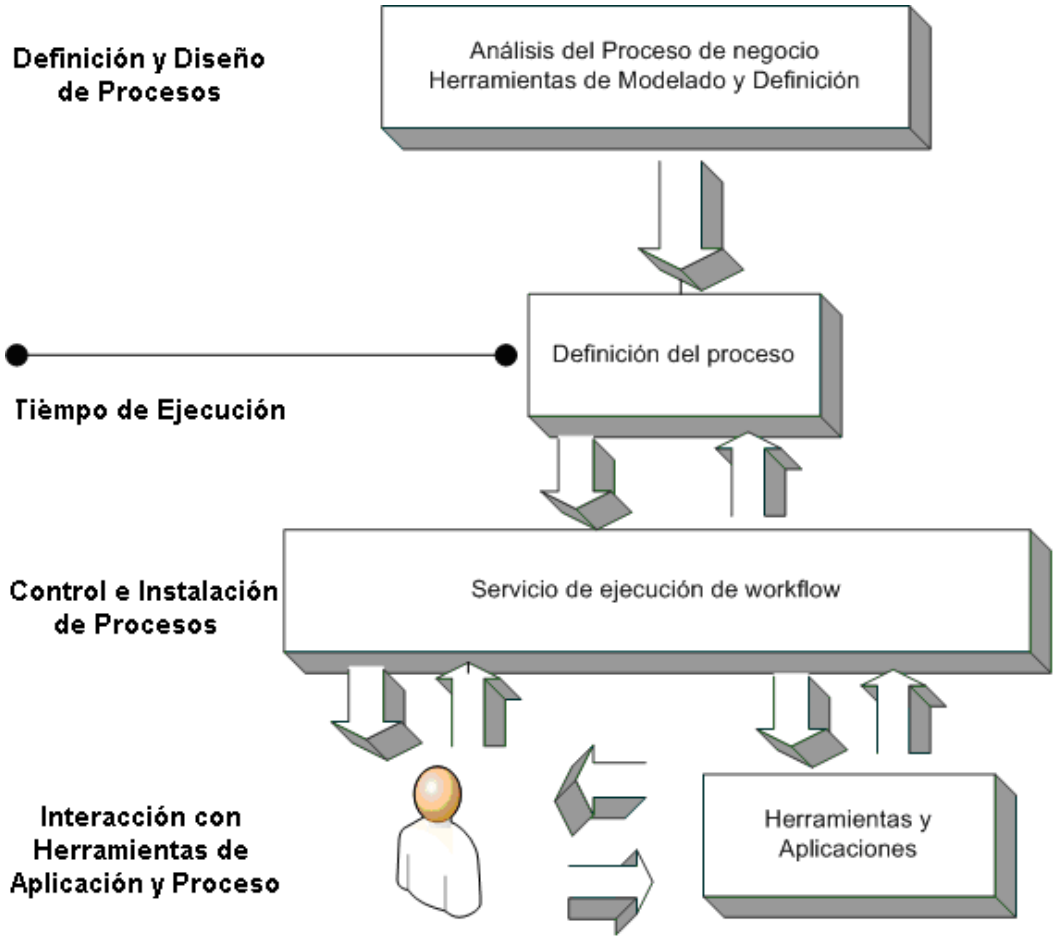

Figura 1. Caracterización de los WMS por la WfMC.

A raíz de esto se han desarrollado una gran diversidad de sistemas de gestión y herramientas de Workflow, que surgieron utilizando diferentes lenguajes y paradigmas. Esto creó la necesidad de proponer un estándar que permita a los desarrolladores de aplicaciones Workflow, seguir un modelo común. En este sentido, la WfMC propuso *El Modelo de Referencia* [14], este modelo intenta reunir las características comunes a cualquier WMS, con el fin de poder alcanzar la interoperabilidad entre ellos a través de estándares comunes para cada una de las funciones que se puedan realizar. Debido a que el objetivo de la WfMC es definir estándares y guías globales para el desarrollo de WMS, su documentación es de carácter genérico y no describe en detalle ningún WMS en particular, sino que brinda un marco general para la construcción de los mismos.

Este modelo de referencia identifica las características comunes de un sistema Workflow y define cinco interfaces discretas funcionales a través de las cuales un sistema Workflow interactúa con su entorno (empleados, herramientas y aplicaciones informáticas, otros servicios de software, etc.) Dichas interfaces están distribuidas como puede verse en la figura 2, donde se presentan los componentes del modelo de referencia WfMC y son descritos a continuación [14].

- El componente central del modelo lo conforma el *Servicio de Ejecución del Workflow,* el cual es el encargado de crear, ejecutar y gestionar cada una de las actividades que expone el flujo de trabajo de acuerdo a las reglas definidas en el proceso de negocio.

- La *Herramientas de Definición del Proceso*, es el componente que permite definir y modelar un proceso de negocio, en este componente se establecen las actividades, sus relaciones, también las reglas de usuarios y roles que estos tienen dentro del proceso de negocio.

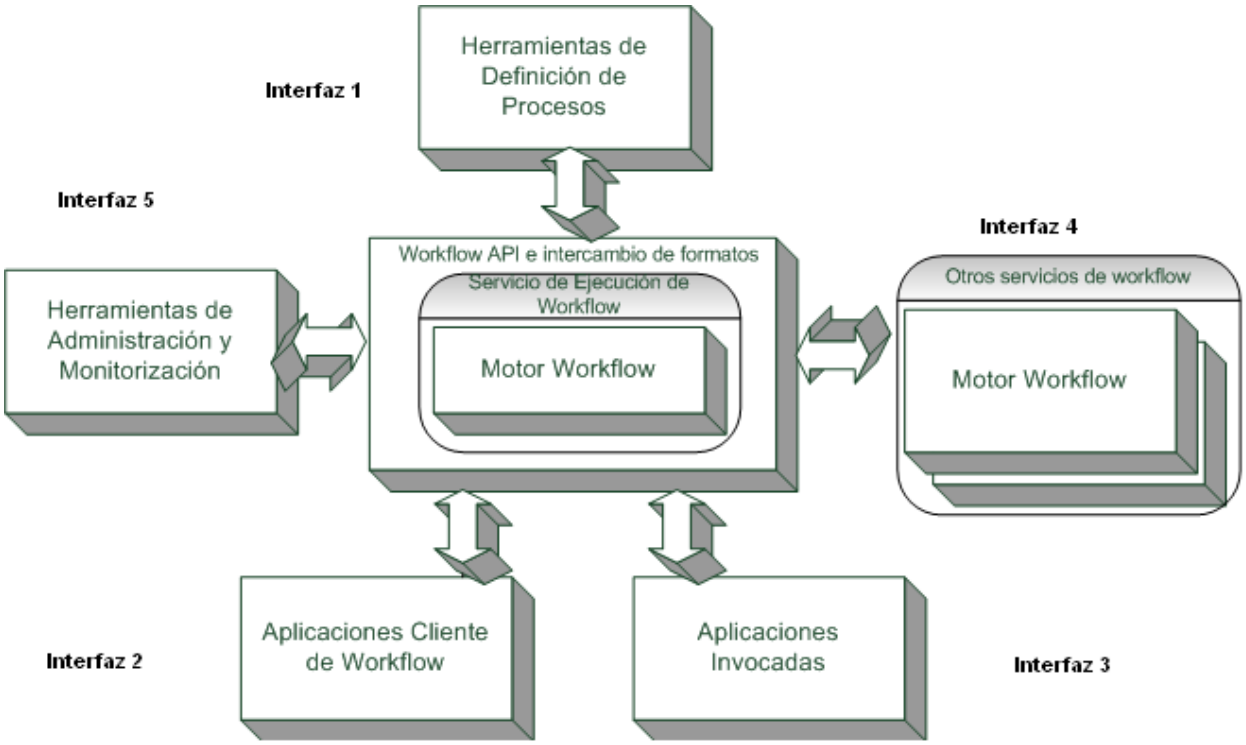

Figura 2. Modelo de referencia de la WfMC.

- El componente de *Aplicaciones Cliente del Workflow*, permite tener una interacción entre los usuarios de la aplicación y el motor de ejecución, con el objetivo de cumplir con las tareas que estos tienen dentro de la ejecución del Workflow.

- El componente de *Aplicaciones Invocadas*, el cual maneja las aplicaciones software que el motor de ejecución puede utilizar para realizar ciertas actividades, con el fin de cumplir con la ejecución del proceso de negocio de acuerdo a las reglas de negocio ya establecidas.

- El sistema WMS puede interactuar con otros motores de ejecución, este tipo de interoperabilidad está representado por el componente denominado *Otros Servicios de Ejecución de Workflow*, que permite conexiones simples de actividades, definición de procesos y datos entre otros.

- Finalmente, el componente *Herramientas de Administración y Monitorización*, nos permite realizar varias tareas de gestión como monitorización del sistema, supervisión del estado de ejecución del proceso y otras muchas más funcionalidades.

2.1.2 Gestión de Procesos de Negocio (Business Process Management - BPM)

BPM [15] surge como la evolución natural de los sistemas de Workflow y de los procesos de negocio de las empresas. Esto es debido a que la evolución del término proceso ha cambiado en el interior de las organizaciones; muchos de los procesos de las empresas actuales no se apoyan solo sobre una aplicación o un conjunto de aplicaciones internas, como sucede con los sistemas de Workflow tradicionales. Por esto BPM comprende todas las actividades que son parte del ciclo de vida de un proceso de negocios, tales como el descubrimiento, diseño, simulación, ejecución, interacción, monitoreo, control, análisis y optimización del proceso de negocios.

BPM esta soportado sobre tecnología de información para automatizar las actividades que son parte del ciclo de vida de un proceso de negocio y dar agilidad a los cambios requeridos por la empresa. La tecnología que posibilita la implantación y adopción de BPM constituye una categoría nueva de sistemas informáticos denominada Sistema de Gestión de Procesos de Negocio (Business Process Management System - BPMS). A diferencia de los sistemas de información tradicionales basados en la gestión de datos, estos sistemas se especializan en la gestión de procesos de negocio.

2.1.2.1 Sistema de gestión de procesos de negocio (*Business Process Management System* - BPMS)

Un BPMS [16] puede ser definido como un conjunto de utilidades de software para definir, implementar y mejorar procesos de negocio que cumplen con un grupo de características técnicas necesarias para aplicar el concepto de BPM.

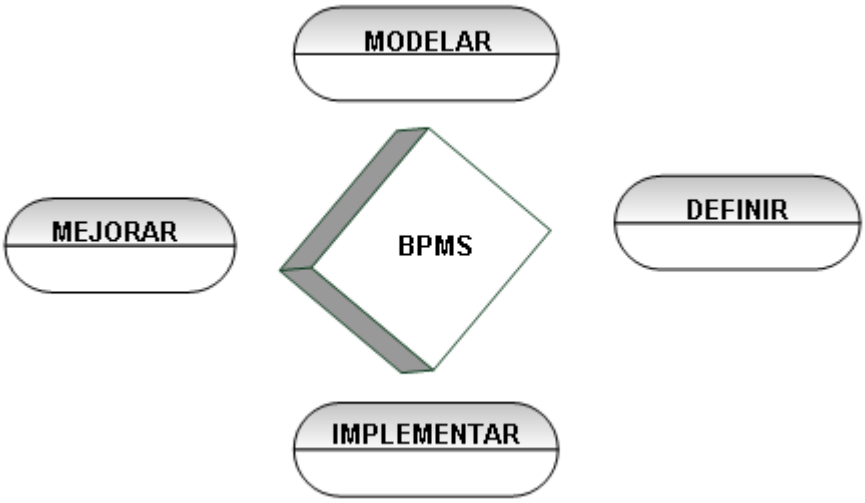

Figura 3. Aspectos de un Sistema de Gestión de Procesos de Negocio.

Estos sistemas permiten manejar el ciclo de vida del proceso a través de características funcionales y no funcionales que posibilitan definir, modelar, implementar y mejorar el proceso durante su operación (Figura 3). Un sistema BPMS está en capacidad de realizar las siguientes operaciones, entre otras:

- Modelamiento de procesos de negocio.
- Generación, actualización y publicación de documentación de procesos.
- Simulación de procesos de negocio para evaluar su comportamiento en situaciones de carga exigidas en determinados momentos del proceso.
- Integración de información proveniente de otros sistemas de negocio.
- Automatización de procesos.
- Colaboración entre las empresas que participan en la cadena productiva de la organización.
- Despliegue de aplicaciones que soportan el proceso, en condiciones tales que no se requiere mayor conocimiento y experiencia de un usuario final.
- Análisis de procesos y comportamiento de la operación.

- Gestión de ciclo de generación, publicación y consumo del conocimiento generado en la operación del proceso.

Estas características constituyen la base sobre la cual se desarrolla el modelamiento, simulación e implementación de procesos en una compañía. La flexibilidad y agilidad en el diseño de procesos, se basan en la abstracción de la realidad que plasma el arquitecto de negocio y las posibilidades del sistema para representar esta realidad de manera gráfica.

2.1.3 Servicios Web

Actualmente los servicios Web juegan un papel importante en la definición y ejecución de procesos de negocio en ambientes distribuidos, por consiguiente, a continuación definiremos algunos de los aspectos necesarios para entender el funcionamiento de los procesos de negocio a través de servicios web.

Inicialmente tenemos que los servicios Web son definidos como aplicaciones software modulares e independientes , las cuales pueden ser descritas, publicadas, localizadas e invocadas a través de una red, estas buscan la interoperabilidad entre distintas aplicaciones [28]. La arquitectura de un servicio Web está compuesta por una serie de protocolos. Entre los más importantes esta SOAP, el cual es un protocolo basado en XML para el intercambio de información en un ambiente distribuido, WSDL que define un esquema XML para la descripción de la interfaz, semántica y administración de llamada a un servicio web y UDDI definen como publicar y descubrir información acerca de los servicios Web.

Dentro de esta arquitectura también encontramos la *composición de servicios* Web, que permite la implementación de procesos de negocio interoperables. La composición de servicios Web implica hallar mecanismos que permitan a dos o más servicios cooperar entre sí para resolver requerimientos que van más allá del alcance de sus capacidades individuales [36], dos términos son comúnmente usados al referirse a composición o colaboración de servicios, estos son *Orquestación de servicios* y *Coreografía de servicios*.

Se considera orquestación de servicios, cuando los servicios Web son controlados por una sola entidad, en este tipo de proceso la entidad que esta orquestando es la única que conoce la información y el flujo del proceso orquestado. En si un solo proceso utiliza y manipula la información de los diferentes servicios, por ejemplo la salida de uno, la puede convertir en la entrada de otro.

La coreografía de servicios se da cuando se define colaboración entre diferentes aplicaciones sin importar la plataforma en que se esté ejecutando, o el lenguaje en que este definida. A diferencia de la orquestación, en la coreografía una entidad no controla a todos los participantes, aquí se define un comportamiento común que todos los participantes deben conocer.

2.1.4 Estándares

Con el fin promover el uso de WMS y BPM mediante el establecimiento de estándares que faciliten su creación, una serie de organizaciones [13], [17] y [18] han unido esfuerzos para definir estándares que logren este fin. Estos se dividen en estándares para la notación de procesos y para la ejecución de procesos, los cuales se describen a continuación.

2.1.4.1 Estándares para la notación de procesos

Tradicionalmente el modelado de procesos de negocio se ha utilizado en la industria para obtener una visión global de los procesos mediante actividades de soporte, control y monitorización y para otras actividades como el procesado automático de documentos [25], para obtener está visión global se han definido unos estándares para proveer una representación gráfica de los procesos de negocio. De estos estándares se destacan dos que son: UML [20] y BPMN [19] los cuales son los más reconocidos en este área.

UML es un lenguaje para especificar, construir y documentar los artefactos que modelan un sistema, se puede utilizar para diseñar aplicaciones basadas en objetos o en componentes de diversos dominios. Los Diagramas de Actividad, los Diagramas de Estado y los Diagramas de Secuencia son utilizados por UML para la descripción del comportamiento dinámico de un sistema. Estos permiten describir flujos de trabajo y procesos de negocio, junto con su lógica procedimental [20], [37], lo que da una herramienta para modelar procesos de negocio.

Sin embargo, en numerosas ocasiones es mejor contar con un lenguaje más específico para modelar y representar los conceptos de ciertos dominios particulares. Por esta razón surge la Notación para el Modelado de Procesos de Negocio (Business Process Modeling Notation - BPMN), de la cual a continuación gráfica se hará una descripción más detallada, debido a que los elementos básicos de esta notación fueron escogidos por el presente proyecto para la representación visual de los diferentes procesos de negocio.

Notación para el Modelado de Procesos de Negocio (Business Process Modeling Notation - BPMN)

Esta notación es una iniciativa clave para representar de manera gráfica los procesos de negocio en un Workflow, fue desarrollada por el grupo de la Iniciativa para la Gestión de Procesos de Negocio (Business Process Management Initiative - BPMI) [17], y es mantenida actualmente por el Grupo para la Gestión de Objetos (Object Management Group OMG) [18]. El principal objetivo de BPMN es proveer una notación estándar que sea fácilmente entendible por todos los usuarios de negocios: para los analistas de negocios que crean los borradores iníciales de los procesos, para los desarrolladores responsables de la implementación, y finalmente para el personal de negocios que administra y monitorea los procesos.

La notación BPMN consiste básicamente en un diagrama llamado Diagrama de Procesos de Negocio (*Business Process Diagram - BPD*), el cual está basado en técnicas de diagramas de flujo para crear modelos gráficos de operaciones de procesos de negocios. Un BPD es creado a partir de un conjunto de elementos gráficos, que presentan diagramas que resulten fáciles de comprender tanto a los analistas de negocio como a los de sistema. Los conjuntos de diagramas BPD se dividen en [21]:

- **Objetos de Flujo**: Consiste solamente de tres elementos principales. Los cuales se describen a continuación:

    - **Eventos (*Events*):** Algo que ocurre durante el transcurso de un proceso de negocio, estos eventos afectan al flujo del proceso y tienen generalmente una causa o un impacto Pueden ser de tres tipos, de Inicio (*Start*), Intermedio (*Intermediate*) y de Finalización (*End*), presentados en la figura 2.

Figura 4. Objetos de flujo BPMN tipo Eventos.

    - **Actividades (*Activity*):** El término genérico para denominar cualquier trabajo que realiza la compañía, una actividad se representa con un rectángulo redondeado como el presentado en la figura 3.

Figura 5. Objeto de flujo BPMN tipo Actividad.

    - **Pasarela (Gateway):** Se usa para controlar la divergencia o convergencia de la secuencia de flujo. Así determina las tradicionales decisiones para la creación de nuevos

caminos, la fusión de estos o la unión. Se representan por la típica figura de diamante (figura 6), los marcadores internos indican el tipo de control de comportamiento.

Figura 6. Objeto de flujo BPMN tipo Pasarela

- **Conectores (*Connecting Objects*):** Los conectores son elementos que sirven para conectar los diferentes objetos de flujo, con el fin de crear el esqueleto estructural básico del proceso de negocio. Estos se describen a continuación.

    - **Flujo de secuencia (Secuence Flow):** Para indicar el orden en el cual son ejecutadas las actividades del proceso de negocio.

Figura 7. Objeto Conector BPMN tipo flujo de secuencia.

    - **Flujo de mensaje (Message Flow):** Para mostrar el intercambio de mensajes entre dos participantes (entidades de negocio o roles).

Figura 8. Objeto Conector BPMN tipo flujo de mensaje.

    - **Asociación (Association):** Para asociar los diferentes objetos del flujo.

Figura 9. Objeto Conector BPMN tipo asociación.

- **Diagramas de Calles (Swinlanes)**: Las calles son un mecanismo que permite clasificar las actividades de manera visual para ilustrar las distintas categorías o responsabilidades. Las distintas clases de calles son:

    - **Piscinas (Pool):** Para indicar los participantes en el proceso.

Figura 10. Objeto Piscina.

    - **Carriles (Lane):** Es una partición de Pool, ya sea vertical u horizontal que nos va a permitir clasificar las actividades.

Figura 11. Objeto Carril.

2.1.4.2 Estándares para la ejecución de procesos

Para la ejecución de procesos de negocio se han definido varios estándares y lenguajes entre los cuales se destacan *Electronic Business XML (ebXML)* [30], *Yet Another Workflow Language* (YAWL) [29][28]**,** Web Services Choreography Description Language (WS – CDL) [43], Business Process Modeling Language (BPML) [27], Web Services Flow Language (WSFL) [44] y Business Process Modeling Language (BPEL) [26].

De estos estándares el presente proyecto de grado selecciono el lenguaje BPEL debido a que 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, todo lo anterior con el fin de poder integrar varias funcionalidades y resolver un objetivo de negocio. Además este lenguaje fue usado con éxito en proyectos como [3], [6], [8] y [11] los cuales fueron bases fundamentales para el desarrollo de este proyecto.

Lenguaje para la ejecución de proceso de negocio (Business Process Execution Language – BPEL)

BPEL es un lenguaje para la ejecución de procesos de negocio [26], que permite definir una notación estándar, para especificar el comportamiento de un proceso basado en servicios. En sus inicios era considerado un lenguaje que solamente manejaba el secuenciamiento de un solo proceso. Donde los procesos de negocios de las empresas eran, menos complejos que los que hoy en día operan, debido a que antes las organizaciones solo interactuaban dentro de su intranet. Posteriormente, las organizaciones comenzaron a definir actividades más complejas en las cuales más de un participante estaba incluido en el proceso de negocio, estos participantes exponían al entorno web lo que podían aportar haciendo uso de servicios web. Es por ello que BPEL migra hacia su versión BPEL para servicios web o como su sigla lo indica, BPEL4WS, el cual es usado para el presente proyecto.

BPEL4WS consiste en un lenguaje de orquestación basado en XML diseñado para el control centralizado de la invocación de diferentes servicios Web, que define el orden preciso en el cual un servicio Web debe ser invocado, generando que los procesos descritos en BPEL4WS exporten e importen funcionalidades usando solo interfaces de Servicios Web.

BPEL4WS permite especificar *procesos abstractos* y *procesos ejecutables*. Los procesos abstractos especifican la secuencia de mensajes que deben ser intercambiados por varias entidades que participan en un proceso de negocio, pero sin revelar aspectos internos de implementación en cada entidad. Es decir los procesos abstractos especifican coordinaciones de servicios Web. Un proceso ejecutable, por el contrario, especifica el comportamiento interno de una entidad, de tal forma que pueda ser ejecutado automáticamente por un motor de ejecución, por lo tanto los procesos ejecutables especifican composiciones de servicios Web.

En la figura 12 se presentan varios componentes definidos en BPEL, cada uno tiene su propia funcionalidad.

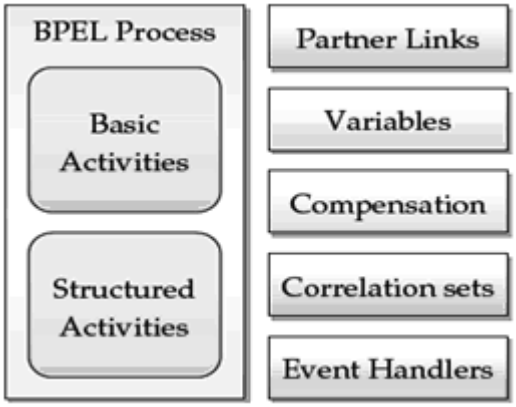

Figura 12. Elementos principales de un proceso BPEL

- **Actividades:** La actividades en BPEL4WS realizan la lógica del proceso, estas representan el tipo de interacciones que se llevan a cabo internamente en la empresa, las cuales se dividen en dos tipos, actividades básicas y estructuradas.

    - **Actividades Básicas:** Las actividades básicas describen pasos elementales del comportamiento del proceso. Estas son:

        - **<receive>:** Espera por la llegada de un mensaje externo de petición para una operación de un servicio Web proporcionado por el propio proceso.

        - **<invoke>:** Invocación de una operación de un servicio Web proporcionado por una entidad externa al proceso.

        - **<assing>:** Asigna un valor a una o más variables.

        - **<wait>:** Permite a un proceso de negocio especificar un tiempo de espera, antes de realizar una acción, por ejemplo, hasta la invocación de un servicio web específico.

        - **<terminate>:** Termina la ejecución del proceso de negocio.

        - **<empty>:** Permite a un proceso de negocio especificar un tiempo durante el cual no se haga nada.

    - **Actividades Estructuradas:** Las actividades estructuradas contienen internamente a otras actividades y simplemente especifican restricciones que aplican a dichas actividades. Estas actividades, permiten estructurar el proceso BPEL ayudando a mantener el flujo del proceso. Algunas actividades estructuradas son:

        - **<sequence>:** Contiene una colección de actividades a ser ejecutadas secuencialmente.

        - **<flow>:** Especifica un conjunto de actividades que deben ser ejecutadas concurrentemente.

        - **<pick>:** La actividad pick espera que ocurra uno de varios eventos y ejecuta la actividad asociada a dicho evento.

- **<while>:** Ejecuta iterativamente una o varias actividades, mientras se cumpla una condición.

- **<switch>:** Permite la ejecución de actividades siempre y cuando se cumplan ciertas condiciones.

- **PartnerLinks**: BPEL permite describir relación entre los socios del proceso (procesos que interactúan con otros). Los servicios Web siempre son modelados como partnerLink, cada proceso BPEL tiene al menos un partner, el cual generalmente tiene que ser un cliente que invoca el proceso. La etiqueta <partnerLinks> es usada para describir los servicios socios.

- **Variables:** Son un conjunto de variables empleadas durante la ejecución del proceso BPEL. Estas se asocian con tipos de mensajes que pueden ser especificados como entradas o salidas para las actividades receive, invoke y reply, por medio de las variables se mantienen los mensajes intercambiados entre socios del negocio, como también mensajes usados solamente dentro del proceso.

- **Compensation:** Proceso por el cual se invierte o se provee una alternativa para restaurar los datos que una actividad hubiera realizado. Por medio de **Compensation** se da marcha atrás a procesos de negocio completados o actividades ejecutadas, cuando ocurren problemas.

- **Correlation Sets:** Permite la asociación de un mensaje entrante con una instancia del proceso, esto es posible gracias a la existencia de identificadores únicos para cada solicitud o mensaje recibido. También permite especificar grupos de operaciones correlacionada en una instancia de un servicio web.

- **Event Handlers:** Cuando un evento específico ocurre, el proceso cambia al código que maneja el evento. Hay dos clases de eventos a los que un proceso BPEL puede reaccionar:

    - Eventos de Mensaje (*Message events*) `<onEvent>`, puede ser comparado a la etiqueta <receive> en la cual se está a la espera de un mensaje entrante.

    - Eventos Alarm (*Alarm events*), `<onAlarm>`, es usada cuando se necesita esperar cierta cantidad de tiempo antes de que algo ocurra.

  Los manejadores de eventos se diferencias de las actividades normales en que ellos son ejecutados solamente cuando ocurre un evento. Si no ocurre un evento el proceso lo abandona y continúa la ejecución.

- **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. Cada rol especifica exactamente un WSDL portType. La etiqueta <partnerLinkType> representa el rol de cada socio del servicio en la comunicación.

- **Port Types**

Es un concepto ligado al estándar WSDL, que muestra las operaciones soportadas y ofrecidas por un 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.

En el resto del presente documento se escribe BPEL para referirse a la versión BPEL4WS que es la utilizada para el presente trabajo de grado.

2.2 TRABAJOS RELACIONADOS

Existen diferentes trabajos que abordan los núcleos temáticos del presente proyecto, donde han presentado propuestas interesantes alrededor de diferentes aspectos que implica la ejecución distribuida de procesos de negocio en sistemas móviles de información.

A continuación se da una descripción de estos trabajos, los cuales brindaron las bases teóricas y prácticas para el desarrollo de este proyecto, estos son agrupados de acuerdo a los temas tratados en el presente trabajo de grado.

2.2.1 Particionamiento de Procesos de Negocio

Uno de los aspectos más importantes en la ejecución distribuida de procesos de negocio en sistemas móviles de información, es el particionamiento, este consiste en dividir un Proceso de Negocio en varios subProcesos sincronizados.

A continuación se presentan los trabajos más relevantes que tratan el particionamiento de procesos de negocio.

2.2.1.1 Particionamiento de Workflow en Sistemas Móviles de Información

En [8] se plantea una metodología para controlar la distribución de un proceso de negocio, por medio del particionamiento de un Workflow en subworkflows, cada uno de estos últimos es asignado a un controlador diferente. En donde se definen unas reglas de particionamiento que transforman un Workflow principal en diferentes subworkflows.

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 [8] y [9] los autores inicialmente plantean una trasformación de la descripción XML de BPEL a un grafo, con el objetivo de trasladar un problema de partición de procesos de negocios a un problema matemático de particionamiento de grafos, a través de las reglas formales establecidas, con el fin de realizar la ejecución de los subworkflows en sistemas móviles de información.

Para el particionamiento, se plantea la inclusión de 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.

La metodología propuesta en esta investigación permitió al presente trabajo de grado establecer y definir los mecanismos y algoritmos necesarios para llevar a cabo el particionamiento y sincronización de procesos de negocio BPEL. Las reglas de particionamiento son presentadas en el capítulo 4 del presente documento, debido a la importancia para el tema tratado en dicho capitulo.

2.2.1.2 Hacia una orquestación distribuida BPEL

En esta investigación [10] se introduce la idea de orquestación distribuida de procesos de negocio, presentando una propuesta que combina BPEL y la ejecución distribuida de procesos en ambientes móviles. Al igual que [8], su principal enfoque esta en el particionamiento, junto con algunos aspectos de sincronización de procesos de negocio a través de un metamodelo que permite representar un proceso de negocio en BPEL para su ejecución en sistemas móviles de información.

Esto es posible gracias a la definición de un metamodelo explícito y a una transformación de grafos, que brindan los fundamentos para obtener el conjunto de subprocesos relacionados y adicionar la

infraestructura de comunicación entre los sub procesos creados**.** La figura 13 presenta los elementos básicos del metamodelo.

El componente clave de todo el metamodelo es un proceso BPEL que consta de varios nodos básicos *Basic Node*, los cuales corresponden a las actividades básicas que se pueden ser ejecutadas por un proceso de BPEL, como invoke y receive. Los nodos estructurados *Estrucutred Node*, son una especialización de los nodos básicos, corresponden a estructuras típicas tales como `switch, flow, sequence,` y `pick`. Cada nodo estructurado se trata por medio de dos nodos básicos para identificar el primer y último elemento de la declaración. Los `Scopes` y `Handlers` son tratados como nodos estructurados de propósito especial ya que ambos integran un conjunto de nodos básicos que conforman las estructuras, es decir `Scopes` y `Handlers` en BPEL crean los vínculos entre los nodos básicos que integran las estructuras.

Por otra parte los nodos están conectados a través de flujos, los cuales se caracterizan por un orquestador que es el encargado de ejecutarlos. Estos orquestadores a su vez, se definen como *FlowLinks* que establecen el orden de ejecución entre los nodos, los *DataLinks* establecen dependencias de datos entre los nodos y los nodos *Dependences* se refieren las actividades básicas para la sincronización de subprocesos (`invoke/receive`).

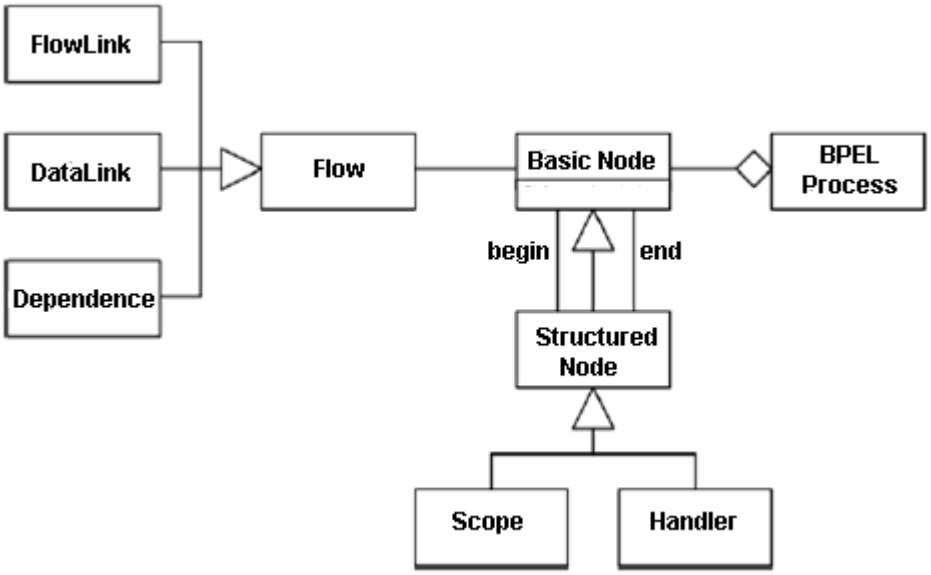

Figura 13. Meta modelo para el particionamiento y sincronización de nodos

La transformación de grafos y las reglas para el particionamiento tratadas en este proyecto, son descritas en [8] y más en detalle para WS-BPEL en [9] . Debido a que esta investigación es considerada una evolución de los resultados obtenidos en [8] y [9], es útil para el presente proyecto de grado, ya que permite definir y aclarar aspectos como particionamiento y sincronización de procesos de negocio en ambientes móviles.

Estos dos trabajos ***Particionamiento de Workflow en Sistemas Móviles de Información (MOBIS)*** y ***Hacia una orquestación distribuida BPEL***, se diferencian del presente proyecto, en que no se tratan aspectos como distribución, monitoreo y ejecución de procesos de negocio en sistemas móviles de información.

2.2.1.3 BDMobIS Arquitectura para la Distribución de Procesos de Negocio en Sistemas Móviles de Información

Este proyecto presentado en [3], [4], [5] y [6] desarrolla una arquitectura para la distribución de procesos de negocio en dispositivos móviles, enfocándose principalmente en la optimización del particionamiento de procesos de negocio, a través de un algoritmo bioinspirado en colonia de hormigas.

Con el fin de realizar este proceso de particionamiento de forma óptima (reduciendo las aristas entre los sub-flujos), se transforma un proceso BPEL a un modelo de grafos. Para realizar dicha transformación en la arquitectura BDMobis se define el módulo *Parser BPEL-Grafos,* que transforma un documento BPEL a un grafo y viceversa.

También se plantea un módulo *Particionador de Grafos* que contiene un algoritmo bioinspirado en colonias de hormigas [3], [4] para realizar el particionamiento, el cual además de particionar permite reducir el número de comunicaciones entre las partes del proceso.

Una vez realizado el particionamiento, se efectúa la sincronización de lo subprocesos resultantes, para mantener el flujo de ejecución del proceso de negocio particionado [28], por lo cual se plantea que este proceso sea realizado por un módulo de sincronización. Donde a cada uno de los subprocesos representados por grafos se le agreguen dos nodos que corresponden a actividades de sincronización (una actividad invoke junto con una actividad receive)

En esta arquitectura también se plantea un módulo de *Distribución,* para enviar a los dispositivos móviles las subpartes del proceso BPEL obtenidas por el particionamiento.

Vale la pena resaltar que esta arquitectura fue punto de partida para el presente trabajo, ya que brinda grandes aportes en el particionamiento-sincronización de procesos de negocio y en la distribución de proceso de negocio en sistemas móviles de información.

Por otra parte, en BDMOBIS no tratan aspectos como el monitoreo, ni profundizan en temas como distribución y ejecución de subprocesos de negocio en sistemas móviles de información, a diferencia del presente trabajado de grado, donde se propone una arquitectura que describe detalladamente cómo tratar el particionamiento, la distribución, ejecución y monitoreo de procesos de negocio de manera integral.

2.2.2 Distribución y monitoreo de los procesos de negocio

Diferentes proyectos plantean el uso de Agentes móviles para el control y ejecución de Workflows y tareas distribuidas. A continuación se describen las investigaciones más importantes para este trabajo de grado, las cuales además de tratar el tema de agentes brindaron bases para la distribución y monitoreo de los procesos de negocio.

2.2.2.1 Arquitectura basada en computación en grilla de un framework colaborativo para dispositivos móviles

En estos trabajos [39] y [40] los autores proponen una arquitectura basada en un modelo de computación en Grilla (Grid Computing), que permite dividir una tarea de gran capacidad computacional en tareas más pequeñas, y distribuir estas pequeñas tareas en varios dispositivos móviles, con el fin de poder realizar un procesamiento más rápido de la tarea principal.

En esta arquitectura se modela cada dispositivo móvil como un agente inteligente y autónomo que facilita la resolución de un problema. Cada dispositivo debe tener una aplicación cliente que implemente un protocolo de comunicación para el intercambio de información con el Sistema Central, en este caso la comunicación se realiza mediante un *Servicio Intermediario* (*Brokering Service*) de la grilla. Cualquier dispositivo que ejecuta esta aplicación cliente es llamado *Subordinado (Subordinate)* y puede pedir su participación en la solución parcial de una tarea de alto grado de complejidad computacional de otro

dispositivo. Un dispositivo móvil que sea *Subordinado* puede llegar a ser un *Iniciador* (*Initiator*) de una tarea distribuida. Este caso se da cuando el usuario necesite ejecutar una tarea de alto grado de complejidad que no puede realizar su dispositivo debido a sus recursos limitados. El *Iniciador* envía esta tarea al *Servicio Intermediario*, el cual junto con otros componentes de la infraestructura, coordina la ejecución distribuida de las tareas y facilita la comunicación entre los dispositivos participantes en las actividades de la grilla.

