Evaluación de Tecnologías de Procesos de Negocio centrado en motores BPEL

Informe de Proyecto de Grado

Año 2010

Autores: Daniel Bonini, Federico Parins

Tutor: Andrea Delgado
Co-Tutor: Pablo Miranda

Resumen del trabajo

Un Proceso de Negocio (Business Process) es un conjunto de actividades que son realizadas en coordinación en entorno organizacional y técnico. Por lo general, estas actividades son realizadas manualmente aunque las organizaciones obtienen beneficios adicionales si utilizan sistemas software para coordinarlas. Es ahí donde entran los Business Process Management Systems (BPMS), los cuales permiten modelar, ejecutar y evaluar procesos de negocio mediante la integración de diversas tecnologías como ser: sistemas de Workflow, sistema de gestión de reglas de negocio, herramientas para el monitoreo y análisis de las actividades, entre otros.

A los efectos de tener un panorama más claro en cuanto a las tecnologías disponibles, el objetivo de este proyecto, enmarcado en un proyecto de investigación más general, será realizar una evaluación en profundidad de motores BPEL con licencia Open Source y su interoperabilidad con otras herramientas como pueden ser herramientas de modelado y motores de reglas. Es de especial interés evaluar dos aspectos principales de dicha interoperabilidad: la ejecución de archivos WS-BPEL creados en otras herramientas, y el esfuerzo requerido para implementar la invocación a sistemas y componentes existentes desde el flujo de ejecución del Proceso de Negocio definido. Asimismo, resultará de interés la evaluación de interoperabilidad entre logs de ejecución generados por el motor WS-BPEL, su carga y análisis en otras herramientas como puede ser ProM (Process Mining http://prom.win.tue.nl/tools/prom/).

Palabras Claves

Proceso de Negocio, Management, Integración, Interoperabilidad, Gestión, Web-Service, Patrones, Modelado, Workflow, Deploy, Servidor, ESB

Tabla de contenido

1. Descripción del Proyecto

Este trabajo se enmarca en el trabajo de investigación del grupo COAL y del grupo de Métodos Formales del InCo, en la línea recientemente incorporada de Procesos de Negocio. Entre los objetivos se incluye el estudio del marco teórico de Procesos de Negocio, su inclusión en el desarrollo de software así como su relación con otros paradigmas como Model Driven Development (MDD) y Service Oriented Computing (SOC), estándares asociados y aplicaciones. Asimismo, este trabajo se encuentra vinculado con las colaboraciones en esta línea de investigación de Procesos de Negocio con el Grupo de investigación ALARCOS de la Universidad de Castilla – La Mancha, y los doctorados en curso.

1.1. Objetivos

El principal objetivo de este proyecto es realizar una evaluación a fondo de motores de WS-BPEL con licencia Open Source, y su interoperabilidad con otras herramientas como pueden ser herramientas de modelado y motores de reglas.

Los objetivos parciales del proyecto fueron los siguientes:

1. Realizar el estado del arte en Procesos de Negocio, en particular en lo que refiere a la fase de ejecución, motores de procesos y características

2. Definir criterios a evaluar en motores de procesos, incluyendo revisión de metodologías, técnicas y trabajos existentes al respecto (por ej. implementación de Patrones de Procesos o Workflow)

3. Seleccionar motores de procesos a evaluar (mínimo 6 motores WS-BPEL)

4. Evaluar los motores de procesos seleccionados (6 herramientas) incluyendo instalación, pruebas, implementación y ejecución de prototipos

    4.1. Se debe hacer especial énfasis en las evaluaciones sobre la interoperabilidad de cada motor con herramientas de diseño de procesos
    4.2. Se debe hacer especial énfasis en las evaluaciones sobre la interoperabilidad de logs de ejecución cada motor con la herramienta ProM – Process Mining (ProM, 2011)

5. Realizar el caso de estudio de aplicación con herramientas seleccionadas (de las evaluadas previamente) que podrá ser de laboratorio o real en alguna organización afín

6. Realizar el Informe final del proyecto de grado por capítulos, incluyendo correcciones y defensa

1.2. Cumplimiento de objetivos

Se puede afirmar que se ha logrado cumplir con el objetivo principal del proyecto, mediante el cumplimiento de los objetivos parciales del proyecto, los cuales fueron mencionados en la

sección anterior. A continuación, se presenta la forma en que se cumplieron los objetivos parciales del proyecto:

1. La realización del estado del arte fue llevada a cabo mediante una investigación exhaustiva sobre Procesos de Negocio, la cual generó como entregable final el Informe del Estado del Arte, con todo el marco teórico necesario para el proyecto.

2. La Definición de los Criterios para evaluar las herramientas fue llevada a cabo determinando aspectos de especial interés para evaluar, definiendo también, escalas para la evaluación y los procedimientos a través de los cuales se evaluaran los criterios. El entregable final generado para este objetivo fue un documento de Definición de Criterios.

3. La selección de motores fue realizada mediante una investigación en amplitud de herramientas de Procesos de Negocio, en la cual el entregable final generado fue la lista de los motores que implementan el estándar WS-BPEL y sus características. De esta lista se seleccionaron 6 motores a ser evaluados.

4. La evaluación de los motores fue llevada a cabo mediante la ejecución de los procedimientos definidos. Dependiendo de los resultados obtenidos con dichas ejecuciones, se le asignó a cada motor valor dentro de la escala definida. El entregable relacionado a este objetivo fue el documento de Evaluación de Motores.

    4.1. El estudio de la interoperabilidad con herramientas de diseño de procesos fue realizada utilizando eClarus, como dicha herramienta. Para ello, se realizó el deploy de los procesos diseñados en eClarus en cada motor, documentando los errores que se produjeron y como se solucionaron o los errores remanentes.
    4.2. El estudio de la interoperabilidad de logs de ejecución con la herramienta ProM fue llevada a cabo mediante la importación de dichos logs en el entorno de la herramienta, indicando en cada caso la forma de importación y los resultados obtenidos.

5. El caso de estudio fue realizado utilizando el motor más adecuado, según las evaluaciones realizadas, implementando un proceso real cuyo escenario de ejecución es un hospital. El entregable final asociado a dicho objetivo fue la implementación y documentación de dicho proceso, en el motor determinado.

6. El informe final del proyecto, fue realizado mediante la recopilación de toda la información generada en el trascurso del mismo. Al mismo se le agregó información sobre la gestión del proyecto y las conclusiones del mismo.

1.3. Organización del documento

En cuanto a la organización del presente informe, en el Capitulo 2, se encuentra el Estado del Arte, con el relevamiento realizado sobre información existente y relacionada al trabajo con BPM y ejecución de Procesos de Negocio con BPEL. En el Capitulo 3 se presentan los Criterios para Evaluación, incluyendo los motores que se seleccionaron para el resto del trabajo. El Capitulo 4 se centra en las Evaluaciones de los Motores, presentando los resultados obtenidos y sus debidas conclusiones. El Capitulo 5 presenta el Caso de Estudio realizado y el Capitulo 6,

la información pertinente a la Gestión del trabajo realizado. Finalmente en el Capitulo 7, se presentan las Conclusiones del trabajo, incluyendo también los posibles trabajos a futuro.

2. Estado del Arte

2.1. Introducción

Según (Weske, 2007), la industria del software cada vez más está enfocándose en los temas relacionados con los Procesos de Negocio. Se desea lograr con esto, un desarrollo cada vez más escalable y robusto de aplicaciones. Cabe aclarar también que este enfoque tiene un interés bastante importante en actores relacionados con aspectos administrativos del negocio. Estos actores buscan una mejora en sus negocios, en aspectos tales como mejora de la satisfacción de los clientes, reducción de costos, entre otros.

Para saber cómo y por qué se llegó a estas necesidades e intereses debemos recordar cómo era el desarrollo en sus comienzos. Los primeros sistemas de información eran demasiado monolíticos, implementaban todos los servicios que necesitaban y controlaban el acceso a los datos (que en un principio era en archivos). Con el advenimiento de los sistemas operativos y sobre todo de las bases de datos, muchos de estos problemas fueron resueltos. El abanico de los posibles sistemas que se podía desarrollar era mucho mayor. Si a esto último, le agregamos tecnologías de programación orientada a objetos y separación en capas, las posibilidades eran aún mayores.

Sin embargo, como menciona (Weske, 2007), con el tiempo surgió la necesidad de integrar estos sistemas, con lo cual, aparecieron problemas nuevos, entre ellos la semántica y redundancia de los datos, así como también su heterogeneidad. Para solucionar estos inconvenientes, se crearon los sistemas ERP (Enterprise Resource Planning), entre otros (CRM, etc.). Este tipo de sistemas tiene como principal ventaja que provee una base de datos integrada, resolviendo así muchos de los problemas antes mencionados.

Pero el mantenimiento de los distintos sistemas se hacía cada vez más inmanejable. Realizar cambios en los flujos de los procesos representaba cambios de código, y esto repercutía en los demás sistemas. Es entonces que surgen las tecnologías de manejo de Workflows, las cuales permiten facilitar dichas modificaciones en los flujos de procesos, debido a la no necesidad de codificación extra.

Los Procesos de Negocio surgen de la observación de los productos que generan las organizaciones, enfocándose en las actividades involucradas en dicho proceso. De esta forma y con herramientas adecuadas, se puede organizar de forma más provechosa dichas actividades y mejorarlas. Debido a que los Workflows resuelven apenas la parte de ejecución del proceso, BPM (Business Process Management) busca tener una perspectiva amplia de todo el proceso. Para lograr dicha visión, BPM agrega otros aspectos relacionados al análisis tales como simulación, verificación, monitoreo y configuración. (Weske, 2007)

A determinado nivel organizacional, los Procesos de Negocio son vitales para entender como las organizaciones trabajan, así como también, juegan un rol sumamente importante en el diseño e implementación de Sistemas de Información flexibles. Dichos Procesos de Negocio, están fuertemente influenciados por conceptos y tecnologías de 2 grandes áreas, la administrativa de la organización y el área técnica, debiéndose esto a las distintas necesidades de ambos. El área administrativa busca mejorar sus métodos operacionales, generando mejores productos, mientras que el área técnica procura implementar software más robustos, flexibles y escalables.

Para entender mejor los conceptos que rodean a los Procesos de Negocio, se presentan en este capítulo, los principales conceptos relacionados (definiciones, ciclo de vida, etc.), así como la descripción de los principales estándares y tecnologías que se relacionan o son de especial interés para este trabajo. Para complementar dicha información, también se presentan los respectivos Anexos, que se indican en cada tema desarrollado.

Este capítulo está organizado como se describe a continuación: en la sección 2.2 se presentan las definiciones importantes sobre BPM incluyendo el Ciclo de Vida de los Procesos de Negocio, la sección 2.3 profundiza en el Modelado de Procesos y estándares existentes al respecto, en la sección 2.4 se describen los Lenguajes de Ejecución/Interpretación de procesos de negocio, en la sección 2.5 se describen diversos Motores de Ejecución de procesos y finalmente en la sección 2.6 se presenta un resumen del trabajo.

2.2. Conceptos de BPM

Para entender el paradigma BPM, en (Weske, 2007) se presentan, a continuación, algunas definiciones que son de carácter fundamental.

En primera instancia, es necesario entender que es un **Proceso de Negocio** o **Business Process**, para ello, (Weske, 2007) menciona que *es un conjunto de actividades que se ejecutan coordinadamente en un ambiente organizacional y técnico. Juntas, estas actividades logran el objetivo del negocio. Dichas actividades se pueden ejecutar de forma individual por cada organización, así como también en conjunto con otras*.

Una vez definidos los procesos tales como se mencionaron anteriormente, es necesario realizar una representación de los mismos. Con este objetivo, (Weske, 2007) introduce a **BPM (Business Process Management)**, teniendo éste como primordial objetivo el de tener una *explícita representación del negocio que se desea modelar, para que el mismo pueda ser ejecutado respetando fielmente las características del proceso en cuestión*. Además de su objetivo principal, engloba conceptos, métodos y técnicas para diseño, administración, soporte y demás actividades que involucren al proceso de negocio. Una vez que las actividades se encuentran bien definidas se ven sujetas al análisis, mejoras y ejecución de las mismas.

Tradicionalmente estos procesos son ejecutados manualmente por personas que conocen el negocio. Sin embargo, para tener más beneficios las organizaciones están comenzando a utilizar software que automatice la ejecución de procesos. Dicho software se denomina **BPMS**

(Business Process Management System) y es el encargado de coordinar la ejecución de los procesos y sus respectivas actividades.

Otro concepto importante respecto a BPM, es el de **Ciclo de vida** de los Procesos. El mismo consiste en fases que están relacionadas unas con las otras y que están organizadas en forma de ciclo. Sin embargo no existe un orden temporal estricto de ejecución, pudiendo muchas actividades de diseño e implementación ser ejecutadas simultáneamente.

A continuación se muestra el diagrama que presenta (Weske, 2007) donde se puede observar dichas fases:

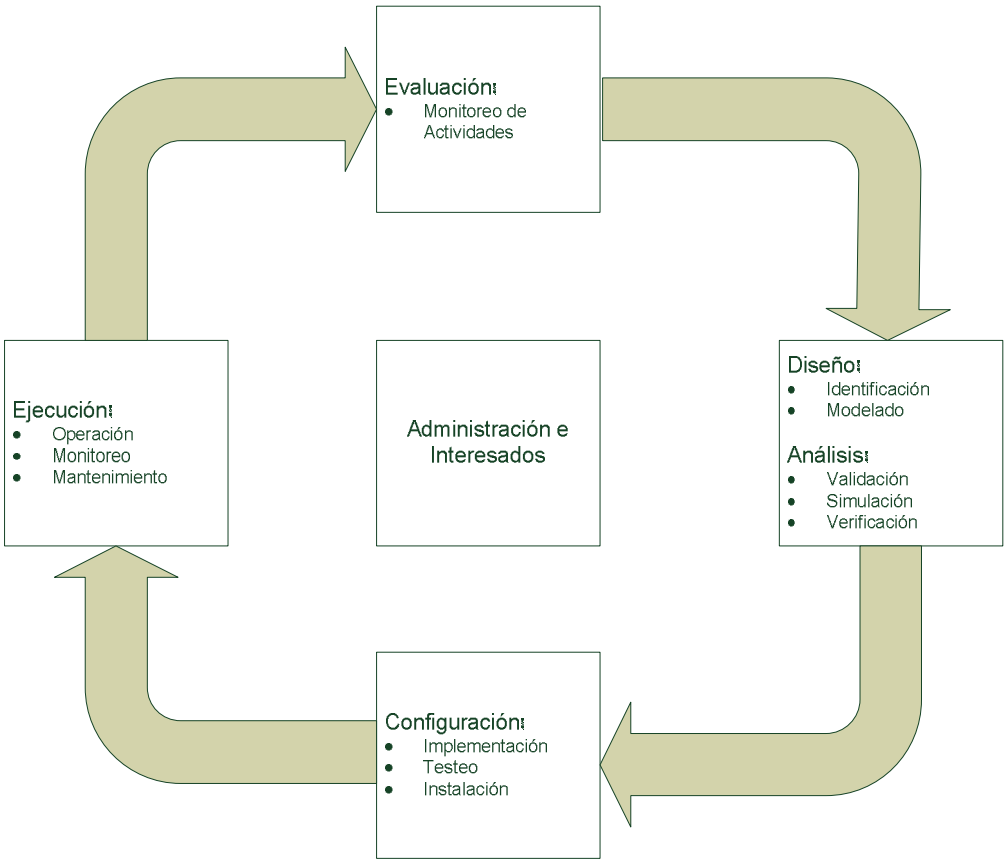

Figura 1 - Ciclo de Vida (Weske, 2007)

- **Diseño y análisis**

  Se llevan a cabo estudios técnicos y organizacionales en los cuales los Procesos de Negocio son identificados, revisados, validados y representados. Se busca realizar modelos de forma clara y explícita, facilitando así a los interesados en la organización entenderlos y mejorarlos. Una vez realizado el desarrollo del proceso, es necesario validarlo. Como técnicas de validación podemos mencionar la del *Workshop*, en donde cada participante del mismo, puede discutir con los demás todas las instancias diseñadas en ese modelo. Otra técnica de especial interés es la de Simulación, esto se debe a que simulando dichos procesos, se puede observar de forma más directa aspectos no deseados del diseño.

- **Configuración**

Una vez diseñado y verificado el Proceso de Negocio, es necesario implementarlo utilizando las políticas y procedimientos propios de la organización, o utilizando software, BPMS. En el caso de utilizar un BPMS, es necesario realizar todos los aspectos de configuración que se relacionen con la organización, pudiéndose hacer esto paralelamente con la implementación. Es en esta etapa también, en que se puede enlazar las actividades de los procesos con Sistemas Legados y todo lo relacionado a la transaccionalidad de las mismas. Por último se encuentra el testeo, en donde se percibe si el Proceso de Negocio tiene el comportamiento esperado, o si se lo puede mejorar aun mas.

- **Ejecución**

  Esta fase es la que le sigue a la de Configuración, en este momento se pone en ejecución el Proceso de Negocio, generalmente con la ayuda del software BPMS, el cual controla la ejecución de las instancias definidas en el modelo, garantizando que las actividades se están llevando a cabo según las restricciones impuestas. También en esta fase se encuentra el monitoreo, que permite obtener información relevante del proceso.

- **Evaluación**

  Para realizar esta fase se necesita información relevante del proceso, la cual se puede llegar a utilizar para mejorar dicho proceso u obtener información de indicadores.

Siguiendo con la terminología presentada por (Weske, 2007), se presenta el concepto de la **Orquestación**. La misma actúa como el motor que dirige el Proceso de Negocio, debido a que posee las actividades y sus relaciones del proceso en cuestión, dependiendo de la realidad que se esté modelando. Es importante tener en cuenta que la Orquestación debe ser independiente de cualquier lenguaje de ejecución de procesos (Capitulo 4). Otro aspecto a ser tenido en cuenta es la existencia de patrones que se pueden utilizar como criterio para desarrollar una buena Orquestación. Estos patrones se desarrollan con mayor detenimiento en el Capitulo 5.

Por otra parte, la **Coreografía** (Weske, 2007) representa la interacción entre distintos procesos, pudiendo incluso ser procesos de organizaciones diferentes. Este tipo de interacción se suele llamar *Business-To-Business*.

2.3. Modelado de Procesos

Entre las diversas actividades que se llevan a cabo durante la fase de Análisis y Diseño del ciclo de vida de un Proceso de Negocio, se encuentra el modelado de procesos. En esta actividad se busca representar dichos procesos de la organización en algún lenguaje gráfico y de forma que sea fácil de entender por todos los actores involucrado. En esta representación gráfica de un Proceso de Negocio se incluyen las diferentes partes que definen el proceso, como actividades, reglas e información de control que permita manipular el flujo del proceso.

En la actualidad existe una gran variedad de herramientas, metodologías y lenguajes para el modelado de procesos de negocio, entre los cuales se encuentra EPC (Event-driven Process

Chain) el cual es un lenguaje para describir procesos de negocio introducido en (Keller, y otros, 1992). Además de EPC podemos encontrar otros lenguajes de modelado de procesos como IDEFØ (de la familia de métodos IDEF desarrollado por Knowledge Based Systems, Inc.), redes de Petri y YAWL (Yet Another Workflow Language) (van der Aalst, y otros), el cual fue especificado en base a las Redes de Petri y los Patrones de Workflow, con la intención de crear una especificación que soportara todos los patrones.

Si bien la gama de lenguajes de modelado es variada, ninguno de ellos ha logrado la adopción como estándar al nivel que BPMN lo ha hecho. En lo que resta de la sección se presenta una breve introducción a BPMN.