El *Servicio Intermediario* interactúa con dos importantes repositorios de información planteados por la arquitectura, los cuales son el Repositorio de Agentes Activos (*Active Agent Repository* - AAR) y la *Tabla de localización de Tareas* (*Task Allocation Table* - TAT). El *AAR* contiene información acerca de todos los dispositivos *Subordinados* que están actualmente disponibles dentro de la grilla, almacena información que incluye: cantidad de memoria física disponible, clase de CPU y nivel de batería actual. La *Tabla de localización de Tareas* contiene información de la ejecución distribuida de cada tarea, esta información incluye: código de la tarea, la distribución de la tarea entre los *Subordinados* y resultados parciales disponibles. La información almacenada en la *Tabla de localización de Tareas* cambia dinámicamente cuando la tareas se acerca a su fin, cuando un *Subordinado* completa su tarea parcial respectiva, el envía los resultados al *Servicio Intermediario* para su almacenamiento en la *Tabla de localización de Tareas*.

Este proyecto a diferencia del presente proyecto de grado no trata concretamente procesos de negocio distribuidos, sino manejo de tareas computacionales que son distribuidas para facilitar su procesamiento, a pesar de esto brinda grandes aportes en aspectos como comunicación, distribución de tareas a dispositivos móviles y registro de tareas por medio de repositorios. Los cuales fueron utilizados en el presente trabajo de grado.

2.2.2.2 Arquitectura de Workflow Distribuidos Agentes Inteligentes JITIK

En este artículo [41] se propone la arquitectura de *Workflow llamada* JITIK (Just-in-Time Información and Knowledge) basada en agentes inteligentes que mapea la estructura distribuida de una organización y la descentralización del control del flujo de procesos al nivel de los agentes. Este enfoque promete una mejor escalabilidad y robustez que sistemas centralizados de W*orkflow* basados en agentes.

Para cumplir con este objetivo se definen varios mecanismos que permiten la ejecución descentralizada de procesos. Para ello se plantea el particionamiento de la ejecución del procesos de negocio, en unidades pequeñas que se encuentran en “sitios” de ejecución manejadas por agentes inteligentes (véase Figura 14 y 15), de tal forma que dichos agentes reflejen la estructura y la manera en que se controla el flujo de los procesos de forma descentralizada. Ésta estructura contiene tres tipos de agente que se describen a continuación:

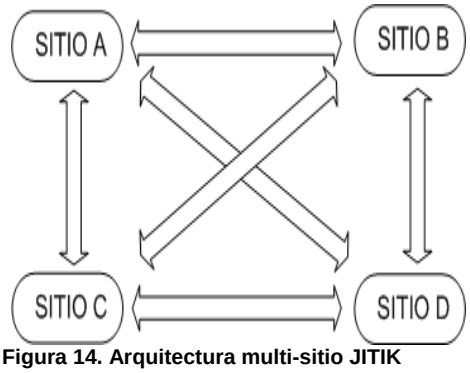

Figura 14. Arquitectura multi-sitio JITIK

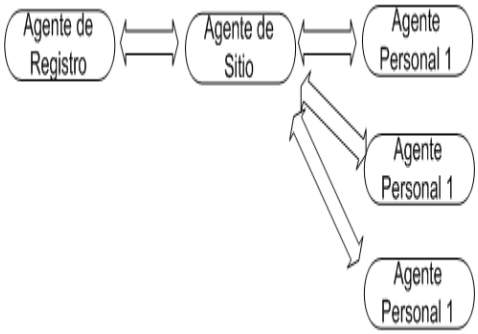

Figura 15. Arquitectura multi-Agente de sitio JITIK

- **Agente Personal:** Este agente ayuda a cada usuario a realizar sus tareas dentro del proceso de negocio, para ello mantiene una lista con las descripciones de las tareas que un usuario puede realizar, es importante destacar que el control de la ejecución del flujo, es llevado a cabo mediante este agente.

- **Agente de Sitio:** Es el encargado de recibir las peticiones de cualquier agente para propagación de mensajes hacia los Agentes Personales, con el objetivo de mantener la ejecución del flujo del proceso descentralizado.

- **Agente de Registro:** En esta arquitectura definen un mecanismo de monitoreo de procesos de negocios descentralizados, el cual está compuesto por un agente que lleva un registro de lo que está sucediendo durante la ejecución de todas las tarea. Las actividades más importantes de este agente son la distribución inicial de tareas, la creación de instancias de procesos y el mantenimiento de un registro de la ejecución de las instancias actuales de procesos.

Según lo anterior, podemos observar que en esta arquitectura el agente de registro tiene asignadas varias tareas importantes en la ejecución descentralizada de un proceso de negocio, entre las cuales se define claramente un mecanismo de registro y monitoreo de las actividades, razón por la cual este mecanismo establece las bases para la definición y el desarrollo del mecanismo de monitoreo de procesos de negocio utilizado en el presente proyecto.

2.2.3 Ejecución de procesos de negocio en dispositivos móviles

Otro aspecto importante tratado en el presente trabajo de grado es la ejecución de los procesos de negocio en los dispositivos móviles. A continuación se da una descripción de investigaciones han trabajado este aspecto.

2.2.3.1 BondFlow: Un sistema para la coordinación distribuida de Workflows sobre servicios Web

BondFlow [33] es 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.

Las características básicas que permite el sistema BondFlow son las siguientes:

- Programación fácil de la configuración del Workflow.

- Encapsulación automática de servicios Web heterogéneos, a través de los proxy wrappers.

- Coordinación distribuida de los Workflows.

Los usuarios de este sistema, pueden configurar sus workflows seleccionando los apropiados servicios Web y enlazándolos para ser dependientes.

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 este proyecto, *BondFlow* no utiliza un motor BPEL de orquestación de procesos de negocio para orquestar los servicios Web, se basan en interconectar los servicios Web a través de enlaces Web y no trata los aspectos particionamiento, distribución y monitoreo.

2.2.3.2 WEBV2

WebV2 [32] es una plataforma que permite la ejecución de subprocesos de negocio independientes que pueden ser usados en dispositivos con pocos recursos computacionales, es escrito en java por lo tanto se puede ejecutar en plataformas heterogéneas incluyendo dispositivos móviles.
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 ya sea embebidos en un dispositivo móvil, envolviendo una aplicación legada o ejecutándose dentro de un servidor de aplicaciones.

El proyecto WebV2 no maneja los aspectos de particionamiento, sincronización, monitoreo tratados en este proyecto, la plataforma WebV2 se centra principalmente en aspectos de ejecución de los procesos de negocio BPEL, por lo cual se considero para este proyecto como una opción para la ejecución de procesos en dispositivos móviles. No se utilizo debido a que es una solución de software propietario y no brinda librería ni fuentes para su uso académico.

2.2.3.3 Motor BPEL para la ejecución de Procesos Workflow en dispositivos móviles SLIVER

Sliver [11] es un Motor BPEL para la ejecución de procesos de negocio, el cual soporta gran variedad de dispositivos que van desde Teléfonos móviles a Computadores de escritorio. Aunque es más utilizado para la ejecución en dispositivos móviles, debido a que fue desarrollado teniendo en cuenta las restricciones que tienen estos dispositivos para ofrecer un tiempo de ejecución razonable.

La arquitectura de Sliver ofrece varias características tales como:

- Tener un adecuado almacenamiento y pequeño espacio en la memoria, incluyendo todas las librerías que conforman el motor.

- Depender sólo de la API de Java que están disponibles en todos los dispositivos.

- Soportar una amplia variedad de medios de comunicación y protocolos. Aspecto que manejan características propias de los dispositivos móviles.

La plataforma inicialmente propone una *capa de transporte (Transport)*. Esta capa envuelve diversos protocolos de comunicación y de red, junto con una interfaz consistente que permite comunicarse con las capas superiores. La *capa de Transporte* es la responsable de la transmisión de los mensajes producidos en las capas superiores, esto lo realiza por medio del intercambio de mensajes de objetos en forma cadenas XML serializadas.

Todos los mensajes intercambiados sobre la *capa de transporte* se codifican en formato XML. Para realizar esto Sliver utiliza las capas *XML Parser y SOAP Parser,* estás hacen uso de las librerías Kxml [47], para la codificación y decodificación de los mensajes, para lo cual el motor Bpel Siver codifica los mensajes de acuerdo con el protocolo SOAP, haciendo uso de las librerías kSOAP [48], que al igual que kXML utilizan poco espacio debido a que son librerías pensadas para la ejecución en dispositivos móviles.

Debido a que Sliver maneja las peticiones y repuestas encapsuladas como objetos SOAP, este proceso es realizado por la capa *SOAP server*. Cuando llegan los mensajes sin serialización desde las capas *XML Parser* y *SOAP Parser*, el *SOAP server* los direcciona al servicio correspondiente. La respuesta del servicio se serializa a través de las capas *XML Parser* y *SOAP Parser* y luego se envía sobre la red a través de la capa *SOAP Server*.

Finalmente tenemos el Servidor BPEL, el cual está representado por la clase *BPELServer*, esta clase aloja los procesos BPEL que el *Bpel Parser* genera, estos procesos usan un esquema XML estandarizado para describir las interacciones entre otros servicios SOAP en donde Sliver representa cada etiqueta de los procesos con una clase Java correspondiente, por ejemplo, la etiqueta <assign> es manejado por la clase Assign*.* Estas tareas son realizadas por la capa *Bpel Parser.*

Por otra parte el *BPELServer* obtiene funcionalidades de *SOAPServer* para invocar estos procesos en lugar de los servicios SOAP. BPELServer también ayuda a cada proceso con el envío y recepción de mensajes SOAP. A diferencia de la SOAPServer, la BPELServer analiza estos mensajes utilizando el kDOM parser.

El proyecto de investigación SLIVER es dirigido principalmente a la ejecución de procesos BPEL en dispositivos móviles, por lo cual se seleccionó este motor para ser integrado en la plataforma propuesta en el presente proyecto, también se tuvo en consideración que está bajo la licencia *GNU Lesser General Public License* y es *OpenSource*, además ha sido probado en diferentes proyectos.

2.2.4 Manejo de workflows distribuidos

La investigación presentada a continuación brinda aportes teóricos sobre cómo trabajar los Workflows ejecutándose de manera distribuida.

2.2.4.1 Workflows Distribuidos: Un framework para comercio electrónico

Este proyecto [35] está enfocado en 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. Describe el diseño del modelo, así como la arquitectura para proveer soporte de transacciones avanzadas de workflows distribuidos, orientado al comercio electrónico inter-empresas, es decir Negocio a Negocio (Bussines-to-Bussines - B2B). Además, soporta estudios de modelos de seguridad para dichos workflows

En este trabajo además se presentan las funciones requeridas para el manejo de operaciones de Workflows en ambientes distribuido, las cuales son nombradas a continuación:

- 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 datos que se requieren 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 plantea maneras de gestionar la ejecución de Workflows en ambientes distribuidos, presentando diferentes consideraciones a tener en cuenta para la ejecución de estos workflows. Pero no tiene en cuenta aspectos como el particionamiento, distribución y ejecución sincronizada en sistemas móviles de información, temas que son tratados a lo largo del presente trabajo.

Resumen

En este capítulo se expusieron los conceptos como Procesos de Negocio Web y Workflows, Gestión de Procesos de Negocio, Servicios Web así como los Estándares para la notación y ejecución de procesos de negocio, los cuales brindaron las bases conceptuales para el desarrollo del presente proyecto.

Igualmente en este capítulo se presentaron los trabajos relacionados a la presente propuesta, agrupados según los aspectos que manejan. De estos trabajos se obtuvo lo siguiente:

- Particionamiento de Procesos de Negocio

  Los trabajos presentados en la sección de particionamiento de procesos de negocio [3], [4], [5], [8], [9] y [10], permitieron identificar y definir mecanismos para dividir un proceso de negocio BPEL en varios subprocesos BPEL sincronizados, que representan el proceso original.

- Distribución y monitoreo de los procesos de negocio

  Los trabajos que manejan la distribución y monitoreo de procesos de negocio[39], [40] y [41], brindaron aportes en aspectos como, distribución de tareas y registro de tareas los cuales permiten definir como realizar la distribución y el monitoreo de los subproceso particionados.

- Ejecución de procesos de negocio en dispositivos móviles

  Las diferentes investigaciones que manejan la ejecución de procesos de negocio BPEL en dispositivos móviles [11], [32] y [33]establecen maneras de realizar la ejecución de procesos BPEL en la plataforma, de estos se selecciono la propuesta [11] por las diferentes características que posee.

- Manejo de workflows distribuidos

  Finalmente el proyecto presentado en [35] permitió aclarar aspectos sobre maneras de gestionar la ejecución de Workflows en ambientes distribuidos.

En el siguiente capítulo se presenta la arquitectura que da soporte a la plataforma propuesta en el presente trabajo de grado, en la cual también se referencian algunos de estos trabajos que ya que establecieron las consideraciones previas a definición de la arquitectura.

Capítulo III

ARQUITECTURA DE LA PLATAFORMA MobFlow

En este capítulo se presenta y describe la arquitectura que da soporte a la plataforma desarrollada, y los resultados del proceso de ingeniería llevado a cabo para este desarrollo, junto con los artefactos más importantes obtenidos en dicho proceso.

La plataforma propuesta en el presente trabajo de grado titulada **MobFlow** (Mobile Flow), está compuesta de una aplicación Web y una aplicación Móvil. La primera realiza la carga de procesos de negocio empresariales, con el fin de particionarlos y generar subprocesos, los cuales son distribuidos entre los dispositivos móviles de los actores que juegan un rol dentro de los procesos de negocio. La aplicación web también permite gestionar la información de los participantes o actores y además brinda un soporte para la ejecución distribuida de los subprocesos en los dispositivos móviles, junto con una interfaz para monitorear esta ejecución. La segunda aplicación llamada **cliente móvil MobFlow**, permite la ejecución desde los dispositivos móviles de las actividades que componen los subprocesos.

Para el desarrollo de las aplicaciones software que conforman la plataforma propuesta en el presente trabajo de grado, se utilizó la metodología propuesta en el articulo *"Desarrollo de Aplicaciones WEB basadas en Servicios WEB XML. Un Caso Práctico"* [37].

La metodología mencionada consta de tres (3) etapas principales, Modelo Conceptual, Arquitectura, Diseño e Implementación, las cuales a su vez constan de subetapas. Para este proyecto la etapa de Diseño e implementación se desarrolló en iteraciones incrementales incluyendo actividades de Verificación y Validación al final de cada iteración.

Para el desarrollo del proyecto se plantearon las siguientes iteraciones:

- **Iteración 1:** Inicio de sesión, Registro de usuarios, Implementación de los módulos para la gestión de datos: Actores, Roles, Usuarios del sistema, Verificación y Validación.
- **Iteración 2:** Implementación de un módulo para el particionamiento de los procesos de negocio, Vinculación de actores, Verificación y Validación.
- **Iteración 3:** Desarrollo del cliente móvil: Inserción de motor BPEL en dispositivo móvil, Implementación de la aplicación para dispositivo móvil, Interacción motor BPEL con Aplicación para dispositivos móviles, Verificación y Validación.
- **Iteración 4:** Implementación de los mecanismos para control de la ejecución de los procesos distribuidos en los dispositivos móviles, Implementación de protocolo de comunicación entre aplicación móvil y aplicación Web, Implementación mecanismos de Invocación de Servicio Web, Verificación y Validación.
- **Iteración 5:** Desarrollo del módulo de visualización y monitoreo de procesos de negocio, Verificación y Validación.

3.1 MODELO CONCEPTUAL

En esta etapa se obtuvieron las bases conceptuales, la investigación de alternativas y demás actividades que permitieron estructurar el trabajo a desarrollarse en las siguientes fases, ésta estuvo compuesta de cuatro (4) actividades base: Investigación documental, Captura y análisis de requisitos funcionales, Identificación de Casos de Uso y Construcción del Esquema Conceptual.

3.1.1 Investigación Documental

A partir de la investigación documental se presento la información acerca de las bases teóricas y los trabajos relacionados del capítulo anterior, de estas investigaciones surgieron diferentes consideraciones a tener en cuenta en el desarrollo de la plataforma. Estas consideraciones se describen a continuación:

Consideraciones previas a la definición de la arquitectura y desarrollo de la plataforma

Particionamiento, Distribución y Ejecución móvil

En la arquitectura propuesta en [3], los autores definieron una aproximación a la distribución de procesos de negocio en sistemas móviles de información considerando varios prototipos para validar la arquitectura propuesta [4], [5], [6]. De dicho trabajo se tuvieron en cuenta las siguientes consideraciones para la definición de la arquitectura:

- Distribuir las actividades de un proceso de negocio descrito en BPEL, entre los dispositivos móviles de los actores siguiendo el enfoque propuesto en [8], [10][1] de manera automática, requiere inicialmente mecanismos de manejo de grafos, los cuales son la base para dividir procesos de negocio descritos en BPEL en varios subprocesos, ya que para muchas aplicaciones de cálculo científico, el problema de descomponer la ejecución de un proceso entre varios procesadores se puede describir convenientemente usando la teoría de grafos [33].

  Por consiguiente se considera que la plataforma debe contar con un módulo que permita transformar un documento BPEL a un modelo de grafos, y a partir de este modelo de grafos realizar el particionamiento del proceso. En este caso el grafo que representa un documento BPEL es particionado con el ánimo de obtener subgrafos, cuyos nodos corresponden a tareas a ejecutar por los actores que intervienen en el proceso de negocio.

  Dado que los subgrafos son transformados a subprocesos BPEL y distribuidos a los respectivos dispositivos móviles de los actores en mención, la plataforma debe contar con un módulo para realizar la distribución de dichos subprocesos.

- En [3] los autores proponen la inclusión de un motor BPEL que facilita la ejecución de dichos procesos desde dispositivos móviles. Tomando en cuenta esta consideración el presente proyecto de grado plantea el desarrollo de una aplicación para dispositivos móviles (tal como lo aconseja [14]) que permita al usuario interactuar con el motor BPEL, con el ánimo de ejecutar los subprocesos que han sido distribuidos.

Sincronización

De lo presentado en [8], [9] y [10] se tiene que:

El proceso de particionar un documento BPEL descrito en un modelo de grafos, requiere de reglas especiales de particionamiento, según las actividades estructuradas que posea el documento BPEL.

- Con el fin de sincronizar la ejecución de estos subprocesos, la plataforma debe tomar en consideración actividades de sincronización que permitan conservar una correcta ejecución del flujo de control del proceso de negocio.

- Los Workflows no tiene variables globales y todas las variables son pasadas por parámetros entre diferentes actores, por consiguiente se plantea que los diferentes datos de las variables usadas en la ejecución de los procesos de manera distribuida, deben estar disponible para todos los actores del proceso.

Monitoreo

También es importante, como se describe en el modelo de referencia de la WfMC en la interfaz 5 [14], manejar mecanismos de monitoreo del proceso, en este caso la plataforma debe monitorear la ejecución distribuida del proceso de negocio en los dispositivos móviles de los participantes.

[1] Enfoque seguido por el presente trabajo de grado

La plataforma debe manejar un mecanismo de registro a medida que se ejecutan las actividades desde los dispositivos móviles, para realizar el monitoreo de la ejecución distribuida, como se plantea en [41].

Comunicación

La aplicación para los dispositivos móviles, debe implementar un protocolo de comunicación para el intercambio de información con el Sistema Central como se plantea en [39] y [40]. De aquí también se obtiene que se deben manejar todos los aspectos de comunicación de entre los móviles y el sistema central a través de un solo módulo.

3.1.2 Captura y Análisis de Requisitos Funcionales

3.1.2.1 Alcance del sistema

La plataforma MobFlow proporciona a las empresas una herramienta para el monitoreo y la ejecución distribuida de procesos de negocio, en los dispositivos móviles de los actores involucrados en los procesos de la organización. A continuación se presenta el alcance del sistema para cada una de las dos aplicaciones software que conforman la plataforma.

MobFlow

Es una aplicación Web que permite cargar procesos de negocio empresariales descritos en la especificación BPEL, de estos procesos se puede generar subprocesos BPEL con las tareas correspondientes a los actores, para su posterior distribución.

Los subprocesos generados deben ser vinculados con los actores de acuerdo al rol que éstos desempeñan en el proceso, solo es posible vincular un actor por cada subproceso generado, no se puede asignar a un actor dos o más subprocesos.

Estos procesos de negocio deben ser definidos en el lenguaje para la ejecución de procesos de negocio WSBPEL 2.0 - BPEL 2.0 [26] en formato BPEL ejecutable, y deben contar con el archivo WSDL en la versión 1.1 (WSDL 1.1) [23] correspondiente al proceso BPEL, la especificación de cómo deben estar formado estos archivos para el correcto funcionamiento en la plataforma es descrita en el **anexo C**.

Los procesos cargados en la plataforma se pueden visualizar de manera gráfica mediante una interfaz Web, que muestra la representación gráfica de los procesos BPEL utilizando los elementos básicos de la notación BPMN. La plataforma no soporta características avanzadas de los procesos BPEL, como *Serializable Scopes* y *Compensation,* debido a que el motor BPEL móvil utilizado para la ejecución de los subprocesos generados tampoco las soporta.

Si el proceso BPEL está mal formado, o posee actividades que no son soportadas por la plataforma no podrán ser cargados. Las actividades soportadas por la plataforma MobFlow son presentadas en el **anexo C**. Es decir la plataforma solo soporta procesos con la estructura que se presentan en dicho anexo.

El sistema permite monitorear la ejecución global del proceso en los dispositivos móviles, visualizando la información de las actividades ejecutadas. Esta información corresponde a: Nombre de la actividad, subprocesos al cual pertenece, actor, fecha y hora de la ejecución, además muestra un diagrama básico del proceso que señala las actividades ejecutadas.

En la plataforma MobFlow, todas las actividades definidas en procesos de negocio, son ejecutadas solamente en los dispositivos móviles de los actores involucrados en el proceso. Los subprocesos de negocio BPEL generados por la aplicación Web, solamente podrán ser ejecutados desde dispositivos móviles, junto con todas las tareas o actividades definidas en ellos.

La plataforma requiere que los servicios Web definidos para la composición del proceso de negocio estén desplegados como Java Web Service (jws) [23].

Cliente móvil MobFlow

El cliente móvil provee los mecanismos para descargar los correspondientes subprocesos de negocio a los dispositivos móviles de los actores, y permite ejecutar sus actividades descritas en el proceso a través de interfaces gráficas básicas generadas a partir de las actividades definidas en el subproceso de negocio obtenido.

Un actor del proceso en el dispositivo móvil solo puede ejecutar un proceso a la vez y los procesos se ejecutan según el orden establecido en la definición del proceso BPEL general. No es posible ejecutar un subproceso de negocio si este requiere información de la ejecución de otro subproceso, y esté ultimo no ha realizado o terminado su ejecución.

La aplicación cliente móvil MobFlow, debe ser instalada de manera manual en los dispositivos móviles de los actores de los procesos de negocio.

Respecto a los usuarios que interactúan con la plataforma, se pueden distinguir los siguientes tres:

- Administrador del sistema: Representa la persona encargada de administrar toda la plataforma MobFlow, este usuario crea, modifica y elimina los diferentes usuarios, carga a la aplicación los procesos empresariales, genera los subproceso, y si es el caso ejecuta desde el sistema móvil de información procesos de negocio.

- Usuario general: Representa la persona que tiene privilegios de consulta en la plataforma, como visualizar los diferentes procesos cargados y también visualizar las actividades ejecutadas (monitoreo).

- Actor del proceso de negocio: Representa la persona encargada de ejecutar determinado subproceso de negocio desde el dispositivo móvil, es decir son los diferentes actores del proceso de negocio.

3.1.2.2 Requisitos funcionales

Los requisitos funcionales identificados para la plataforma son los siguientes:

Cargar procesos de negocio a la plataforma. La plataforma debe permitir al usuario cargar los procesos de negocio de las empresas, descrito en BPEL y con su respectivo WSDL.

Visualizar el proceso de negocio: Se debe poder visualizar de manera gráfica los procesos de negocio cargados en la plataforma.

Generar subprocesos BPEL a partir de un proceso BPEL. Debe permitir generar diferentes subprocesos representados por documentos BPEL, a partir de los procesos BPEL cargados en la plataforma y estos subprocesos deben contener las actividades propias de los actores involucrados.

Gestionar los usuarios de la plataforma. Este requisito funcional debe permitir crear los usuarios que pueden acceder a la plataforma y establecer sus privilegios.

Gestionar los actores de la plataforma. Este requisito funcional debe permitir crear los diferentes actores encargados de la ejecución de los subprocesos, también considerados como participantes de los procesos empresariales y que interactúan con la plataforma mediante sus sistemas móviles de información.

Vincular los actores a los subprocesos generados. Debe permitir vincular los actores de los procesos de negocio a sus respectivos subprocesos.

Descargar el subproceso de negocio en los dispositivos móviles. Debe permitir al actor descargar su respectivo subproceso de negocio en su sistema móvil de información para ser almacenado y ejecutado cuando el usuario lo requiera.

Ejecutar el proceso de negocio desde el dispositivo móvil. Este requisito funcional debe permitir al actor de determinado proceso de negocio, ejecutar sus actividades correspondientes al proceso empresarial desde su sistema móvil de información.

Eliminar proceso de negocio almacenado en el dispositivo móvil. Este requisito funcional debe permitir al actor usuario del dispositivo móvil, eliminar el proceso de negocio almacenado es su dispositivo, en caso de que necesite actualizar el proceso o ejecutar otro proceso de negocio.

Ejecución de los procesos distribuidos. La ejecución distribuida los subprocesos de negocio en los sistemas móviles de información debe realizarse en directa concordancia con la definición del proceso general.

Monitoreo de la ejecución distribuida del proceso de negocio global. Con respecto al monitoreo, debe presentar información acerca del estado actual de la ejecución, a medida que cada actor ejecuta sus actividades desde los sistemas móviles de información.

3.1.3 Identificación de Casos de Uso

En esta sección se presentan los diagramas de casos de uso y su descripción para las dos herramientas software desarrolladas en el presente trabajo de grado, MobFlow y cliente móvil MobFlow. Estos casos de uso se identificaron a partir de las funcionalidades que ofrece el sistema a los diferentes actores, descritos anteriormente.

A continuación se presentan los diagramas de casos de uso, junto con una breve explicación de ellos, los caso de uso en formato extendido y formato de caso de uso real se presentan en el **anexo A.**

3.1.3.1 Diagramas de casos de Uso Plataforma MobFlow

3.1.3.1.1 Diagrama de Casos de Uso MobFlow

En la figura 16, se presentan los diagramas de casos de uso para MobFlow, a continuación se realiza una breve descripción de ellos.

Iniciar Sesión: A través de él, los diferentes actores del sistema tienen la posibilidad de ingresar a la plataforma, para realizar las diferentes tareas según los privilegios de usuario.

Gestionar Información: Este caso de uso permite al administrador del sistemas realizar la adición, modificación, consulta y eliminación de la información de los usuarios, actores y roles que maneja la plataforma.

Gestionar Usuarios: En este caso de uso el usuario administrador puede crear, consultar, editar y eliminar los diferentes usuarios que tienen acceso a la plataforma vía Web.

Gestionar Actores: Este caso de uso permite al usuario administrador de la plataforma crear, consultar, editar y eliminar los diferentes actores o participantes de los procesos de negocio que maneje la organización.

Gestionar Roles: Este caso de uso permite al usuario administrador de la plataforma consultar y editar los diferentes roles de los actores que ejecutan los procesos de negocio.

Figura 16. Diagrama de Casos de Uso MobFlow

Visualizar Proceso: Mediante este caso de uso, los diferentes usuarios pueden visualizar los diferentes procesos de negocio cargados en la plataforma, de una manera gráfica.

Monitorear Proceso: Este caso de uso permite a los diferentes usuarios de la plataforma, visualizar de manera gráfica y textual el estado de ejecución del proceso de negocio, de acuerdo a la ejecución realizada por los diferentes actores.

Control de Ejecución: El fin de este caso de uso es el de permitir controlar la ejecución de los procesos distribuidos.

Distribuir: Este caso de uso, permite distribuir los subproceso de negocio a los dispositivos móviles para que los actores puedan ejecutarlos posteriormente.

Sincronizar: Este caso de uso, permite sincronizar los subprocesos ejecutados en los dispositivos móviles, de forma que la ejecución de los subprocesos corresponda a la ejecución del proceso global.

Monitorear: Este caso de uso, permite llevar un monitoreo de las actividades que han sido ejecutadas en los dispositivos móviles.

3.1.3.1.2 Diagrama de Caso de Uso cliente móvil MobFlow

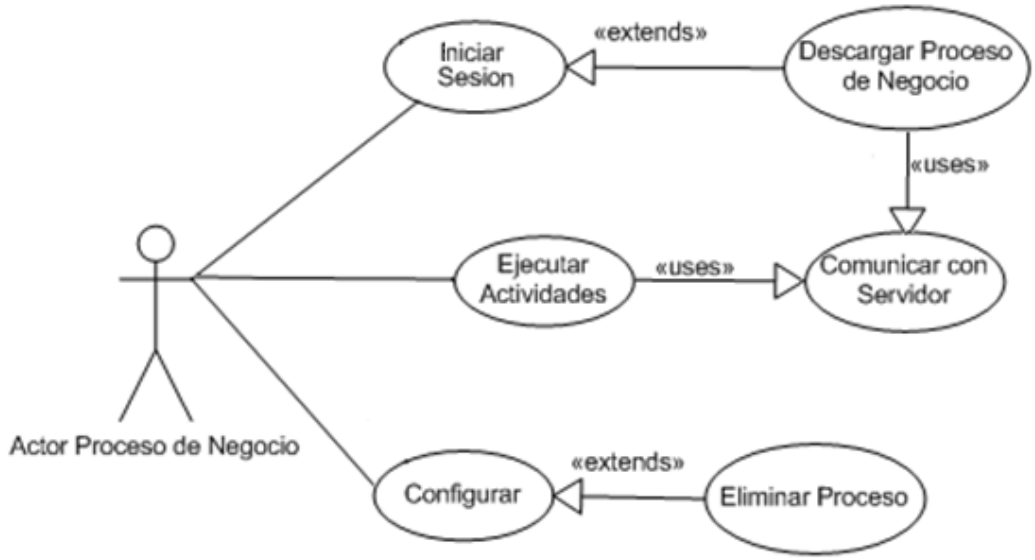

Figura 17. Diagrama de casos de cliente móvil MobFlow.

En la figura 17, se presentan los diagramas de casos de uso para la aplicación cliente móvil MobFlow, a continuación se da una descripción de ellos.

Iniciar Sesión: Con este caso de uso, el usuario actor ingresa sus datos en el sistema móvil de información para validar sus datos en la aplicación Web MobFlow.

Descargar Proceso de Negocio: Con este caso de uso, el usuario actor de determinado proceso de negocio accede a la plataforma una vez validados sus datos y obtiene su respectivo proceso de negocio descrito en BPEL, junto con el archivo WSDL del proceso de negocio en general. Este proceso queda almacenado de manera persistente en el dispositivo móvil.

Ejecutar Actividades: Este caso de uso permite al usuario, realizar las actividades correspondientes a la ejecución del proceso de negocio BPEL (sus tareas del proceso de negocio), a partir del proceso descargado.

Comunicar con Servidor: Este caso de uso permite realizar una conexión y comunicación con el servidor donde está alojada la aplicación Web MobFlow.

Configurar: El usuario puede realizar las diferentes opciones de configuración disponibles en la plataforma.

Eliminar Proceso Mediante este caso de uso, el usuario actor del proceso de negocio elimina el subproceso de negocio almacenado en el dispositivo móvil.

3.1.4 Construcción del Esquema Conceptual

La construcción del esquema conceptual de la plataforma MobFlow, se realizo usando los diagramas de clase, diagrama de paquetes y diagramas de secuencia.

Estos diagramas como los anteriores artefactos se presentan para las dos aplicaciones desarrolladas en el presente proyecto de grado que son: La aplicación Web MobFlow y la aplicación cliente móvil MobFlow.

3.1.4.1 Diagramas de clases y Diagramas de paquetes plataforma MobFlow

En esta sección se presentan los diagramas de clases, a estos se le ocultaron los atributos y métodos para poder ser presentado de forma legible en este documento. También en esta sección se presentan los diagramas de paquetes, los cuales definen el agrupamiento de las diferentes clases y reflejan la organización de módulos de software dentro del entorno del desarrollo.

La descripción de las clases se presenta en los diagramas de paquetes, donde también se describe las funciones que realiza cada paquete.

3.1.4.1.1 Diagrama de clases para MobFlow

El diagrama de clase de la figura 18 presenta las clases implementadas en la aplicación Web MobFlow. Es importante mencionar que el diseño de la aplicación se basa en dos patrones de diseño: el patrón *Session Facade* [47] y el patrón *Objeto de Acceso a Datos* (DAO) [50] y la aplicación Web se implementa utilizando la tecnología de Java Servlet Faces, por lo que sigue el Modelo Vista-Controlador [51].

Las clases cuyos nombres terminan en –***Bean***, implementan la lógica de las interfaces de usuario, encargadas de brindar las funciones permitidas a los usuarios dentro de la plataforma.

Las clases con nombres que terminan en **-Impl**, representan las diferentes clases que manejan la lógica de la aplicación para funcionalidades como monitoreo, particionamiento entre otras. Estas clases a su vez se relacionan con las clases de lógica de presentación y persistencia de datos.

Finalmente, las clases de persistencia que contienen los datos obtenidos de una fuente de datos, o los datos que serán almacenados en una fuente de datos, están definidas con el nombre de la tabla de la base de datos. El uso de estas clases se realiza en conjunción con las clases que hace las implementaciones concretas para el acceso a una fuente de datos determinada, el nombre de estas clases terminan en **–Dao**.

Por ejemplo la clase **ProcesodeNegocioImpl**, se asocia con una clase **ProcesodeNegocio** y **ProcesodeNegocioDao** para el acceso a datos, con esto se logra una separación entre la lógica de la aplicación y el acceso a datos. Estas clases asociadas no son presentadas en el diagrama por razones de espacio y legibilidad.

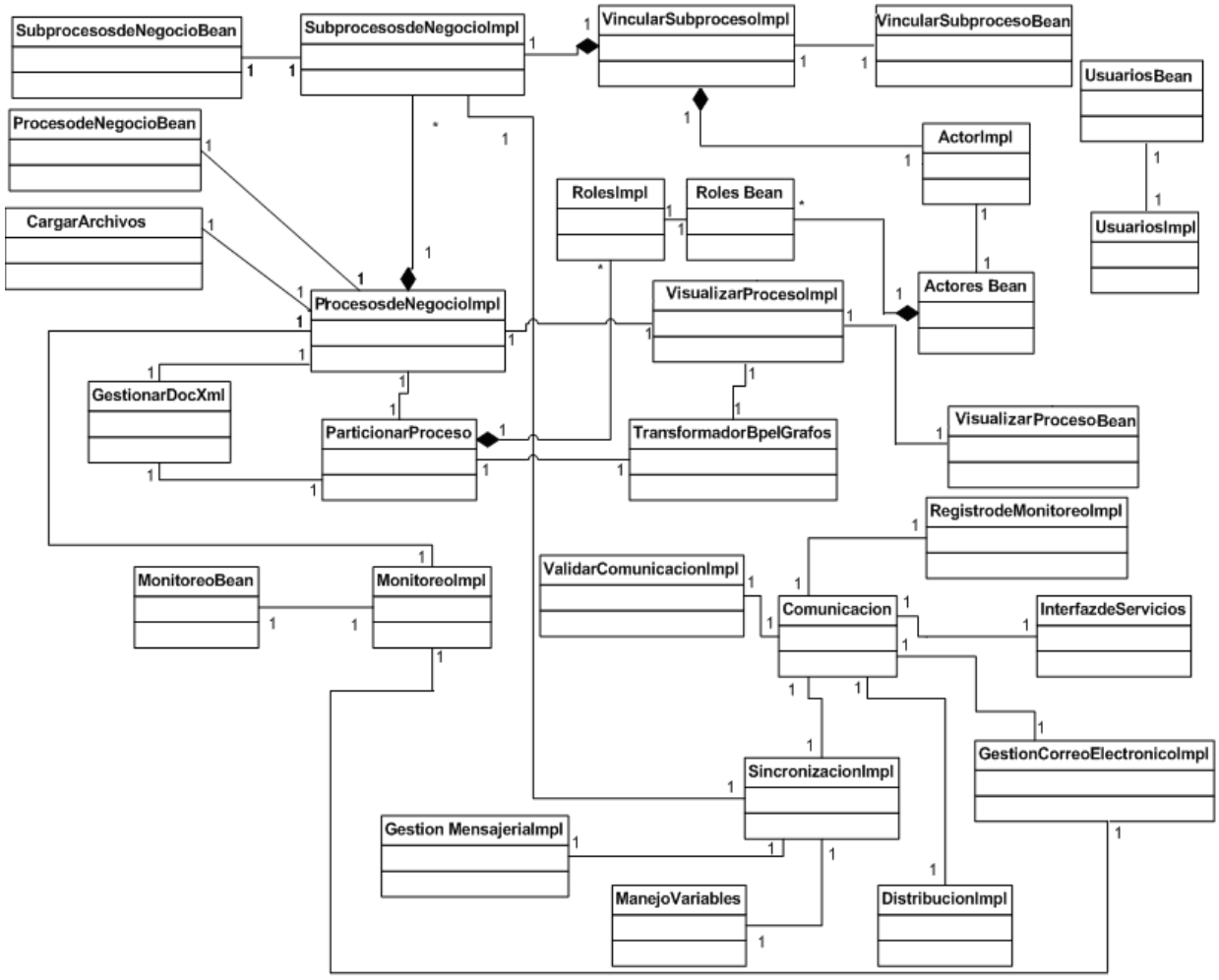

Figura 18. Diagrama de clases aplicación Web MOBFLOW.

3.1.4.1.2 Diagrama de Paquetes MobFlow

El diagrama de paquetes muestra la agrupación de las clases que a la vez reflejan la organización de subsistemas dentro del entorno de desarrollo. Los subsistemas se organizan en capas jerárquicas, y cada capa proporciona una interfaz bien definida a sus capas superiores, en la figura 19 se presenta el diagrama de paquetes general de MobFlow. A continuación se describen los paquetes junto con las clases contenidas.

Capa de Presentación

- Carga de Procesos Este paquete contiene las clases de lógica de presentación encargadas de gestionar los procesos de negocio. Compuesto por la siguiente clase:

    - *ProcesodeNegocioBean*: Esta clase presenta una interfaz gráfica de usuario para realizar la gestión de Procesos de Negocio, con esta se puede cargar, listar, y eliminar los documentos de los procesos BPEL.

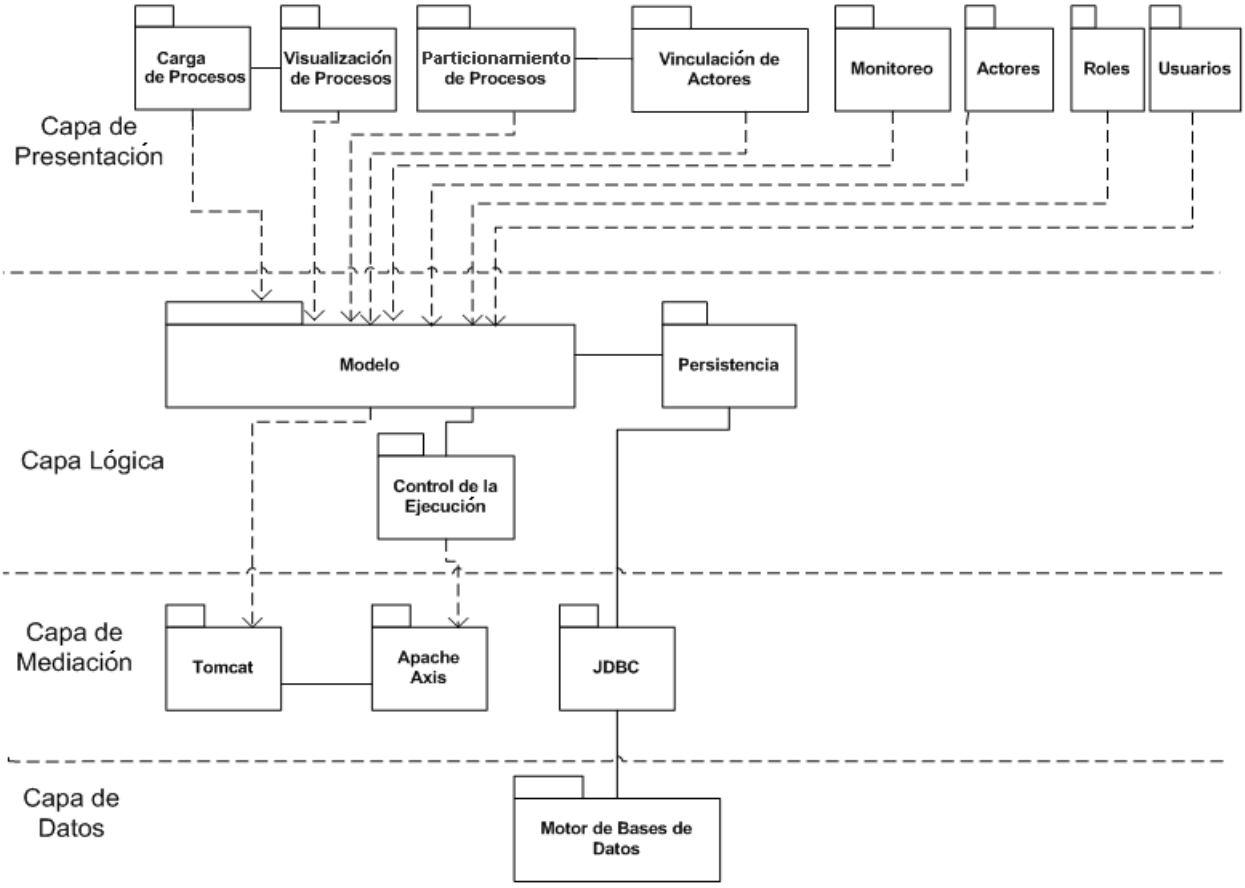

Figura 19. Diagrama de paquetes de la aplicación Web MobFlow.

- Visualización de Procesos: Este paquete contiene las clases de lógica de presentación encargadas de mostrar de forma visual los procesos de negocio. Compuesto por la siguiente clase:
    - *VisualizarProcesoBean*: Esta clase permite visualizar los procesos de negocio cargados en la plataforma.
- Particionamiento de procesos: Este paquete contiene las clases de lógica de presentación encargadas de particionar los procesos de negocio, está compuesto por la siguiente clase:
    - *SubProcesosdeNegocioBean*: Esta clase implementa las funcionalidades de mostrar los Subprocesos generados en el particionamiento.
- Vinculación de actores: Este paquete contiene las clases de lógica de presentación encargadas de vincular los actores con los subprocesos generados en el particionamiento, contiene la siguiente clase:
    - *VincularSubprocesoBean*: Esta clase implementa una interfaz gráfica de usuario que permite vincular los actores con los subprocesos generados.
- Monitoreo: Este paquete contiene las clases de lógica de presentación encargadas de mostrar el monitoreo de los procesos ejecutados, contiene la siguiente clase:
    - *MonitoreoBean*: Esta clase permite administrar las funcionalidades básicas de visualización y monitoreo de los procesos en ejecución.

- Actores: Este paquete contiene las clases de lógica de presentación encargadas de manejar la información de los actores, contiene la siguiente clase:

    - *ActoresBean:* Esta clase implementa una interfaz gráfica de usuario para realizar las funcionalidades básicas de crear, editar, consultar y eliminar los actores de los procesos de negocio.

- Roles: Este paquete contiene las clases de lógica de presentación encargadas de manejar la información de los Roles, contiene la siguiente clase:

    - *RolesBean:* Esta clase implementa una interfaz gráfica de usuario para realizar las funcionalidades básicas consultar y editar los roles de los procesos de negocio.

- Usuarios: Este paquete contiene las clases de lógica de presentación encargadas de manejar la información de los usuarios de la plataforma, contiene la siguiente clase:

    - *UsuariosBean:* Esta clase implementa una interfaz gráfica de usuario para realizar las funcionalidades básicas de crear, editar, consultar y eliminar los usuarios de la plataforma.

Capa Lógica

- Modelo: Este paquete contiene toda la lógica que permite realizar las funciones relacionadas con los procesos de negocio, como son la visualización de estos, el particionamiento de procesos y la vinculación de los actores con los subprocesos generados en el particionamiento. Las clases contenidas en este paquete son:

    - *ProcesodeNegocioImpl:* Clase encargada de realizar la carga de archivos del proceso, guardar la información del proceso, consultar y llamar a otras clases para desplegar, monitorear y particionar los procesos de negocio.

    - *CargarArchivos:* Clase que implementa la lógica necesaria para cargar (subir) archivos a la plataforma y almacenarlo en un directorio específico.

    - *VisualizarProcesoImpl:* Clase encargada de generar una imagen de monitoreo o visualización de los procesos de negocio cargados en la plataforma.

    - *TransformadorBpelGrafos:* Encargada de transformar un documento BPEL en una notación de grafos. Esta clase es de vital importancia ya que facilita los procesos de visualización, particionamiento y validación de procesos de negocio.

    - *ParticionarProceso*: Encargada de manejar la lógica de la aplicación para el particionamiento de procesos de negocio en subprocesos de negocio, utilizando para ello las clases *gestionarDocXml* y *TransformadorBpelGrafos*.

    - *GestionarDocXml*: Esta clase contiene las funciones que permiten la manipulación de documentos XML

    - *SubprocesosdenegocioImpl*: Esta clase se encarga de la lógica para manejar la información de los subprocesos particionados.

    - *VincularSubprocesoImpl*: Encargada de manejar la lógica para la vinculación de los actores a cada subproceso generado.

    - *MonitoreoImpl*: Encargada de manejar la lógica para obtener la información del estado de los procesos en ejecución.

- RolesImpl: Esta clase se encarga de la lógica para manejar la información de los roles de los procesos.

- ActorImpl: Esta clase se encarga de la lógica para manejar la información de los actores de los procesos.

- UsuariosImpl: Esta clase se encarga de la lógica para manejar la información de los usuarios de la plataforma Web.

- Control de la Ejecución: Este paquete contiene, las clases que permiten interactuar entre el dispositivo móvil y la plataforma. Las funciones involucradas son: control de la sincronización de la ejecución de procesos, control del monitoreo de procesos, distribución de subprocesos a los dispositivos móviles y llamado dinámico a servicios Web, las clases que contiene este paquete son:

    - *Comunicacion:* Esta clase permite la comunicación entre la aplicación para el dispositivo móvil y la aplicación Web. Esta se comunica con las otras clases de este paquete según la petición enviada por el dispositivo (sincronización, monitoreo, distribución, ejecución de servicios Web).

    - ValidarComunicaciónImlp: Por medio de esta clase, se valida si el actor que realiza determinada petición desde el sistema móvil de información, es un usuario registrado en la plataforma. Con esta validación, también se obtiene el identificador del proceso de negocio al cual está asociado el subproceso ejecutado por el actor que realiza la petición, y se lo comunica a las otras clases de este paquete a través de la clase *Comunicacion*. De esta manera las otras clases realizan sus funciones de acuerdo al proceso de negocio identificado.

    - *SincronizaciónImpl:* Encargada de manejar la lógica de la aplicación para controlar la ejecución de los subprocesos de negocio, con el fin de gestionar aspectos de la sincronización de ejecución. Para lograr este propósito esta clase registra y consulta los datos de sincronización (variables) recibidos a través de la clase *Comunicacion*.

    - *ManejoVariables:* Esta clase se encarga de manejar la información de las variables utilizadas en la ejecución de los subprocesos de negocio.

    - *GestionMensajeria*: Cuando un dispositivo móvil termina la ejecución de un proceso de negocio, es necesario indicarle al sistema móvil de información que continua con la ejecución del proceso. Esta clase obtiene el número de teléfono del siguiente dispositivo involucrado en la ejecución de un proceso.

    - *RegistrodeMonitoreoImpl*: Clase que permite registrar la información de las actividades ejecutadas en los dispositivos móviles, por medio de los datos recibidos de la clase de *Comunicacion*.

    - *DistribucionImpl*: Encargada de distribuir por medio de la clase de *Comunicacion* los subprocesos generados a los correspondientes actores en sus sistemas móviles de información.

    - *InterfazdeServicios*: Encargada de realizar la invocación de los servicios Web, requeridos en la ejecución de los procesos de negocio en los sistemas móviles de información. Esta clase captura la información enviada por estos y la transforma a un formato apropiado para la invocación. Finalmente retorna el resultado de esta invocación en un formato comprensible por el motor BPEL móvil.

    - *GestionCorreoElectronicoImpl*: Luego de realizado el particionamiento, se encarga de obtener la información del correo electrónico de los actores que se acaban de asociar al

proceso particionado, y envía un correo electrónico que indica que pueden descargar el subproceso

- Persistencia: Este paquete contiene las clases encargadas de manejar el acceso a la base de datos.
- *ProcesodeNegocio*: Esta clase representa datos persistentes para almacenar la información del Proceso de negocio.
- *ProcesodeNegocioDAO*: Esta clase se usa en conjunto con la clase *ProcesodeNegocio*, y hace las implementaciones concretas para el acceso a la fuente de datos.
- *VisualizarProceso:* Esta clase representa datos persistentes para almacenar la información requerida para visualizar el Proceso de negocio
- *VisualizarProcesoDAO*: Esta clase se usa en conjunto con la clase *VisualizarProceso*, y hace las implementaciones concretas para el acceso a la fuente de datos.
- *ValidarComunicacion:* Esta clase representa datos persistentes para almacenar la información requerida para Validar la Comunicación.
- *ValidarComunicacionDAO:* Esta clase se usa en conjunto con la clase *ValidarComunicacion*, y hace las implementaciones concretas para el acceso a la fuente de datos.
- *Distribucion:* Esta clase representa datos persistentes para almacenar la información requerida para la Distribución.
- *DistribucionDAO:* Esta clase se usa en conjunto con la clase *Distribucion*, y hace las implementaciones concretas para el acceso a la fuente de datos.
- *RegistrodeMonitoreo:* Esta clase representa datos persistentes para almacenar la información requerida para el Monitoreo.
- *RegistrodeMonitoreoDAO:* Esta clase se usa en conjunto con la clase *RegistrodeMonitoreo*, y hace las implementaciones concretas para el acceso a la fuente de datos.
- Sincronizacion: Esta clase representa datos persistentes para almacenar la información requerida para la sincronización.
- SincronizacionDAO: Esta clase se usa en conjunto con la clase Sincronizacion, y hace las implementaciones concretas para el acceso a la fuente de datos.
- *SubprocesosdeNegocio*: Esta clase representa datos persistentes para almacenar la información requerida para el manejo de los subprocesos de negocio.
- *SubprocesosdeNegocioDAO*: Esta clase se usa en conjunto con la clase *SubprocesosdeNegocio*, y hace las implementaciones concretas para el acceso a la fuente de datos.
- *VincularSubproceso*: Esta clase representa datos persistentes para almacenar la información sobre la vinculación de los actores y subprocesos.
- *VincularSubprocesoDAO*: Esta clase se usa en conjunto con la clase *VincularSubproceso*, y hace las implementaciones concretas para el acceso a la fuente de datos.

- *Monitoreo*: Esta clase representa datos persistentes para obtener la información sobre el Monitoreo.
- *MonitoreoDAO*: Esta clase se usa en conjunto con la clase *Monitoreo*, y hace las implementaciones concretas para el acceso a la fuente de datos.
- *GestionMensajeria:* Esta clase representa datos persistentes para almacenar y obtener la información sobre el número de teléfono del dispositivo requerido.
- *GestionMensajeriaDAO*: Esta clase se usa en conjunto con la clase *GestionMensajeria*, y hace las implementaciones concretas para el acceso a la fuente de datos.
- *GestionCorreoElectronico:* Esta clase representa datos persistentes para almacenar y obtener la información sobre el correo de electrónico de los actores del proceso requerido.
- *GestionCorreoElectronicoDAO:* Esta clase se usa en conjunto con la clase *GestionCorreoElectronico*, y hace las implementaciones concretas para el acceso a la fuente de datos.
- *Actor:* Esta clase representa datos persistentes para almacenar y obtener información los actores del proceso requerido.
- *ActorDAO:* Esta clase se usa en conjunto con la clase *Actor*, y hace las implementaciones concretas para el acceso a la fuente de datos.
- *Roles:* Esta clase representa datos persistentes para almacenar y obtener la información sobre los roles del proceso.
- *RolesDAO:* Esta clase se usa en conjunto con la clase *Roles*, y hace las implementaciones concretas para el acceso a la fuente de datos.
- *Usuarios:* Esta clase representa datos persistentes para almacenar y obtener la información sobre los usuarios de la plataforma.
- *UsuariosDAO:* Esta clase se usa en conjunto con la clase *Usuarios*, y hace las implementaciones concretas para el acceso a la fuente de datos.

Capa de mediación

- Tomcat: Representa al servidor Web y de aplicaciones, necesarios para que la plataforma esté desplegada constantemente en la Web y permita las peticiones de los diferentes dispositivos móviles de manera concurrente.
- Apache Axis: Es una implementación OpenSource de SOAP que presenta un entorno para la ejecución de servicio Web implementados en java. Apache Axis permite realizar la invocación de servicios Web como la realiza un motor BPEL [24].
- JDBC: Permite la comunicación entre los componentes de capas superiores con el motor de base de datos.

Capa de datos

- Motor de base de datos: Este paquete representa el motor de base datos donde son almacenados los datos requeridos por la aplicación Web.

3.1.4.1.3 Diagrama de clases cliente móvil MobFlow

La figura 20 presenta el diagrama de las clases implementadas en la aplicación cliente móvil MobFlow. Las clases que inician con *frm*, son la clases que presenta una interfaz de usuario que permiten a los actores del proceso realizar sus actividades y comunicarse con MobFlow, las clase que inician con *manejo* representa clase que permiten realizar algunas actividades BPEL y fueron implementadas para complementar algunos aspectos del motor BPEL utilizado y que es representado por la clase *motorBPEL*, más adelante en este documento se explica porque fue necesario implementar las clases contenidas en este paquete. Las otras clases permiten que la aplicación móvil realice los aspectos de ejecución, comunicación y almacenamiento necesarios.

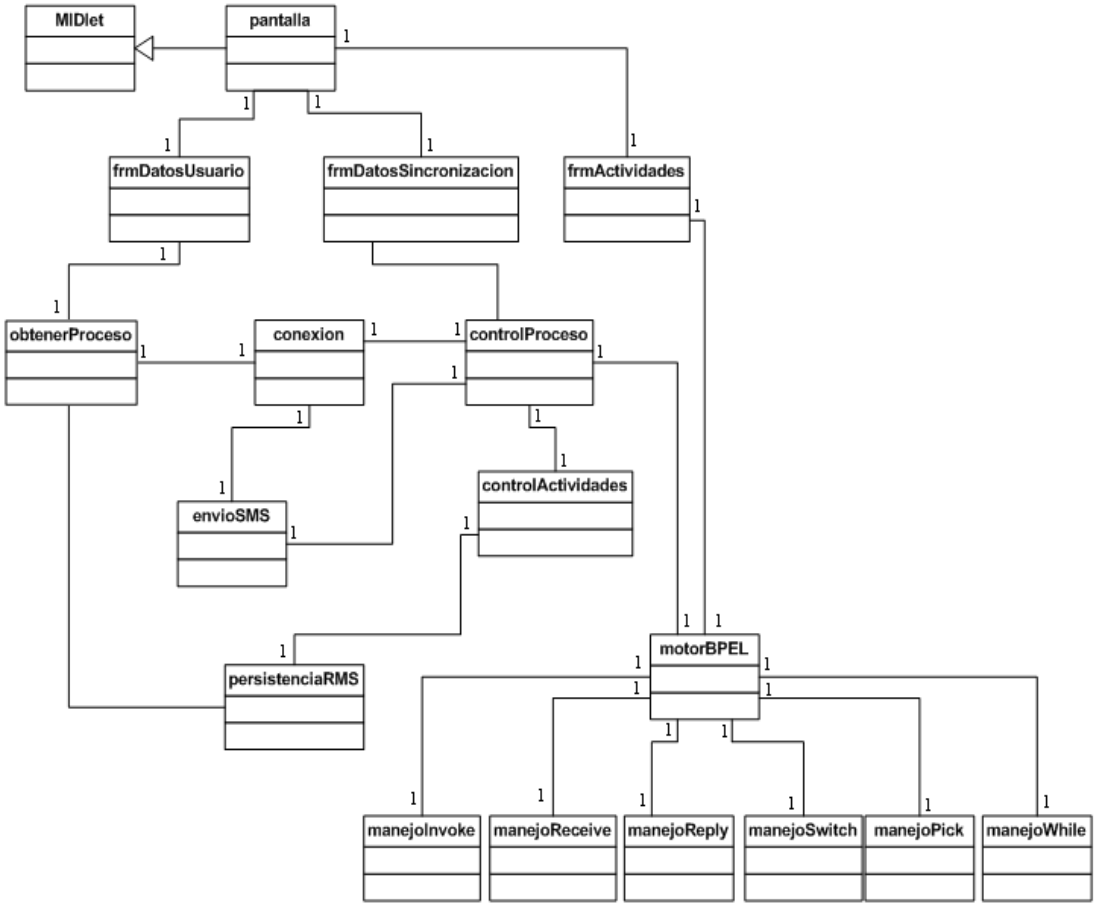

Figura 20. Diagrama de clases cliente móvil MobFlow.

3.1.4.1.4 Diagramas de paquetes cliente móvil MobFlow

La figura 21, presenta el diagrama de paquete de la aplicación cliente móvil MobFlow, a continuación se describen cada uno de ellos junto con las clases contenidas y que fueron presentadas en la figura 20.

Capa de Presentación

- Vista**:** Contiene las clases que implementan las interfaces gráficas de usuario, que permiten al actor del dispositivo móvil realizar sus tareas del proceso de negocio. Está compuesto de las siguientes clases.

    - *frmDatosUsuario***:** Clase que implementa la Interfaz gráfica que permite al usuario ingresar sus datos con el fin de comunicarse con la aplicación Web MobFlow y descargar su respectivo proceso de negocio.

    - *frmDatosSincronizacion***:** Presenta los datos de sincronización que recibe el usuario al iniciar un proceso.

    - *frmActividades***:** Genera una interfaz de usuario dinámica para que el actor del proceso realice sus actividades o tareas respectivas.

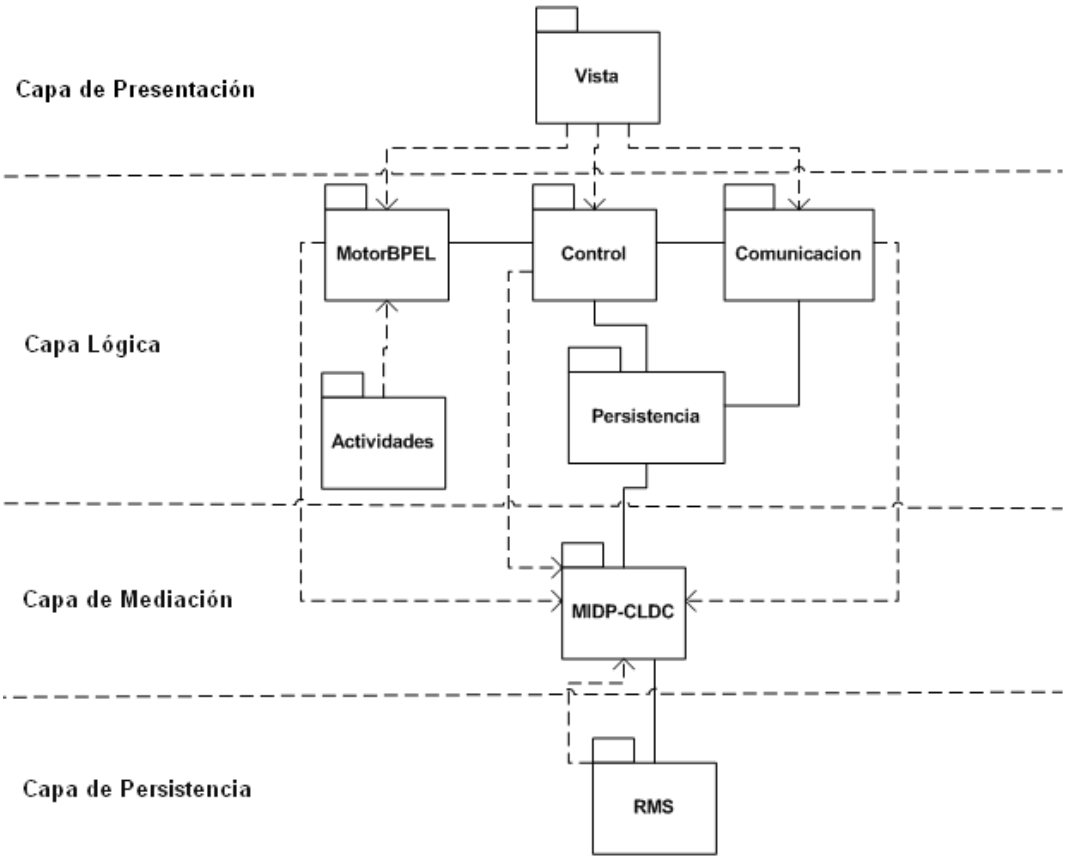

Figura 21. Diagrama de paquetes cliente móvil MobFlow.

Capa Lógica

- Comunicación**:** Este paquete maneja las clases encargadas de todos los aspectos de comunicación que requiere el sistema móvil de información para la ejecución satisfactoria de los procesos de negocio.

    - *conexion:* Esta clase realiza el proceso de conexión con el servidor, con el fin de establecer la comunicación que requiere la ejecución de los procesos de negocio, tales

como: invocación a Servicios Web, recibir información de variables para la sincronización**,** u obtener el respectivo proceso de negocio del usuario.

- *obtenerProceso***:** Obtiene los datos de usuario y los parametriza para ser enviados al servidor en forma de una petición HTTP. Cuando se obtiene una respuesta del servidor, por medio de esta clase se serializa y deserializa la información enviada por esté, para formar los archivo del proceso correspondiente.

- *envioSMS:* Esta clase permite enviar un mensaje de texto al siguiente dispositivo en la ejecución una vez finalizada la ejecución del subproceso.

- Motor BPEL**:** Contiene las clases que componen el motor BPEL Sliver utilizado por la aplicación cliente móvil MOBFLOW, para procesar los documentos BPEL descargado por los usuarios. Contiene la siguiente clase.

  - *motorBPEL***:** Representa el motor BPEL que interpreta y ejecuta los subprocesos BPEL obtenidos por el usuario. Como ya se mencionó se utiliza el motor BPEL para dispositivos móviles Sliver.

- Actividades**:** Contiene las clases que complementan el motor BPEL Sliver, para poder realizar el procesamiento de los procesos BPEL generados por MOBFLOW e interactuar con la misma.(Estas clases se implementaron debido a algunas restricciones del motor Sliver descritas más adelante, para otras actividades BPEL no fue necesario la implementación). Contienen las siguiente clases:

  - *manejoInvoke***:** Clase que interactúa con el paquete motor BPEL, para ejecutar actividades de tipo <`invoke`> en el proceso BPEL.

  - *manejoReceive***:** Clase que interactúa con el paquete motor BPEL, para ejecutar actividades de tipo <`receive`> en el proceso BPEL.

  - *manejoReply***:** Clase que interactúa con el paquete motor BPEL, para ejecutar actividades de tipo <`reply`> en el proceso BPEL.

  - *manejoSwitch***:** Clase que interactúa con el paquete motor BPEL, para ejecutar actividades estructuradas de tipo <`switch`> en el proceso BPEL.

  - *manejoPick***:** Clase que interactúa con el paquete motor BPEL, para ejecutar actividades estructuradas de tipo <`pick`> en el proceso BPEL.

  - *manejoWhile***:** Clase que interactúa con el paquete motor BPEL, para ejecutar actividades estructuradas de tipo <`while`> en el proceso BPEL.

- Control**:** Este paquete contiene las clases encargadas de controlar la ejecución del proceso para lo cual se interactúa con la aplicación Web. A continuación se describe cada una de ellas:

  - *controlActividades:* Clase que recorre el proceso BPEL y permite establecer el orden de ejecución de las actividades del proceso.

  - *controlEjecucion:* Clase que permite realizar la ejecución controlada de las diferentes actividades definidas del proceso, lo hace en conjunto con la clase del paquete ***motor BPEL.***

- Persistencia**:** Maneja las clases que implementan el almacenamiento persistente en la aplicación móvil a través de registros. Está compuesto por la siguiente clase:

- *persistenciaRMS:* Se encarga de almacenar y obtener los archivos del proceso a través de registros, para que sean cargados y procesados por el motor BPEL.

Capa de Mediación

- MIDP-CLDC**:** Representa la estructura de la aplicación en configuraciones y perfiles. Una configuración CLDC contiene la maquina virtual Java (KVM) y un conjunto de clases genéricas ("configuración") que pueden ser empleadas en un amplio rango de dispositivos [63]. Por su parte, el "perfil" MIDP contiene las clases que permiten el desarrollo de las aplicaciones en un tipo de dispositivo determinado [62].

- *MIDlet*: La clase MIDlet representa la clase que hereda de la clase abstracta *javax.microedition.midlet.MIDlet* y permite el control de las aplicaciones para dispositivos móviles desarrolladas bajo el perfil MIDP-CLDC.

- *pantalla:* Cada MIDlet tiene una referencia a este objeto. Recupera información acerca del despliegue actual (Ej. Nro. colores) e incluye métodos para que los objetos sean ubicados en la pantalla.

Capa de Persistencia

- RMS**:** Representa el sistema de almacenamiento, en este caso se usa el Sistema Administrador de Registros RMS (*Record Management System*) [52]. Este proporciona un mecanismo que permite almacenar datos de forma persistente para su futura recuperación. Este mecanismo está implementado sobre una base de datos basada en registros.

En el **anexo A** se presentan los diagramas de secuencia tanto de la plataforma MOBFLOW como de la aplicación cliente móvil MOBFLOW obtenidos en esta fase.

3.2 DEFINICIÓN DE LA ARQUITECTURA

Para la plataforma se definió una arquitectura que consta de diferentes módulos cada uno de estos se encarga de alguna funcionalidad especifica. En la figura 22 se presenta la arquitectura que da soporte a la plataforma MobFlow. A continuación se da una breve descripción de cada uno de los módulos que la componen.

Archivos del Proceso

Estos archivos representan la definición del proceso de negocio en el lenguaje BPEL, los archivos del proceso para la plataforma MobFlow deben proveer un archivo BPEL y WSDL. Estos archivos en la definición de un proceso descrito en BPEL, contienen toda la información necesaria acerca de los procesos, incluye información de comienzo de actividades, condiciones y reglas de navegación. Además modelan la navegación entre los procesos, proveen información acerca de entradas a estos y criterios a tomar en cada paso de la navegación, junto con la asignación de tareas a los participantes del proceso

Módulo de Gestión de Información de la Organización

Este módulo es el encargado de realizar todas las gestiones relacionadas con la información de la empresa. Dicha información es requerida por la plataforma para poder realizar el despliegue distribuido de los procesos de negocio en los sistemas móviles de información.

En este módulo Intervienen los paquetes *Actores*, *Roles*, *Usuarios* para el manejo grafico de la información respectiva haciendo uso de las clases *ActoresBean*, *RolesBean* y *UsuariosBean*. También interviene el paquete Modelo para el manejo de la lógica de dicha información a través de las clases *RolesImpl*, *ActorImpl* y *UsuariosImpl*, junto con las clases *Roles, Actor, y Usuarios* del paquete Persistencia las cuales complementan dicha lógica.

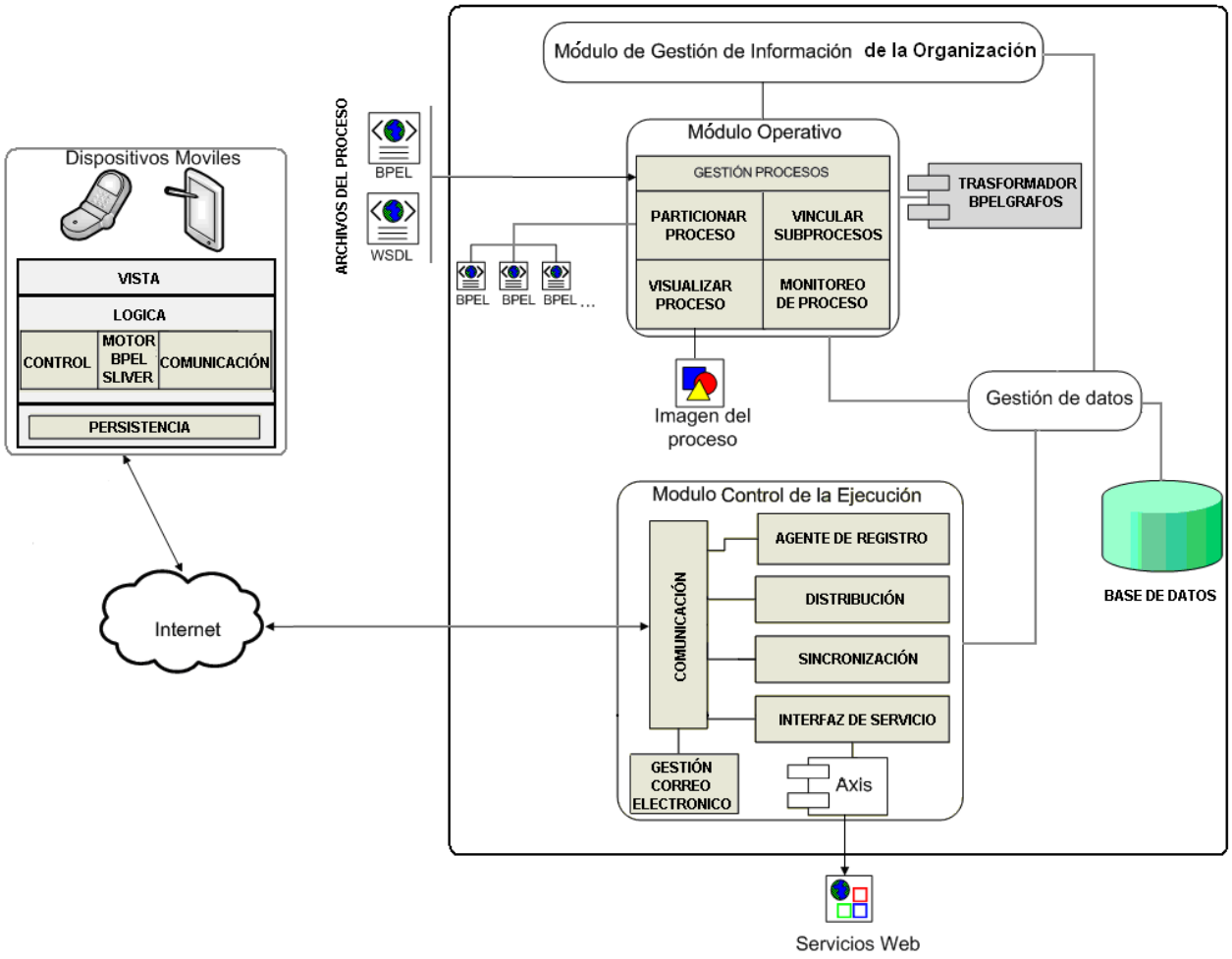

Figura 22. Arquitectura de la plataforma MobFlow

Transformador BPEL-Grafos: Este componente es el encargado de realizar el proceso de transformación de un documento BPEL a un modelo de grafos [42].

Se escogió la representación de un proceso BPEL en un modelo de grafos, basados en los resultados del análisis realizado en [3], donde se comprueba que la representación en Grafos es la más eficiente cuando se requiere realizar el particionamiento de procesos de negocios.

En este módulo interviene el paquete Modelo para la transformación descrita en los párrafos anteriores (la clase utilizada en esta transformación es: *TransformadorBpelGrafos* que representa las clases del componente utilizado para esta transformación).

Módulo Operativo

Módulo encargado de manejar los diferentes aspectos operativos de la aplicación. Los diferentes submódulos que lo componen a excepción del submódulo **Vincular Subproceso** requieren del componente **Transformador BPEL-Grafos.** El módulo operativo está compuesto por los siguientes submódulos:

- **Gestión de Proceso de negocio:** Módulo encargado de cargar los documentos BPEL y WSDL correspondientes a la descripción de un proceso específico y determina que estén bien formados, además realiza las funciones de editar, consultar, eliminar los procesos cargados.

En este módulo interviene el paquete **Carga de Procesos** para presentar las interfaces gráficas de usuario que permiten esta carga y los paquetes **Modelo** y **Persistencia** con las clases *ProcesodeNegocioImpl, CargarArchivos* y *ProcesodeNegocio,* respectivamente, para manejar la lógica requerida para esta funcionalidad.

- **Visualizar Proceso:** Por medio de este módulo se presenta de manera visual, los diferentes procesos cargados en la plataforma, este módulo genera una imagen creada a partir del modelo de grafos que representa el proceso y es presentada en una interfaz Web.

  En este módulo interviene el paquete **Visualizar Proceso** para la presentación de la interfaz Web que muestra la imagen del proceso a través de la clase *VisualizarProcesoBean*, también interviene las clases *VisualizarProcesoImpl* del paquete **Modelo** para manejar la lógica que requiere esta visualización, junto con *VisualizarProceso* del paquete **Persistencia.**

- **Particionar Proceso:** Este módulo realiza el particionamiento del proceso de negocio en varios subproceso, y crea los documentos BPEL correspondientes, considerando actividades especiales que permiten sincronizar la ejecución de los subprocesos distribuidos.

  Para el particionamiento en este módulo participa la clase *SubProcesodeNegocioBean* del paquete **Particionamiento de Procesos** para el manejo de las interfaces gráficas de usuario de esta funcionalidad**.** Las clases *ParticionarProceso, GestionarDocXml* y *SubprocesosdeNegocioImpl* del paquete **Modelo** y la clase *SubProcesodeNegocio* del paquete **Persistencia** manejan la lógica que requiere el particionar un proceso.

- **Vincular Subprocesos:** Permite vincular los actores con los subprocesos generados en el módulo de **Particionar Proceso**, de acuerdo al rol que esté desempeñe dentro de la ejecución del proceso.

  En este módulo participa el paquete **Vinculación de actores** que presenta las interfaces gráfica de usuario para permitir la vinculación de los actores a los subprocesos y el paquete **Modelo** con la clase *VincularSubprocesoImpl,* la cual junto con su clase asociada *VincularProceso* del paquete **Persistencia** manejan la lógica de esta funcionalidad.

- **Monitoreo Proceso:** Este módulo presenta el estado actual de ejecución de los procesos de negocio, a medida que los subprocesos generados son ejecutados en los dispositivos móviles, permitiendo mostrar la información del monitoreo, así como una gráfica de las actividades que han sido ejecutadas.