2.3.1. BPMN

BPMN es una notación estándar para modelado de procesos de negocio desarrollada inicialmente por la BPMI (Business Process Management Initiative) y es mantenida en la actualidad por la OMG (Object Management Group), luego de la fusión de ambas organizaciones. La versión más utilizada de este estándar es la 1.2 (OMG, 2008) pero ya se encuentra disponible la versión 2.0 (OMG, 2009), y está teniendo una fuerte inserción en la industria.

Además de BPMN, BPMI desarrolló otro estándar denominado BPML (Business Process Management Language) el cual pretendía ser un lenguaje de ejecución de Procesos de Negocios que se mapeara directamente con un diagrama BPMN, pero este estándar no prosperó frente a otros como WS-BPEL o XPDL.

BPMN fue creado con dos objetivos, el primero de ellos es que sea entendible por todos los Stakeholders, desde los analistas de negocio hasta el personal técnico. El segundo objetivo es asegurar que los lenguajes XML desarrollados para ejecutar los procesos de negocio, como BPEL, XPDL y BMPL tengan una notación común para poder ser expresados visualmente. Es más, BPMN ha sido especificado para mapearse directamente con BPML y cualquier otro lenguaje de ejecución que no haya sido desarrollado por BPMI.

BPMN consiste en un conjunto de construcciones gráficas que permiten modelar las diferentes partes de un proceso de negocio y que juntas forman un BPD (Business Process Diagram). Aunque este diagrama fue desarrollado con el fin de ser fácil de entender y usar, permite modelar procesos de negocio complejos.

Este estándar provee construcciones de varios tipos para modelar las diferentes características de un proceso de negocio como procesos, sub-procesos, tareas, eventos, decisiones, mensajes dentro de una organización y entre organizaciones. Para modelar el flujo de un Proceso de Negocio, se modelan los eventos que disparan el comienzo del proceso, las actividades que se desarrollan durante la ejecución del proceso, y el resultado final del mismo.

En la Figura 2 se presenta a modo de ejemplo el diagrama de flujo de un proceso de solicitud de crédito. En la misma se pueden observar algunas de las construcciones que provee el estándar BPMN 1.2. Para entrar más en detalle de los elementos de BPMN, se recomienda ver el Anexo 1 – Estado del Arte, en su sección 1.1.

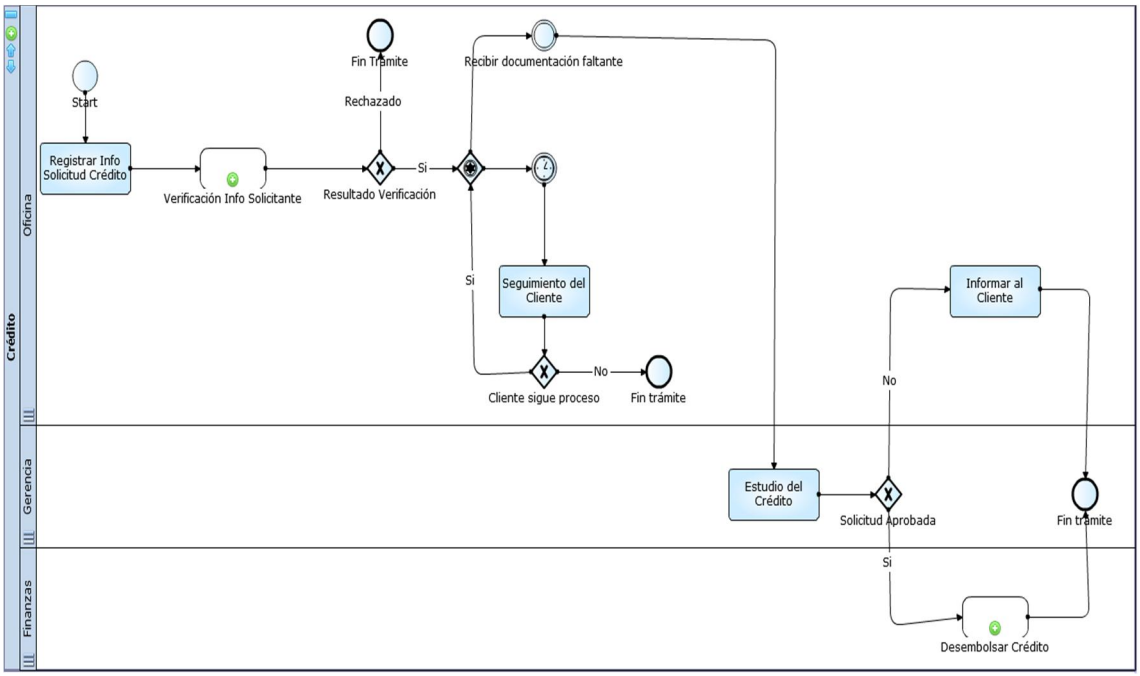

Figura 2 - Ejemplo de proceso en BPMN

2.3.2. Patrones de Procesos

Para el diseño de Procesos de Negocio existen una serie de Patrones de Procesos o Patrones de Workflow, los cuales presentan soluciones a diferentes problemas comunes en el diseño de procesos. El objetivo de esta iniciativa es investigar y detectar los diferentes problemas recurrentes que surgen en el diseño de Procesos de Negocio y que deberían ser soportados por los motores de Workflow y lenguajes de modelado de procesos. Estos problemas recurrentes y sus soluciones son presentados de forma estructurada, similar a los patrones de diseño en Orientación a Objetos. Una descripción más detallada de los patrones se encuentra en el Anexo 1 – Estado del Arte, en su sección 1.2.

2.4. Lenguajes de Ejecución/Interpretación

Siguiendo con el ciclo de vida de un Proceso de Negocio, en la sección anterior se ha atacado la etapa de Análisis de dicho ciclo, mencionando los distintos estándares en cuanto a lenguajes de modelado de procesos. En esta sección se seguirá con la siguiente etapa, la de configuración, con lo cual se estarán describiendo los lenguajes más importantes para ejecución/interpretación de procesos, como son WS-BPEL y XPDL.

2.4.1. WS-BPEL

2.4.1.1. Motivación

Antes de comenzar con la descripción de las características de WS-BPEL, es conveniente introducir algunos conceptos importantes como son SOA (Service Oriented Architecture) y Web services para entender mejor este lenguaje.

Según el modelo de referencia de SOA (OASIS, 2006), se puede definir como un paradigma de organización y distribución de capacidades y características (a través de servicios) en distintos dominios (por ej. organizaciones o empresas distintas). Esto permite una mayor y mejor integración entre sistemas, posibilitando incluso manejar de forma más sencilla, los cambios que pueda tener cada una.

El uso de Web Services tiene como objetivo lograr interoperabilidad entre sistemas (por ej. permitiendo el acceso a servicios), utilizando un modelo de bajo acoplamiento, permitiendo que sistemas completamente heterogéneos puedan interactuar. Para ello se tienen 3 especificaciones que logran dicho objetivo, SOAP (W3C, 2007), WSDL (W3C, 2001) y UDDI (OASIS). SOAP define el protocolo de mensajería para la interoperabilidad de los servicios, WSDL brinda la semántica necesaria de cada servicio y UDDI provee la infraestructura para publicar y utilizar los servicios.

2.4.1.2. Descripción

Integrar sistemas requiere más que simplemente tener protocolos de comunicación para su interacción. Se requiere que los procesos sean capaces de integrar su lógica compleja (con lo cual no son suficientes los archivos WSDL) usando estándares de integración, y eso es a lo que apunta WS-BPEL (OASIS, 2007).

El mismo es un estándar de OASIS (Organization for the Advancement of Structured Information Standards), basándose en lenguajes desarrollados por empresas como IBM (con su lenguaje WSFL), Microsoft (con su lenguaje XLANG) y BEA Systems (con su lenguaje PD4J).

Con el desarrollo de este lenguaje, se eliminaron muchos problemas de interoperabilidad, así como también se potencia la adopción de tecnologías basadas en SOA y BPM. Además, brinda un nivel mayor de flexibilidad y libertad de elección de herramientas, debido a que es un estándar. El mismo está basado en XML, y permite a los desarrolladores crear programas que automatizan las interacciones entre los servicios Web, actividad conocida como Orquestación de servicios Web.

Cuando se define un proceso usando WS-BPEL, el mismo consta de 2 partes (como se muestra en la Figura 3), una primera parte con los archivos WSDL que conforman los Web Services que se van a utilizar en el proceso, y una segunda parte conformada por los archivos BPEL que conforman el proceso en sí, con definiciones del proceso junto con sus actividades, links, variables, excepciones y demás aspectos que hacen al mismo. Luego, estas 2 partes antes mencionadas son utilizadas por el motor de ejecución de BPEL.

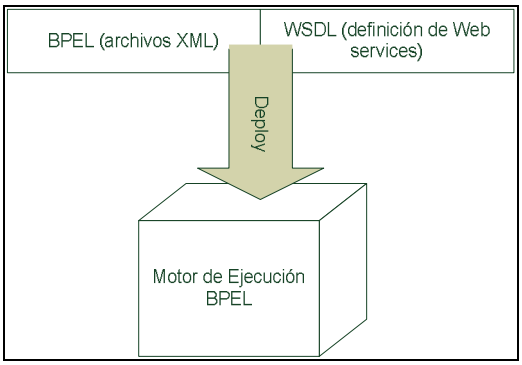

Figura 3 - Definición de proceso en WS-BPEL

2.4.1.3. Elementos de WS-BPEL

Los elementos más importantes que componen un proceso (y una breve descripción de cada uno) interpretado utilizando WS-BPEL, según el estándar (OASIS, 2007), se pueden observar en mayor detalle en el Anexo 1 - Estado del Arte, en la sección 1.3.

2.4.2. XPDL

2.4.2.1. Motivación y descripción

Como se menciona al comienzo de esta sección, XPDL (XML Process Definition Language), es otro de los lenguajes de interpretación o ejecución para Procesos de Negocio.