  En este módulo participa el paquete **Monitoreo** para presentar la información del monitoreo y los paquetes **Modelo y Persistencia** con las clases *MonitoreoImpl y Monitoreo* respectivamente las cuales realizan el registro y obtención de las actividades ejecutadas en los dispositivos móviles, además de la lógica requerida para esta funcionalidad.

  Vale la pena mencionar que para esta funcionalidad también se hace uso de la clase *VisualizarProcesoImpl* del paquete modelo para presentar de manera gráfica el proceso de negocio monitoreado.

Módulo de Control de la Ejecución

Este módulo, es el encargado de realizar el control de la ejecución de los subprocesos distribuidos en los sistemas móviles de información. Está compuesto por:

- **Módulo de Comunicación:** Este módulo se encarga de manejar todos los aspectos de comunicación entre la aplicación móvil y la aplicación Web.

  En este módulo intervienen el paquete **Control de la Ejecución** con las clases *Comunicacion* y *ValidarComunicacionImpl,* para manejar toda la lógica requerida para la comunicación, e

interviene el paquete **Tomcat** de la capa de mediación, el cual maneja las peticiones enviadas por los sistemas móviles de información.

- **Distribución:** Este módulo se encarga de distribuir los subproceso de negocio a los correspondientes actores en sus sistemas móviles de información, esta distribución es lograda luego de que los sistemas móviles realizan la petición de descarga por medio del módulo de **Comunicación**.

  En este módulo interviene el paquete **Control de la Ejecución** con la clase *DistribucionImpl* y el paquete **Persistencia** con la clase *Distribucion*, las cual implementan la lógica para lograr esta funcionalidad.

- **Sincronización:** Este módulo se encarga de manejar todos los aspectos de sincronización que se requieren para la ejecución distribuida de los diferentes subprocesos.

  En este módulo interviene el paquete **Control de la Ejecución** con las clases *SincronizacionImpl, ManejoVariables y GestionMensajeria* y el paquete **Persistencia** con la clase *Sincronizacion.* Estas clases implementan dichos aspectos de sincronización requeridos por los sistemas móviles información para la ejecución distribuida de los subprocesos de negocio.

- **Agente de Monitoreo:** Este módulo permite registrar la información de las actividades ejecutadas en los sistemas móviles información, con el objetivo de que esta información sea utilizada por el módulo de **Monitoreo Proceso,** para poder presentar el estado de ejecución del proceso general**.**

  En este módulo intervienen los paquetes **Control de la Ejecución** con la clase *RegistrodeMonitoreoImpl* y el paquete **Persistencia** con la clase *RegistrodeMonitoreo.*

- **Interfaz de servicios:** Este módulo obtiene los datos enviados desde el móvil cuando requiere invocar un servicio Web, y los convierte en un formato adecuado para que sean invocados utilizando Apache Axis. También se encarga de recibir el resultado de la invocación del servicio Web y retórnalo al sistema móvil de información.

  En este módulo intervienen los paquetes **Control de la Ejecución** con la clase *interfazdeServicio* y el paquete **Apache Axis** de la capa de Mediación

- **Gestión Correo Electrónico:** Este módulo es encargado de enviar un correo electrónico a los participantes del proceso, luego de que se ha realizado la vinculación de un actor con el subproceso generado, para indicarles a estos que se encuentra pendiente la descarga y ejecución de subprocesos. Participan los paquetes **Control de la Ejecución** y **Persistencia** con las clases *GestionMensajeriImpl* y *GestionMensajeria* respectivamente.

- **Apache Axis:** Es una implementación OpenSource de SOAP que presenta un entorno para la ejecución de servicio Web implementados en java, AXIS se encuentra diseñado para ser ejecutado dentro de un servidor de aplicaciones como Tomcat, lo que permite utilizar muchas características de esos contenedores como la seguridad, los repositorios de recursos, multi-hilos, etc. [48]. Apache Axis es utilizado por la plataforma para realizar el proceso de invocación dinámica de servicios Web cuando los requiere la ejecución de un proceso de negocio desde el sistema móvil de información.

- **Módulo Gestión de Datos**

Este módulo es el encargado de realizar el acceso a la base de datos del sistema, ya sea para almacenar u obtener información requerida por la aplicación. Interviene el paquete **Persistencia** con las clases *Usuarios, UsuariosDAO, Actor, ActorDAO, Roles, RolesDAO, ProcesodeNegocio, ProcesodeNegocioDAO, Visualizar, VisualizarProcesoDAO, SubProcesodeNegocio, SubProcesodeNegocioDAO, VincularProceso, VincularProcesoDAO, Monitoreo, MonitoreoDAO,*

VisualizarProceso, VisualizarProcesoDAO, ValidarComunicacion, ValidarComunicacionDAO, Distribucion, DistribucionDAO, Sincronizacion, SincronizacionDAO, RegistrodeMonitoreo, RegistrodeMonitoreoDAO, GestionMensajeria y GestionMensajeriaDAO. Estas clases realizan el acceso a datos requeridos por los diferentes módulos que conforman la plataforma. También intervienen los paquetes **JDBC** que realiza el acceso físico a la Base de Datos y el paquete **Motor de Bases de Datos** donde se encuentra la base de datos del sistema.

Dispositivos Móviles

Los dispositivos móviles son utilizados por los actores del proceso para realizar la ejecución de sus actividades, estos la llevan a cabo gracias a la aplicación cliente móvil MobFlow que permite la descarga y ejecución de los subprocesos generados por la aplicación Web MobFlow.

La aplicación cliente móvil MOBFLOW maneja tres capas como se presenta en la figura 23.

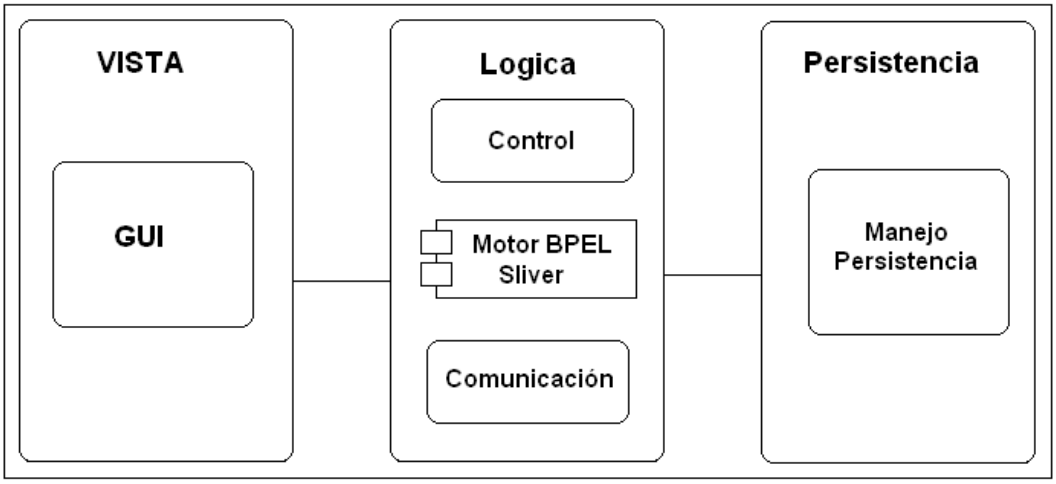

Figura 23. Arquitectura aplicación cliente móvil MobFlow.

VISTA

- GUI (Graphics User Interface – Interfaces Gráficas de Usuario): Permiten al actor del sistema móvil de información interactuar con las interfaces para poder realizar sus correspondientes actividades del proceso de negocio desde el dispositivo móvil. Interviene el paquete **Vista** con las clases *frmDatosUsuario*, *frmDatosSincronizacion y frmActividades* que presentan las interfaces gráficas y el paquete **MIDP-CLDC** que permite crear esta interfaces.

Lógica

- Comunicación: Este módulo se encarga de manejar todos los aspectos de comunicación del sistema móvil con la aplicación Web MobFlow y con otros sistemas móviles de información. Intervienen todas las clases del paquete **Comunicación** y el paquete **MIDP-CLDC**.

- Control: Las clases de este módulo son las encargadas de controlar todos los aspectos de la ejecución de la aplicación cliente móvil, comunicando las diferentes capas e interactuando con ellas, una de las principales funciones es también interactuar con el motor BPEL. Conformado por todas las clases del paquete **Control.**

- Motor BPEL Sliver: Representa el motor BPEL que interpreta y ejecuta los subprocesos BPEL descargados por el actor, para esta aplicación se utilizo el motor BPEL para dispositivos móviles Sliver. En este módulo intervienen los paquetes **MotorBPEL** con la clase *motorBPEL* que representa las diferentes clases de Sliver y el paquete **Actividades** que permite complementar al motor BPEL Sliver para lograr la funcionalidad requerida por la plataforma.

Persistencia

- Manejo Persistencia: Maneja el sistema de almacenamiento del dispositivo móvil, donde se almacena la información requerida para la ejecución del subproceso. Esta información será guardada en el dispositivo en una zona de memoria dedicada para este propósito. La cantidad de memoria y la zona asignada para ello dependerá de cada dispositivo. Se almacena el proceso correspondiente (BPEL-WSDL) y los datos del usuario, como se mencionó anteriormente utilizando el sistema administrador de registros RMS.

  Intervienen los paquetes **Persistencia** que realiza la lógica para almacenar la información y los paquetes **MIDP-CLDC**, **RMS** para lograr el acceso y almacenamiento en el sistema de registros RMS.

3.2.1 Diagrama de Despliegue de Plataforma.

El diagrama de despliegue indica físicamente en donde estarán ubicados cada uno de los componentes de la arquitectura, y permite visualizar los protocolos de comunicación usados. La figura 24 presenta como se realiza el despliegue de la plataforma MobFlow.

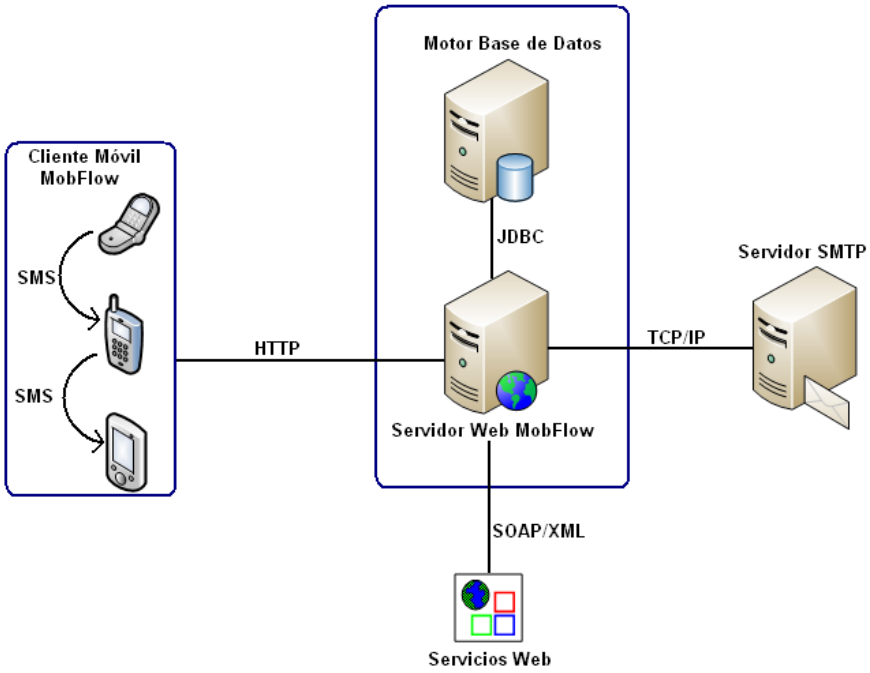

Figura 24. Diagrama de Despliegue Plataforma.

La totalidad de aplicación Web MobFlow es alojada en un solo equipo de cómputo que actúa como servidor Web, el cual además contiene el motor de base de datos. Los archivos BPEL y WSDL que representan los procesos de negocio BPEL cargados en la plataforma, quedan almacenados en los directorios de este servidor web, al igual que los archivos BPEL de los subprocesos generados por la plataforma.

Las solicitudes provenientes de los dispositivos móviles al Servidor Web son realizadas a través de HTTP. La invocación de los servicios Web requeridos por los subprocesos de negocio se hace desde el servidor Web, mediante una comunicación SOAP/XML. La aplicación Web hace uso de un servidor SMTP para enviarles un correo electrónico a los actores de los procesos empresariales, indicándoles que

Cuando un sistema móvil de información termina la ejecución de un proceso, le notifica al siguiente dispositivo en la ejecución que debe realizar sus actividades, esta notificación se hace mediante un mensaje de textos (SMS) enviado automáticamente desde el dispositivo que termina la ejecución. A continuación se hace una descripción de los componentes y protocolos presentado en el diagrama de despliegue.

Servidores

- **Servidor Web MobFlow**
  Un Servidor Web es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. El servidor Web se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página Web o información de todo tipo de acuerdo a los comandos solicitados.
- **Servidor de Base de Datos o Motor de Base de Datos**
  Componente que sirve como medio de almacenamiento de los datos que deben persistir dentro de una base de datos para el correcto funcionamiento de la plataforma.

- **Servidor SMTP** (SMTP: Simple Mail Transfer Protocol - Protocolo Simple de Transferencia de Correo)
  Componente que sirve para el envió de correos electrónicos a través de la red usando el protocolo SMTP que permite el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA's, teléfonos móviles, etc.)

Dispositivos Móviles
Los dispositivos móviles lo utilizan los actores de los procesos de negocio para ejecutar sus actividades correspondientes.

Protocolos de comunicación y conexión.
Los protocolos de comunicación son el conjunto de reglas que especifican el intercambio de mensajes durante la comunicación entre las entidades que forman parte de una red. Los utilizados por la plataforma MobFlow son los siguientes:

- **HTTP:** (Protocolo de transferencia de hipertexto - HyperText Transfer Protocol) este protocolo define la sintaxis y la semántica que utilizan los elementos software de la arquitectura web (clientes, servidores) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

- **SOAP:** (Protocolo Simple de Acceso a Objetos - Simple Object Access Protocol) este protocolo define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

- **XML:** (eXtensible Markup Language - lenguaje de marcas extensible) es un Lenguaje de Etiquetado Extensible, que juega un papel fundamental en el intercambio de una gran variedad de datos. Su función principal es describir datos y permite la lectura de datos a través de diferentes aplicaciones. XML sirve para estructurar, almacenar e intercambiar información.

- **Servicio Web:** Un servicio Web es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones software desarrolladas en lenguajes de programación diferentes y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios Web para intercambiar datos en redes de computadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos, como XML.

- **SMS:** (Short Message Service - Servicio de Mensajes Cortos) es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos (también conocidos como mensajes de texto) entre teléfonos móviles, teléfonos fijos y otros dispositivos de mano.

- **JDBC:** (Java Database Connectivity) Es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede.

Resumen

En este capítulo, se presentó la arquitectura que da soporte a la plataforma para despliegue distribuido de procesos de negocio en sistemas móviles de información MobFlow. También se mostraron los Diagramas de Casos de Uso, Diagramas de Clases y Diagramas de Paquetes obtenidos en el desarrollo de las dos aplicaciones software que conforman la plataforma y los requisitos funcionales de está.

Entre los aspectos teóricos expuestos en esté capitulo se describieron las consideraciones previas a la definición de la arquitectura y desarrollo de la plataforma. Finalmente se presento el diagrama de despliegue de la plataforma MobFlow.

En el siguiente capítulo se presentará una descripción detallada de cómo se realiza la distribución, sincronización, ejecución y monitoreo de los procesos de negocio en los sistemas móviles de información.

Capítulo IV

DESPLIEGUE DISTRIBUIDO DE PROCESOS DE NEGOCIO EN SISTEMAS MÓVILES DE INFORMACIÓN

En este capítulo se describen de manera detallada los diferentes mecanismos y algoritmos que permiten a la plataforma MobFlow, lograr el despliegue distribuido de los procesos de negocio en los sistemas móviles de información. Para realizar dicho despliegue se consideran claves los siguientes aspectos.

- Particionamiento y sincronización de los procesos de negocio.
- Distribución de los respectivos subprocesos de negocio en los correspondientes sistemas móviles de información de los actores.
- Ejecución sincronizada de los subprocesos de negocio.
- Monitorear el estado de la ejecución distribuida de los subprocesos de negocio.

A continuación se presenta la descripción de cómo la plataforma MobFlow realiza dichas funciones.

4.1 PARTICIONAMIENTO Y SINCRONIZACIÓN DE LOS PROCESOS DE NEGOCIO

El particionamiento y la sincronización de los procesos de negocio, consiste en generar de un proceso de negocio descrito en BPEL, varios subprocesos de negocio con las actividades propias de los actores participantes en esté. Y a estos subprocesos insertarles actividades especiales de sincronización con el fin de llevar el control del flujo del proceso, de manera que la ejecución distribuida de los subprocesos de negocio sea equivalente a la ejecución del proceso de negocio original.

4.1.1 Bases Teóricas

Para el procesos de particionamiento se utiliza un modelo de grafos y se particiona a partir de éste, por consiguiente se debe hacer una transformación del proceso BPEL a un modelo de grafos. Debido a que en este proyecto se utiliza la propuesta especificada en [42] para realizar esta transformación, en esta sección se describe de manera resumida como dicha propuesta realiza la transformación de BPEL a grafos. En esta sección también se presentan las bases teóricas, que permitieron establecer como generar los subprocesos mediante la utilización de reglas de particionamiento y sincronización.

Transformador BPEL-Grafos

El transformador BPEL-Grafos toma como entrada un archivo BPEL y genera un modelo de grafos, este transformador es un proyecto desarrollado en Java [42]. A continuación se describen las clases y paquetes más importantes que exponen su funcionamiento.

El transformador mediante el paquete `bpm.bpel.model` almacena todo lo concerniente a una representación en objetos del documento BPEL cargado. Para cada parte del archivo de entrada se extraen sus componentes y se crean objetos que representan al proceso de negocio. Otro paquete elemental en el transformador *BPEL-Grafos* es el `bpm.graph.model`, cuya funcionalidad primordial es la de almacenar la estructura BPEL resultante en un metamodelo de objetos. Esté paquete contiene la clase `ProcessGraph`, la cual representa un grafo mediante arcos, funciones, conectores y vértices `start/end`. 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. Finalmente los arcos (aristas) hacen posible atar los vértices que se encuentran entre ellos.

Uno de los paquetes más importante del transformador es el paquete `bpm.graph.transform`, éste es el responsable de transformar el metamodelo de BPEL en un objeto de tipo *ProcessGraph*. Este paquete

tiene la clase *BpelTransform*, la cual tiene las funciones de transformación de cada actividad BPEL, además contiene la clase *AbstractActivityTransform*, que implementa las estrategias de transformación comunes a todas las actividades.

Finalizado el proceso de transformación se tiene un modelo de grafos que representa el proceso BPEL, los diferentes nodos del grafo representan actividades y poseen información de los atributos de la actividad, e información de la siguiente actividad a ejecutarse. La primera actividad *sequence* que contiene el proceso es representada por los nodos *Start* (que tiene la información la siguiente actividad a ejecutarse después de ella) *y End* (que marca el fin del proceso). Las actividades estructuradas tipos *While, Switch, Pick* entre otras poseen unos nodos asociados que marcan el inicio y fin de la actividad.

```
1 <process name="Operaciones" targetNamespace="http://xmlns.oracle.com/ Operaciones "
2       xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3       xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
4       xmlns:ns1="http://localhost:8090/axis/Operadora.jws" xmlns:ns3="http://localhost:8090/axis/Restadora.jws"
5       xmlns:ns2="http://localhost:8090/axis/Sumadora.jws" xmlns:as="http://xmlns.oracle.com/Promedio2Nums">

6   <partnerLinks>
7    <partnerLink myRole="Operar" name="Operacion" partnerLinkType="ns1:Operacion_PL"/>
8    <partnerLink myRole="Sumar" name="Sumadora" partnerLinkType="ns2:Sumadora_PL"/>
9    <partnerLink myRole="Restar" name=" Sustractora" partnerLinkType="ns3:Restadora_PL"/>
10 </partnerLinks>

11 <variables>
12   <variable messageType="as:Input" name="Entrada" />
13 </variables>

14   <sequence name="main">
15     <receive name="EntradaDatos" partnerLink=" Operacion " portType="as:Opera" operation="operar" variable="Entrada"/>
16       <switch name="DatoEntrada">
17           <case condition="bpws:getVariableData('inputVariable','num1') > bpws:getVariableData('inputVariable','num2')  ">
18               <invoke name="Suma" partnerLink="Sumadora" portType="as:Suma" operation="sumar" variable="Entrada"/>
19           </case>
20           <otherwise>
21                <invoke name="Resta" partnerLink="Sustractora" portType="as:Resta" operation="restar" variable="Entrada"/>
22            </otherwise>
23     </switch>
24   </sequence>
25 </process>
```

Figura 25. Ejemplo de un proceso BPEL que realiza operaciones de suma o resta según los datos de entrada Operaciones.bpel

En la figura 25 se presenta un ejemplo de un proceso BPEL, esté recibe como entrada dos parámetros numéricos y según la entrada realiza la invocación de una suma o una resta. Al realizar la transformación de este proceso a un modelo de grafos utilizando en transformador BPEL –Grafos [42], se obtiene una representación en grafos de este proceso, como la que se presenta en la figura 26.

Como se menciono anteriormente cada uno de los nodos de este grafo representa una actividad del proceso BPEL y contiene los atributos característicos de cada actividad, por ejemplo para el nodo receive se obtendrían los siguientes atributos: *Tipos actividad*: *receive*, *nombre actividad*: *EntradaDatos*, *partnerLink*: *Operadora*, *portType*: *Opera*, *operación*: *operar, variable*: *Entrada*. De igual manera para cada nodo se tiene una serie de atributos según el tipo de actividad que represente.

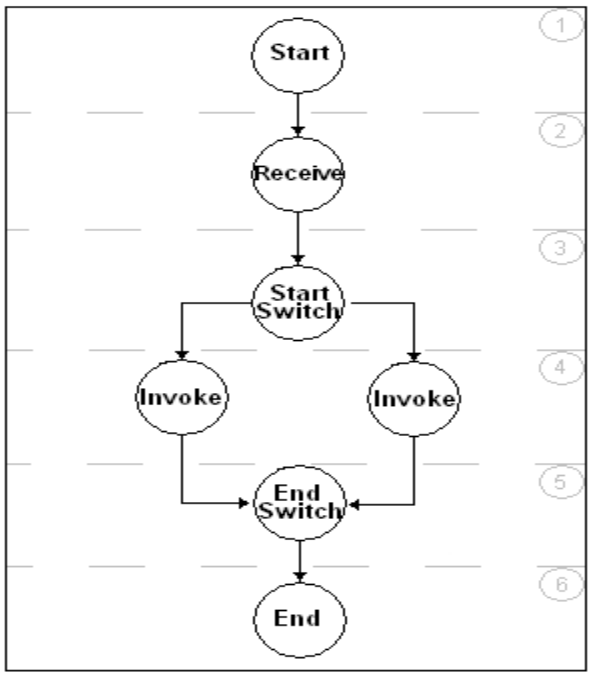

Figura 26. Grafo que representa el proceso BPEL presentado en la figura 25

Reglas de particionamiento y sincronización

Este proyecto de grado utiliza las reglas formales de particionamiento y sincronización propuestas en [8] y [9]. Estas investigaciones proponen que el particionamiento sea realizado utilizando un grafo que represente al proceso BPEL, y sobre este grafo se realicen los aspectos de particionamiento y sincronización. El fin de estas reglas es crear para cada actor involucrado en el control de la ejecución, una vista del proceso local teniendo en cuenta las actividades relacionadas con el rol que el actor cumple.

La plataforma MobFlow particiona los procesos de negocio BPEL considerando como actores a los *partnerLinks* del proceso, y aplica las reglas de particionamiento y sincronización a partir de los *partnerLinks*.

De lo anterior se considera que los *partnerLinks* juegan el papel principal en la definición del proceso de negocio, por lo cual en el presente proyecto se generan los diferentes subprocesos por cada *partnerLink* del proceso BPEL. Consecuentemente, el número de subprocesos generados en el particionamiento es igual al número de *partnerLinks* definidos en el proceso, y cada uno de estos subprocesos generados cuenta con las actividades propias del *partnerLink*. A continuación se presentan las reglas para el particionamiento y sincronización propuestas en [8]:

1. Es necesario establecer un controlador para cada actividad estructurada (con excepción de la actividad estructurada *<sequence>*), de tal manera que cada una tenga un *StarController* y un *EndController*. En el caso de la actividad estructurada *<while>* los controladores deben ser ejecutados por el mismo actor. Lo que buscan 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.

2. 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.

3. 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.

4. La ejecución del flujo de control de un conjunto específico de tareas se puede asignar solamente a un dispositivo.

5. 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.

Estas reglas presentan los aspectos principales que se deben considerar para realizar el particionamiento y sincronización, utilizando la representación del proceso en grafos. Por cada tipo de actividad especialmente las actividades estructuradas se definen reglas específicas para el particionamiento (no se presenta en este documento por razones de espacio, consultar [8], [9] y [10]).

Es importante aclarar que la regla 5, no se utiliza para generar los subprocesos del particionamiento. Esta regla se tuvo en consideración para realizar el otro proceso que inmiscuye la sincronización, que es el de distribuir la información de las variables requerida entre los actores del proceso para que puedan realizar la ejecución. Para la sincronización a continuación se profundizan los aspectos expuestos en la regla 2, que son tenidos en cuenta en el presente proyecto.

Sincronización

En el momento en que una actividad marque el fin del subproceso y el inicio de otro se procede a ingresar las actividades de sincronización, para ello se plantea una regla en la cual se identifican las actividades sucesivas que pertenezcan a dos subprocesos diferentes, con el fin de incluir actividades de sincronización en el momento en que se hace el particionamiento de procesos de negocio.

Estas actividades son: Una actividad tipo *<receive>* para recibir los datos de las variables de un proceso anterior y una actividad tipo *<invoke>* para enviar o registrar los datos de las variables al siguiente proceso.

Utilizando estas reglas de particionamiento y sincronización se obtienen subprocesos para los diferentes participantes, donde por ejemplo para un actor A, el subproceso incluye solamente la parte del proceso que tiene que ser controlada por A. En este caso el actor es representado por el partnerLink A, por lo cual para generar cada subproceso se remueven todas las actividades cuya ejecución no sea controlada por el partnerLink A. Garantizando la ejecución sincronizada de los subprocesos de negocio de forma similar a la ejecución del proceso inicial.

A continuación se describe como la plataforma realiza este proceso de particionamiento y sincronización, presentado los mecanismos y algoritmos utilizados para dicho fin.

4.1.2 Mecanismos y Algoritmos para el Particionamiento y Sincronización

Para llevar a cabo el particionamiento y sincronización debe haber sido cargado el documento BPEL en la plataforma. De este documento, inicialmente se obtienen todos los partnerLinks definidos en él y se almacenan la base de datos del sistema.

De aquí se procede a crear los documentos BPEL correspondientes con los subprocesos de negocio, para esto se plantearon dos algoritmos que realizan el particionamiento y sincronización, en estos se presenta paso a paso cómo se particiona y sincroniza, los algoritmos planteados son llamados: ***Inicio del Particionamiento*** y ***Reglas de Particionamiento***.

El algoritmo 1 Inicio del Particionamiento se presenta en pseudocódigo e inicia el particionamiento del proceso de negocio. Tiene como entrada un documento BPEL (*BPEL$_{XML}$*), como salida se obtienen varios objetos XML (*SubBPEL$_{XML}$*) con la información correspondiente a las actividades de cada subproceso.

Un objeto XML representa una estructura especial que contiene elementos XML (atributos, comentarios, instrucciones de procesamiento entre otros) y lleva el formato requerido por él. Esta estructura se define teniendo en cuenta que un proceso BPEL es totalmente en XML.

Algoritmo 1: Inicio del Particionamiento

```
INPUT: BPELXML
OUTPUTS: SubBPELXML1, Sub BPELXML2…SubBPELXMLN
1 BEGIN
2  Obtener todos los partnerLinks del proceso XMLBPEL
3  FOR Cada partnerLink Obtenido del proceso XMLBPEL
4        Obtener el atributo nombre del partnerLinki
5        Crear objeto XML (SubBPELXMLi) para contener actividades del subproceso
6        Convertir BPELXML a Modelo de Grafos
7        Ubicar el primer nodo en el grafo con atributo partnerLink igual nombre  partnerLinki
         (Representa el primer nodo  del   subproceso a generar)
8        IF NO es el primer subproceso a crear THEN
9              Insertar actividad receive de sincronización en SubXMLBPELi con los siguientes datos
                    TipoActividad = Receive
                    Nombre Actividad = sincronizacion
                    PartnerLink = Obtener atributo PartnerLink Nodo actual
                    PortType = Obtener atributo PartnerLink Nodo actual
                    Operación = Obtener atributo PartnerLink Nodo actual
                    Variable = Obtener atributo Variable Nodo actual
                    (Si existen más atributos obtenerlos del nodo  y ponerlo con  la etiqueta)
10       END IF
11       SubBPELXMLi ← Llamar Reglas Particionamiento
12  END FOR
13 END
```

En primer lugar se obtienen todos los `partnerLinks` del proceso (Línea 2) y se efectúan diferentes iteraciones por cada uno (Línea 3), donde se realizan los siguientes pasos:

- En cada interacción se obtiene el atributo nombre de cada `partnerLink` del proceso (Línea 4).
- A continuación se crea un objeto XML que contendrá todas las actividades propias del `partnerLink` obtenido en el paso anterior (Línea 5).
- De aquí se procede a realizar la transformación del documento BPEL a un modelo de grafos, como ya se menciono utilizando el transformador propuesto en [42] (Línea 6).
- Obtenido el grafo se ubica en el primer nodo con el atributo `partnerLink` igual al nombre del `partnerLink` sobre el cual se está realizando el particionamiento (Línea 7).
- Ubicado este nodo, se verifica si el subproceso a crear es el iniciador del proceso de negocio (es decir, si está definido después de la primera actividad del proceso representada en la figura 2 con el nodo `Start`) (Línea 8), si esto es falso se debe insertar una actividad tipo `receive de sincronización` (al primer subproceso a crear no se le genera actividad `receive de sincronización` debido a que es el primero en ejecutarse en el proceso general), para esto se realiza lo siguiente:

  Sobre el nodo ubicado, se obtiene la información de él y se crea una actividad `receive` con la siguiente información: El atributo *nombre de la actividad* igual a `sincronizacion`, los atributos *partnerLink, portType, operation* y *variable* se asignan según la información obtenida del nodo.

Esta actividad se agrega al objeto XML para que el subproceso inicie su ejecución desde esta actividad `receive de sincronización`. En el caso de que sea el primer subproceso a crear no se inserta actividad de sincronización y se realiza el paso de la Línea 11.

- A continuación se procede a llamar al algoritmo ***Reglas de Particionamiento*** (Línea 11).

El algoritmo 2 Reglas de Particionamiento se presenta en pseudocódigo y expone los pasos para realizar el particionamiento del proceso basado en las reglas básicas de particionamiento y sincronización descritas en [8].

El algoritmo ***Reglas de Particionamiento*** es llamado por cada iteración ejecutada en la línea 3 del algoritmo 1. Las entradas del algoritmo 2 son el Grafo (G_{BPEL}) que representa el proceso BPEL, el nodo ($Nodo_i$) que representa la actividad que se va a analizar y el objeto XML ($SubBPEL_{XMLi}$) correspondiente a las actividades del `partnerLink` sobre el cual se está particionando. Como salida se obtiene un objeto XML ($SubBPEL_{XMLi}$) con las actividades propias del `partnerLink` correspondientes al subproceso.

Algoritmo 2: Reglas de Particionamiento

INPUTS: G_{BPEL}, $Nodo_{i...j}$, $SubBPEL_{XML1...n}$.
OUTPUT: $SubBPEL_{XML}$

```
1 BEGIN
2   IF Tipo actividad que representa Nodoi es una actividad tipo End THEN
3         Finalizar → ir a END
4   ELSE
5       IF Tipo actividad que representa Nodoi es una actividad básica THEN
6           IF Atributo partnerLink Nodoi = Nombre subproceso a particionar Ó Nodoi no tiene atributo partnerLink  THEN
7                       Obtener información correspondiente a etiqueta del  Nodoi (nodo actual)
8                       Insertar información obtenida  en SubBPELXMLi
9                       Ubicar siguiente nodo
10                      Llamar a Reglas Particionamiento con nodo ubicado en línea anterior
                          (Los otros parámetros  iguales)
11          ELSE
12                Insertar actividad invoke de sincronización en SubBPELXMLi con los siguientes datos:
                      TipoActividad = Invoke
                      Nombre Actividad = sincronizacion
                      PartnerLink = Obtener nombre del PartnerLink del Nodo
                      PortType = Obtener atributo PartnerLink Nodo actual
                      Operación = Obtener atributo PartnerLink Nodo actual
13                Finalizar → ir a END
14          END IF
15      ELSE
16          SWITCH Tipo actividad que representa Nodoi
17            CASE "Switch"
18                  Crear objeto XML para contener las actividades del Switch (XMLSwitch)
19                  Recorrer desde StarSwitch hasta EndSwitch y obtener actividades, aplicando las reglas definidas del Switch
20                  Insertar actividades en objetoXML Switch
21                  Insertar de XMLSwitch al objeto XML SubBPELXMLi
22            CASE "While"
                                …
23            CASE "Pick"
                                …
24            CASE "Flow"
                                …
25          END SWITCH
26      END IF
27      Ubicar siguiente nodo
28      Llamar a  Reglas Particionamiento con  nodo ubicado  (Los otros parámetros iguales)
29    END IF
30 END
```

El algoritmo hace un llamado recursivo que permite hacer un recorrido del grafo y realiza acciones específicas según el nodo del grafo donde se está ubicado. La función de salida de esta recursividad se da cuando se encuentra un nodo que represente una actividad de finalización (Línea 2).

- Si no se cumple la condición de salida, se verifica si el nodo representa una actividad básica (Línea 5). Si esta condición se cumple se procede a verificar si el atributo `partnerLink` de este nodo corresponde al `partnerLink` del subproceso que se está particionando (Línea 6), esto se hace con el objetivo de discriminar si la actividad que representa el nodo pertenece al subproceso. Si esto se cumple se obtiene toda la información del nodo, se crea la etiqueta correspondiente con la información obtenida y se agrega al *objeto XML* que representa el subproceso. Esto también se realiza en caso de que la actividad básica no tenga un `partnerLink` asociado (como por ejemplo en una actividad `<assing>`) Línea 7 a 10.

- Por el contrario, si el atributo `partnerLink` del nodo no corresponde al `partnerLink` del subproceso que se está particionando, se crea una actividad tipo `invoke de sincronización` con la siguiente información: El atributo *nombre de la actividad* igual a `sincronizacion`, los atributos *partnerLink, portType, operation* y *variable* se obtienen de la información del nodo. Esta actividad se agrega al *objeto XML*, para poder llamar al subproceso siguiente en la ejecución (Línea 11 a 13).

- Si se está ubicado sobre un nodo que representa una actividad estructurada, se aplican las reglas de particionamiento correspondientes a esa actividad. Para ello se recorren los nodos desde el `StarController` hasta `EndController,` y según las actividades que representan los nodos encontrados en esté recorrido, se van aplicando las reglas como se explicó anteriormente (Línea 16).

  Por ejemplo si es un nodo que representa la actividad estructurada `<switch>,` se crea un *objeto XML* con la información del nodo `Switch,` y se recorren todos los nodos, desde el nodo *Start Switch* hasta el nodo *End Switch,* agregando las actividades correspondientes, al *objeto XML del Switch*. Así por cada nodo en este recorrido se verifica el tipo de actividad que representa; además se verifica si está relacionado con el `partnerLink` sobre el cual se está particionando y si es el caso se agregan las actividades de sincronización, estas condiciones son similares a las presentadas en las líneas 5 a 14.

  Cuando termina el recorrido de los nodos de la actividad estructurada, se agrega el *objeto XML del Switch* al *objeto XML* que contiene todas las actividades del subproceso. De la misma forma se procede para las diferentes actividades estructuradas, considerando las reglas correspondientes (Línea 21).

  En las líneas 16 a 25 se presentan únicamente los pasos a seguir para la actividad estructurada `<switch>,` debido a la similaridad de estos para cada tipo de actividad.

Vale la pena aclarar que estos dos algoritmos requieren uno del otro para poder realizar el proceso de particionamiento, debido a que para obtener las salidas del algoritmo 1 se requiere realizar las sentencias expuestas en el algoritmo 2 y de igual manera el algoritmo 2 requiere de las entradas que son generadas en el algoritmo 1.

Además es importante mencionar que con los algoritmos Inicio del Particionamiento y Reglas de Particionamiento se obtienen subprocesos con las actividades propias del `partnerLink` en formato XML mediante los objetos XML, pero para la correcta ejecución de los subprocesos en un motor BPEL, estos deben contener información de cabeceras o inicio del proceso similar a las del proceso original (como la que se presenta en las líneas 1 a 13 de la figura 25) y posteriormente estos objetos deben ser almacenados como archivos BPEL.

Para crear la cabeceras luego de obtenido el *objeto XML* del subproceso, se procede a leer nuevamente el documento BPEL y se extrae la información de la etiqueta `<process>,` la cual contiene la cabecera

con los *espacios de nombres (namespace)*, `partnerLinks` y `variables`, que se insertan en cada uno de estos *objetos XML*. Finalmente se crea el archivo BPEL correspondiente al *objeto XML* obtenido y se guarda según el nombre del `partnerLink` sobre el cual fue particionado.

En la figura 25, se presenta un ejemplo de un proceso BPEL, el cual recibe como entrada dos datos numéricos, y según una comparación de estos datos, realiza la invocación a un servicio Web para sumarlos o restarlos. Al aplicarle a este proceso BPEL los algoritmos y mecanismo descritos anteriormente, se obtiene los subprocesos presentados en la figuras 27, 28 y 29 cuyos nombres son *Operacion, Sumadora y Sustractora* respectivamente correspondientes con los nombres de los partnerLinks definidos en el proceso principal.

Cada uno de estos subprocesos contiene las actividades respectivas a cada `partnerLink`, donde se nota que el primer subproceso *Operacion* (Figura 27) no tiene definida una actividad `receive de sincronización` debido a que es el iniciador del proceso. Además se presenta que por cada actividad que no estaba relacionada con este `partnerLink`, se ingresan actividades `invoke de sincronización` con la información necesaria para llamar al siguiente proceso en la ejecución. Los subprocesos *Sumadora* (Figura 28) y *Sustractora* (Figura 29) a diferencia del subproceso *Operacion* si cuentan con actividades `receive de sincronización`, ya que requieren de la información del proceso anterior para realizar sus actividades. En **anexo D** se presenta un ejemplo detallado del funcionamiento de este algoritmo con dicha entrada.

Finalmente se tiene que para el particionamiento y sincronización interviene en la plataforma MobFlow presentada en la figura 22, el módulo ***Operativo*** específicamente el módulo **Particionar Proceso,** en cuyas clases se implementan los algoritmos Inicio Particionamiento y Reglas Particionamiento. También para este mecanismo interviene el componente transformador BPEL-Grafos y el módulo Gestión de Datos para la persistencia.

```
<process name="Promedio2Nums" targetNamespace="http://xmlns.oracle.com/Promedio2Nums"
            xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
            xmlns:ns1="http://localhost:8090/axis/Operadora.jws" xmlns:ns3="http://localhost:8090/axis/Restadora.jws"
            xmlns:ns2="http://localhost:8090/axis/Sumadora.jws" xmlns:as="http://xmlns.oracle.com/Promedio2Nums">
<partnerLinks>
   <partnerLink myRole="Operar" name="Operacion" partnerLinkType="ns1:Operacion_PL"/>
   <partnerLink myRole="Sumar" name="Sumadora" partnerLinkType="ns2:Sumadora_PL"/>
   <partnerLink myRole="Restar" name="Sustractora" partnerLinkType="ns3:Restadora_PL"/>
</partnerLinks>
<variables>
   <variable messageType="as:Request" name="Entrada" />
</variables>
<sequence name="main">
    <receive name="EntradaDatos" partnerLink="Operadora" portType="as:Opera" operation="operar" variable="Entrada"/>
      <switch name="DatoEntrada">
         <case condition="bpws:getVariableData('inputVariable','num1') > bpws:getVariableData('inputVariable','num1') ">
              <invoke name="sincronizacion" partnerLink="Sumadora" portType="as:Suma" operation="sumar" variable="Entrada"/>
         </case>
         <otherwise>
           <invoke name=" sincronizacion " partnerLink="Sustractora" portType="as:Resta" operation="restar" variable="Entrada"/>
         </otherwise>
      </switch>
</sequence>
</process>
```

Figura 27. Primer Subproceso resultante del particionamiento y sincronización Operacion.bpel

```
<process name="Promedio2Nums" targetNamespace="http://xmlns.oracle.com/Promedio2Nums"
            xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
            xmlns:ns1="http://localhost:8090/axis/Operadora.jws" xmlns:ns3="http://localhost:8090/axis/Restadora.jws"
            xmlns:ns2="http://localhost:8090/axis/Sumadora.jws" xmlns:as="http://xmlns.oracle.com/Promedio2Nums">
 <partnerLinks>
   <partnerLink myRole="Operar" name="Operacion" partnerLinkType="ns1:Operacion_PL"/>
   <partnerLink myRole="Sumar" name="Sumadora" partnerLinkType="ns2:Sumadora_PL"/>
   <partnerLink myRole="Restar" name="Sustractora" partnerLinkType="ns3:Restadora_PL"/>
 </partnerLinks>
<variables>
  <variable messageType="as:Request" name="Entrada" />
</variables>
<sequence name="main">
    <receive name="sincronizacion" partnerLink=" Sumadora " portType=" as:Suma " operation=" sumar " variable="Entrada"/>
    <invoke name="Suma" partnerLink="Sumadora" portType="as:Suma" operation="sumar" variable="Entrada"/>
 </sequence>
</process>
```

Figura 28. Segundo Subproceso resultante del particionamiento y sincronización Sumadora.bpel

```
<process name="Promedio2Nums" targetNamespace="http://xmlns.oracle.com/Promedio2Nums"
            xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
            xmlns:ns1="http://localhost:8090/axis/Operadora.jws" xmlns:ns3="http://localhost:8090/axis/Restadora.jws"
            xmlns:ns2="http://localhost:8090/axis/Sumadora.jws" xmlns:as="http://xmlns.oracle.com/Promedio2Nums">
  <partnerLinks>
    <partnerLink myRole="Operar" name="Operacion" partnerLinkType="ns1:Operacion_PL"/>
    <partnerLink myRole="Sumar" name="Sumadora" partnerLinkType="ns2:Sumadora_PL"/>
    <partnerLink myRole="Restar" name=" Sustractora " partnerLinkType="ns3:Restadora_PL"/>
 </partnerLinks>
 <variables>
   <variable messageType="as:Request" name="Entrada" />
 </variables>
 <sequence name="main">
     <receive name="sincronizacion" partnerLink=" Sustractora " portType=" as:Resta " operation=" restar " variable="Entrada"/>
     <invoke name="Resta" partnerLink=" Sustractora " portType="as:Resta" operation="restar" variable="Entrada"/>
  </sequence>
</process>
```

Figura 29. Tercer Subproceso resultante del particionamiento y sincronización Sustractora.bpel

4.2 DISTRIBUCIÓN DE SUBPROCESOS DE NEGOCIO EN SISTEMAS MÓVILES DE INFORMACIÓN

En la plataforma MobFlow la distribución de subprocesos de negocio en sistemas móviles de información, consiste en enviar a cada actor del proceso de negocio el respectivo subproceso para que este lo pueda ejecutar desde su Terminal móvil. Dicho subproceso contiene las actividades o tareas que debe ejecutar el actor en el proceso de negocio general. Para cumplir con este objetivo se creó un mecanismo de distribución que permite vincular un actor con un subproceso, para que este posteriormente pueda ser descargado.

Debido a que el mecanismo de distribución no requiere de funciones complejas (ya que la mayoría de aspectos de esté mecanismo implica acceso a base de datos) para este no se plantearon algoritmos como los expuestos en la sección anterior. A continuación se realiza una descripción del mecanismo planteado para la distribución de subprocesos de negocio en sistemas móviles de información, empezando por las bases teóricas que permitieron su definición.

4.2.1 Bases Teóricas

En esta sección se exponen algunas consideraciones adicionales con respecto a los *partnerLinks* de un proceso BPEL, ya que estos son primordiales para realizar aspectos como generación de subprocesos, vinculación de actores con subprocesos y descarga a los sistemas móviles de información, que a la vez componen las funciones requeridas para la distribución de subprocesos de negocio.

Elementos principales en la definición de procesos BPEL

La Definición de Procesos forma parte de los datos del Workflow, contiene, toda la información necesaria acerca de los procesos, incluye información de comienzo de actividades, condiciones y reglas de navegación. Mediante la definición de proceso, se puede modelar la navegación entre los procesos, proveer información acerca de entradas a procesos y criterios a tomar en cada paso de la navegación, asignación de tareas a usuarios entre otras. Los procesos de negocios son definidos 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 tenemos 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.

- Ejecutantes de las actividades. Estos vienen representados especialmente por 2 grupos. El primero esta representado los usuarios del servicio y el segundo por los servicios web. Estos dos grupos 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.

En la definición de procesos BPEL se puede observar que los *partnerLinks* juegan un papel importante, ya que se definen como ejecutantes de las actividades del proceso, por lo tanto los *partnerLinks* van directamente ligados a las actividades básicas y son quienes ejecutan dichas actividades, de esta manera se puede definir el conjunto de actividades que realizan determinados socios del proceso. Por esta razón, son utilizados para el presente trabajo de grado en el particionamiento de procesos, la vinculación de sus actores y posterior distribución en los dispositivos móviles.

***Particionamiento de procesos por* `partnerLinks`**

Como se mencionó en la sección anterior para el proceso de particionamiento en la plataforma MobFlow se utiliza los *partnerLinks*, donde el número de subprocesos generados es igual al número de *partnerLinks*, lo que conlleva a decir que los subproceso de negocio, representan las actividades que corresponde a un *partnerLink* específico. Teniendo en cuenta esto se puede observar que los *partnerLinks* además de permitir el particionamiento representan los ejecutantes de las actividades de cada subproceso, por esta razón en este proyecto el *rol* que juega cada *partnerLink*, es relacionado con el *rol* que cumple cada actor del subproceso, para que este pueda ser vinculado con el subproceso que va a ejecutar.

4.2.2 Mecanismos para la Distribución de Subprocesos de Negocio en Sistemas Móviles de Información

En esta sección se describe cómo se realiza la distribución de los subprocesos de negocio en los sistemas móviles de información. Para realizar esta distribución, se requiere que los subprocesos hayan sido generados mediante el particionamiento; de igual manera para este proceso de distribución se necesita vincular subprocesos con los actores que los ejecutan, y descargar los subprocesos de los actores en los dispositivos móviles. A continuación se da una descripción de cómo se realizan estas tareas.

Vinculación de actores a los subprocesos de negocio

El mecanismo de distribución inicia cuando se efectúa el particionamiento del proceso de negocio, dando lugar a la generación de subprocesos, los cuales son iguales al número de `partnerLinks` que contiene el proceso global. Los archivos generados correspondientes a los subprocesos son guardados en un directorio específico del servidor Web, el cual es creado según el nombre del proceso. La ubicación de los archivos, es almacenada en la tabla que contiene los datos de los subprocesos generados.

Posteriormente por cada `partnerLink` se obtiene el *rol* que juega en el proceso mediante el atributo "*myRole*" (obtenido de la etiqueta donde se define el `partnerLink`), cada *rol* se almacena igualmente en una tabla roles de la base de datos, los cuales corresponden en este mecanismo a los roles que cada actor cumple en la ejecución del proceso. Por consiguiente se procede a vincular los roles generados con sus respectivos responsables.

Para este proceso de vinculación se deben crear los diferentes actores del proceso, los cuales se consideran los trabajadores de la organización y son quienes realizarán la ejecución de los subprocesos desde los sistemas móviles de información. Estos actores se deben asociar con los roles creados en el proceso de generación de subprocesos de negocio, de esta manera se relaciona un actor con el *rol* que le corresponda desempeñar en la ejecución del proceso. La creación de estos actores es realizada de manera manual por el administrador del sistema.

Creados los actores se debe proceder a vincular cada uno de ellos con sus respectivos subprocesos de negocio de acuerdo al *rol* que estos desempeñan. Actividad que realiza el administrador del sistema.

Gracias a esta vinculación es posible consultar la información del subproceso, su ubicación física, proceso perteneciente y el actor encargado de su ejecución. Realizada la vinculación se les envía automáticamente a los actores participantes, un correo electrónico indicado que está disponible el subproceso para su descarga en el sistema móvil de información.

En la vinculación de los actores con subprocesos de negocio interviene el ***Módulo Gestión de Información de la Organización*** de la plataforma MobFlow presentada en la figura 22, que mediante las clases *ActoresBean*, *RolesBean* de este módulo, presentan las interfaces gráficas de usuario que permiten ingresar datos de los actores de la organización y asociarlos al rol que desempeñan en la ejecución del proceso. También interviene el ***Módulo Operativo*** y específicamente el módulo ***Vincular Subproceso***, que mediante la clase *VincularSubprocesoBean* presenta la interfaz gráfica de usuario para poder realizar la vinculación de cada actor con los subprocesos generados. La clase *VincularSubprocesoImpl* interactúa con las clases *RolesImpl*, *ActorImpl* y *SubprocersodeNegocioImp* para implementar la lógica requerida para esta funcionalidad. Finalmente las clases *Roles*, *Actor*, *SubprocersodeNegocio*, *VincularSubproceso* las cuales junto a las clases *RolesDAO*, *ActorDAO*, *SubprocersodeNegocioDAO* y *VincularSubprocesoDAO* del módulo ***Gestión de Datos*** almacenan en la base de datos la información requerida.

Una vez vinculado el subproceso mediante el ***Módulo Control de la Ejecución*** y específicamente el módulo **Gestión Correo Electrónico,** se hace el envió del mensaje de correo electrónico mencionado.

Descarga de subproceso de negocio a dispositivo móvil

Luego de realizado el proceso de vinculación de actores a subprocesos, es posible que estos obtengan los correspondientes procesos de negocio en sus sistemas móviles de información.

Para la descarga del subproceso, el actor del proceso debe hacer un registro desde el sistema móvil de información a la plataforma Web. El registro se hace mediante una interfaz gráfica de usuario (ver figura 30).

El actor desde la aplicación cliente móvil MobFlow, realiza el ingreso de su información de usuario (nombre de usuario y contraseña los cuales son establecidos cuando se crea el actor), estos datos son

enviados mediante HTTP a la aplicación Web MobFlow, la cual los recibe y los valida. Si la información remitida es correcta, se procede a realizar la búsqueda del subproceso de negocio asignado a este actor.

Posteriormente se obtiene la información de la ubicación del directorio, donde se encuentra almacenado el archivo BPEL del subproceso correspondiente a las tareas del actor que realizó la petición desde el móvil. En esta operación también se obtiene la ubicación del archivo WSDL del proceso general (requerido para la ejecución del subproceso).

Figura 30. Interfaz gráfica de usuario del dispositivo móvil para registro en la plataforma y descarga del correspondiente subproceso.

La información de la ubicación de estos archivos se convierte en formato de URL y es enviada a la aplicación cliente móvil MobFlow mediante HTTP, la cual realiza la descarga de estos archivos a partir de las URL obtenidas.

Finalizada la descarga de los archivos, estos son almacenados de manera persistente en el sistema móvil de información, utilizando el sistema de registros RMS [52]. Lo que permite que el actor no deba estar permanentemente conectándose al servidor Web MobFlow para descargar su subproceso.

En el sistema móvil de información también se almacena información del usuario (nombre de usuario y contraseña). Con esta información de usuario es posible obtener el proceso de negocio correspondiente al subproceso que ejecuta un actor específico, debido a que en la plataforma MobFlow pueden cargarse varios procesos de negocio y para cada uno se generan varios subprocesos.

Para realizar toda esta funcionalidad interviene el **Módulo Control de la Ejecución** de la plataforma MobFlow presentada en la figura 22, el cual mediante el submódulo de **Comunicación** y concretamente con la clase *Comunicacion* que implementa un *Servlet*, recibe todas las peticiones enviadas desde los sistemas móviles de información (a través del cliente móvil MobFlow), en éste caso son peticiones de registro y descarga. Estas son enviadas a la clase *ValidarComunicacionImpl* que valida los datos del actor que hace la petición y retorna el identificador del proceso, si la información no es válida, se retorna a través de la clase *Comunicacion* un mensaje de error al sistema móvil. Por otro lado si la información es válida, la clase *DistribucionImpl* del submódulo **Distribución** obtiene la ubicación física de los archivos correspondientes al actor y la devuelve por medio de la clase *Comunicacion* en formato URL al dispositivo para que pueda ser descargado por el mismo.

El acceso a la base de datos se realiza utilizando las clases *ValidarComunicacion* y *Distribucion*, junto con las clases *ValidarComunicacionDAO* y *DistribucionDAO* del módulo **Gestión de Datos**.

Para la descarga de los subprocesos a los dispositivos móviles en la aplicación cliente móvil, interviene el módulo ***Vista*** con la clase *frmDatosUsuario* del submódulo **GUI**, está presenta la interfaz de usuario para que el actor del proceso ingrese los datos de usuario y los envié.

También interviene el módulo **Lógica** con el submódulo **Comunicación** el cual a través de la clase *conexion* realiza la conexión con la aplicación Web y con la clase *obtenerProceso* descarga los archivos correspondientes al actor mediante la URL obtenida. Finalmente interviene la clase *persistenciaRMS* de módulo **Persistencia** que almacena la información requerida en el sistema móvil de información.

Las funciones descritas componen el mecanismo para distribución de procesos de negocio en los sistemas móviles de información; con estos se logra distribuirle a cada actor el respectivo subproceso de negocio, que contiene las tareas propias a ejecutar por dicho actor.

4.3 EJECUCIÓN SINCRONIZADA DE LOS SUBPROCESOS DE NEGOCIO

Una vez generados los subprocesos de negocio y distribuidos a los sistemas móviles de información, es posible iniciar la ejecución distribuida de los subprocesos. Esta ejecución debe seguir un flujo con un orden específico según como se definió en el proceso general, lo cual es posible lograr gracias a la inclusión de las actividades de sincronización en la generación de los subprocesos, como lo propone la regla 2 presentada en las bases teóricas de la sección anterior 4.1.1 (Reglas de particionamiento y sincronización). La cual plantea que cada vez que una actividad controlada por un actor A1 sea seguida por otra actividad controlada por un actor A2, se introducen nuevas tareas para el control del proceso de delegación.

En esta sección se describe los mecanismos y algoritmos planteados para realizar la ejecución de las actividades de sincronización incluidas en los diferentes subprocesos de negocio, que garantizan la ejecución sincronizada de los subprocesos de negocio de forma similar a la ejecución del proceso inicial. Además se describe como se envía la información de las variables requeridas por los diferentes dispositivos para ejecutar los subprocesos de negocio.

Estos dos aspectos se realizan con el fin de lograr así la sincronización de los sub-procesos de negocios, que han sido particionados y distribuidos por la plataforma (ejecución sincronizada de los subprocesos de negocio). A continuación se hace una descripción de los algoritmos y mecanismos planteados para la ejecución sincronizada de subprocesos de negocio, empezando por las bases teóricas que permitieron definirlos.

4.3.1 Bases Teóricas

Sincronización

La sincronización de la ejecución distribuida de procesos de negocio en los sistemas móviles de información, es uno de los puntos más críticos del sistema, ya que es responsable 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, el proceso pierde su estructura normal, por lo cual en [4], [5], [8] y [9] sugieren una sincronización basadas en la inclusión de dos actividades especiales Receive e Invoke, con las cuales se conserva de manera consistente el estado del proceso, junto con el estado de sus variables mediante un método de comunicación apropiado. 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 adecuada, el sistema pierde su secuencia y el objetivo de negocio se verá imposible de cumplir. Por esta razón, en la ejecución de subprocesos, los participantes deben entenderse de forma que se siga el flujo de la ejecución, considerando que la información de las variables del proceso sea enviada entre cada subproceso que las requiera, utilizando para ello medios eficientes de comunicación y ampliamente utilizados por la mayoría de los dispositivos móviles.

Variables y mensajes en BPEL

En un proceso BPEL los mensajes y variables locales usadas por el flujo son en formato XML [53]. Estos mensaje y variables son definidos con esquemas XML, usualmente en el archivo WSDL (como lo requiere los procesos cargados en la plataforma MobFlow) para el flujo mismo o para los servicios que el invoca.

Por ejemplo para el proceso presentado en la figura 25, la variable *Entrada* se define en un esquema XML tipo *Input* en el archivo WSDL de dicho proceso, tal como se visualiza de la línea 8 a la 11 de la figura 31, la cual presenta parte del archivo WSDL correspondiente al proceso BPEL presentado en la figura 25.

En este caso la variable *Entrada* consta de dos parámetros numéricos tipo *double*, llamados *num1* y *num2*. Al realizar el proceso de asignación de datos a la variable *Entrada,* internamente el motor BPEL genera un documento XML con la información respectiva a esta asignación, este documento XML es similar al presentado en la figura 32, donde el primer parámetro numérico de la variable *Entrada* (*num1)* tiene asignado el valor de 6 y al segundo parámetro numérico (*num2)* tiene el valor de 4.

En el caso de una ejecución distribuida de los subprocesos de negocio presentado en las figuras 27,28 y 29, el primero en la ejecución es el subproceso *Operación.bpel* (figura 27). En la ejecución de esté se realiza el proceso de asignación de la variable E*ntrada*. Para el inicio de la ejecución ya sea del subproceso *Sumadora.bpel* (figura 28) o *Sustractora.bpel* (figura 29), los datos asignados a la variable *Entrada* es decir el documento XML de la figura 32, debe ser enviado al sistema móvil de información del actor que va a ejecutar dichos subprocesos, para así poder iniciar la ejecución.

```
1<wsdl:definitions name=" Operaciones " targetNamespace="http://xmlns.oracle.com/ Operaciones /"
2    xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/" xmlns:tns="http://xmlns.oracle.com/ Operaciones /"
4    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
6    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/">

8    <wsdl:message name="Input">
9       <wsdl:part name="num1" type="xsd:double" />
11      <wsdl:part name="num2" type="xsd:double" />
11   </wsdl:message>

12   <wsdl:message name="Output">
13      <wsdl:part name="res" type="xsd:double" />
14   </wsdl:message>

15   <wsdl:portType name="Opera">
16      <wsdl:operation name="operar">
17          <wsdl:input message="tns: Input "/>
18          <wsdl:output message="tns: Output "/>
19      </wsdl:operation>
20   </wsdl:portType>
.
.
.
```

Figura 31. Ejemplo de archivo WSDL correspondiente al proceso BPEL presentado en la figura 22

```
<n0: operar xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:n0="http://xmlns.oracle.com/Operaciones/">
            <num1 xsi:type="xsd:double">2</num1>
            <num2 xsi:type="xsd:double">6</num2>
</n0: operar>
```

Figura 32. Documento XML de asignación de datos a una variable

4.3.2 Ejecución Sincronizada de los Subprocesos de Negocio

Una vez los actores han descargado su subprocesos de negocio, pueden iniciar la ejecución de sus tareas en el sistema móvil de información. La ejecución de cada subprocesos debe realizarse en un orden específico hasta culminar el proceso global, esto se logra gracias a las actividades de sincronización ingresadas en los subprocesos. Para esta ejecución se propone el algoritmo 3: ***Ejecución actividades sincronización***, el cual se presenta en pseudocódigo y expone los pasos seguidos para realizar la ejecución de los subprocesos en el sistema móvil de información, principalmente enfocándose en tratar los aspectos requeridos para la sincronización.

El algoritmo recibe como entrada el subproceso BPEL y el archivo WSDL del proceso general (*XML_{BPEL}, XML_{WSDL})*. Como salida se obtiene un mensaje para imprimir en pantalla el resultado de la ejecución del subproceso (*Operación Exitosa – Operación NO Exitosa*). A continuación se describe paso a paso el funcionamiento del algoritmo que hace parte del mecanismo de ejecución sincronizada.

Algoritmo 3: Ejecución actividades sincronización

INPUT: BPEL **XML**, WSDL**XML**
OUTPUTS: String Mensaje
1 **BEGIN**
2 Inicializar variables: *VariableRequerida* (Objeto XML)← NULO.
MensajeSalida (String) ← NULO.

3 Obtener la primera actividad del proceso BPEL **XML**
4 **IF** (Tipo de primera Actividad del proceso BPEL **XML** = *receive* **Y** Atributo **Nombre** primera Actividad del proceso BPEL **XML** = *sincronizacion*)
5 Obtener atributo **variable** de primera Actividad del proceso BPEL **XML**
6 Enviar petición a plataforma Web solicitando variables de sincronización agregando como parámetro en la petición atributo **variable** obtenido en el paso anterior
7 Obtener respuesta de plataforma Web
8 **IF** (Respuesta obtenida NO es mensaje de ERROR)
9 Asignar a *VariableRequerida* mensaje obtenido de plataforma Web
VariableRequerida ← Respuesta obtenida de plataforma Web
10 **ELSE**
11 Asignar a *MensajeSalida* valor de "Error de ejecución"
MensajeSalida ← "Error de Ejecución"
12 Imprimir en pantalla *MensajeSalida*
13 Finalizar Algoritmo. Ir a línea 24
14 **END IF**
15 **END IF**
16 Ejecutar actividades del proceso BPEL **XML** junto con WSDL**XML** en motor BPEL teniendo en consideración que:
- Si fue valida la condición de la sentencia 4, ejecutar Actividad 2 del proceso BPEL **XML** asignando lo obtenido en *VariableRequerida*. a los datos de variable de esta actividad.
- Si existen procesos de asignación de datos a variable en la ejecución de actividades del proceso BPEL **XML** registrarlos en la plataforma Web.
- Al ejecutar una actividad *invoke o receive* (no de sincronización) o reply que requiera invocación de un Servicio Web enviar petición a plataforma Web solicitando ejecución del servicio, obteniendo los siguiente parámetros para el envió en la petición.
  - **URL** donde está desplegado el Servicio Web
  - Atributo *operation* de la etiqueta
  - Parametros obtenidos de los datos asignados a la variable relacionada con la actividad invoke actual.
- Si actividad es invoke de sincronización ir a línea 17.

17 **IF** (Tipo Actividad del proceso BPEL **XML** = *invoke* **Y** Atributo **nombre** de Actividad =*sincronizacion*)
18 Obtener atributo **PartnerLink** de ActividadActual
19 Enviar petición a plataforma Web solicitando número de dispositivo de siguiente en la ejecución agregando parámetro PartnerLink
20 Número ← Obtener Respuesta de plataforma Web
21 Enviar SMS a Número Obtenido indicando que debe iniciar la ejecución de su subproceso
22 **END IF**
23 Finalizada la ejecución del proceso presentar mensaje de ejecución exitosa
MensajeSalida ← "*Ejecución Exitosa*"
Imprimir en pantalla *MensajeSalida*
24 **END**

- En primer lugar se crean las variables *VariableRequerida* y *MensajeSalida*. Donde la variable *VariableRequerida* se define como tipo *Objeto XML*, y es usada para recibir la información de las variables necesarias al dar inicio la ejecución del subproceso (en el proceso BPEL estas variables se manejan en formato XML, por esta razón se crea un *Objeto XML* para almacenar su información). Por otra parte la variable *MensajeSalida* se define como String (cadena de caracteres) y es usada en la salida del algoritmo, la cual indica si fue EXITOSA o NO la ejecución de un subproceso. Estas variables se inicializan con valores nulos (Línea 2).

- Luego se obtiene la primera actividad del proceso BPEL (BPEL **$_{XML}$)** (Línea 3).

- En caso que la primera actividad del subproceso BPEL (BPEL **$_{XML}$)**, sea una actividad tipo `receive de sincronización` (cumple la condición de la línea 4), se procede a obtener el atributo variable asociado a esa actividad, el cual contiene el nombre de la variable que requiere el subproceso para dar inicio a su ejecución (Línea 5). Posteriormente se realiza una petición al servidor Web MobFlow con el fin de obtener la información de la variable requerida, para esto se envía como parámetro el atributo *variable* de la actividad `receive` (Línea 6) y se espera por la respuesta de la plataforma (Línea 7). (En la ejecución de los subproceso los valores que toman las variables se envían el servidor web MobFlow para su registro, esto se menciona más adelante en esta sección).

- Si la respuesta del servidor Web MobFlow NO genera un mensaje de Error (Línea 8), se ingresa la información obtenida a la variable *VariableRequerida,* para que luego sea utilizada por las diferentes actividades que la requieran (Línea 9).

- En caso que la plataforma retorne un mensaje de Error, se asigna a la variable *MensajeSalida* ← "*No es momento de iniciar la ejecución"* (Mensaje de Error) con el fin de que sea presentada en pantalla, para indicarle al usuario que no ha sido notificado por el subproceso anterior para dar inicio a su ejecución (Líneas 11 - 13). (Cada vez que un subproceso termina la ejecución, le notifica a través de un mensaje de texto al siguiente móvil que debe iniciar su ejecución. Esto se explica más adelante).

- La línea 16 representa la ejecución del proceso de negocio BPEL en el motor, teniendo en cuenta que para esto se requiere el archivo WSDL asociado. Este proceso de ejecución no se describe completamente debido a su complejidad, extensibilidad y a que está ligada directamente al motor BPEL, por esta razón únicamente se mencionan los aspectos tenidos en consideración para lograr la ejecución sincronizada de los subprocesos de negocio, los cuales son los siguientes:

    - Si fue válida la condición de la línea, 4 se procede a ejecutar la siguiente actividad del proceso BPEL pasándole los datos de la variable obtenidos, los cuales se requiere para que sea ejecutada la actividad.

    - En los procesos de asignación de datos a variables, se realiza el envió de la información a la plataforma Web para su registro, de manera tal que pueda ser utilizada en la ejecución de otro subproceso (si es requerido). La información enviada corresponde al documento XML con los datos de la variable y el nombre de esta.

    - Para llevar a cabo la invocación de servicios Web por el proceso en ejecución, se hace una petición a la plataforma Web con los siguientes parámetros.

        - ✓ **URL** donde está desplegado el Servicio Web.

        - ✓ Atributo `operation` de la etiqueta correspondiente a la actividad que requiere la invocación.

        - ✓ Parámetros para la invocación, obtenidos de la variable asociada a la actividad que hace la invocación.

- En el momento de ejecutar una actividad *invoke de sincronización*, se procede a obtener el atributo *partnerLink* relacionado a la actividad, y se realiza una petición a la plataforma Web, con el fin de obtener el número del dispositivo móvil siguiente en la ejecución. Obtenido el numero se hace el envió de un mensaje de texto para indicarle al siguiente dispositivo que debe iniciar la ejecución de su subproceso (Línea 17 - 21).

- Finalmente se presenta en pantalla la información de la variable *MensajeSalida*, para indicar al usuario si fue exitosa o no la ejecución del subproceso en el sistema móvil de información.

Si se tiene como entrada al algoritmo 3 el subproceso BPEL presentado en la figura 27 Operacion.bpel, al ejecutar la primera actividad (*<receive name="EntradaDatos" partnerLink="Operadora" portType="as:Opera" operation="operar" variable="Entrada"/>*) ya que no es una actividad de sincronización, es manejada por el motor BPEL, en el caso de esta actividad se realiza un proceso de asignación a la variable "*Entrada*", según los datos introducidos por el usuario, los cuales son enviados al Servidor Web para su registro. Luego se continua con la ejecución del subproceso y debido a la actividad *<Switch>,* según la entrada de datos de la primera actividad *(<receive name="EntradaDatos"…*) se ejecutan alguna de las dos actividades de sincronización (*<invoke name="sincronizacion" partnerLink="Sumadora" portType="as:Suma" operation="sumar"* ó *<invoke name=" sincronizacion " partnerLink="Sustractora" portType="as:Resta" operation="restar" variable="Entrada"/>)*.La ejecución de estas actividades cumple con la condición de la línea 18 del algoritmo 3, por lo cual se obtiene el atributo *partnerLink* de la actividad en cuestión (en este caso sería *Sumadora* ó *Sustractora* según la entrada de datos*)* el cual es enviado al Servidor Web para obtener el numero de dispositivo móvil asociado a este *partnerLink*, con el fin enviar el mensaje de texto a dicho dispositivo. En el **anexo D** se presenta un ejemplo detallado del funcionamiento de este algoritmo.

A continuación se describe el mecanismo para la ejecución sincronizada de los subprocesos de negocio, el cual expone la manera como interactúa la aplicación cliente móvil y la aplicación Web para lograr la ejecución. Este mecanismo se presenta a partir de 3 escenarios de ejecución que son: *escenario de inicio de la ejecución del proceso, escenario de ejecución no Inicial* y *escenario de Invocación de servicio Web*, estos se describen teniendo en cuenta los aspectos tratado en el algoritmo 3.

- **Escenario de Inicio de la ejecución del proceso.** Este caso se da cuando un actor es el encargado de iniciar la ejecución de un proceso de negocio (el primer subproceso), en donde la primera actividad definida en su subproceso no es de sincronización, sino una actividad *<receive>* para el ingreso de los datos iniciales del proceso, en este caso la aplicación cliente móvil MobFlow utilizando el motor BPEL Sliver genera una interfaz gráfica correspondiente a estos datos requeridos.

  Después que el usuario ingresa los datos que corresponden a sus actividades respectivas, estos son asignados a una variable para continuar con la ejecución de las tareas del proceso correspondiente. Cada vez que se realiza la asignación de datos a variables, ya sea por ingreso de datos del usuario, por asignación mediante la etiqueta *<assing>*, o por la ejecución de una actividad básica, la información de los datos asignados es enviada por el cliente móvil a la plataforma Web, la cual los almacena en la base de datos, con el fin, de que luego pueda ser obtenida por los diferentes sistemas móviles que la requieran al iniciar la ejecución de sus subprocesos. Para minimizar el número de conexiones entre el cliente móvil con el servidor Web, los datos de las variables son enviadas al servidor Web cuando se realiza la ejecución de las actividades <receive>, <invoke> y <reply>, en el momento que estas soliciten una conexión con la plataforma Web.

  El subproceso termina cuando el motor BPEL ejecuta una actividad de *sincronización de tipo invoke*, (la cual cumple la condición planteada en la Línea 18 del algoritmo), en ese momento la aplicación cliente móvil MobFlow realiza una petición al servidor Web MobFlow enviando la información del atributo *partnerLink* relacionado a la actividad, para obtener el número del dispositivo móvil que continúa en la ejecución del subproceso. Posteriormente se envía un mensaje de texto a dicho dispositivo anunciándole que debe iniciar la ejecución de su subproceso.

En resumen, cuando se termina la ejecución del subproceso, se le envía un mensaje de texto al dispositivo móvil del siguiente actor en el flujo, indicándole que debe iniciar la ejecución de su subprocesos de negocio.

- **Escenario de Ejecución no Inicial.** Este caso se da cuando un actor no es el encargado de iniciar la ejecución del proceso y requiere datos de variables asignados en la ejecución de procesos anteriores. La primera actividad definida en su subproceso es una actividad de sincronización tipo receive (la cual cumple la condición planteada en la Línea 6 del algoritmo).

  Al ejecutar esté tipo de actividad la aplicación móvil obtiene la información de los atributos *partnerLink* y *variable* de la etiqueta de la actividad. De aquí procede a realizar una petición HTTP al servidor Web MobFlow enviando la información.

  Cuando el servidor Web MobFlow recibe una petición de este tipo, realiza la búsqueda en las variables que han sido registradas en la ejecución de subprocesos previos, para que devuelva los datos asignados de la variable solicitada. Esta búsqueda se realiza por el nombre de la variable

  Si es encontrada la información requerida se obtiene los datos asignados a dicha variable, es decir el documento XML que contiene la información de la asignación y se envía a la aplicación cliente móvil MobFlow que hizo la petición. En caso de que no sea encontrada, se devuelve un mensaje de error indicando problemas en la ejecución, esto último generalmente sucede cuando se realiza la ejecución del subproceso sin la previa notificación, es decir se ejecuta el subproceso antes del orden establecido por el flujo.

  Una vez el sistema móvil de información obtiene los datos de variable requeridos, le presenta al usuario una interfaz gráfica con esta información y con las opciones para seguir con la ejecución de su subproceso.

  Finalmente, si el subproceso al terminar de ejecutar las actividades del flujo NO encuentra una actividad de sincronización tipo *invoke de sincronización*, la aplicación móvil determina que ha finalizado la ejecución del proceso distribuido y le envía información a la plataforma indicándole que finalizo la ejecución de este proceso, de igual manera se realiza cuando al ejecutar una actividad *<terminate>*. En caso contrario envía el mensaje de texto al siguiente dispositivo como ya se ha explicado.

- **Escenario de Invocación de servicio Web.** Este caso se da cuando una actividad básica (`<receive>` `<invoke>` y `<reply>`) del proceso en la ejecución, solicita el llamado a un servicio Web, para lo cual la aplicación cliente móvil MobFlow obtiene la *URL* donde está desplegado el servicio Web, la operación a realizar con ese servicio y los parámetros que se requieren. (Información obtenida del subproceso BPEL y del archivo WSDL).

  Con esta información hace una petición a MobFlow con el fin de que realice dicha invocación y devuelva el resultado en un formato adecuado para continuar la ejecución del subproceso. Una vez MobFlow recibe esta información utiliza *Apache Axis*, el cual permite realizar la invocación a diferentes servicios Web. Esto es posible gracias a la información enviada desde el móvil (*URL* del Servicio Web a invocar, la operación del servicio y los parámetros de entrada a este servicio Web).

  Obtenido el resultado de la invocación de un servicio Web, esté se convierte en formato XML para que lo entienda el motor BPEL, el formato es similar al presentado en la figura 33. La información del resultado de la invocación en formato XML, se almacena en la base de datos en caso de que algún otro sistema móvil de información lo requiera para la ejecución de su subproceso.

  Finalmente se le envía esta información al dispositivo móvil que hizo la petición para que continúe con la ejecución de su subproceso de negocio. Cuando la invocación del servicio Web es de naturaleza asíncrona no es necesario realizar este proceso, solo se hace la invocación al servicio como se describió anteriormente.

La invocación al los servicios Web se realiza desde MobFlow, debido a que el motor BPEL Sliver para dispositivos móviles[2], solo hace llamados a servicios que estén desplegados en otros dispositivos móviles que también tengan el motor BPEL Sliver instalado. En donde estas invocaciones son realizadas mediante comunicaciones tipo sockets/TCP y Bluetooth, lo que presenta una limitación del motor BPEL utilizado. Por lo que se considero que la invocación de los servicio Web requeridos por los subprocesos se realice desde la aplicación Web MobFlow, lo que permite llamadas a servicios Web que estén localizados en cualquier lugar de la red. (Por limitaciones del motor BPEL también se implementaron las clases que contiene el paquete actividades presentado en la sección 3.1.4.1.4).