Según la WfMC (Workflow Management Coalition) en su estándar (WfMC, 2008), es la interfaz entre las herramientas de definición de procesos y los servicios de ejecución. La idea es exportar los procesos diseñados por las herramientas, en formato XPDL y cargarlos en algún servicio de ejecución.

2.4.2.2. Elementos de XPDL

Los elementos más importantes para XPDL según (Havey, 2005), se pueden observar en el Anexo 1 - Estado del Arte, en la sección 1.4.

2.5. Motores de Ejecución

Una vez que contamos con la especificación de un proceso de negocio en algún lenguaje de ejecución/interpretación, como BPEL o XPDL, es necesario contar con alguna herramienta de software capaz de interpretar la especificación del proceso para llevarla a cabo. Es aquí donde se emplean los motores de procesos los cuales proveen el entorno necesario para ejecutar los procesos de negocio durante la etapa de ejecución del ciclo de vida visto en la sección **Error! Reference source not found.**.

2.5.1. Motores de Workflow

La WfMC ha definido, en su glosario de términos, el concepto de Workflow como *"La automatización de un Proceso de Negocio, en su totalidad o en parte, durante el cual documentos, información o tareas son pasadas de un participante a otra para actuar sobre él, de acuerdo con un conjunto de normas de procedimiento."* (Terminology & Glossary, 1999).

Los Sistemas de Administración de Workflows (WMS de sus siglas en ingles Workflow Management System) son sistemas empleados para la automatización de Procesos de Negocio mediante la secuenciación de las diferentes actividades que conforman dicho proceso. Durante la ejecución de esta secuencia de actividades se ven involucrados desde recursos humanos (por ejemplo para la toma de decisiones ante situaciones anormales), bases de datos, sistemas de información e incluso otros WMS.

Existe una gran variedad de tipos de suites WMS, cada una de las cuales utiliza un tipo diferente de lenguaje basado en diferentes conceptos, como XPDL o BPEL. Corren sobre diversas plataformas y cada una cumple con requerimientos diferentes. A continuación presentamos una categorización de los WMS introducida en (Allen, 2001).

- **Production Worflows:** Estos motores de Workflows están diseñados para manejar procesos extremadamente complejos y aumentar la productividad de la organización. Por lo general son embebidos en grandes aplicaciones donde actúan como motores de reglas de negocio. Para ello se busca reducir al mínimo la intervención humana, salvo que ocurra alguna excepción. Estos Workflows se pueden dividir a su vez en Autonomous Workflow Engines y Embedded Workflow.
- **Administrative:** La característica más importante de los Administratvie Workflows es la facilidad para definir un proceso. En este tipo de Workflows es más importante la flexibilidad para definir un proceso que la productividad, lo que ocasiona que el número de instancias por hora que estos sistemas manejan sea uno o dos órdenes de magnitud inferior al de un Production Workflow System.
- **Collaborative:** Estos Workflows se centran en equipos trabajando en conjunto en busca de objetivos comunes. El uso efectivo de trabajo colaborativo es considerado hoy día un elemento clave en el éxito de cualquier empresa. Sin embargo para estos sistemas la performance y la flexibilidad en la definición de procesos no es tan importante como en otros sistemas.

- **Ad-Hoc:** Los Ad-Hoc Workflows permiten a los usuarios crear y corregir definiciones de procesos en forma rápida y fácil frente a circunstancias inesperadas que puedan surgir. Mientras que en un Production Workflow System la empresa es dueña del Proceso de Negocio, en los Ad-Hoc Worflow es el usuario quien es propietario de los procesos.

Los motores de Workflow muy rara vez trabajan en forma aislada, por lo que desde un principio cuando la WfMC comenzó a trabajar en busca de estándares para el sector, el objetivo principal era alcanzar un nivel importante en la interoperabilidad de estos sistemas. El resultado de este trabajo fue la creación del Modelo de Referencia de Workflows en el cual se describen 5 interfaces. A partir de este punto la coalición formo equipos de trabajo para escribir la definición de cada una de las interfaces. En la Figura 4 se pueden observar los diferentes componentes e interfaces definidos en el Workflow Reference Model (Hollingsworth, 1995).

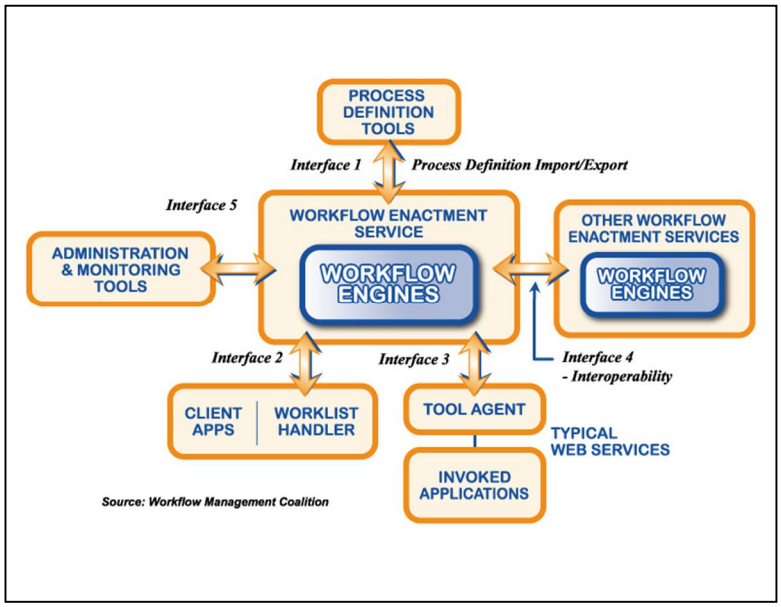

Figura 4 - Workflow Reference Model (Hollingsworth, 1995)

- **Interface 1:** Esta interface está encargada de importar definiciones de procesos de herramientas externas hacia el motor de Workflow donde serán ejecutadas. El objetivo de esta interfaz es desacoplar el lenguaje de modelado del motor de ejecución de los procesos permitiéndole así a los usuarios emplear diferentes modeladores de procesos con cada WMS.
- **Interface 2 y 3:** Estas interfaces combinadas son las que permiten la comunicación con otros sistemas de información, aplicaciones, herramientas, etc.
- **Interface 4:** Esta interface define los mecanismos que deben ser implementados por los fabricantes de Workflows para permitirle a un WMS instanciar y ejecutar un proceso de negocio manejado por otro WMS.
- **Interface 5:** Esta interface permite el análisis por parte de herramientas de auditoría de Workflows mediante información de ejecución generada por el WMS.

2.5.2. Motores BPEL

Si bien BPEL permite la especificación formal de los Procesos de Negocio y automatización de los mismos, no solo dentro de una compañía sino también entre compañías, carece de construcciones que permitan definir interacciones humanas con el proceso, aspecto no menor, teniendo en cuenta que este tipo de interacciones suelen ser frecuentes. Para poder integrar personas a los Procesos de Negocio en motores BPEL se requiere desarrollar soluciones individuales por ejemplo a través de Web Services como actividades externas al proceso y que involucran la interacción humana (Holmes, y otros, 2007). En este sentido IBM, SAP, Adobe, Active Endpoints, BEA y Oracle definieron dos especificaciones, BPEL4People (IBM and SAP, 2005) (Agrawal, y otros, 2007) y WS-Human Task (Amend, y otros, 2007), en busca de incorporar interacciones humanas a los procesos BPEL en forma similar a como lo hacen los motores de Workflow.

Existe una gran gama de motores de procesos cada uno de ellos con diferentes niveles de madurez y aceptación. En la sección 3.2.4.1 de Motores relevados se puede acceder a una lista con todos los motores BPEL.

2.6. Resumen

Esta sección se centró en el estudio del paradigma BPM, definiendo en primera instancia, conceptos claves sobre el mismo, como Proceso de Negocio, Orquestación y Coreografía, pasando por el ciclo de vida de los Procesos, siendo este último el hilo conductor del capítulo.

Luego, teniendo en cuenta la etapa de análisis del ciclo de vida de los procesos, se mencionó como estándar de notación para BPM a BPMN. Del mismo se mencionaron las características más generales del mismo, incluyendo elementos del estándar, pero profundizando sobre patrones para el diseño de los procesos.

Siguiendo con la etapa de Configuración (la siguiente etapa en el ciclo de vida de los procesos), se profundizó sobre los lenguajes de Ejecución/Interpretación más importantes en el mercado, WS-BPEL y XPDL, haciendo mayor énfasis en el primero. Para los mismos, se mencionaron aspectos motivacionales y descriptivos, así como los elementos más importantes de cada uno, incluyendo ejemplos. Para finalizar esta etapa, se presentó la relación entre los Patrones de Procesos y los lenguajes en cuestión, la cual se encuentra en el Anexo 1 – Estado del Arte, en la sección 1.5.

Para profundizar en la etapa de Ejecución del ciclo de vida, se presentó un estudio sobre los motores de Workflow, haciendo especial énfasis en los relacionados con BPEL.

3. Selección de Criterios y Motores BPEL para Evaluación

3.1. Introducción

Es importante basarse en criterios claros que tomen en cuenta tanto las características de la organización (debido a que no todas las organizaciones tienen los mismos requerimientos, como por ejemplo, cantidad de usuarios, presupuesto, sector del mercado), así como también las características propias del motor como ser performance, seguridad y escalabilidad, entre otras.

En el marco de este proyecto se cuenta con una organización de gran porte, como es un hospital, como organización objetivo. Los criterios que se emplearan se dividen en 2 grandes categorías, los criterios de corte y los de evaluación. Los criterios de corte refieren a las características imprescindibles que deben cumplir los motores a ser seleccionados. Una vez seleccionados los motores se emplearán los criterios de evaluación y sus métricas para poder comparar dichos motores.