Es importante aclarar que en todos los procesos de comunicación realizados entre el sistema móvil de información y el servidor Web, se envían siempre los datos de usuario junto con la información enviada requerida por el mecanismo, de esta manera se identifica a qué proceso en ejecución pertenece el actor mediante la identificación del proceso.

Para este mecanismo de ejecución sincronizada, en la aplicación cliente móvil MobFlow intervienen todos los módulos de dicha aplicación los cuales fueron presentados en la figura 23. Los subproceso que previamente habían sido almacenados, son obtenidos a través de la clase *persistenciaRMS* del módulo **persistencia**. Cuando se inicia la ejecución del subproceso, con las clases *motorBPEL, manejoWhile, manejoPick, manejoSwitch, manejoReply, manejoReceive* y *manejoInvoke* del submódulo **Motor BPEL Sliver,** junto con las clases *controlEjecucion controlActividades* del submódulo **Control** (submódulos contenidos dentro del módulo **Lógica**) se manejan los aspectos lógicos para realizar la ejecución y sincronización de los subprocesos BPEL (con archivo WSDL), la clase *controlEjecucion* implementa el algoritmo 3. Estas clases interactúan con las clases *frmDatosSincronizacion, frmActividades* del módulo **Vista**. Estas a su vez presentan los datos de sincronización que recibe el usuario al iniciar un proceso y generan una interfaz de usuario dinámica para la realización de las tareas concernientes por parte del actor. Las clases del submódulo **Motor BPEL Sliver** también interactúan con las clases *conexión* y *envioSMS* del submódulo **Comunicación** las cuales establecen la comunicación con el servidor Web y envían mensaje de texto al siguiente móvil en la ejecución respectivamente.

En la aplicación Web MobFlow interviene el **Módulo Control de la Ejecución** de la plataforma presentada en la figura 22, el cual mediante el submódulo de **Comunicación** y concretamente con la clase *Comunicacion* que implementa un *Servlet*, recibe todas las peticiones enviadas desde los sistemas móviles de información, en éste caso son peticiones de sincronización. Estas son remitidas a la clase *ValidarComunicacionImpl* que con el nombre de usuario y contraseña (enviados junto con la petición de sincronización), se valida y determina a que proceso pertenece el subproceso ejecutado por el usuario que realiza la petición. Una vez efectuada esta validación, la clase *SincronizacionImpl* del submódulo **Sincronización** registra u obtiene las variables manejadas por los diferentes subprocesos, y con clase M*anejoVariable* se manipulan estas variables para obtener información de ellas. Del submódulo de **Sincronización** también interviene la clase *GestionMensajeriaImpl,* la cual obtiene el número del dispositivo siguiente en la ejecución luego que el dispositivo realiza una petición de sincronización, igualmente interviene el submódulo **Interfaz de Servicios,** que con la clase *interfazdeServicio* y el componente ***Apache Axis,*** realizan la invocación de servicios Web. El submódulo de **Sincronización** accede a la base de datos utilizando las clases *Sincronizacion* y *GestionMensajeria*, junto con las clases *SincronizacionDAO, GestionMensajeriaDAO* del módulo **Gestión de Datos**.

[2] Existen librerías de Sliver para computadores de escritorio, las utilizadas en este proyecto son para la ejecución de procesos BPEL únicamente en dispositivos móviles

4.4 MONITOREO DEL ESTADO DE LA EJECUCIÓN DISTRIBUIDA DE LOS SUBPROCESOS DE NEGOCIO

Una vez ejecutados los subprocesos de negocio en los dispositivos móviles es posible llevar a cabo el monitoreo del proceso. Este monitoreo permite visualizar el estado actual de la ejecución distribuida del proceso de negocio global, es decir presenta qué actividades de los subprocesos se han ejecutado y quien las ejecutó.

4.4.1 Bases Teóricas

Para definir el mecanismo de monitoreo de la ejecución distribuida de los subprocesos de negocio, se siguió la metodología de registro de actividades propuesta en la arquitectura de Workflow distribuidos JITIK (Just-in-Time Informatión and Knowledge) [41]. Adicionalmente se definió un mecanismo de despliegue visual de los procesos en ejecución, basados en unas técnicas de visualización de grafos y elementos del lenguaje de definición de procesos de negocio BPMN.

Visualización del proceso

Para la visualización de los procesos de negocio se hace una representación en grafos de estos y se utilizan algunos aspectos importantes en la visualización de grafos presentados en [54]. La visualización de grafos enfrenta el problema de construir representaciones geométricas de los mismos respetando determinados criterios estéticos y restricciones impuestas por el usuario. Hay varios criterios estándar que se utilizan en la visualización de grafos, usualmente los vértices son representados por símbolos tales como cajas o puntos y las aristas por curvas abiertas de Jordan, conectando los símbolos que representan a los vértices asociados, sin embargo estos estándares pueden variar según la aplicación.

En el presente trabajo la representación del grafo como ya se menciono está ligada a la notación gráfica de procesos de negocio BPMN, también llamados diagramas BPD (*Business Process Diagram*) que están formados por una serie de elementos fundamentales utilizados para crear un esqueleto estructural elemental del proceso, que se compone de objetos de flujo y conectores los cuales definen los símbolos que pueden ser utilizados para su despliegue.

Para dibujar el proceso BPEL utilizando su representación en grafos se sigue la metodología Jerárquica [54] la cual define los siguientes pasos:

- Asignación de capas o niveles: El objetivo de este paso es asignar una coordenada (Y) a cada vértice. En otras palabras es asignarle un nivel a cada nodo.

- Reducción de cruces: En este paso se ordena los vértices dentro de cada capa para minimizar el número de cruces entre los arcos.

- Se eliminan los ciclos dentro del grafo.

- Asignación de la coordenada horizontal: Se determina para cada vértice un par de coordenadas (X, Y).

- Se dibuja el grafo de acuerdo a la representación BPMN de cada nodo y a sus coordenadas (X, Y) ya establecidas.

Es importante aclarar que solamente se utilizaron algunos elementos básicos de la notación BMPN, los cuales fueron necesarios para tener una representación visual básica del proceso

Monitoreo con agentes de registro

Como ya se mencionó el mecanismo de monitoreo propuesto en el presente trabajo de grado sigue la metodología propuesta en la arquitectura de Workflow JITIK, la cual trabaja con un agente de registro para el monitoreo de procesos. En el agente de registro utilizado en este proyecto se definen varios

mecanismos que permiten la ejecución descentralizada de procesos con agentes inteligentes, entre la que se encuentra el monitoreo de procesos de negocio. A continuación se describirá el mecanismo de monitoreo utilizado en la arquitectura JITIK.

Cuando un trabajador está a punto de iniciar la ejecución de una tarea a través de su Agente Personal, éste envía un mensaje al Agente de Registro indicando la instancia y la tarea en particular a ser ejecutada. Posteriormente el Agente de Registro envía al Agente Personal un mensaje que contiene la descripción de una tarea, junto con su identificador y el identificador del proceso al que pertenece, información que es almacenada por cada Agente Personal en una lista. Este mecanismo de coordinación no es importante para llevar a cabo el control de flujos de procesos de manera descentralizada, sin embargo sí es necesario para monitorear procesos en ejecución. En la Figura 33 se presenta el diagrama de secuencia que corresponde a este mecanismo.

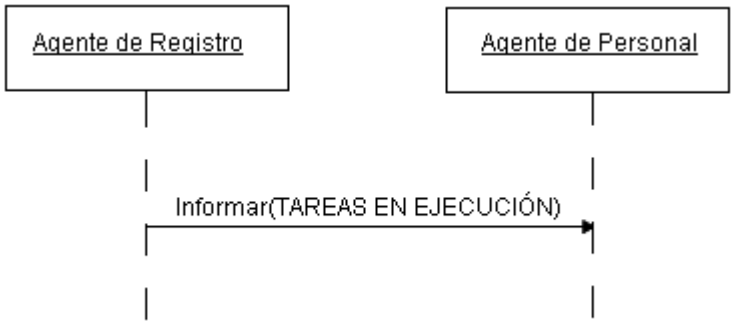

Figura 33. Diagrama de secuencia Informar sobre la ejecución de una tarea JITIK.

4.4.2 Mecanismo de Monitoreo

Mediante el mecanismo de monitoreo se presenta la información correspondiente a la ejecución de los subprocesos de negocio distribuidos en los sistemas móviles de información, la cual es posible consultar de manera gráfica y textual. A continuación se describen los dos procesos que componen este mecanismo.

Proceso 1: Registro de las actividades que han sido ejecutadas

Este proceso sucede a medida que un actor ejecuta sus tareas en el sistema móvil de información, en ese instante la aplicación móvil envía un mensaje al servidor Web MobFlow con la información correspondiente a la actividad ejecutada. Esto se hace a través de una serie de comunicaciones del dispositivo móvil y servidor Web MobFlow.

Para realizar el proceso de registro de las actividades, MobFlow obtiene la información de la actividad ejecutada y registra en la base de datos del sistema los datos correspondientes al identificador del subproceso, Actividad, Fecha y hora de ejecución. Con esto es posible consultar la información de monitoreo (Proceso, subprocesos, Actores, actividades ejecutadas, fecha y hora de ejecución). De igual manera se registra la ejecución de las actividades de sincronización.

Vale la pena recordar que en cada una de estas comunicaciones se envía información del usuario, la cual permite establecer a qué proceso de negocio, cargado en la plataforma pertenece el subproceso que se está ejecutando.

Proceso 2: Despliegue del monitoreo del proceso de negocio

Para consultar el estado de la ejecución del proceso de negocio, se establecieron dos formas de hacerlo: ***Consulta textual de las actividades ejecutadas por los subprocesos*** y ***Consulta visual de las actividades ejecutadas***. Estas se describen a continuación:

- ***Consulta textual de las actividades ejecut adas***

  Una vez registrada las actividades que han sido ejecutadas, es posible consultar en la Base de Datos de la plataforma MobFlow la información almacenada respecto al monitoreo, donde se puede obtener lo siguiente: Nombre del proceso y subprocesos de negocio, actores y nombre de las actividades ejecutadas así como la hora y fecha de ejecución.

  Mediante la consulta textual de actividades (básicas, estructuradas y de sincronización) también es posible visualizar el momento en que un subproceso inicio o término la ejecución de sus tareas en el sistema móvil de información, teniendo en cuenta que las actividades de sincronización indican el inicio o fin de la ejecución de un subproceso. Toda esta información es presentada mediante una interfaz Web.

- ***Consulta visual de las actividades ejecutadas***

  Para la consulta visual de las actividades ejecutadas se propone el algoritmo 4: ***Visualización del proceso***, el cual se presenta en pseudocódigo y describe paso a paso como generar una imagen del monitoreo de la ejecución de un proceso.

  Además se utilizo este algoritmo que genera una imagen solamente del flujo del proceso, función utilizada por la plataforma para desplegar visualmente el proceso de negocio una vez cargado, esto es posible variando los parámetros de entrada del algoritmo.

  En la generación de la imagen del proceso se plantea el uso de los elementos básicos de la notación BPMN, con el fin de obtener una representación gráfica entendible del proceso, ya que está provee una notación fácil y clara para todos los usuarios del negocio [21], además permite que los lenguajes XML diseñados para la ejecución de procesos de negocio tales como BPEL4WS, puedan ser visualizados con una notación orientada al negocio.

  El algoritmo ***Visualización del proceso*** tiene como entradas el proceso BPEL ($BPEL_{XML}$), la lista de las actividades ejecutadas ($LISTA_{ACTIVIDADES}$), que se obtienen a partir de una consulta en la base de datos de la plataforma, la lista de Subprocesos ($LISTA_{SUBPROCESOS}$) asociados al proceso BPEL de entrada ($BPEL_{XML}$), la lista de Actores ($LISTA_{ACTORES}$) asociados igualmente al proceso BPEL de entrada y por ultimo una variable booleana tipo bandera (b*oolean*: BanderaMonitoreo), la cual establece si el algoritmo retorna una imagen con el monitoreo del proceso o solamente con el flujo del proceso (IMG_{BPEL}).

  Esté algoritmo permite generar una imagen del estado de la ejecución distribuida del proceso de negocio ($\mathbf{BPEL_{XML}}$), además mediante este algoritmo también es posible generar la imagen del flujo del proceso de entrada, para lo cual se requiere un cambio en el parámetro de entrada ***BanderaMonitoreo***, de manera tal que cuando este parámetro tenga asignado el valor ***TRUE*** el algoritmo genera una imagen para el monitoreo del proceso (para lo que se requiere los parámetros $\boldsymbol{LISTA_{ACTIVIDADES}}$, $\boldsymbol{LISTA_{SUBPROCESOS}}$, $\boldsymbol{LISTA_{ACTORES}}$). En caso que se quiera generar la imagen de solo el flujo del proceso, se necesita que el parámetro de entrada ***BanderaMonitoreo*** tenga asignado el valor de ***FALSE*** (para este caso los parámetros $\boldsymbol{LISTA_{ACTIVIDADES}}$, $\boldsymbol{LISTA_{SUBPROCESOS}}$, $\boldsymbol{LISTA_{ACTORES}}$ tendrían el valor de *NULL*).

Algoritmo 4: Visualización del proceso

INPUT: $BPEL_{XML}$, $LISTA_{SUBPROCESOS}$, $LISTA_{ACTORES}$, $LISTA_{ACTIVIDADES}$, b*oolean*: BanderaMonitoreo
OUTPUTS: IMG_{BPEL}
1 **BEGIN**
2 Crear imagen en blanco **IMG_{BPEL}**
3 Convertir **$BPEL_{XML}$** a Modelo de Grafos se obtiene **G_{BPEL}**
4 **IF** BanderaMonitoreo = TRUE **THEN**
5 Asignar de manera aleatoria un color a cada actor y subproceso relacionado, contenidos en $LISTA_{ACTORES}$ y $LISTA_{SUBPROCESOS}$ respectivamente
6 Asignar un color a cada Nodo del grafo de acuerdo al color del subprocesa a que pertenece
7 Dibujar una tabla en **IMG_{BPEL}** con los subprocesos contenidos en $LISTA_{SUBPROCESOS}$ y con el color asignado a cada subproceso.
8 Dibujar una tabla en **IMG_{BPEL}** con los actores contenidos en $LISTA_{ACTORES}$ y con el color asignado a cada actor.
9 Señalar el subproceso que ejecutó la última actividad según lo indicado en $LISTA_{ACTIVIDADES}$
10 **END IF**
11 Reorganizar los arcos de **G_{BPEL}** para eliminar los ciclos
12 Crear una estructura de datos en forma de arreglos bidimensionales (**$Matriz_{ACTIVIDADES}$**), para contener las actividades del proceso.
13 Recorrer **G_{BPEL}** por niveles y agregar cada nodo a una fila de **$Matriz_{ACTIVIDADES}$** según el nivel correspondiente, manteniendo la relación de los nodo padre-hijo.
Por ejemplo los nodos del nivel 1, en la fila 1 de **$Matriz_{ACTIVIDADES}$**, nodos del nivel 2, en la fila 2 de **$Matriz_{ACTIVIDADES}$…**
14 A cada nodo de la matriz **$Matriz_{ACTIVIDADES}$**, asignarle una coordenadas (X, Y) según su nivel y posición(fila,columna), Teniendo en consideración el tamaño de la imagen creada **IMG_{BPEL}**.
15 **FOR** entero **i** ← 1 hasta **i** = numero de Filas de **$Matriz_{ACTIVIDADES}$** **DO**
16 **FOR** entero **j** ← 1 hasta **j** = numero de Columnas de **$Matriz_{ACTIVIDADES}$** **DO**
17 Obtener el nodo con la información de la actividad, que representa nodo contenido en la **$Matriz_{ACTIVIDADES}$** en la fila **i** columna **j**
18 Obtener el icono de la actividad correspondiente al nodo en notación BPMN e insertar un símbolo para las actividades BPEL
19 **IF** BanderaMonitoreo = TRUE **THEN**
20 **IF** La actividad obtenida en el nodo se encuentra en la **$LISTA_{ACTIVIDADES}$** **THEN**
21 Marcar el icono de la actividad como ejecutada.
22 **END IF**
23 Marcar el icono de la actividad con el color de correspondiente al subproceso que pertenece
24 **END IF**
25 Obtener posición (X, Y) asignada a nodo contenido en **$Matriz_{ACTIVIDADES}$** en la fila **i** columna **j**.
26 Pintar icono obtenido con nombre de actividad (si se requiere) en posición (X, Y) de **IMG_{BPEL}**.
27 Obtener la poción de los nodos hijos asociados al nodo contenido en **$Matriz_{ACTIVIDADES}$**
28 Crear las líneas de los arcos correspondientes entre nodo Padre e Hijo, evitando la superposición entre las líneas de los arcos.
29 Pintar las líneas en IMG_{BPEL}
30 **END FOR**
31 **END FOR**

El algoritmo ***Visualización del Proceso*** utiliza una representación en grafos del proceso y los elementos BPMN necesarios. El algoritmo realiza los siguientes pasos:

- El algoritmo en primer lugar crea una imagen en blanco (IMG_{BPEL}), en donde se pintarán las diferentes actividades del proceso (Línea 2).
- Luego se procede a realiza la conversión del proceso BPEL a un modelo de grafos. Para esta transformación utiliza el transformador propuesto en [42] (Línea 3).
- Si el valor de la bandera *BanderaMonitoreo* está en verdadero (TRUE) (Línea 4), como ya se menciono se crea la imagen de monitoreo. Por consiguiente se siguen los pasos descritos en las líneas 5, 6,7, 8 y 9.
- Para la imagen de monitoreo del proceso, a cada subproceso se le asigna un color el cual es igualmente asignado al actor que lo ejecuta (Línea 5).

- Posteriormente se asigna un color a los nodos del grafo teniendo en cuenta el color del subproceso a que pertenece, con esto se tiene que los nodos de cada subproceso tendrán un color diferente (Línea 6).

- Una vez realizados los pasos descritos en las líneas 5 y 6, se pintan en la imagen de monitoreo (IMG_{BPEL}) las tablas de actores y subprocesos con el color correspondiente (Líneas 7 y 8).

- Para indicar el subproceso que se encuentra en ejecución se señala en la tabla de subprocesos, el subproceso que ejecutó la última actividad, buscado de acuerdo a la fecha y hora de ejecución en la lista de actividades $LISTA_{ACTIVIDADES}$ (Línea 9).

- En el modelo de grafos se realiza el proceso de eliminación de los ciclos del grafo; en este caso para un proceso BPEL los ciclos corresponden a los nodos del grafo de actividades repetitivas, como por ejemplo la actividad `<while>`. Para esto se reorganiza la orientación de los arcos necesarios para que el grafo sea acíclico (sin ciclos) (Línea 11).

- De aquí se crea una estructura de datos en forma de arreglos bidimensionales (Matriz), la cual contendrá las diferentes actividades del proceso BPEL. En cada campo de la matriz se guarda un nodo del grafo (Línea 12).

- A continuación se inicia un recorrido del grafo por niveles (Línea 13). Por ejemplo en el proceso BPEL presentado en modelo de grafos en la figura 26, los diferentes niveles del grafo se exponen separados con las líneas punteadas y son numerados. En donde el nodo *Start* corresponde al nivel 1 (1), el nodo *invoke* nivel 2 (2), el nodo *End* nivel 6 (6). En un grafo NO acíclico no sería posible recorrer el grafo de esta manera, por lo cual se realizó el paso descrito en la línea 11.

  Los nodos de un mismo nivel se guardan en la misma fila de la matriz según su correspondencia, por ejemplo el nodo *Start* de la figura 26 es almacenado en la fila 1 columna 1 de la matriz, los nodos *invoke* del nivel 4 serán contenidos en la fila 4 columnas 1 y 2 de la matriz respectivamente.

- A cada uno de los nodos se le asigna una posición (X, Y) que corresponde a las coordenadas en la imagen donde estará graficada la actividad (Línea 14). Para esto se debe considerar el tamaño de la imagen creada y la ubicación de los nodos dentro de la matriz (filas y columnas).

  Por ejemplo si la fila 1 de la matriz solo contiene un nodo, las coordenadas son asignadas de manera tal que al pintar en la imagen del proceso la actividad que representa el nodo, quede en la parte superior de la imagen y centrada.
  Si la fila 3 que contiene varios nodos, se les asignará las coordenadas (X, Y) de forma que queden alineados en un posición (Y), y a distancias proporcionales en el eje (X).

- En la líneas 15 a 30 se recorre cada uno de los nodos de la matriz ($\mathbf{Matriz}_{ACTIVIDADES}$), se obtiene la información de cada actividad que representan y se pintan dentro del proceso.

- Para cada iteración, se obtiene el nodo con la información de la actividad que representa (Línea 17).

- Luego se obtiene el icono de la actividad correspondiente al nodo en notación BPMN y además se inserta un símbolo para las actividades básicas en BPEL (Línea 18).

- Si el valor de la bandera *BanderaMonitoreo* está en verdadero (TRUE) (Línea 20) cumple la condición para generar la imagen de monitoreo. Por otra parte si la actividad que se está evaluando se encuentra en la lista $\mathbf{LISTA}_{ACTIVIDADES}$, quiere decir que ya ha sido ejecutada por un dispositivo móvil, por lo cual se obtiene el icono obtenido en la línea 18 y se marca como una actividad ejecutada, (Línea 21). Cada actividad es marcada con el color correspondiente al subproceso que pertenece (Línea 23).

- Luego se pinta la actividad en las coordenadas (X, Y) de la imagen (IMG_{BPEL}) con el nombre de la actividad (el nombre solo en caso que sea una actividad básica) (Línea 25 y 26).

- También de cada campo se obtiene la información de los nodos hijos, con el objetivo de pintar los arcos que unen a estos nodos evitando que estas se superpongan (Línea 27, 28 y 29).

- Finalmente se obtiene una imagen con el flujo del proceso (IMG_{BPEL}).

Con este algoritmo se genera una imagen que representa la ejecución del proceso, llevado a cabo desde los dispositivos móviles que contienen los subprocesos generados.

Es importante recalcar que si se necesita tener solo una representación del flujo del proceso de negocio, este algoritmo también puede ser utilizado con este fin, para ello solo se requiere cambiar la bandera de entrada *BanderaMonitoreo* y colocarlo a falso para que el algoritmo retorne solo el flujo del proceso sin tener en cuenta la ejecución del mismo.

Por ejemplo si en el algoritmo 4, el proceso BPEL de entrada fuera el presentado en la figura 25, con *BanderaMonitoreo* = TRUE, y se diera el caso que se esté realizando la ejecución distribuida de los subproceso presentados en las figura 27, 28 y 29, teniendo finalizada la ejecución del subproceso Operacion.bpel se obtendría una imagen como la de la figura 34. En este caso el grafo sobre el cual trabaja esté algoritmo es el presentado en la figura 26.

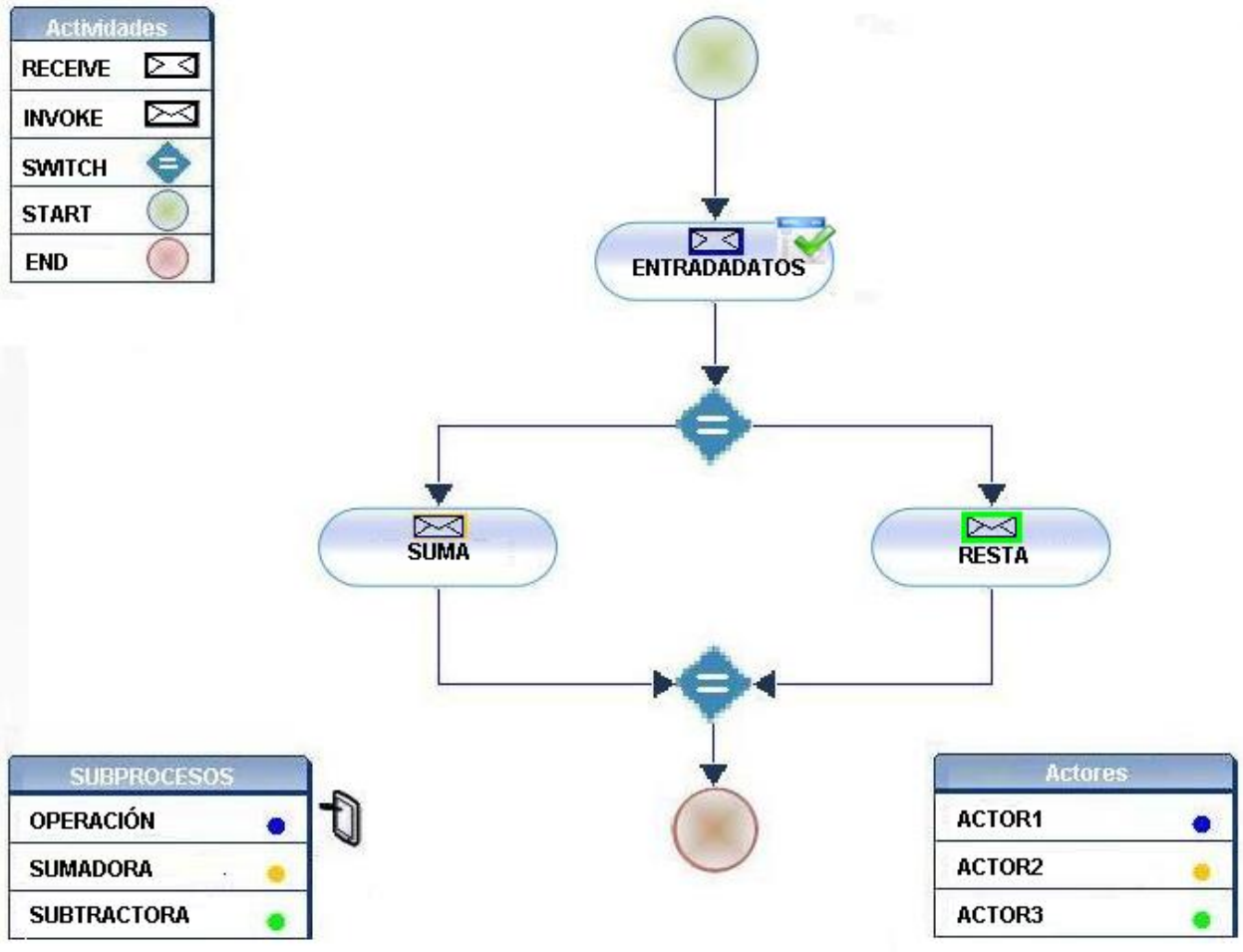

Figura 34. Imagen obtenida del monitoreo del proceso luego de la ejecución de una actividad

En esta imagen de monitoreo se muestran dos tablas en la parte inferior, las cuales presentan la información de los subprocesos generados en el particionamiento y los actores que ejecutan cada subproceso en los dispositivos móviles. Estos se relacionan entre sí por un color específico, que tiene como fin identificar los actores y subprocesos que hacen parte de la ejecución distribuida del proceso de negocio, además permite identificar las actividades dentro del flujo que corresponden a cada subproceso, tal como se puede apreciar, las actividades están marcadas con un color de acuerdo al subproceso al que pertenecen. Adicionalmente en la tabla subprocesos se muestra un icono de un dispositivo móvil, el cual indica cual fue el ultimo subproceso en la ejecución (en el caso de la figura 34, el dispositivo móvil que ejecuta subproceso *Operacion* es el ultimo en haber ejecutado una actividad del proceso). Respecto a las tablas, finalmente en la esquina superior izquierda se presenta una tabla que expone el tipo de actividades BPEL que se muestran en el flujo, los cual permiten tener una idea de que operaciones se pueden estar ejecutando.

Por otro lado, en el flujo los nodos que marcan el inicio y fin (*Start*, *End*) del proceso son representados por el círculo amarillo y el círculo rojo respectivamente, en donde el nodo *Start* tiene como hijo el nodo *receive* (nivel 1) relación que se simboliza con una flecha entre actividades. Los nodos de las actividades básicas son representados por rectángulos redondeados de color azul-blanco en los cuales se pone el nombre de la actividad, también contiene un icono de la actividad BPEL que representa, con el fin de que la imagen del flujo sea más descriptiva. Las actividades estructuradas <s*witch*> se representan con los rombos azules que marcan el inicio y fin de la actividad. Todas estas actividades son representadas teniendo en consideración la notación BPMN.

Finalmente la actividad *Entrada de Datos,* muestra un icono que representa una actividad que ha sido ejecutada por un dispositivo móvil, por lo cual dicha información debió registrarse previamente en la plataforma Web, para ser obtenida por el algoritmo mediante la LISTA$_{\text{ACTIVIDADES}}$. Como se puede observar esta actividad tiene un icono en la parte superior derecha, que distingue a esta actividad básica de las demás. (En el **anexo D** se presenta un ejemplo detallado del funcionamiento de este algoritmo).

Para el Registro de información de monitoreo interviene el módulo **Control de la Ejecución** y específicamente el submódulo **Agente de Monitoreo** que con la clase *RegistrodeMonitoreoImpl* realiza el registro de las actividades que han sido ejecutadas en los dispositivos móviles, obtenidas gracias al **Módulo de comunicación**.

En el momento de desplegar el monitoreo del proceso de negocio interviene el **módulo Operativo** con el submódulo **Monitoreo**, los cuales son los encargados de presentar la información de forma textual con las clases *Monitoreo* y *MonitoreoImpl* estas consultan en la base de datos las actividades ejecutadas por el proceso y presentan en una tabla esta información. Para la consulta visual del monitoreo se hace uso del **submódulo de Visualizar Proceso** para generar y presentar la imagen del monitoreo, para ello las clases *MonitoreoImpl* llama a la clase *VisualizarProcesoImpl*, que implementa el algoritmo 4, el cual genera una imagen con el estado actual de la ejecución del proceso. Esta imagen así como el informe textual es presentada en una interfaz Web usando la clase *MonitoreoBean*.

Por otra parte para el despliegue visual del flujo del proceso interviene el **módulo Operativo** con el submódulo **Visualizar Proceso,** el cual por medio de las clases ProcesodeNegocioImp, *ProcesodeNegocioBean* llama a la clase del módulo *VisualizarProcesoImpl,* para desplegar una imagen con el flujo del proceso. Los diferentes accesos a la base de datos para el mecanismo de monitoreo se logran a través de las clase *VisualizarProceso, Monitoreo, MonitoreoImpl, RegistrodeMonitoreoImpl,* y *RegistrodeMonitoreo* son realizados junto con las clases *MonitoreoDAO* y *RegistrodeMonitoreoDAO* del módulo **Gestión de Datos**.

En la aplicación cliente móvil MobFlow interviene el módulo **Lógica** y el submódulo **Control** que con las clases *controlEjecucion* y *controlActividades* establecen las actividades ejecutadas en el subproceso, esta clases interactúan con la clase *conexión* para el envió de la información requerida para el monitoreo al servidor Web MobFlow.

Resumen

En este capítulo se presentó en detalle como la plataforma MobFlow realiza los aspectos más críticos e importantes que permiten el despliegue distribuido de procesos de negocio en sistemas móviles de información, estos son:

- Particionamiento y sincronización de los procesos de negocio, consiste en generar de un proceso de negocio descrito en BPEL, varios subprocesos de negocio sincronizados, con las actividades propias de los actores participantes en esté.

- Distribución de los subprocesos de negocio en los sistemas móviles de información, consiste en enviar a cada actor del proceso de negocio el respectivo subproceso para que esté lo pueda ejecutar desde su Terminal móvil.

- Ejecución sincronizada de los subprocesos, consiste en realizar la ejecución de los subprocesos de negocio en los sistemas móviles de información de manera sincronizada, de forma tal que la ejecución distribuida de estos subprocesos sea equivalente a la ejecución del proceso de negocio original.

- Monitoreo de la ejecución distribuida, consiste en realizar la visualización del estado actual de la ejecución distribuida del proceso de negocio global, es decir se presentan que actividades se han ejecutado del proceso global y quien las ejecuto.

Para esto se describieron diferentes mecanismo y algoritmo que fueron utilizados para lograr dichos aspectos. En el **anexo D** como ya se ha mencionado se exponen ejemplos detallados del funcionamiento de los algoritmo expuestos en este capítulo.

En el siguiente capítulo se presentará una descripción del prototipo implementado, el caso de estudio y las diferentes pruebas realizadas a la plataforma.

CAPITULO V

CASO DE ESTUDIO, PROTOTIPO Y EXPERIMENTACIÓN

En este capítulo se describe el caso de estudio seleccionado para probar la plataforma MobFlow y se expone el prototipo implementado para su validación, presentando cada una de las funcionalidades y servicios ofrecidos, que permiten lograr el despliegue distribuido de los procesos de negocio en sistemas móviles de información. Finalmente, en este capítulo se presentan las diferentes pruebas realizadas a la plataforma que permiten medir el grado de rendimiento y estabilidad.

5.1. CASO DE ESTUDIO

Con el objetivo de validar la plataforma propuesta en el presente trabajo de grado, se seleccionó un caso de estudio que permite probar la funcionalidad total de la plataforma MobFlow. El caso de estudio seleccionado es el proceso de venta de un artículo en un almacén de electrodomésticos (*Electrocréditos de Cauca* [71]), este proceso es desarrollado basado en datos reales obtenidos a través una serie de entrevistas con diferentes actores del establecimiento, principalmente con el ingeniero encargado del departamento de sistemas. A continuación se presentan las consideraciones tenidas en cuenta para la selección de este caso de estudio, y posteriormente se hace una descripción detalla de él.

5.1.1. Identificación del Escenario de Funcionamiento

Para seleccionar el proceso del negocio se tuvieron en cuenta las siguientes consideraciones: los actores involucrados en la ejecución de los procesos requieren en algún momento la necesidad de movilidad y las funcionalidades que cumplen los actores pueden ser expuestas como servicios Web.

La empresa seleccionada como ya se menciono es *Electrocréditos del Cauca*, está cuenta con diversas sucursales dentro y fuera de la ciudad donde se manejan varios procesos de negocio, de estos se seleccionó el proceso de negocio venta de un producto, el cual involucra vendedores que pueden realizar ventas locales puerta a puerta o en diferentes pueblos del departamento. Los demás actores involucrados como el personal de facturación y el ingeniero encargado del sistema central necesitan estar en constante movilidad debido a las funciones que desempeñan.

Además las actividades que realiza cada actor en el proceso de ventas, pueden manejarse como módulos independientes que interactúan entre sí, lo cual permite que puedan ser implementadas como servicios Web.

Por consiguiente este proceso de negocio cumple con las consideraciones necesarias para implementar un caso de estudio sobre la plataforma MobFlow. A continuación se presenta una descripción detallada del proceso de negocio de ventas.

5.1.2. Descripción General del Proceso de Negocio

Para el proceso de negocio de ventas se plantea el flujo de trabajo presentado en la figura 35 y viene descrito de la siguiente manera. Inicialmente el cliente informa el electrodoméstico a comprar y el vendedor le presenta las formas de pago (Crédito o Contado) junto con los diferentes descuentos según el tipo de pago.

Luego el cliente indica el tipo de pago que va a realizar, si este pago es a contado significa que el cliente paga el artículo en efectivo, recién reciba el articulo y la venta es aprobada inmediatamente. En caso que el tipo de pago sea a crédito, el cliente debe indicar a cuantas cuotas realizará el pago del artículo y tendrá que diligenciar junto con el vendedor un formulario de solicitud de crédito, el cual debe incluir fotocopias de diversos documentos y deberá conseguir un fiador, este proceso lo ayuda a gestionar el vendedor.

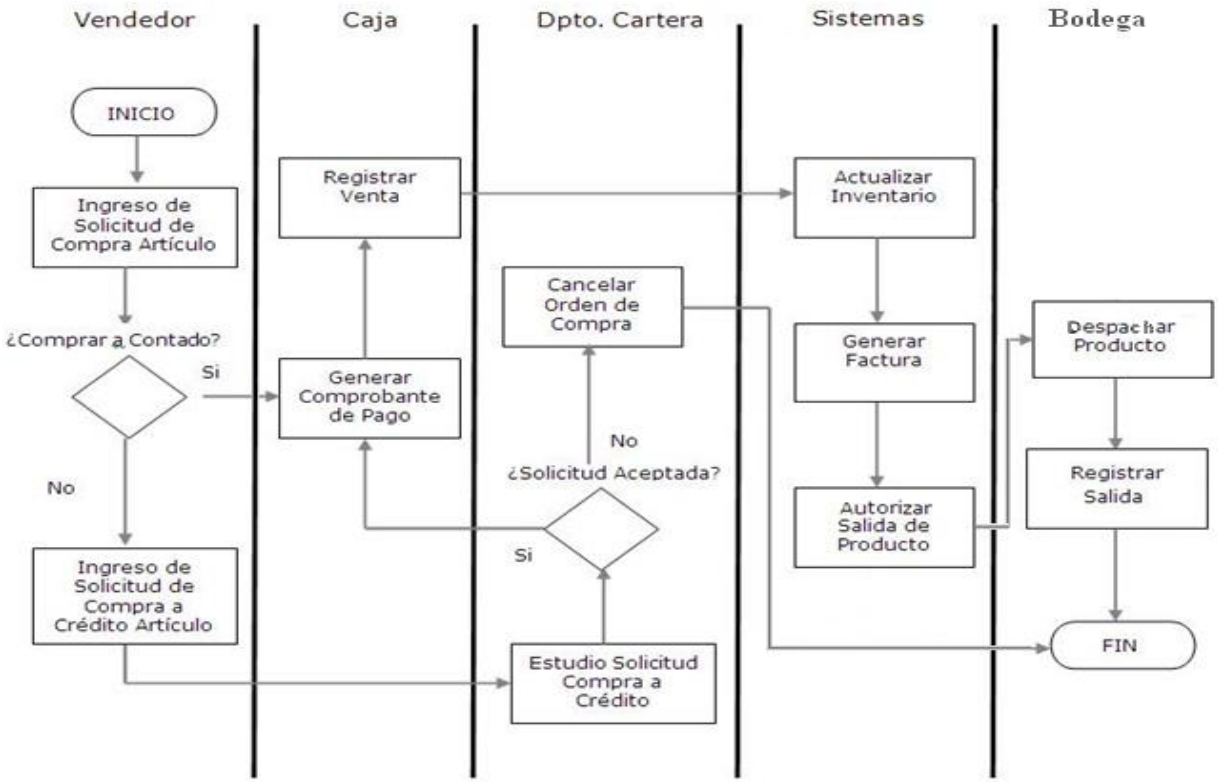

Figura 35. Flujo de trabajo del proceso de ventas

Si la compra es realizada a contado se procede a remitir la información de dicha venta a la Caja del almacén, aquí se realiza el pago del artículo comprado, se crea un comprobante de pago y se procede a notificar a la central la venta realizada.