A continuación se muestra cómo se organiza este capítulo. En la sección 3.2, se presentan los Criterios para la evaluación de los motores de procesos. En primer lugar en la sección 3.2.1 se presentan criterios existentes (y trabajos relacionados) sobre Motores de Procesos, luego en 3.2.2, se describen los Criterios de Corte que se emplearan para seleccionar los motores a evaluar, los Criterios de Evaluación con las métricas que se definirán para comparar los motores seleccionados y los procedimientos mediantes los cuales se evaluaran los criterios. En la sección 3.3 se tendrán los motores relevados y seleccionados para la evaluación y finalmente en la sección 3.4 se presentan conclusiones sobre el trabajo realizado en esta etapa del proyecto.

3.2. Selección de Criterios para Evaluación

3.2.1. Antecedentes

Seleccionar un motor de procesos adecuado requiere comprender los requerimientos del negocio, así como también las diferentes alternativas que ofrece el mercado. En este sentido existen diversos trabajos a nivel académico y comercial sobre evaluación de motores basados en diferentes criterios, y algunos de ellos están enfocados en áreas de negocio específicas como comercio electrónico e investigación científica.

Algunos trabajos utilizan como criterio de evaluación el soporte para los patrones de Workflow, donde el objetivo es determinar para cada patrón si el motor provee soporte directo (mediante alguna construcción), soporte limitado (mediante alguna construcción pero que impone una restricción sobre el proceso) o simplemente no provee soporte (sin introducir una modificación en el motor). Se pueden mencionar tres trabajos principales al respecto:

- En (Wohed, y otros, 2007) se realizó una evaluación basada en todos los patrones (control de flujo, recursos y datos) de 3 BPMS código abierto jBPM (jBPM, 2011), OpenWFE (OpenWFE, 2011) y Eyhndra Shark (Shark, 2011).
- Un trabajo similar, también realizado por un equipo encabezado por Wil M. P. van der Aalst es el que se encuentra en (Aalst, 2009). El mismo consiste en realizar una evaluación de Windows Workflow Foundation basada en los patrones de Workflow de control de flujo.
- Por su parte (Wohed, y otros, 2003) realizaron una evaluación con similares características a los indicados anteriormente pero enfocado a determinar el soporte que provee el lenguaje BPEL4WS a los diferentes patrones.

Otro aspecto que ha sido tomado en cuenta para la evaluación de motores es la implementación del modelo de referencia para Workflows de la WfMC (Hollingsworth, 1995).

- En (Garcês, y otros, 2008) se evalúan 10 motores código abierto basados en 22 criterios los cuales son divididos en dos grandes categorías: conformidad con el modelo de referencia antes mencionado y aspectos funcionales. Para el cumplimiento del modelo de referencia se estudia en cada caso cuáles interfaces del modelo son implementadas, mientras que para los aspectos funcionales se encuentran criterios tales como tiempo de instalación, documentación, integración con DBMS y soporte de transacciones entre otros.

Si bien pueden existir criterios que sean independientes de la organización como pueden ser facilidad de instalación, documentación adecuada, soporte, etc., existen requerimientos específicos de cada área de negocio que deben ser tenidos en cuenta. Algunos trabajos enfocan su investigación en un área en particular como es el caso de (Tsalgatidou, 2007) donde podemos encontrar un estudio de algunos criterios para la selección de Workflows y herramientas de modelado para comercio electrónico. Estos criterios se dividen en cuatro aspectos: Interfaz de Usuario, Modelado, Análisis y Validación, y por último aspectos Técnicos. Algunos de estos criterios son aplicables tanto a la selección del modelador como a la elección del WfMS.

- Ejemplos de estos criterios podemos encontrar en (Tsalgatidou, 2007):

Criterio	**Descripción**
Adaptabilidad y portabilidad de la interfaz de usuario	Los usuarios podrían acceder a las interfaces desde diferentes tipos de dispositivos como PC's de escritorio o PDA's
Análisis y validación	Tanto sean estáticos como dinámicos deben permitir determinar algunas métricas del proceso como costos de utilización de recursos, basándose en algoritmos y técnicas de simulación de procesos
Interoperabilidad vertical	Interoperabilidad del motor con diferentes modeladores
Interoperabilidad horizontal	Interoperabilidad del motor con otros motores y otros sistemas que formen parte del proceso
Monitoreo de procesos	Permitir en cualquier momento ver el estado de un proceso en ejecución
Asignación de recursos automática	El motor debería asignar tareas en función de la carga de trabajo de otros recursos como personas o grupos de

	personas
Reportes estadísticos	Permitan estudiar y sacar conclusiones del desempeño de un proceso
Procesamiento paralelo	En un Proceso de Negocio se ven involucrados muchos usuarios y en ocasiones el trabajo de estos puede ser paralelizado
Manejo de Deadlines	Es imprescindible poder fijar fechas límites tanto para la finalización de tareas como de actividades para un empleado en particular
Integración	Herramientas de comunicación como agentes de correo electrónico

Tabla 1 - Criterios según (Tsalgatidou, 2007)

- Por su parte (Eswaran, y otros, 2006) presentan su trabajo sobre Workflows aplicados a la investigación científica y la posibilidad de adaptar algunos motores comerciales como Biztalk Server, Windows Workflow Foundation y Oracle BPEL Process Manager. En este trabajo podemos encontrar requerimientos tales como:

Criterio	**Descripción**
Manipulación de datos	Los motores empleados en investigación científica deben manejar un gran volumen de datos que deben ser pasados de una tarea o proceso a otro de forma eficiente y confiable
Seguridad	Se deben acreditar a los usuarios los permisos necesarios para poder ejecutar un proceso
Interfaz de usuario	Debe proveer una interface de usuario amigable con el motor
Escalabilidad	Debe ser posible escalar la cantidad de Wokflows ejecutándose en el motor así como también la cantidad de tareas dentro de un Workflow
Heterogeneidad e Interoperabilidad	El motor debe proveer las mismas funcionalidades para diferentes plataformas y obviamente se espera que estas sean interoperables
Tracking	Es necesario contar con la capacidad de logging y monitoreo para permitirle al usuario determinar de qué forma se obtuvo un resultado particular durante cualquier fase de la ejecución del Workflow
Tolerancia a fallas	Debe incluir técnicas como check-pointing, rollback y selección dinámica de servicios dentro de un cierto tipo de servicios en caso de que uno similar llegue a fallar
Persistencia	Es deseable que el motor persista el estado de ejecución del Workflow para prevenir pérdidas por fallas

Tabla 2 - Criterios según (Eswaran, y otros, 2006)

- Por otro lado en (Quon, 2008) se puede encontrar una evaluación de motores para procesamiento de imágenes por resonancia magnética funcional (fMRI). Esta evaluación se basó en la definición de criterios como:

Criterio	**Descripción**
Computación distribuida	Debido al gran número de imágenes que pueden llegar a ser procesadas y analizadas
Buscar y agregar nuevos paquetes	Para facilitar el procesamiento de datos para

	fMRI
Registro	Capacidad para registrar los pasos mediante los cuales se llegó a un conjunto de datos que permitan luego reproducir los resultados obtenidos
Integridad de la información	Similar a criterios antes expuestos en anteriores trabajos
Interoperabilidad	
Tolerancia a fallas	

Tabla 3 - Criterios según (Quon, 2008)

Respecto al ámbito comercial se puede encontrar compañías como TEC-Technology Evaluation (www.technologyevaluation.com) dedicadas a la consultoría sobre distintos tipos de soluciones IT en diversas áreas de negocio. La propuesta abarca criterios diversos agrupados por categorías como:

- Características de la organización
  - Sector de la industria (Salud, Banking, Finanzas, etc)
  - Cantidad de empleados
  - Si es multinacional o nacional
  - Ingresos anuales de la organización
  - Presupuesto destinado a el proyecto
- Características del proveedor
  - Servicios requeridos (mantenimiento, capacitación, soporte, etc.)
  - Idiomas soportados (inglés, español, etc.)
  - Localización del vendedor
- Características del producto
  - Plataforma de los servidores (Unix, Windows, IBM, etc.)
  - Plataforma DBMS (Oracle, MySQL, SQL Server, Informix, etc.)
  - Orientación a tarea u objetivos
  - Los workflows pueden tener múltiples estados concurrentes.
  - Una tarea de un workflow puede ser activada automáticamente al completarse las tareas anteriores.
  - Una tarea del workflow puede ser activada por triggers de tiempo.
  - Las tareas pueden activarse incluso si las anteriores no han terminado.
  - Se pueden integrar aplicaciones de terceros o incluso motores externos.
  - Se distingue entre días laborales, fines de semana, vacaciones y feriados
  - Se pueden importar workflows modelados con herramientas externas.

3.2.2. Criterios, Escalas y Procedimientos para la Evaluación

Esta sección presenta los criterios definidos, tanto para seleccionar los motores, así como también, los que se emplearon para realizar las evaluaciones. Para estos últimos, también se presentan las escalas que se definieron, las cuales se emplearon para categorizar el grado cumplimiento de cada uno. Completando la sección, se presentaran los procedimientos mediante los cuales, se evaluaran los criterios.

3.2.2.1. Criterios y sus escalas

En primera instancia, se presentan los criterios que se emplearán para seleccionar los motores de proceso de entre todos los relevados. Los llamaremos **Criterios de Corte**.

Dichos criterios tendrán como objetivo limitar los motores a ser seleccionados y obtener información suficiente para decidir qué motores son de mayor interés para ser evaluados y cuáles no, ellos son:

- **Licencia**. Los motores a evaluar deberán ser conseguidos bajo licencias de código abierto tales como GPL o LGPL, aunque se incluyen algunos motores comerciales. Se dará prioridad a aquellos que sean código abierto.
- **Versión estable**. Se trabajará con versiones estables de los motores para evitar problemas debido a la inestabilidad o falta de características de las versiones beta.
- **Documentación**. Será necesario contar con documentación actualizada en lo referente a instalación, configuración y guías de usuario. La misma debe presentarse en español y/o inglés.
- **Desarrollo sostenido y activo**. Será de mayor interés evaluar motores que hayan tenido un desarrollo constante, sobre aquellos que sean obsoletos.
- **Versión del estándar de BPEL soportado**. Un soporte a versiones más recientes será de mayor interés que las demás.
- **Lenguaje de programación que utiliza**. Es de interés que soporte la mayor cantidad de lenguajes de programación posible.
- **Soporte de comunidad de desarrolladores y usuarios**. El soporte es de especial interés para tener conocimiento sobre errores previamente detectados, así como también poder realizar preguntas.

Una vez seleccionados los motores con los Criterios de Corte, se definen los **Criterios de Evaluación**, con el objetivo de determinar las mejores características de cada motor y también realizar comparaciones entre ellos. Estos criterios fueron definidos en base a los distintos trabajos relevados y que fueron presentados en la sección 3.**Error! Reference source not found.**.1. También se presentan las escalas que se utilizarán con cada criterio para determinar en qué grado el motor cumple con los mismos.

Estos criterios tratan de medir características deseables de los motores de procesos a nivel de funcionalidades que posean. Se definen dos escalas para evaluar el grado de cumplimiento de cada criterio. La primera será una escala numérica de tres valores (0-No soportado, 1-Soportado parcial y 2-Soportado), mientras que la segunda escala será de dos valores (Si/No).

Cuando a un criterio se le asigne el valor de 0 – No soportado, se deberá a que el motor no provee las mínimas características deseables sobre el criterio en cuestión. En caso que se le asigne el valor 1 – Soportado parcial, esto implica que el motor posee la mínima característica deseable sobre el criterio, o la misma se puede mejorar mediante algún cambio o sistema externo que trabaje en conjunto con el motor en cuestión. Finalmente, cuando a un criterio se le asigne el valor 2 – Soportado, implica que el criterio cumple con las características más deseables y esperadas para el motor.

Los criterios de Si/No, se definen debido a las características de las funcionalidades que se desean evaluar en esta instancia, ya que únicamente es de interés saber si poseen o no dichas funcionalidades.

Teniendo presente como organización objetivo una de gran porte, como ser un hospital, se definieron criterios en base a temáticas relacionadas con la misma, para la evaluación. El enfoque fue primordialmente sobre características deseables en cuanto a la administración de procesos (deploy, control de ejecución, fallas, entre otras), seguridad (manejo de usuarios y permisos) y obtención de información (reportes, indicadores, entre otros). Debido a que la interoperabilidad con herramientas de diseño de procesos, y la integración con herramientas de logs de ejecución eran 2 objetivos del proyecto, los mismos tendrán criterios específicos para evaluarlos.

Para el caso de la interoperabilidad con herramientas de diseño, se utilizará (eClarus, 2011) para realizar las pruebas, mientras que para la integración con logs de ejecución, la herramienta a utilizarse será (ProM, 2011).

En la Tabla 4 se presentan los criterios con los cuales se empleará la escala de tres valores, mientras que en la Tabla 5 se presentan los criterios a los que se les aplicará la escala Si/No.

Criterio	0-No soportado	1-Soportado parcial	2-Soportado
Facilidad de instalación	Instalación completamente manual con archivos de configuración (por ejemplo xml), sin ningún tipo de asistencia automática	Instalación parcialmente asistida (por ejemplo uso de Ant), con cambios en archivos de configuración	Instalación completamente asistida mediante wizard
Asistencia en el deploy	Deploy completamente manual	Deploy asistido por la incorporación de plug-ins o herramientas de terceros	Deploy completamente asistido
Ejecución de múltiples versiones de un proceso	No pueden coexistir dos versiones de un proceso. Se deben terminar de ejecutar todas las instancias de la versión anterior para hacer deploy de la nueva versión	Se puede hacer deploy de una nueva versión del proceso, pero el motor espera a que terminen de ejecutarse las instancias de la versión vieja para	Se permite más de una versión de un proceso ejecutándose. Cuando finalizan las anteriores, las nuevas instancias ejecutan la nueva

		comenzar con la nueva versión	versión
Tolerancia a fallas	No se provee ningún mecanismo automático para recuperarse ante fallas y tampoco se puede integrar ningún sistema que lo haga	Se debe implementar la tolerancia a fallas o se permite integrar con algún sistema que lo haga	El motor provee un sistema integrado que permite recuperarse frente a fallas
Suspender o hacer rollback en un proceso	No se permite la suspensión ni tampoco el rollback en un proceso	Se permite la suspensión o el rollback de un proceso pero no ambos	Se permite tanto la suspensión como el rollback de un proceso
Cantidad de procesos ejecutándose	Se permite hasta 100 procesos ejecutándose simultáneamente	Se permite hasta 500 procesos ejecutándose simultáneamente	Se permite más de 500 procesos ejecutándose simultáneamente
Cantidad de instancias por proceso ejecutándose	Se permite hasta 10 instancias ejecutándose simultáneamente	Se permite hasta 50 instancias ejecutándose simultáneamente	Se permite más de 50 instancias ejecutándose simultáneamente
Soporte para grandes cantidades de usuarios	Se permite hasta 100 usuarios (caso de una organización de pequeño porte)	Se permite hasta 500 usuarios (caso de una organización de mediano porte)	Se permite más de 500 usuarios (caso de una organización de gran porte)
Auditoría	No posee un sistema de auditoría y tampoco permite integrarse con uno	No posee un sistema de auditoría pero permite la integración con uno o su implementación	Posee un sistema de auditoría integrado
Reportes	No permite la generación de reportes para monitoreo del motor	Permite la generación de reportes pero no su exportación en distintos formatos	Permite la generación de reportes con toda la información pertinente y su exportación a distintos formatos.
Integración de distintas bases de datos	No permite la integración con más de una base de datos, el motor trae la propia	Permite la integración con un grupo acotado de bases de datos	Permite la integración con cualquier base de datos
Obtención de indicadores de ejecución como cantidad de procesos finalizados por unidad de tiempo,	No es posible obtener indicadores.	Se pueden obtener indicadores de ejecución mediante la incorporación de herramientas de terceros (ProM,	El motor permite la obtención de indicadores de ejecución y además se integra sin problemas con

etc.		2011) o modificación.	(ProM, 2011)
Interoperabilidad con herramienta de modelado (eClarus, 2011)	No se puede lograr el deploy ni la ejecución de un proceso diseñado en la herramienta eClarus	Mediante la modificación de los archivos generados por el diseñador eClarus, se puede lograr el deploy/ejecución del proceso	Tanto el deploy como la ejecución, se logran de forma directa, para procesos diseñados en la herramienta eClarus

Tabla 4 - Criterios de evaluación con escala de tres valores

Criterio	No	Si
Importación archivos BPEL	No es posible la importación de archivos BPEL exportados desde otros motores	El motor permite la importación de archivos BPEL exportados desde otros motores
Exportación de archivos BPEL	No es posible exportar a archivos BPEL procesos definidos en el motor	El motor provee la capacidad de exportar los procesos a archivos BPEL
Importación desde herramientas de modelado	No es posible importar procesos definidos en diferentes modeladores	Se pueden importar procesos definidos en más de un modelador
Usuarios	El motor no dispone de un sistema para manejo de usuarios	El motor dispone de un sistema para manejo de usuarios
Perfiles	El motor no dispone de un sistema para manejo de perfiles de usuarios	El motor dispone de un sistema para manejo de perfiles de usuarios
Roles	El motor no dispone de un sistema para manejo de roles	El motor dispone de un sistema para manejo de roles
Simulación de procesos	No es posible realizar simulaciones de los procesos.	Es posible simular la ejecución de los procesos sin necesidad de realizar el deploy de los mismos en el motor

Tabla 5 - Criterios de evaluación con escala Si/No

Además de los criterios funcionales antes presentados, se definieron **Criterios Técnicos.** Estos tienen la intención de informar características técnicas de cada motor de procesos y el ambiente en el cual fueron ejecutadas las evaluaciones. En este caso no se definen métricas para evaluarlos sino que los resultados serán de carácter informativo.

- Requerimientos
  - Hardware: Procesador, memoria, espacio en disco.
  - Software: Sistema operativo, paquetes pre instalados como versión de Java, .Net, entre otros.

3.2.2.2. Procedimientos de evaluación

En esta sección se presentan los procedimientos, mediante los cuales se llevaran a cabo las evaluaciones de los criterios, cuyos resultados se presentan en el Capitulo 4, Evaluación de motores BPEL seleccionados. A continuación, en la Tabla 6, se presentan los procedimientos antes mencionados.