En caso de que la compra sea realizada a crédito, se le envía al departamento de Cartera los datos del cliente y de la compra a realizarse, junto con el formulario que este diligencia y sus respectivos documentos. Una vez estos datos están en el departamento de Cartera, se procede a realizar un estudio crediticio del cliente que desea realizar la compra, dicho estudio consta de una revisión de créditos anteriores del cliente, información del fiador, bienes raíces, entre otros. Esto con el fin de realizar o no la aprobación del crédito.

Si el crédito es aprobado, estos datos son envidados a la Caja del almacén donde se encargan de manejar las cuotas y crear un comprobante de pago, junto con la notificación de la venta a la central, tal y como se hace con una compra a contado, este estudio de crédito no puede demorar más de 2 días por políticas del almacén. Si el crédito no es aprobado la venta es cancelada.

Después de que la información de la venta ha sido utilizada por la Caja del almacén, esta información empieza a ser manipulada por la Central (Dpto. Sistemas del Almacén), aquí se hace uso de estos datos para establecer que producto fue vendido y realizar la actualización del catalogo de productos, luego se realiza una factura en papel (para el cliente) y digital de la venta, para finalmente ordenar la salida de bodega del artículo. Finalizando así el proceso de ventas del almacén.

Entes participantes en el proceso

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

Cliente. Persona que realiza la compra de un electrodoméstico, esta compra la efectúa ya sea a contado pagando en efectivo en el instante de la compra el valor total del artículo, o la realiza a crédito donde

definen unas cuotas fijas a pagar mensualmente por un tiempo determinado, presentando los papeles pertinentes para la petición del crédito, antes de esperar su aprobación.

Vendedor. Persona encargada de la atención al cliente y la venta de los electrodomésticos. Entre las actividades correspondientes a la atención al cliente se encuentra, la presentación de los diferentes artículos y ofertas brindados por el almacén, como son producto, marcas, descuentos, rifas, etc. El vendedor también realiza la venta, toma los datos de dicha venta y la remite al almacén, en caso de que la venta sea a crédito ayuda al cliente a gestionar los papeles necesarios para la aprobación del crédito. Muchas veces los vendedores realizan ventas puerta a puerta en diferentes lugares de la ciudad o fuera de ella.

Caja. Se encarga de recibir el pago de la venta de determinado artículo, ya sea el pago completo o por cuotas, de esto se genera la impresión de un recibo de pago para el cliente que a la vez es una constancia del pago. También aquí se registra y se confirma la venta realizada al sistema central del almacén.

Dpto. de Cartera. Este ente es el encargado de realizar el estudio de crédito, en el caso de que un cliente desee comprar un artículo de esta forma, para esto se hace una revisión de los papeles requeridos para el estudio de crédito (fotocopias de cedula, renta mensual, formulario, fotocopia cedula fiador, renta mensual fiador), también se hace una revisión de los anteriores crédito realizados por el cliente que solicita el crédito, realiza una breve visita domiciliaria para confirmar estos datos y finalmente hace la aprobación o denegación del crédito.

Central (Dpto. de Sistemas). Para este proceso de negocio este ente es el encargado de actualizar el inventario del almacén después de realizada una venta, establecer el articulo exacto que fue vendido (Serial del artículo), realizar una factura de dicho artículo, ordenar la salida de determinada bodega de dicho artículo para ser enviado al lugar establecido por el cliente.

Dpto. de Cartera. Este ente es el encargado de realizar el estudio de crédito, en el caso de que un cliente desee comprar un artículo de esta forma. Para esto se hace una revisión de los papeles requeridos para el estudio de crédito (fotocopias para de cedula, renta mensual, formulario, fotocopia cedula fiador, renta mensual fiador), también se hace una revisión de los anteriores crédito realizados por el cliente que solicita el crédito, realiza una breve visita domiciliaria para confirmar estos datos y finalmente hace la aprobación o denegación del crédito.

Bodega. Este ente es el encargado de despachar el artículo establecido por la central al lugar señalado por el cliente y registrar la salida del artículo.

5.2 PROTOTIPO

En esta sección se presenta el prototipo funcional de la plataforma MobFlow, aplicado al caso de estudio de ventas de electrodomésticos descrito en la sección anterior. Inicialmente se presentan las herramientas software utilizadas para desarrollo del prototipo, y posteriormente se hace una descripción funcional.

5.2.1 Selección de las herramientas software utilizadas

Para la selección de las herramientas de desarrollo, se tuvieron en consideración los componentes externos utilizados en la plataforma (Analizador BPEL-Grafos y motor BPEL Sliver), los cuales son cruciales para el desarrollo de este proyecto y están desarrollados en el lenguaje Java. Además, también se consideró la tendencia del grupo de Ingeniería Telemática (GIT) [59] en la utilización de herramientas de libre distribución, junto con la experiencia que tienen los autores del proyecto en el manejo de estas herramientas libres.

En el desarrollo de las dos aplicaciones se utilizó el lenguaje de programación orientado a objetos JAVA en la versión JDK 1.6.

La aplicación Web MobFlow se desarrollo utilizando la tecnología J2EE con el marco de trabajo (framework) JSF (Java Servlet Faces), usando el entorno de desarrollo (IDE) Eclipse en la versión 3.3.2. Esta aplicación usa como Servidor Web Apache Tomcat en la versión 5.5.9.

Para la aplicación cliente móvil MobFlow se uso la tecnología J2ME usando como IDE NetBeans en la versión 6.5. El motor de base de datos utilizado fue el PostgreSQL 8.2, el cual es un sistema de gestión de base de datos relacional de software libre, con amplio reconocimiento a nivel mundial.

5.2.2 Descripción del prototipo de validación

A continuación se describen los pasos llevados a cabo para la ejecución del prototipo utilizando el caso de estudio descrito anteriormente

- **Paso 1- Diseño e Implementación del proceso**

Definido y entendido el proceso de negocio se realizó el análisis, diseño, implementación y pruebas necesarios para implementar los Servicios Web requeridos para el caso de estudio. Luego se procedió a realizar la composición de estos servicios con el fin de crear un documento BPEL que represente el proceso de negocio de ventas. Para esto se utilizó la herramienta JDeveloper BPEL Designer disponible en la suite Oracle BPEL Process Manager 10.1.2 siguiendo el flujo y la descripción del proceso presentado en la figura 36. De aquí se obtuvieron los archivos BPEL y WSDL que conforman la definición del proceso, los cuales se especificaron cumpliendo los requerimientos mencionados en el **anexo C**.

- **Paso 2 - Cargar el proceso de negocio en la plataforma**

Después de definir el proceso de ventas se procedió a cárgalo en la plataforma. Para esto MobFlow cuenta con una interfaz gráfica de usuario como la presentada en la figura 36.

Figura 36. Carga del proceso de ventas.

En esta interfaz se ingresó la información del proceso de ventas. En los campos Nombre (1), Descripción (2) se ingresaron los valores "*MisVentas*" y "*proceso de ventas*" respectivamente, luego se buscaron los archivos WSDL y BPEL correspondientes al proceso de ventas utilizando los botones *Seleccionar archivo* (3), (4), finalmente se enviaron los datos con el botón Enviar (5). Al enviar los datos del proceso la plataforma lo valida y lo carga como se presenta en la figura 37.

Figura 37. Proceso de ventas cargado en la plataforma.

- **Paso 3 - Visualización del proceso de Ventas**

Después de cargar el proceso BPEL, se visualizó mediante la interfaz Web que proporciona MobFlow la cual se presenta en la figura 38. Esta interfaz expone la información del proceso de ventas, así como su presentación visual.

Figura 38. Consulta y visualización gráfica del proceso de ventas.

- **Paso 4 - Particionamiento del proceso de Ventas**

Una vez cargado el proceso de ventas, se accedió a la interfaz gráfica de usuario que permite realizar el particionamiento del proceso, esta interfaz es presentada en la figura 39, donde se seleccionó el botón PARTICIONAR (1) para obtener los subprocesos.

Figura 39. Particionamiento del proceso de ventas.

Al ejecutar el particionamiento se obtuvieron 4 subprocesos BPEL correspondientes a los partnerLinks del proceso, como se muestra en la figura 40.

Figura 40. Subprocesos resultantes del Particionamiento.

- **Paso 5 – Vinculación de los actores con los subproceso**

Particionado y creados los actores del proceso, se procede a vincularlos con los subprocesos que ejecutarán en sus dispositivos móviles, para ello se seleccionó la opción (1) de la figura 40 que carga la interfaz presentada en la figura 41.

Al cargar la interfaz se despliegan los actores asociados con el rol del subproceso (seleccionado en la anterior interfaz), estos actores se despliegan en la opción (1), de la cual se escogió el actor para vincular con este subproceso, lo que despliega sus datos automáticamente. Luego mediante la opción (2) se ingresó el número de celular del Actor y con el botón Adicionar (3) se hace vinculación. Este proceso se realiza de igual manera para cada uno de los subproceso generados. Realizado el proceso de vinculación los actores pueden descargar sus procesos a los dispositivos móviles.

Figura 41. Interfaz que permite Vincular a los actores del proceso.

- **Paso 6 - Descarga del subproceso desde la aplicación Cliente Móvil MobFlow**

Para la descarga de los subproceso se ejecutó la aplicación Cliente Móvil MobFlow, la cual carga la interfaz que contiene el menú principal presentado en la figura 42, donde se escogió la opción (1), esta opción carga la interfaz gráfica presentada en la figura 43, que permite la descarga de los subprocesos correspondientes a cada actor. Para ello se llenaron los campos Nombre de Usuario (1), Contraseña (2) y se presionó el botón Conectar (3) (Para cada usuario se ejecuta la aplicación Cliente Móvil en un dispositivo diferente).

Figura 42. Menú principal aplicación cliente Móvil MobFlow

Figura 43. Interfaz gráfica para el registro desde el móvil a la plataforma.

- **Paso 8 - Ejecución de las actividades desde la aplicación Cliente Móvil MobFlow**

Obtenidos los correspondientes procesos de negocio, se procedió a iniciar su ejecución, seleccionando la opción 2 del Menú principal (figura 42), que genera dinámicamente una interfaz gráfica de usuario según las actividades definidas en el proceso BPEL descargado. En la figura 44, se presenta la interfaz gráfica de usuario generada por la aplicación móvil para el Vendedor, el cual es el encargado de iniciar el proceso de Ventas correspondiente al subproceso *DatosVenta,* esta interfaz se genera automáticamente según el subproceso que ejecuta el usuario. Para este caso se ingresó la información de una venta a

contado y se envió la información con el botón Send (1) a la plataforma Web. Posteriormente la aplicación móvil envía un mensaje de texto al siguiente dispositivo en la ejecución, el cual es el subproceso de *Caja* llamado *RegistroVenta.*

La figura 45, presenta la interfaz gráfica inicial de la ejecución del subproceso *RegistroVenta,* este requiere la información de las variables registradas del subproceso *DatosVenta* presentadas en la interfaz. En este caso se simulo el encargado de caja que registra los datos de la venta (obtenidos por las variables), para esto se presionó el botón Aceptar (1) para realizar el registro (Este actor además hace un recibo de pago de la venta, lo crea de manera manual basado en los datos presentados en la interfaz). Terminada la ejecución de sus tareas, el siguiente dispositivo en la ejecución recibió un mensaje de texto enviado automáticamente por la aplicación, el cual informaba que podía iniciar su ejecución. El mensaje es enviado al encargado de manejar la información de la Central, cuyo proceso de negocio es *ServicioInventario*.

Figura 44. Interfaz para realizar la Venta de un artículo.

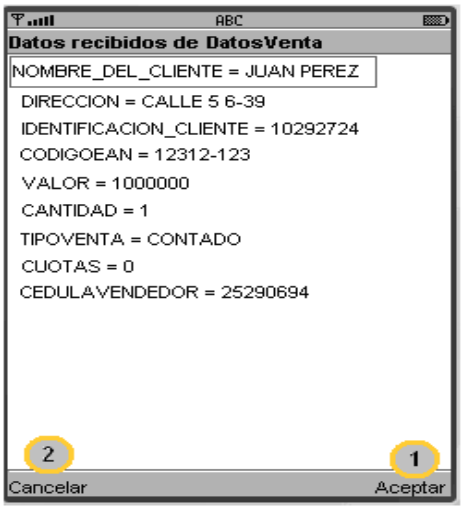

Figura 45. Interfaz registrar la Venta de una articulo.

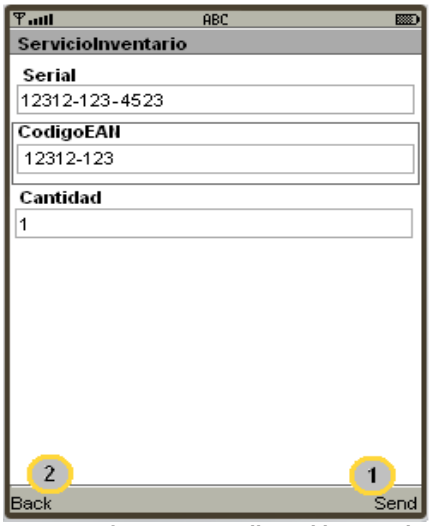

Figura 46. Interfaz para actualizar el inventario

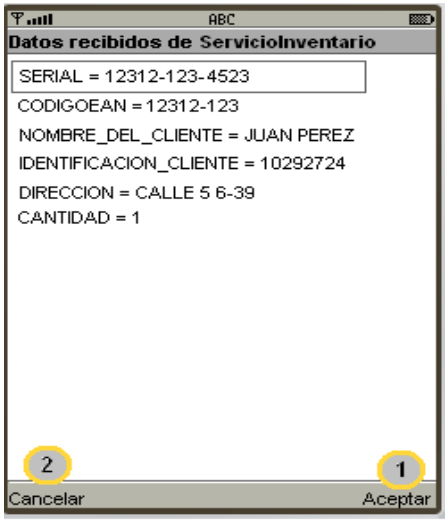

Figura 47. Interfaz para registrar salida de bodega

Continuando con el flujo del proceso, en la figura 46 se presenta la interfaz gráfica del actor encargado de la Central (quien es el que maneja los aspectos del inventario para este proceso), esta figura se visualizan las tareas que ejecuta el actor, donde ingresó la información para actualizar la información del inventario. Finalmente cuando el encargado del inventario termina estas tareas, sigue el turno de ejecución al encargado de la bodega, para el cual se presenta la interfaz de la figura 47 que presenta la

información requerida para despachar un artículo de la bodega y registrar la salida, finalizando así el proceso de ventas.

- **Paso 9 – Monitoreo de la ejecución distribuida del proceso de negocio**

A medida que los actores ejecutan los subprocesos en los dispositivos móviles, se pudo monitorear el estado de la ejecución, mediante una interfaz como la presentada en la figura 48, en esta interfaz se visualizó la información de la ejecución de cada actividad y una vista del proceso en ejecución.

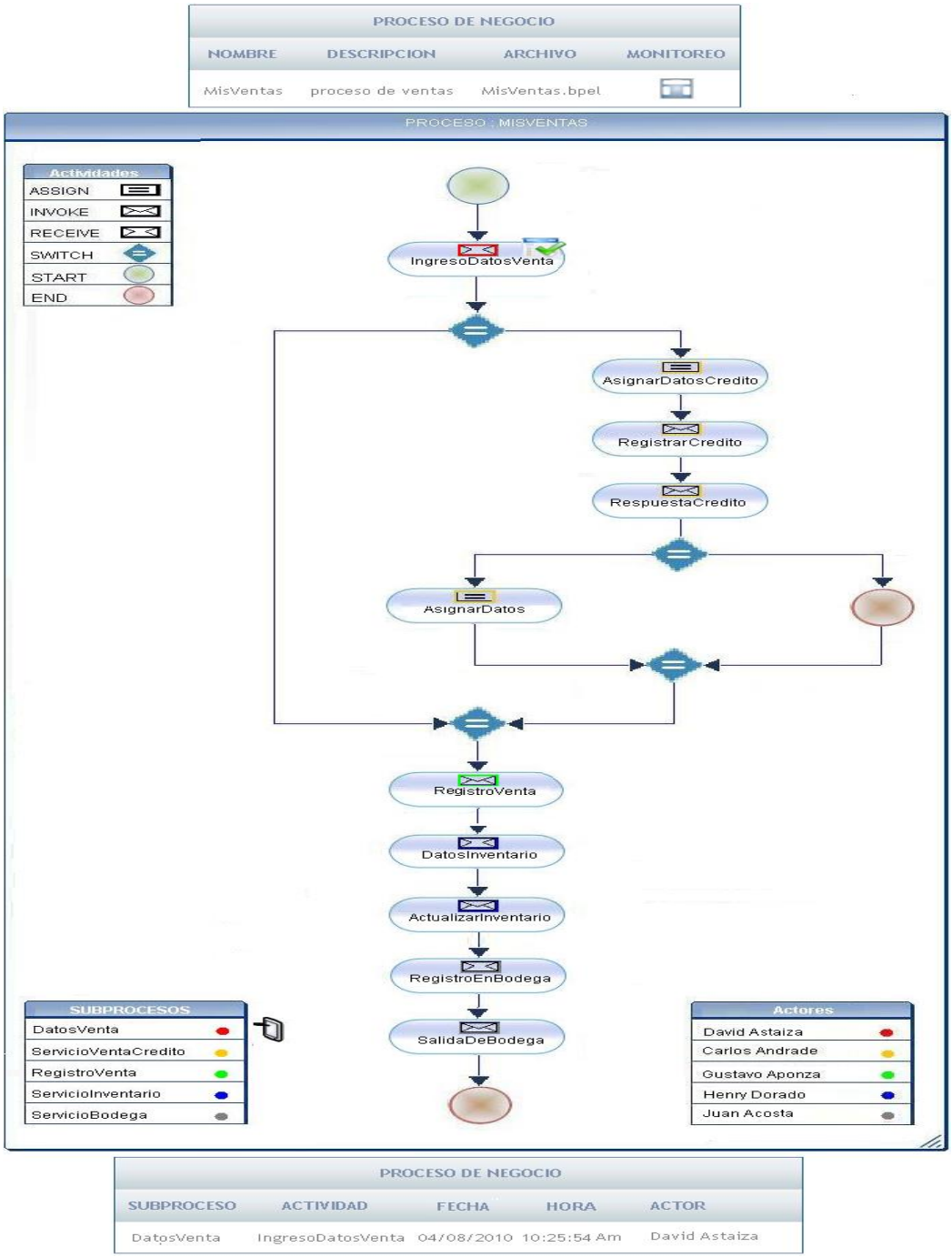

Figura 48. Interfaz gráfica para el monitoreo

5.3. EXPERIMENTACIÓN

En esta sección se presentan la experimentación realizada para comprobar la estabilidad y robustez de la plataforma MobFlow. Inicialmente se expone la metodología y criterios de evaluación, luego se describe el plan de las pruebas y finalmente se presentan el análisis de los resultados, junto con las gráficas asociadas.

5.3.1.Metodología y Criterio de Evaluación

Con el fin de asegurar y garantizar el correcto funcionamiento de la plataforma MobFlow, se realizaron pruebas que permiten determinar el rendimiento y escalabilidad de la plataforma, brindando un servicio a un grupo de usuarios de manera eficiente y sin fallas o retrasos. Es importante mencionar que el rendimiento del sistema está estrictamente ligado con el hardware del equipo que realiza la función de servidor, ya que éste determina el desempeño del mismo.

Para los módulos y componentes funcionales relevantes dentro de la plataforma MobFlow, se definieron pruebas enfocadas a la evaluación de dos aspectos fundamentales: el rendimiento y el soporte de usuarios o estabilidad. El primero, evalúa los tiempos de respuesta a solicitudes específicas realizadas a la plataforma, y el segundo evalúa la capacidad que tiene la plataforma de soportar varios usuarios de manera simultánea sin generar retardos significativos en los tiempos de respuesta.

Por otro lado, para la ejecución de las pruebas de estabilidad se usó la herramienta JMeter [57], la cual provee el mejor servicio y la mejor solución para este tipo de pruebas según una comparación realizada en [56]. Para obtener los tiempos de respuesta de los módulos funcionales que conforman la plataforma MobFlow, se uso la clase nativa de java conocida como *System*.

Finalmente la evaluación y clasificación de los resultados de las pruebas, se realizó a partir de un conjunto de reglas que determinan los criterios de tiempos de respuesta para aplicaciones que están directamente relacionadas a las características cognitivas de los humanos [56]:

- Los usuarios no notan un retardo de menos de 0.1 segundos. Por tanto, un tiempo de respuesta que este bajo el umbral de 0.1 segundos se clasifica como óptimo.

- Un tiempo de respuesta de menos de 1 segundo no interrumpe el flujo del pensamiento de un usuario, pero deja notar un retardo. Por tanto, un tiempo de respuesta que este bajo el umbral de 1 segundo se clasifica como bueno.

- Los usuarios aún esperarán la respuesta si está por debajo del umbral de 10 segundos, pero el retardo es notorio. Por tanto, un tiempo de respuesta que este bajo el umbral de 10 segundos se clasifica como aceptable.

- Después de 10 segundos los usuarios pierden la concentración y no continúan esperando la respuesta del sistema. Por tanto, un tiempo de respuesta que esté por encima del umbral de 10 segundos se clasifica como deficiente.

5.3.2. Plan de Pruebas

La pruebas de rendimiento y estabilidad se hacen sobre los módulos que realizan las funcionalidades más importantes y críticas de la plataforma MobFlow, estas son: particionamiento y sincronización, distribución de los subprocesos, ejecución sincronizada y monitoreo, las cuales fueron descritas en el capítulo anterior.

A continuación en la tabla 1 se describen las pruebas de rendimiento y estabilidad realizadas a la plataforma MobFlow.

PLAN DE PRUEBAS	
Particionamiento y sincronización	• **Prueba de rendimiento PR1:** Determina el tiempo que tarda la plataforma en particionar los procesos de negocio BPEL, según el numero de actividades definidas en él. Para está funcionalidad no se efectúan pruebas de estabilidad debido a que es un proceso que se hace siempre por un solo usuario (administrador), es decir no se presenta el caso que múltiples usuarios realicen el particionamiento al tiempo.
Distribución de los subprocesos	• **Prueba de rendimiento DR1**: Determina el tiempo que tarda la plataforma en validar y dar respuesta al usuario, después de que este realiza el proceso de inicio de sesión desde su sistema móvil de información. Esta prueba se ejecuta utilizando diferentes dispositivos, con el fin de verificar la influencia que tienen las características físicas de cada dispositivo en los tiempos de respuesta. Además, se analiza el rendimiento de la plataforma dependiendo del acceso (Wi-Fi. GPRS) sobre un mismo dispositivo. • **Prueba de rendimiento DR2:** Determina el tiempo que demora un usuario del sistema móvil de información en descargar un proceso de negocio, después de haber validado sus datos y aceptado la descarga. Esta prueba se ejecuta utilizando diferentes dispositivos, con el fin de verificar la influencia que tienen las características físicas de cada dispositivo en los tiempos de respuesta. Además, se analiza el rendimiento de la plataforma dependiendo de la forma de acceso (Wi-Fi. GPRS) sobre un mismo dispositivo. • **Prueba de estabilidad DE1:** Determina el soporte máximo de usuarios de la plataforma con un tiempo estable en el proceso validar y dar respuesta al usuario para la obtención del proceso de negocio, cuando se envían varias solicitudes al tiempo.
Ejecución sincronizada	• **Prueba de rendimiento ER1:** Determina el tiempo que tarda la plataforma en obtener y enviar las variables de inicio de ejecución, requeridas cuando se ejecuta una actividad *receive* de sincronización desde el motor BPEL del dispositivo móvil. Para esta prueba se mide el tiempo desde que el usuario inicia la ejecución del subproceso hasta que el servidor retorna la información de las variables requeridas y son mostradas en la pantalla del dispositivo móvil. Se ejecuta utilizando diferentes dispositivos con el fin de verificar la influencia que tienen las características físicas de cada dispositivo en los tiempos de respuesta. Además, se analiza el rendimiento de la plataforma dependiendo de la forma de acceso (Wi-Fi. GPRS) sobre un mismo dispositivo. • **Prueba de rendimiento ER2:** Determina el tiempo que demora la plataforma en obtener y enviar el resultado de la invocación de un servicio Web al sistema móvil de información, cuando esté requiere de dicha invocación para la ejecución de su proceso de negocio. Para esta prueba se mide el tiempo desde que un cliente móvil hace la petición de invocación de un servicio Web al servidor, hasta cuando el móvil obtiene el

	resultado de la invocación y lo procesa para continuar con la ejecución. Esta prueba se ejecuta utilizando diferentes dispositivos con el fin de verificar la influencia que tienen las características físicas de cada dispositivo en los tiempos de respuesta. Además, se analiza el rendimiento de la plataforma dependiendo de la forma de acceso (Wi-Fi. GPRS) sobre un mismo dispositivo. • **Prueba de estabilidad EE1:** Determina el soporte máximo de usuarios de la plataforma con un tiempo estable de repuesta en el proceso de obtención y envió de las variables de inicio de ejecución requeridas en la ejecución de una actividad *receive* de sincronización desde el motor BPEL del sistema móvil de información, cuando se envía varias solicitudes desde diferentes dispositivos. • **Prueba de estabilidad EE2:** Determina el soporte máximo de usuarios de la plataforma con un tiempo estable de repuesta en el proceso de obtención y envió de los resultados en la invocación a un servicio Web, requerido en la ejecución de un proceso de negocio en el sistema móvil de información, cuando se envía varias solicitudes desde diferentes dispositivos.
Monitoreo	• **Prueba de rendimiento MR1:** Determina el tiempo que tarda la plataforma en presentar la imagen del proceso de negocio según el numero de actividades que tenga definido. Para está funcionalidad no se efectúan pruebas de estabilidad debido a que es un proceso que se realiza siempre por un solo usuario, es decir no se presenta el caso de que múltiples usuarios realicen la visualización del proceso al tiempo.

Tabla 1. Plan de Pruebas

5.3.3. Especificaciones Técnicas del Servidor Central y de los Dispositivos móviles usados en las pruebas.

- **Servidor Central.**

Procesador	RAM	Disco Duro	Sistema Operativo
Genuine Intel(R) T2080 @ 1.73GHZ x2	1 GB	120 GB	Microsoft Windows XP

Tabla 2. Especificaciones técnicas del Servidor Central de MobFlow.

- **Dispositivos Móviles**

Dispositivo	Procesador	RAM	ROM	Tamaño de pantalla	Sistema Operativo
Pocket PC DELL AXIM x51v	Intel PXA270, 520 MHz	64MB	256MB	480 X 640 Pixeles.	Microsoft Windows Mobile 5.0
Nokia N93	Dual ARM 11 332 MHz	64MB	50 MB	128 X 160 Pixeles.	Symbian 9.1

Tabla 3. Especificaciones técnicas de los dispositivos usados para las pruebas.

5.3.4. Resultados

A continuación se presenta el análisis de los resultados obtenido en las diferentes pruebas y las gráficas asociadas a estos los resultados.

Resultados pruebas para el Particionamiento y Sincronización
• **Prueba de rendimiento PR1:** Para esta prueba se definió un proceso BPEL inicial con 10 actividades, esté se cargó en la plataforma y se realizo una prueba inicial que mide el tiempo que demora en particionar dicho proceso. Luego al mismo proceso se le aumentaba el número de actividades de manera coherente y se media el tiempo que toma la plataforma en particionar el proceso con más actividades. En la figura 49 se presentan los resultado obtenidos al realizar esta prueba, se visualiza que para procesos de más 1750 actividades los tiempos de generación de los subprocesos son deficientes. También se visualiza en la figura 49 que este tiempo está ligado directamente a número de actividades razón por la cual obtiene un crecimiento lineal. Es de destacar que muy pocas veces se da el caso que un proceso BPEL cuente con 1750 actividades, caso que no se da un proceso de negocio real.

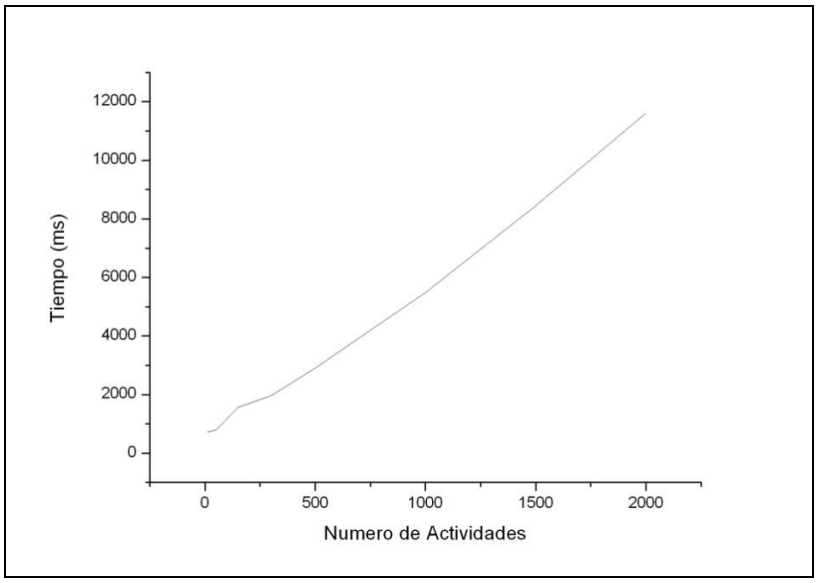

Figura 49. Gráfica de rendimiento del proceso de validación y respuesta en validación del usuario desde el dispositivo móvil.

Tabla 4. Análisis de resultados pruebas para las funcionalidades particionamiento y sincronización

Resultados pruebas para la Distribución
• **Prueba de rendimiento DR1**: En la figura 50 se presenta la gráfica de resultados de las pruebas de rendimiento para la validación y obtención de respuesta en el proceso de registro de usuario requerido en la distribución. El acceso de los dispositivos móviles reales al servidor Web se hizo a través de una conexión inalámbrica. En los dos dispositivos se obtiene tiempos aceptables de repuesta después de realizar el registro desde el móvil.

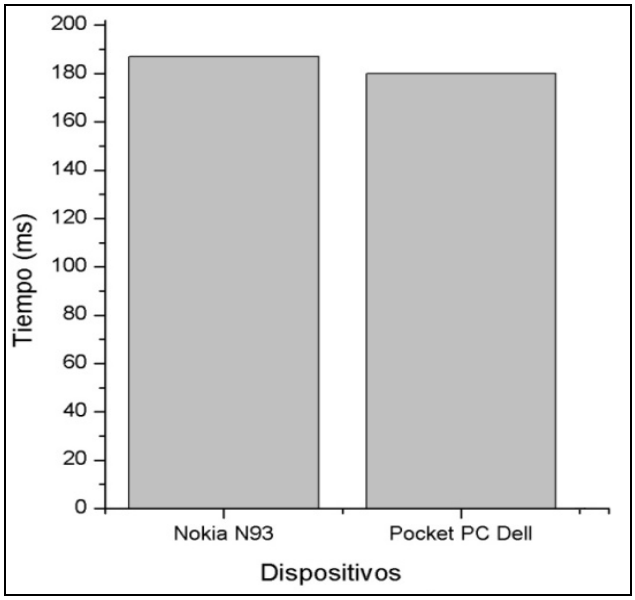

Figura 50. Rendimiento de la plataforma en el proceso de descarga y almacenamiento de procesos de negocio.

Esta prueba también se hizo utilizando la red GPRS y el dispositivo Nokia N93, para el cual se obtuvo un tiempo de 2610 ms, significativamente mayor en comparación al obtenido en la red inalámbrica. Con esto se puede afirmar que en la red GPRS se obtienen tiempos de respuesta aceptables.

- **Prueba de rendimiento DR2:** En la figura 51 se muestra la gráfica de resultados de las pruebas de rendimiento para la descarga y almacenamiento de subprocesos de negocio (archivo BPEL y WSDL). Para esta prueba se generaron varios subprocesos con diferentes tamaños que varían desde 1KB hasta 20 KB, estos archivos fueron descargados desde los dispositivos móviles reales utilizando una conexión inalámbrica. Como se visualiza en la figura 51, los diferentes tiempos de descarga varían según el tamaño del archivo, entre más grande es el archivo más demora la descarga. En la figura 51 también se observa que los tiempos de descarga entre los dispositivos móviles reales son muy similares, principalmente para archivos mayores a 10KB, donde los tiempos de descarga varían en milésimas de segundo.

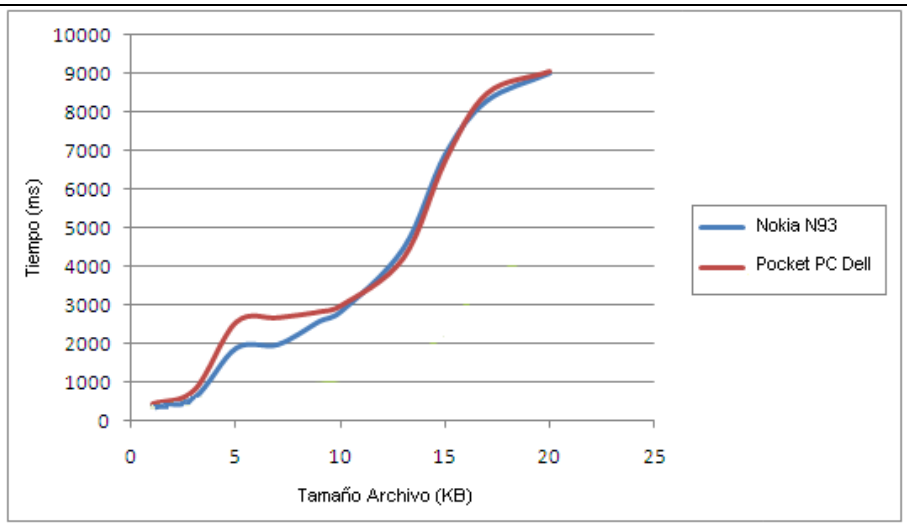

Figura 51. Rendimiento de la plataforma en el proceso de descarga desde diferentes dispositivos

De igual manera se realizo la misma prueba utilizando la red GPRS con el dispositivo móvil Nokia N93 donde se hizo la descarga de los mismos archivos, de aquí se obtuvieron los resultados presentado en la figura 52, observando mayor tiempo de respuesta utilizando la red GPRS. De esto se puede concluir que lo tiempos del proceso de descarga de archivos a los sistemas móviles de información, están ligados al tamaño de los archivos y a la conexión, más que a las capacidades de los dispositivos móviles usados en la prueba.

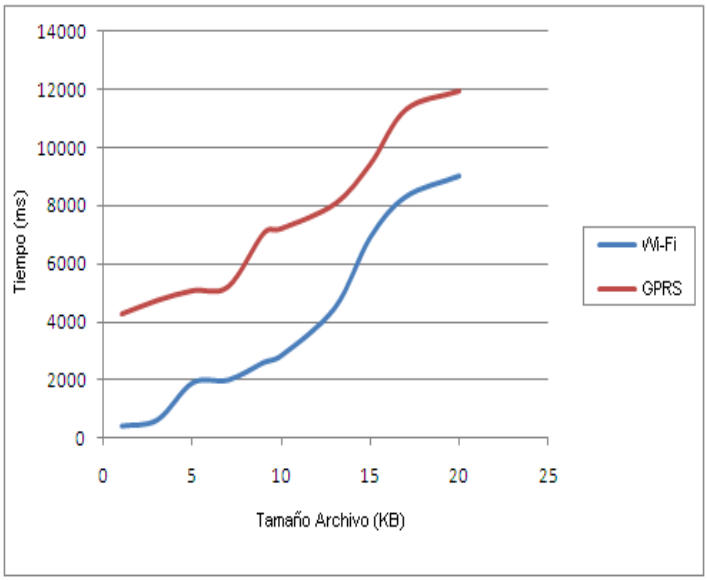

Figura 52. Rendimiento de la plataforma en el proceso de descarga con el acceso desde dos tipos de redes

- **Prueba de estabilidad DE1:** Para determinar la estabilidad que ofrece la plataforma MobFlow cuando varios usuarios acceden al tiempo, durante el proceso de registro y validación de usuarios necesario para la distribución de los subproceso, se creó un cliente Web que simula la ejecución del comando de distribución, tal como se hace desde el sistema móvil de información. De esta manera, con la herramienta JMeter, se tomaron los datos relacionados con la estabilidad del módulo de distribución (figura 53), los cuales indican que a partir de un conjunto de 100 conexiones simultáneas (**Anexo E**) no es posible llevar a cabo el procedimiento solicitado por al menos uno de los clientes remotos, debido a que el tiempo de repuesta de la solicitud supera los 10 segundos.