Criterio	Procedimiento
Facilidad de instalación	Se procede a instalar el motor según se indique en la documentación, documentando aspectos relevantes del mecanismo, errores y dificultades presentadas.
Asistencia en el deploy	Se procede a realizar el deploy del proceso utilizado para las evaluaciones mediante el mecanismo que cada motor disponga. Se documentan aspectos relevantes en cuanto a facilidad y amigabilidad, así como también errores que se encuentren.
Ejecución de múltiples versiones de un proceso	Se procede a realizar el deploy de un mismo proceso 2 o más veces, y ejecutarlos, para luego documentar el comportamiento del motor. Con esta información se procederá a asignarle alguna escala definida anteriormente.
Tolerancia a fallas	Se procede a realizar el deploy y ejecutar un proceso con un error implementado intencionalmente, en este caso un loop infinito y el lanzamiento de una excepción. De esta forma se documentará el comportamiento del motor, para luego asignarle alguna escala.
Suspender o hacer rollback en un proceso	Se procederá a ejecutar estas funcionalidades en un proceso (en caso que el motor posea alguna de estas funcionalidades). Luego se documentará el comportamiento del motor y se le asigna una escala adecuada.
Cantidad de procesos ejecutándose	Se procederá a realizar el deploy de la cantidad de procesos que indica el criterio en la sección anterior, mediante ayuda de scripts, para luego realizar su ejecución. Luego se documenta el comportamiento del motor y se le asigna una escala adecuada.
Cantidad de instancias por proceso ejecutándose	Se procederá a ejecutar tantas instancias de un proceso según indique el criterio en la sección anterior, para poder observar el comportamiento del motor y documentarlo. Luego se le asigna una escala adecuada.
Soporte para grandes cantidades de usuarios	Mediante la documentación, se deberá determinar si el motor posee un mecanismo para soporte de usuarios (LDAP por ejemplo). Luego se documentan los resultados y se le asigna una escala adecuada.
Auditoría	Se procederá a ejecutar un proceso y luego determinar qué información de auditoría se puede obtener con cada motor, ya sea mediante una funcionalidad del mismo o mediante su almacenado en un log. Se documenta el comportamiento y se le asigna una escala.
Reportes	Se procederá a ejecutar la funcionalidad de generar reportes (en caso que el motor la posea) sobre el estado del motor, procesos e instancias. Se valora el poder exportar la información a distintos

	formatos y la calidad de la misma. Se documenta el comportamiento de la funcionalidad y se le asigna una escala.
Integración de distintas bases de datos	Mediante la documentación, se deberá determinar el mecanismo con el cual utiliza bases de datos cada motor. Dependiendo del mecanismo se determinará si se puede integrar con cualquier base de datos y si alguna es más recomendable que otra. Este mecanismo se documenta y se le asigna una escala al criterio.
Obtención de indicadores de ejecución como cantidad de procesos finalizados por unidad de tiempo, etc.	Se procederá a investigar qué integración se puede lograr con la herramienta Process Mining (ProM, 2011) a través de los logs de los distintos motores. Es de importancia obtener de ellos la información de fecha de inicio y fin para cada actividad del proceso, así como también duración del mismo. Para que dicha información pueda ser interpretada por la herramienta (ProM, 2011), la misma debe ser exportada desde los logs en un archivo XML de extensión MXML. Esto se logra mediante la herramienta (ProMImportFramework, 2011), la cual exporta dicha información (que se encuentra en una base de datos) al formato MXML. También es de especial interés que el propio motor provea un mecanismo similar para obtener indicadores. Este comportamiento se documentará y se le asigna una escala adecuada.
Interoperabilidad con herramienta de modelado (eClarus, 2011)	Para evaluar la interoperabilidad, se utilizara la herramienta de modelado (eClarus, 2011). Esta prueba consiste en diseñar un proceso en BPMN y exportarlo a BPEL, para que luego se pueda realizar el deploy del mismo en los motores, para poder evaluar el nivel de integración que se puede lograr con dicha herramienta. Este comportamiento se documentará y se le asigna una escala adecuada.
Importación archivos BPEL	Se procederá a importar procesos de otros motores (archivos BPEL y WSDL) en cada motor, para evaluar la interoperabilidad entre ellos. Este comportamiento se documentará y se le asigna una escala adecuada.
Exportación de archivos BPEL	Se procederá a determinar si cada motor provee algún mecanismo para exportar sus procesos. Este comportamiento se documenta y se le asigna una escala.
Importación desde herramientas de modelado	Se procederá a importar procesos provenientes de otros motores, desde las herramientas de modelado de cada motor (los que las posean). De esta forma se puede observar que tan fácil puede ser editar un proceso de otro motor. Este comportamiento se documentará y se le asigna una escala adecuada.
Usuarios	Se procederá a evaluar si el motor posee algún mecanismo para la administración de usuarios. Esto se documentará y se le asigna una escala adecuada.
Perfiles	Se procederá a evaluar si el motor posee algún mecanismo para la administración de perfiles de usuarios. Esto se documentará y se le asigna una escala adecuada.
Roles	Se procederá a evaluar si el motor posee algún mecanismo para la administración de roles. Esto se documentará y se le asigna una escala adecuada.
Simulación de procesos	Se procederá a evaluar si el motor posee algún mecanismo para la simulación de procesos. Esto se documentará y se le asigna una escala adecuada.

Tabla 6 - Tabla con los procedimientos de evaluación de cada criterio

3.3. Motores Relevados y Seleccionados

3.3.1. Motores relevados

La tarea de realizar el relevamiento de los motores de proceso fue llevada a cabo mediante una investigación de las implementaciones del estándar BPEL existentes. Para dicha investigación, fueron utilizadas como fuentes de información los sitios web de OASIS (estándar de BPEL) e implementaciones sobre Java de dichas herramientas. Los sitios más importantes de los cuales se obtuvieron las herramientas fueron los siguientes:

http://bpel.xml.org/products

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel-implement

http://java-source.net/open-source/workflow-engines

Una vez finalizada la investigación, el resultado obtenido de dicho relevamiento fue el que se muestra a continuación, en la Tabla 7:

Motor	Licencia	Versión	Documentación	Versión BPEL	Lenguaje	Soporte	Sitio Web
Active BPEL	GPL	2.1	Completa. Presenta Manuales para desarrolladores, Guías de instalación y ejemplos.	WS-BPEL 1.1 (WSDL 1.0)	Java	Foros Blogs	http://www.activevos.com/community-open-source.php
Apache ODE	GPL	1.3.4	Completa. Presenta Manuales para desarrolladores, Guías de instalación y ejemplos.	WS-BPEL 2.0	Java	Lista de mails	http://ode.apache.org/index.html
jBPM BPEL	LGPL	3.0	Completa. Presenta Manuales para	WS-BPEL 2.0	Java	Foros Wikis	http://www.jboss.org/jbpm.html

			desarrolladores con ejemplos. La guia de instalacion se encuentra general para jBPM.			Blogs	
Orchestra	LGPL	4.4.0	Completa. Presenta Manuales para desarrolladores con ejemplos y una guia de instalacion.	WS-BPEL 2.0	Java	Foros Lista de mails	http://orchestra.ow2.org/xwiki/bin/view/Main/WebHome
petals	LGPL	3.0	Completa. Presenta Manuales para desarrolladores(con ejemplos) y guias de instalacion.	WS-BPEL 2.0	Java	Foros Lista de mails Blogs	http://petals.ow2.org/index.html
openESB BPEL SE	CDDL	2.2	Completa. Presenta Manuales para desarrolladores (con ejemplos y tutoriales) y Guias de instalacion.	WS-BPEL 2.0	Java	Foros Lista de mails	https://open-esb.dev.java.net/BPELSE.html
Intalio BPMS (basado en Apache ODE)	LGPL (tambien presenta versión Comercial mas completa)	6.0.3	Completa. Presenta Manuales para desarrolladores (con ejemplos y tutoriales) y Guias de instalacion.	WS-BPEL 2.0	Java	Foros Wikis	http://www.intalio.com/bpms
Riftsaw	LGPL	2.2.1	Completa. Presenta Manuales para desarrolladores	WS-BPEL 2.0	Java	Foros Blogs	http://www.jboss.org/riftsaw

			(con ejemplos y tutoriales) y Guias de instalacion.				
Twister	LGPL	0.3	Escasa	No especifica	Java	No presenta	http://swik.net/Agila-BPEL/Blog%3A+Twister+%28twister%29/Twister+Open+Source+WS-BPEL+engine+v0.3+released/mw3
bpel-g	GPL	5.1	Escasa. Presenta Manual de instalacion y Guia rapida para desarrolladores.	WS-BPEL 2.0	Java	Wikis Lista de mails	http://code.google.com/p/bpel-g/
Agila BPEL	GPL	No especifica	Escasa. Presenta Manual de instalacion, Guia de Usuario y Manual para desarroladores	No especificado	Java	No presenta	http://wiki.apache.org/agila/FrontPage
bexee BPEL (Proyecto abandonado)	GPL	0.1	Escasa. Presenta Manual de instalacion, Guia de Usuario y Manual para desarroladores	WS-BPEL 1.1	Java	No presenta	http://wiki.apache.org/agila/FrontPage
Dalma (última versión proveniente del año 2006)	GPL	0.5	Completa. Presenta Guia de instalacion y Manual para desarrolladores.	No especificado	Java C#	Foros	https://dalma.dev.java.net/nonav/maven/index.html
Appian BPEL	Comercial	6.0	Completa (Requiere registro). Presenta Manuales	WS-BPEL 2.0	Java .NET	Foros Lista de mails	http://www.appian.com/bpm-software.jsp

			para desarrolladores, Ejemplos de uso y Tutoriales.				
Microsoft BizTalk Server	Comercial	Año 2009	Completa (MSDN). Presenta Manuales para desarrolladores, Tutoriales y Ejemplos.	WS-BPEL 2.0	.NET	Foros Blogs Telefónico	http://www.microsoft.com/biztalk/en/us/default.aspx
Oracle BPEL Process Manager	Comercial	11g	Completa. Presenta Manuales para desarrolladores, Tutorales y Ejemplos.	WS-BPEL 1.1 (extensiones)	Java	Foros Blogs Telefónico	http://www.oracle.com/technetwork/middleware/bpel/overview/index.html
IBM WebSphere Process Server	Comercial	6.0	Completa. Presenta Manuales para desarrolladores, Guias de instalacion y Tutoriales (con Ejemplos)	WS-BPEL 2.0	Java	Foros Telefónico	http://www-01.ibm.com/software/integration/wps/
IBM Lombardi BPM (basado en WebSphere)	Comercial	7.1	Completa. Presenta Manuales para desarrolladores, Guias de instalacion y Tutoriales (con Ejemplos)	WS-BPEL 2.0	Java	Foros Telefónico	http://www.lombardisoftware.com/enterprise-bpm-software.php
Virtuoso Universal	Comercial	6.x	Completa. Manuales para	WS-BPEL 1.1	Java	Foros Blogs	http://virtuoso.openlinksw.com/