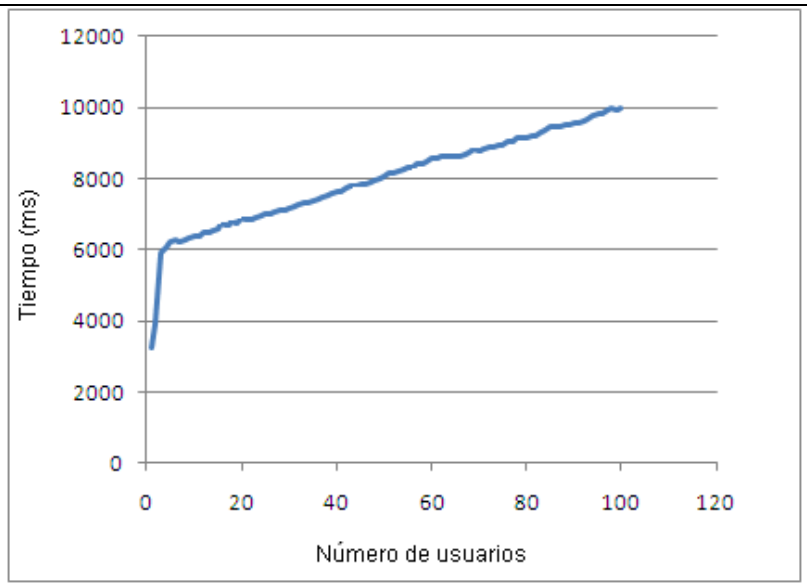

Figura 53. Tiempos de respuesta del proceso de validación y respuesta con acceso de usuarios simultáneos.

Para esta prueba se tuvo en consideración que los tiempos de respuesta de esta solicitud desde un cliente web, varían en muy pocos milisegundos respecto a los tiempos de respuesta obtenidos desde los dispositivos móviles, cuando realizan una solicitud de este tipo conectados a través de una red Wi-Fi. En el caso de los tiempos de respuesta obtenidos al realizar una petición de este tipo desde el emulador, dichos tiempos son prácticamente similares a los realizados desde un cliente web. Por lo cual se considera que los tiempos obtenidos para esta prueba utilizando la herramienta JMeter, varían en pocos milisegundos a los resultados obtenidos en los dispositivos móviles reales, conectándose mediante una red Wi-Fi.

Vale la pena mencionar que 100 conexiones simultaneas para realizar el proceso de distribución, es un caso muy particular, el cual implicaría un escenario que involucra la carga de muchos procesos de negocio que cuenten con varios actores. Caso que no se da en un escenario real.

Tabla 5. Análisis de resultados pruebas para la funcionalidad de distribución

Resultados pruebas para la ejecución sincronizada
• **Prueba de rendimiento ER1:** En la figura 54 se visualiza la gráfica que presenta los resultados de las pruebas de rendimiento, donde se estipulan los tiempos de respuesta promedio que tarda la plataforma en obtener y presentar las variables de sincronización en el sistema móvil de información, proceso requerido para llevar a cabo una ejecución sincronizada del proceso. Para esta prueba se utilizaron dos dispositivos móviles reales, los cuales se conectan al Servidor MobFlow como se ha descrito anteriormente. Con estos se descargaron y ejecutaron los subprocesos de negocio que requerían variables de sincronización para iniciar su ejecución, luego se midió el tiempo de repuesta desde el momento de hacer la petición al servidor hasta obtener la información de las variables en el dispositivo móvil. En los dos dispositivos se logran tiempos de respuesta aceptables al obtener las variables de sincronización (figura 54), las variaciones de tiempo entre el Nokia N93 y la Pocket PC Dell son mínimas. Por otra parte es importante tener en consideración que la información de las variables son pequeños documentos XML los cuales no superan el tamaño de 1KB generalmente, por lo que podemos concluir que este proceso llevado a cabo por la plataforma es optimo para gran parte de los procesos de negocio que pueden ser ejecutados.

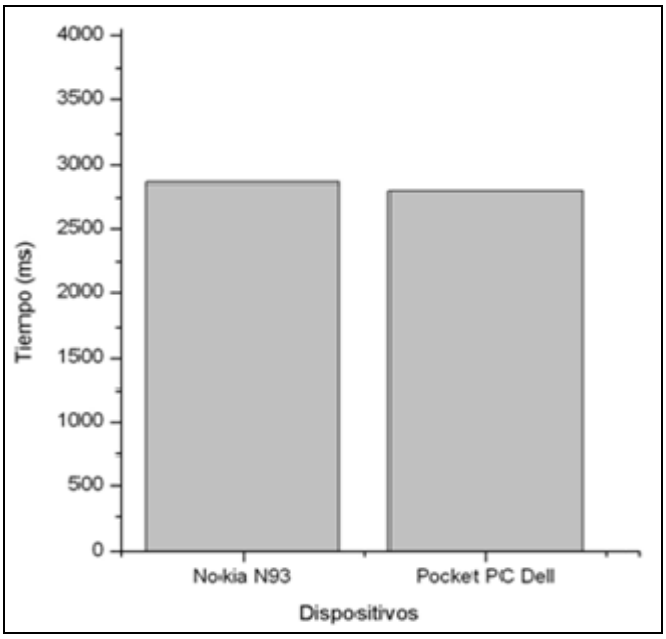

Figura 54. Gráfica de rendimiento del proceso de obtención de variables de sincronización.

Esta prueba se realizó igualmente utilizando diferentes redes inalámbricas con el Nokia N93, los resultados se presentan en la figura 55, donde se observa que a pesar de la diferencia considerable en los tiempos de respuesta entre estas dos redes, dichos tiempos son aceptables.

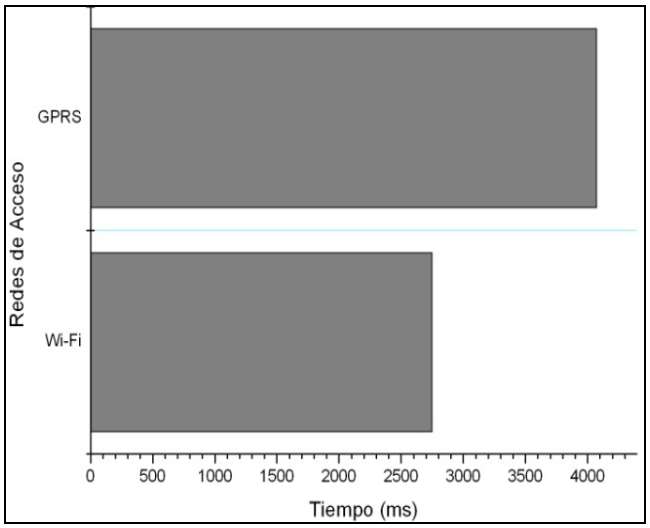

Figura 55. Gráfica de rendimiento del proceso de obtención de variables para la ejecución de un subproceso con acceso desde diferentes redes inalámbricas.

- **Prueba de rendimiento ER2:** El proceso de invocación de un servicio Web, cuyo resultado es requerido para la ejecución de un subproceso de negocio desde el sistema móvil de información, posee una mayor complejidad debido a la cantidad de funciones involucradas. La figura 56 presenta los resultados de las pruebas de rendimiento que determinan el tiempo que demora la plataforma en obtener y enviar el resultado de la invocación de un servicio Web al sistema móvil de información.

Para esta prueba en los dispositivos móviles se descargaron y ejecutaron subprocesos de negocio que requerían invocación de servicios Web, en los cuales se midió el tiempo desde la petición para la invocación de los servicios hasta obtener su respuesta. Se utilizaron los dos dispositivos móviles reales, la evaluación de estos resultados muestra un rendimiento aceptable de la plataforma sobre los dispositivos usados para la prueba.

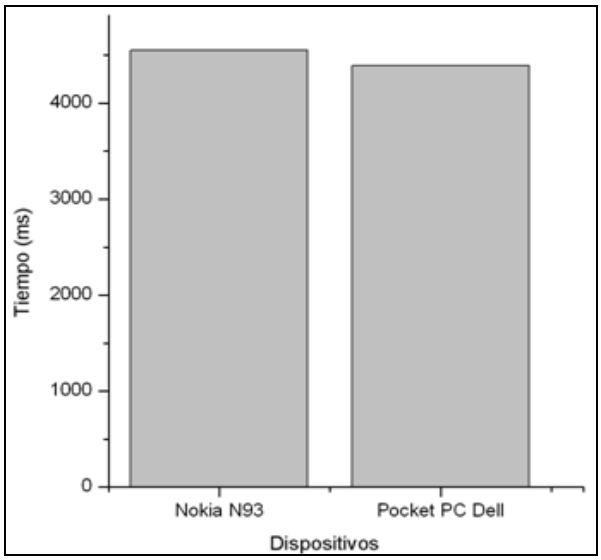

Figura 56 Gráfica de rendimiento del proceso de obtención de respuesta en la invocación de un servicio Web requerido para la ejecución de un subproceso.

Al realizar la prueba sobre diferentes redes inalámbricas se observa que continúa el mismo comportamiento (figura 57), que en las otras pruebas de este tipo, siendo siempre mayor el tiempo de respuesta en las redes GPRS, pero siempre en rangos aceptables.

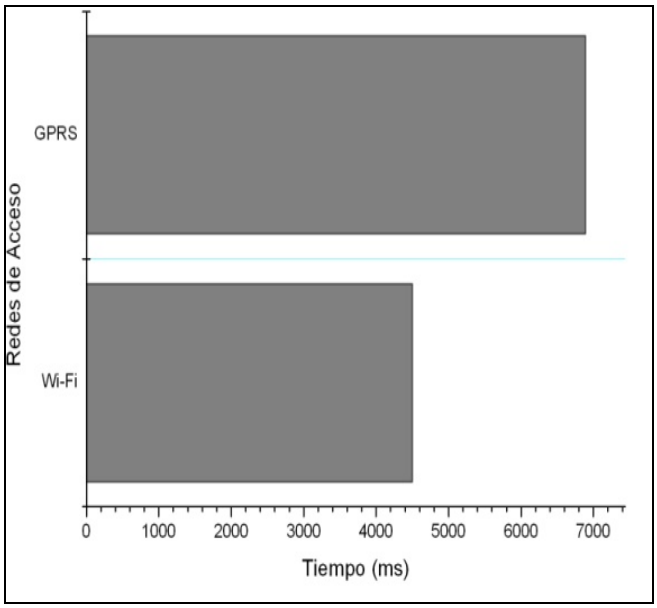

Figura 57. Gráfica de rendimiento en la obtención de respuesta para la invocación de un servicio Web en la ejecución de un subproceso con acceso desde diferentes redes inalámbricas.

Finalmente se observa que los tiempos de respuesta de este proceso aumentan considerablemente respecto a otros procesos ejecutados en la plataforma, pero siempre en rangos aceptables, por lo que se concluye que la plataforma MobFlow tiene un comportamiento tolerable en la invocación de servicios Web.

- **Prueba de estabilidad EE1:** Para determinar la estabilidad que ofrece la plataforma, en el proceso de obtención de las variables de inicio de ejecución, cuando se envía varias solicitudes desde diferentes dispositivos, se creó un cliente Web que simula la ejecución del comando de ejecución sincronizada, tal como se hace desde el sistema móvil de información. De esta manera, con la herramienta JMeter, se tomaron los datos relacionados con la estabilidad del módulo de sincronización, los cuales son presentados en la figura 58, estos indican que a partir de un conjunto de 70 conexiones simultaneas no es posible llevar a cabo el procedimiento solicitado por al menos uno de los clientes remotos (**Anexo E**).

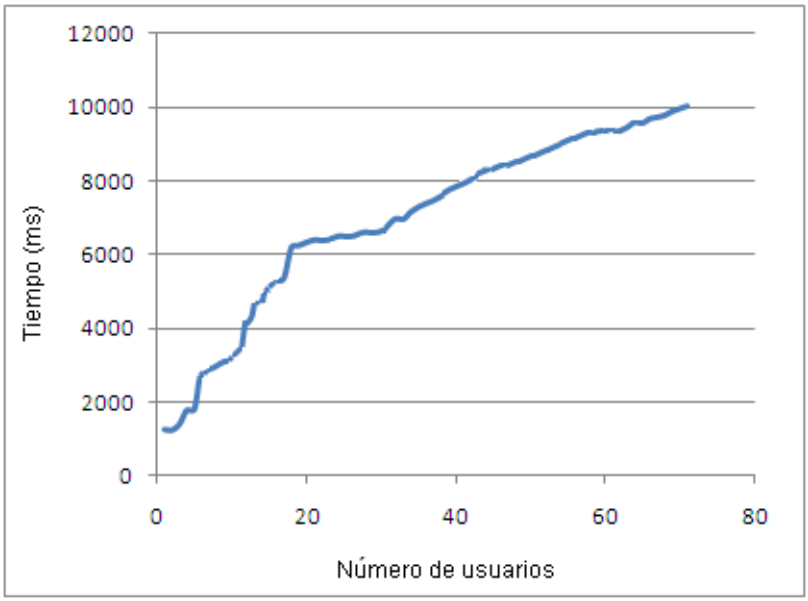

Figura 58. Tiempos de respuesta en la obtención de variables con acceso de usuarios simultáneos.

Un caso de 70 conexiones simultáneas se presentaría en un escenario particular, que no da en un entorno real.

- **Prueba de estabilidad EE2:** Para esta prueba se simularon varios usuarios invocando al tiempo servicios Web por medio de la plataforma. Los datos obtenidos de esta prueba son presentados en la figura 59, e indican que a partir de un conjunto de 30 conexiones simultáneas no es posible llevar a cabo el procedimiento solicitado por al menos uno de los clientes remotos (**Anexo E**).

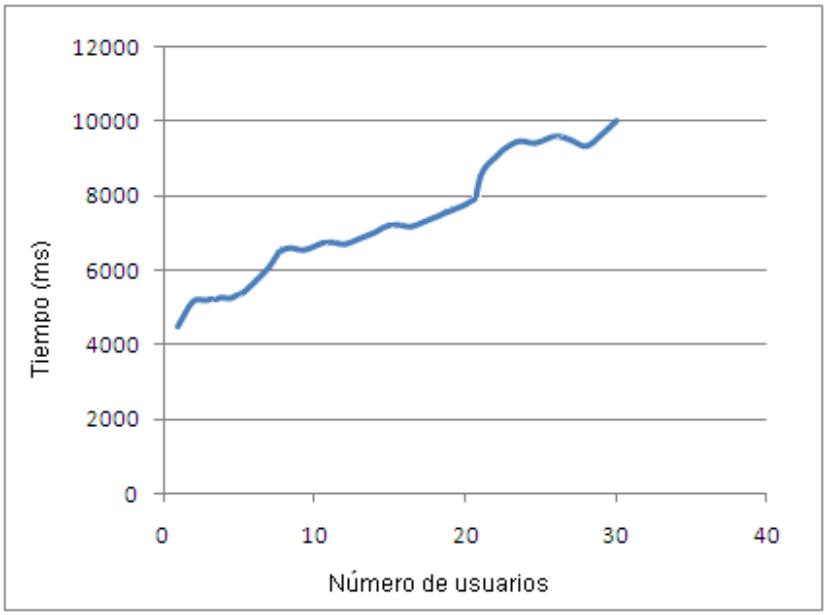

Figura 59. Tiempos de respuesta del proceso de obtención de variables con acceso de usuarios simultáneos.

El número de conexiones simultáneas se reduce significativamente en comparación a las pruebas de estabilidad anteriores, debido a que en este caso se requieren de diversos procesos para realizar la invocación. Además, hay que tener en cuenta que los servicios Web por lo general no se encuentran en la misma y pueden ser expuestos en distintas plataformas lo cual afecta directamente los tiempos de respuesta.

Tabla 6. Análisis de resultados pruebas para la funcionalidad de ejecución distribuida

Resultados pruebas de monitoreo
• **Prueba de rendimiento MR1:** Para esta prueba se definió un proceso BPEL inicial con 10 actividades, donde se hizo una prueba preliminar midiendo el tiempo que se demora la plataforma en presentar la imagen de dicho proceso. Luego a este proceso se agregaron actividades de manera coherente y se procedió a medir el tiempo que toma la plataforma en generar la imagen del proceso con más actividades. En la figura 60 se presentan los resultado obtenidos al realizar esta prueba, donde se puede observar que para procesos de más 550 actividades los tiempos de generación de la imagen son deficientes (**Anexo E**).

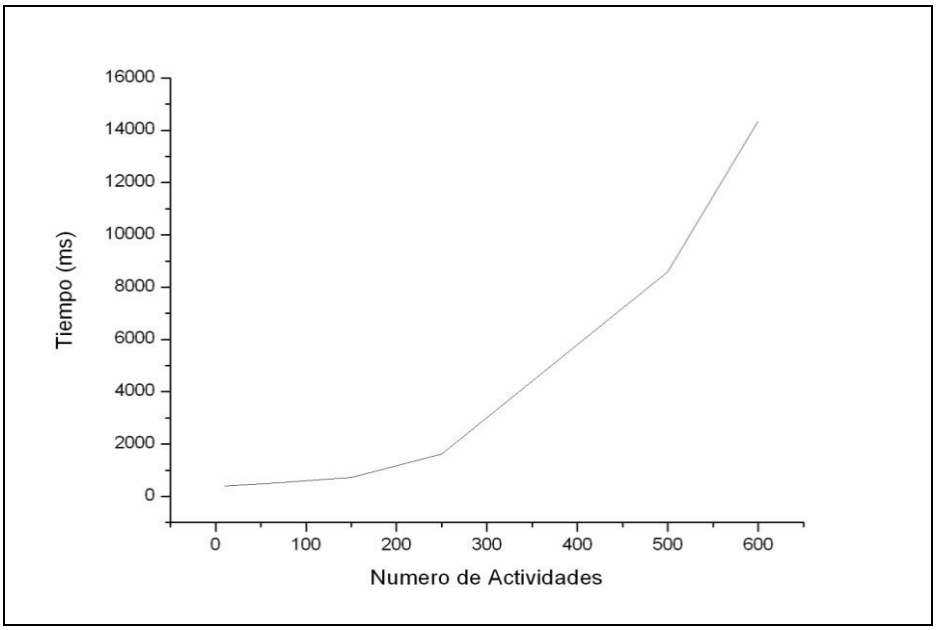

Figura 60. Tiempos de respuesta en la generación de la imagen del proceso según el número de actividades.

También se observa que a partir de 250 actividades se presenta un incremento exponencial en los tiempos de respuestas. Por lo cual es importante destacar que un proceso BPEL que defina más de 250 actividades es un caso poco probable y teniendo en cuenta que la manipulación de imágenes implica una mayor necesidad de procesamiento los tiempos de respuesta para la mayoría de proceso es aceptable.

Tabla 7. Análisis de resultados pruebas para la funcionalidad de monitoreo

Resumen

En este capítulo se describió el caso de estudio seleccionado para probar la plataforma MobFlow. También se presento el prototipo implementado para su validación, exponiendo cada una de las funcionalidades y servicios ofrecidos.

Finalmente, en este capítulo se expusieron las diferentes pruebas realizadas a la plataforma, que permiten medir el grado de rendimientos y estabilidad, con las gráficas y análisis obtenidos de los resultados. De esta se obtuvieron resultados en el rango de óptimo a aceptable, por lo que se considera que la plataforma cumple con las expectativas propuestas y es tolerable para la ejecución de varios procesos de negocio al tiempo (pruebas de estabilidad).

En el siguiente capítulo se presentan las conclusiones del desarrollo de este proyecto y las consideraciones a tener en cuente para el trabajo futuro.

CAPITULO VI

CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se describirán las conclusiones relacionadas al presente trabajo de grado, así como las diferentes propuestas a trabajos futuros alrededor de la distribución de procesos de negocio en sistemas móviles de información.

6.1 CONCLUSIONES

- Si bien en cierto que en trabajos previos [1], [2] se han realizado algunos acercamientos a la distribución de procesos en sistemas móviles de información, el presente trabajo de grado complementó dichas investigaciones proponiendo técnicas de monitoreo y distribución, así como una integración de todos los bloques para la construcción de un sistema integral de distribución y ejecución de procesos de negocios.

- El despliegue distribuido de procesos de negocio en sistemas móviles de información involucra la definición y desarrollo de algoritmos y mecanismos que manipulan el documento BPEL que describe el proceso de negocio. Este trabajo de grado utilizó la transformación de un documento BPEL a un modelo de grafos, para facilitar la visualización, el particionamiento en subprocesos sincronizados y el monitoreo de procesos de negocio en sistemas móviles de información.

- En esta plataforma, los sistemas móviles de información deben contar con una aplicación que interactué con el Motor BPEL que ejecuta los subprocesos. La aplicación móvil incluye el motor de ejecución SLIVER, el cual hace posible ejecutar procesos de negocios en dispositivos móviles. Gracias a esto, la plataforma MobFlow 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.

- La ejecución compartida del proceso de negocio en sistemas móviles de información necesita de una coordinación entre los diferentes ejecutantes de los subprocesos de negocio. Para dar solución a estos requerimientos, MobFlow implementó un mecanismo de sincronización. La sincronización es uno de los puntos más críticos del sistema, ya que es 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 subproceso, 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. Así se logra que el proceso particionado sea igual al proceso de negocio original.

- En el presente trabajo se creó un mecanismo de Monitoreo, el cual guarda toda la información sobre la ejecución de las tarea en los dispositivos móviles, para ser presentadas posteriormente en una interfaz Web que provee la plataforma MobFlow. Este mecanismo brinda varias ventajas como las que se describen a continuación:

    - Debido a que la información del monitoreo es presentada en una interfaz Web, esta puede ser consultada desde Internet por diferentes dispositivos de computo como PC's, PDA's, Celulares entre otros. Lo cual brinda una característica importante a la plataforma, ya que es posible acceder a la interfaz de monitoreo desde cualquier lugar utilizando un dispositivo que tenga acceso a Internet, brindando otro aporte a la movilidad requerida de los trabajadores.

    - Con el monitoreo es posible seguir la ejecución de cada una de las actividades dentro de un proceso, reuniendo la información para evaluar el comportamiento general. Brindando una herramienta importante a la plataforma que permite tomar decisiones acerca del proceso. Como

por ejemplo colocar actividades paralelas que una vez implementadas mejoraran la productividad general.

- En el monitoreo se presenta la información de la ejecución de un proceso de manera textual y gráfica, permitiendo tener una visión global de la ejecución de cada subproceso en los dispositivos móviles.

- La información que almacena la base de datos de la plataforma propuesta, constituye un punto importante para lograr el despliegue distribuido de procesos de negocio en sistemas móviles. En esta se encuentra la información de los subprocesos, los actores, la vinculación de estos con cada subproceso generado, las variables requeridas por la ejecución y las actividades que han sido ejecutadas. Datos que son necesarios para realizar el particionamiento, distribución, ejecución y monitoreo.

- Las pruebas de rendimiento y escalabilidad presentadas en el capítulo V, fueron definidas para medir cada una de las funciones más importantes que comprende la plataforma. En base a los resultados de las pruebas realizadas, se puede concluir que la plataforma se desempeña de una manera eficiente en los aspectos primordiales como particionamiento, distribución, ejecución y monitoreo.

- Gracias al módulo de Interfaz de Servicio y al componente Axis (contenidos dentro del módulo Control de la Ejecución) de la plataforma MobFlow presentada en el capítulo III, se puede lograr la invocación dinámica de servicios Web, permitiendo llamar a cualquier servicio independiente del lugar y la plataforma Web donde se ejecuten, utilizando para ello únicamente la URL donde está alojado el servicio, la operación a realizar y los parámetros requeridos para lograr la invocación, los cuales son obtenidos gracias al archivo WSDL y BPEL del subproceso.

6.2 TRABAJO FUTURO

- Como trabajo futuro queda por establecer mecanismos para manejar fallos (de ejecución, comunicación, distribución entre otros) y realizar la reconfiguración automática de la distribución. Esto con el fin de que al detectar un fallo en un dispositivo, ya sea porque se sale del área de cobertura o simplemente se encuentra apagado, el sistema debe ser capaz de identificar esta situación y proponer dispositivos candidatos para la ejecución de las tareas pendientes para el proceso.

- También se plantean mecanismos para que el motor BPEL instalado en los dispositivos móviles, realice el llamado de Servicios Web desde el mismo dispositivo. De esta manera se reduce el número de comunicaciones entre la aplicación cliente móvil y plataforma.

- Desarrollar un módulo que permita diseñar a partir de una interfaz gráfica los procesos empresariales y generar documentos BPEL y WSDL que sean acordes a los estándares de validación de la plataforma. Es decir genere los BPEL con el formato adecuado para que sean cargados en la plataforma MobFlow.

- Desarrollar mecanismos para que las interfaces gráficas generadas sean acorde a las tareas a ejecutar por cada actor en el dispositivo móvil, manejando estándares de Interfaz Humano-Computador con el fin de que sean más intuitivas, llamativas y agradables a los actores de los procesos.

- Plantear la manera de realizar la distribución de información utilizando múltiples redes como Wi-Fi o WLAN, sobre una gamma más amplia de dispositivos, como lo son los de naturaleza ubicua.

- Realizar la experimentación de la plataforma en un ambiente real, para medir el grado de satisfacción de los usuarios y establecer en cuanto tiempo se reduce la ejecución de los procesos empresariales al ser distribuidos en sistemas móviles de información.

7 BIBLIOGRAFÍA

[1] ALVEZ, Pablo; SCALONE, Marco; FOTI, Patricia. *Generador de aplicaciones orquestadoras. Estado del arte proyecto Batuta*. Facultad de Ingeniería, Universidad de la República, Uruguay, 2005. Disponible en Web: http://peruti.files.wordpress.com/2007/09/toolorquesta.pdf [Consulta: 26-11-2007]

[2] CUERVO, Eva; SOTO, Adolfo. *Nuevas Tendencias en Sistemas de Información: Procesos y Servicios*. Pecvnia, 2 (2006), pp. 129-158 Disponible en Web: http://dialnet.unirioja.es/servlet/fichero_articulo?codigo=1945340&orden=0 [Consulta: 26-11-2007].

[3] FIGUEROA, Cristhian; GUERRERO Andrés. *Arquitectura para la distribución de procesos de negocio en sistemas móviles de información*. Facultad de Ingeniería Electrónica y Telecomunicaciones, Universidad del Cauca, Popayán, Colombia 2008.

[4] FIGUEROA, Cristhian; GUERRERO, Andrés; ORDOÑEZ, Armando; MACA, Mauricio; CORRALES, Juan. *Distribución de Procesos de Negocios en Sistemas Móviles de Información Basada en un Algoritmo de Colonia de Hormigas*, Revista Avances en Sistemas e Informática, vol 4. No 1, pp 67 – 72, Jun. 2007.

[5] FIGUEROA, Cristhian; GUERRERO, Andrés; ORDOÑEZ, Armando; MACA, Mauricio; CORRALES, Juan. *BDMobis un Sistema para la Ejecución de Workflows Distribuidos es Sistemas Móviles de Información*. in Proc. 2007 Congreso Colombiano de Comunicaciones IEEE COLCOM. Articulo No SC02.

[6] ORDÓÑEZ, Armando; FIGUEROA, Cristhian, DORADO, Henry, APONZÁ, Gustavo. *BDMobIS un sistema para la distribución de procesos de negocio en sistemas móviles de información*. in Proc. 2008 Congreso Colombiano de Comunicaciones IEEE COLCOM. Articulo COLCOM AC06

[7] CANTOR, Oskar; MANCILLA, Leonardo. *Arquitectura orientada a comunidades virtuales colaborativas sobre dispositivos móviles: AYLLU.* Facultad de Ingeniería, Pontificia Universidad Javeriana, Bogotá DC, Colombia 2005. Disponible en Web: www.javeriana.edu.co/biblos/tesis/ingenieria/Tesis199.pdf [Consulta: 14-01-2008]

[8] BARESI, Luciano; MAURINO, Andrea; MODAFFERI, Stefano. *Workflow Partitioning in Mobile Information Systems*. In Kluwer, editor, In Proc. of IFIP TC8 Working Conference on Mobile Information Systems , volume 158 of IFIP International Federation for Information Processing, 2004.

[9] MAURINO, Andrea; MODAFFERI, Stefano. *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.

[10] BARESI, Luciano; MAURINO, Andrea; MODAFFERI, Stefano. *Towards Distributed BPEL Orchestrations*. Workshop on Software Evolution through Transformations: Embracing the Change -Electronic Communications of the EASST, Volume 3. 2006.

[11] HACKMANN, Gregory; HAITJEMA, Mart; GILL, Christopher, 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

[12] HACKMANN, Gregory; HAITJEMA, Mart; GILL, Christopher, GRUIA, Catalin. *Extending BPEL for Interoperable Pervasive Computing*. Washington University, Department of Computer Science and Engineering, Tech. Rep. WUCSE-06-37, 2006.

[13] The Workflow Management Coalition. Workflow: http://www.wfmc.org [Consulta 17-01-2008]

[14] Workflow Management Coalition: The Workflow Reference Model www.wfmc.org/standards/referencemodel.htm [Consulta 17-01-2007]

[15] Business Process Management Group (BPMG). http://www.bpmg.org/ [Consulta 01-04-2008]

[16] RAJ, Jog; OWEN, Martin. *BPMN and Business Process Management. Introduction to the New Business Process Modeling Standard*. Popkin Software. http://www.omg.org/bpmn/Documents/6AD5D16960.BPMN_and_BPM.pdf [Consulta 01-04-2008]
[17] Business Process Management Initiative http://www.bpmi.org/ [Consulta 01-08-2008]
[18] Object Management Groups. www.omg.org [Consulta 01-04-2008]
[19] Business Process Modeling Notation. www.bpmn.org [Consulta 02-08-2008]
[20] Unified Modeling Language. www.uml.org [Consulta 01-03-2008]
[21] WHITE, Stephen. *Introduction to BPMN*. Technical report, IBM Corporation, 2004. Disponible en Web : www.bpmn.org/Documents/Introduction%20to%20BPMN.pdf [Consulta 21-01-2008]
[22] http://www.w3.org/TR/webont-req/ [Consulta 01-08-2008]
[23] Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001 http://www.w3.org/TR/wsdl [Consulta 04-06-2008]
[24] Web Services AXIS http://ws.apache.org/axis/ [Consulta 10-04-2009]
[25] PEREZ, Juan. *Notaciones y lenguajes de procesos. Una vision global. Reporte de Investigación*. Department of Computer Languages and Systems. Universidad de Sevilla. Disponible en Web: http://www.lsi.us.es/docs/doctorado/memorias/Perez,%20Juan%20D.pdf [Consulta: 21-01-2008]
[26] Business Process Execution Language Version 1.1. the BPEL4WS Specification Disponible en Web: http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf [Consulta 09-12-2007]
[27] BPML|BPEL4WS: A Convergence Path toward a Standard BPM Stack www.bpmi.org/downloads/BPML-BPEL4WS.pdf [Consulta 09-12-2007]
[28] W3C. "*Guía Breve de Servicios Web"*. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb. [Consulta 10-01-2009].
[29] YAWL Yet Another Workflow Language. http://www.yawl-system.com/ [Consulta 20-10-2008]
[30] ebXML: Electronic Business eXtensible Markup Language. http://www.ebxml.org/ [Consulta 04-02-2008]
[31] Services Architecture W3C - Working Group Note 11 February 2004.Disponible en Web: http://www.w3c.org/TR/2004/NOTE-ws-arch-20040211/ [Consulta 26-11-2008]
[32] A WebV2 Whitepaper. WebV2. www.webv2.com [Consulta 04-10-2008]
[33] B. Hendrickson and R. Leland. *A multilevel algorithm for partitioning graphs.* in Proceeding of Supercomputing '95, ACM, Diciembre 1995.
[34] BALASOORIYA, Janaka; PADHYE, Mohini; PRASAD, Sushil; NAVATHE, Shamkant. BondFlow: *A System for Distributed Coordination of Workflows over Web Services*. Georgia Institute of Technology. Georgia State University, Atlanta.
[35] WIETRZYK, Vland; TAKIZAWA Makoto. *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). Disponible en Web: http://www.iis.sinica.edu.tw/page/jise/2003/200301_02.pdf [Consulta 14-01-2008]
[36] ÁLVAREZ, Pedro; ESPINOZA, Mauricio. *El ciclo de vida de un servicio Web compuesto: virtudes y carencias de las soluciones actuales*. Departamento de Informática e Ingeniería de Sistemas. Universidad de Zaragoza. 2005 Disponible en Web: http://webdiis.unizar.es/~mespinoz/ReporteComposici%F3n.pdf [Consulta: 10-12-2007]
[37] V. Pelechano, M. Ruíz, J. J. Fons, P. Valderas. *Desarrollo de aplicaciones web basadas en servicios WEB XML. Un Caso Práctico*. Disponible en: http://oomethod.dsic.upv.es/anonimo/..%5Cfiles%5CBookChapter%5Ceecw02.pdf [Consulta 20-01-2008].
[38] LARMAN, Craig. *UML y Patrones: Introducción al Análisis y Diseño Orienta a Objetos*. Ed. Prentice Hall. Mexico, 1999.
[39] S. Kurkovsky, Bhagyavati, A. Ray. *Modelling a Grid-Based Problem-Solving Environment for Mobile Devices*. Journal of Digital Information Management. Volume 2. Number 2. June 2004.
[40] S. Kurkovsky, Bhagyavati, A. Ray. *A Collaborative Problem-Solving Framework for Mobile Devices.* Columbus State University. 4225 University Ave. Columbus, GA 31907. 1-706-565-3520.

[41] C. Marin, R. Brena, *Arquitectura de Workflow Distribuido Basada en Agentes Inteligentes*. in Proc. 2º Congreso Internacional en Innovación y Desarrollo Tecnológico 15-19 noviembre 2004, Cuernavaca, Morelos, México

[42] I. Baloche, E. Draperi. *Projet Annuel ISTY 3 Transformation BPEL en Graphe*. Draft Unpublished. pp. 5-32. 2006.

[43] Web Services Choreography Description Language Version 1.0 (WS-CDL) http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/ [Consulta: 23-03-2008]

[44] Web Services Flow Language (WSFL) http://xml.coverpages.org/wsfl.html [Consulta: 02-05-2008]

[45] RICO, Jorge; GOMEZ, Jairo. Documento de estado del arte en SOA y Cálculo Pi. Grupo de Investigación Arquitecturas de Software. Universidad Distrital Francisco José de Caldas. Bogotá. 2007. Disponible en Web: http://www.udistrital.edu.co/comunidad/grupos/arquisoft/index.php?id=78&type=1 [Consulta 04-03-2008].

[46] GINER, Pau; TORRES, Victoria. Una Propuesta Basada en Modelos para la Construcción de Sistemas Ubicuos que den Soporte a Procesos de Negocio. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. España

[47] Core J2EE Patterns - Session Facade. http://java.sun.com/blueprints/corej2eepatterns/Patterns/SessionFacade.html [Consulta: 10-01-2009]

[48] kXML 2. http://kxml.sourceforge.net/kxml2/ [Consulta: 05-03-2009]

[49] kSOAP 2. http://ksoap2.sourceforge.net/ [Consulta: 05-03-2009]

[50] Core J2EE Patterns - Data Access Object (DAO). http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html [Consulta: 10-01-2009]

[51] Model-View-Controller (MVC). http://java.sun.com/blueprints/patterns/MVC.html [Consulta: 10-01-2009]

[52] J2ME record management store. http://www.ibm.com/developerworks/library/wi-rms/ [Consulta: 10-09-2009]

[53] BPEL Tutorial - Tutorial 3: Manipulating XML Documents in BPEL. http://www.oracle.com/technology/products/ias/bpel/pdf/orabpel-Tutorial3-DataManipulationTutorial.pdf [Consulta: 11-03-2009]

[54] MARTIG, Sergio; SEÑAS, Perla. *Grafo Integrador de un Mapa Conceptual Hipermedial.* Grupo InE. Departamento de Ciencias de la Computación. Universidad Nacional del Sur. Avda. Alem 1253 – (8000) Bahía Blanca –Argentina. Disponible en Web: http://www.niee.ufrgs.br/eventos/SBC/2000/pdf/wie/art_completos/wie028.pdf [Consulta 07-09-2009].

[55] Workflow y UML Visión General. http://www.willydev.net/descargas/articulos/general/WorkFlowUML.pdf [Consulta: 22-10-2007]

[56] Joines, S., Willenborg, R., Hygh, K., "Performance Analysis for Java Websites", Addison-Wesley, ISBN-13: 978-0201844542, Septiembre de 2002.

[57] Jakarta, "JMeter UserManual", http://jakarta.apache.org/jmeter/usermanual/index.html [Consulta: 04-10-2009]

[58] Rizzini A. *Mobile Environmental Management System*. Maestría en Ciencias dela Computación del Colegio de Ingeniería. Universidad College Dublin Irlanda. Febrero 15, 2007.

[59] GRUPO INGENIERÍA TELEMÁTICA (GIT) – Facultad de Ingeniería Electrónica y Telecomunicaciones. Universidad del Cauca. http://git.unicauca.edu.co/

[60] Sun Java (TM) Wireless Toolkit 2.5.1 for CLDC http://java.sun.com/products/sjwtoolkit/ [Consulta: 10-02-2009]

[61] ELECTROCREDITOS DEL CAUCA http://www.electrocreditos.com [Consulta: 10-05-2008]

[62] JavaME - Mobile Information Device Profile (MIDP); JSR 118. http://java.sun.com/products/midp/ [Consulta 10-10-2008]

[63] JavaME - Connected Limited Device Configuration (CLDC); JSR 139 http://java.sun.com/products/cldc/ [Consulta 10-10-2008]