Server			desarrolladores, Tutoriales y Guias de instalacion.			Wikis	
SEEBURGER Business Integration	Comercial	No especifica	Disponible mediante registro.	No especifica	Java	No especifica	http://www.seeburger.com/bpm-workflow/
Parasoft BPEL Maestro	Comercial	5.0.1	Disponible mediante registro.	WS-BPEL 2.0	Java	Foros Lista de mails	http://www.parasoft.com/jsp/products/bpel.jsp?itemId=114
Pegasystems SmartBPM	Comercial	6.x	Disponible mediante registro.	No especifica	Java	Foros	http://www.pega.com/Products/business-process-management.asp
SAP NetWeaver BPM	Comercial	7.0	Disponible mediante registro.	WS-BPEL 2.0	Java .NET	Foros Lista de mails	http://www.sap.com/platform/netweaver/index.epx

Tabla 7 - Motores relevados

3.3.2. Motores Seleccionados

Una vez relevados los motores que se presentan en la Tabla 7, es necesario determinar cuáles serán seleccionados para ser evaluados por los criterios de la sección 3.2.3. Los motores que cumplen las condiciones anteriores son los siguientes:

- **Active BPEL**
- **Apache ODE**
- **jBPM BPEL**
- **Orchestra**
- **petals**
- **openESB BPEL SE**
- **Intalio BPMS**
- **Riftsaw**
- **Twister**
- **bpel-g**
- **Agila BPEL**

Los motores fueron seleccionados en base a que cuentan con mayor interés a ser evaluados, debido a sus características, y realizando la selección en conjunto con los tutores del proyecto. A continuación se muestran dichos motores:

- **Riftsaw**
- **Apache ODE**
- **jBPM BPEL**
- **Orchestra**
- **petals**
- **Intalio BPMS - Community Edition**
- **jBPM5**

Es de especial interés informar que jBPM5 no fue incluido en la Tabla 7, debido a que es un motor que implemente el estándar BPMN2, el cual fue liberado a principios del 2011, y que se considera importante evaluar sus características. Si bien algunos criterios son específicos para motores BPEL, la mayoría son de carácter genérico, con lo cual, no se presentaron demasiados inconvenientes con la integración de este motor a la lista.

3.4. Conclusiones

A lo largo de esta sección, se realizó un relevamiento de las características más importantes que serían deseables para un motor de procesos, a efectos de obtener el más adecuado y práctico a la hora del uso. Estos criterios se definieron tanto para evaluar las características de los motores (Criterios de Evaluación), como para filtrar los motores posiblemente más interesantes (Criterios de Corte) del conjunto de todos los relevados.

Para las características de interoperabilidad requeridas, se definieron criterios relacionados a la Exportación de procesos, así como también respecto a la importación de los mismos, proviniendo de otros motores, siendo lo más relevante, la integración con la herramienta de modelado de procesos de negocio (eClarus, 2011).

Los criterios de Auditoría u Obtención de Indicadores, se relacionan mas al ámbito del negocio, siendo de vital importancia también, saber que características poseen los motores al respecto y el grado de integración que se pueda lograr con la herramienta (ProM, 2011).

Para finalizar, se puede decir que se realizó un relevamiento en amplitud de los motores existentes (Tabla 7), para que luego, y a través de los Criterios de Corte, sean seleccionados para su correspondiente evaluación, además de jBPM5, que si bien no ejecuta BPEL, si lo hace con el nuevo estándar BPMN2, siendo por lo tanto, de especial interés evaluar dicho motor.

4. Evaluación de Motores BPEL seleccionados

4.1. Introducción

Luego que en el Capítulo 3 se seleccionaran los motores más relevantes, a través de los Criterios de Corte, y se indicaran los Criterios de Evaluación, sus escalas y procedimientos, en este Capítulo se procederán a presentar los resultados de su aplicación sobre dichos motores.

Si bien, todos los criterios indicados son importantes para evaluar los motores, se hará mayor énfasis en aquellos relacionados a la interoperabilidad con la herramienta de modelado (eClarus, 2011) y entre otros motores, debido a que uno de los objetivos primordiales del proyecto es estudiar que tanto se puede lograr de dicha interoperabilidad y que problemas y errores se generan al respecto. Otro de los objetivos respecto a las evaluaciones se corresponde con la interoperabilidad de logs de ejecuciones de los procesos, con la herramienta ProM. Esto conlleva a que sea se haga especial énfasis en este criterio también.

Las evaluaciones se llevaron a cabo utilizando un proceso de negocio de ejemplo, el cual se puede observar en la Figura 5, implementado en cada motor. El mismo consiste en una aprobación de un préstamo y presenta la mayoría de los elementos del estándar de WS-BPEL que fueron indicados en el Capitulo 2. Además, para realizar la evaluación de interoperabilidad con la herramienta eClarus, no será necesario utilizar otro proceso, debido a que éste consta de la condiciones necesarias para realizar el pasaje de BPMN a BPEL (y viceversa también) según indica (OASIS, 2006). Los resultados obtenidos se comparan con las métricas definidas en el Capítulo 3.

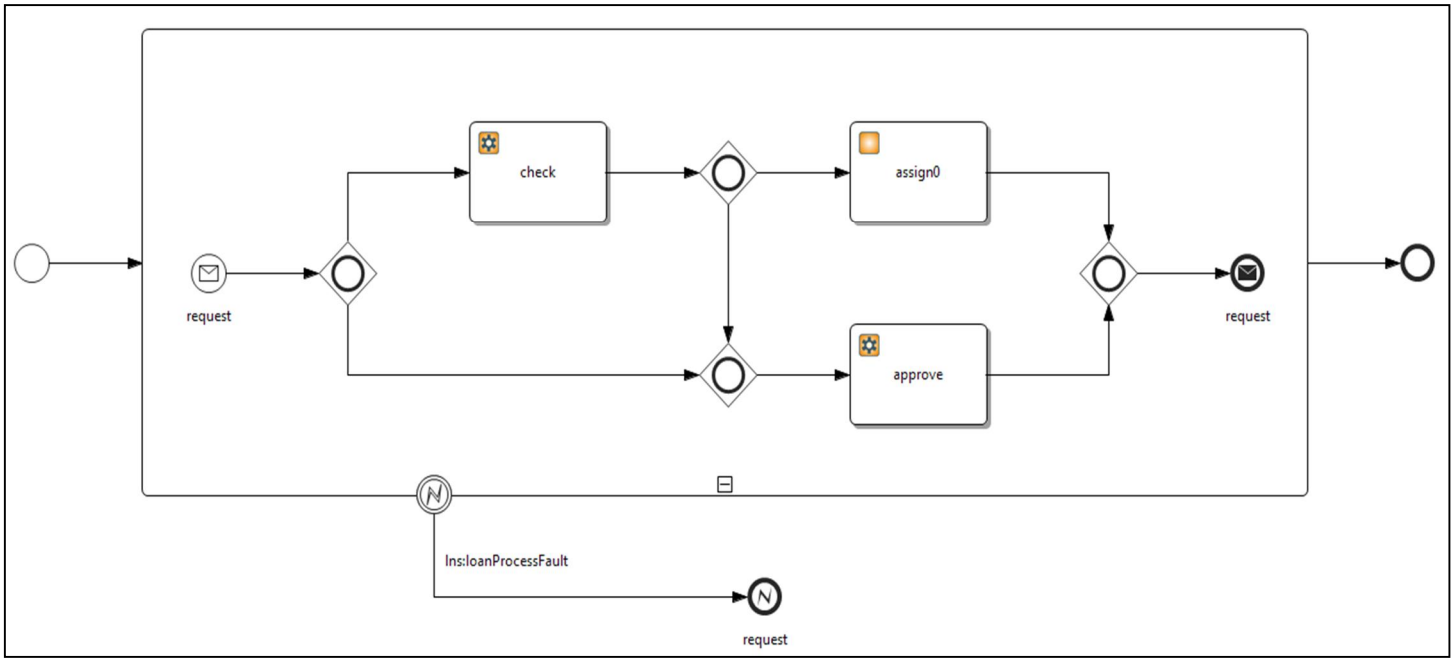

Figura 5 - Proceso utilizado para las evaluaciones

El código BPEL asociado al proceso, respetando el estándar y sin agregar construcciones particulares, se indica a continuación:

```
<process>
	<partnerLinks>
		<partnerLink name="customer"
			partnerLinkType="lns:loanPartnerLinkType"
			myRole="loanService" />
		<partnerLink name="approver"
			partnerLinkType="lns:loanApprovalLinkType"
			partnerRole="approver" />
		<partnerLink name="assessor"
			partnerLinkType="lns:riskAssessmentLinkType"
			partnerRole="assessor" />
	</partnerLinks>
	<variables>
		<variable name="request"
			messageType="lns:creditInformationMessage" />
		<variable name="risk" messageType="lns:riskAssessmentMessage" />
		<variable name="approval" messageType="lns:approvalMessage" />
		<variable name="error" messageType="lns:errorMessage" />
	</variables>
	<faultHandlers>
		<catch faultName="lns:loanProcessFault" faultVariable="error">
			<reply partnerLink="customer" portType="lns:loanServicePT"
				operation="request" variable="error"
				faultName="unableToHandleRequest" />
		</catch>
	</faultHandlers>
	<flow>
		<links>
			<link name="receive-to-assess" />
			<link name="receive-to-approval" />
			<link name="approval-to-reply" />
			<link name="assess-to-setMessage" />
			<link name="setMessage-to-reply" />
			<link name="assess-to-approval" />
		</links>
		<receive partnerLink="customer" portType="lns:loanServicePT"
			operation="request" variable="request"
			createInstance="yes">
			<source linkName="receive-to-assess"

	transitionCondition="bpws:getVariableData('request','amount')&lt; 10000"
/>
			<source linkName="receive-to-approval"

	transitionCondition="bpws:getVariableData('request','amount')>=10000" />
		</receive>
		<invoke partnerLink="assessor" portType="lns:riskAssessmentPT"
			operation="check" inputVariable="request"
outputVariable="risk">
			<target linkName="receive-to-assess" />
			<source linkName="assess-to-setMessage"

	transitionCondition="bpws:getVariableData('risk','level')='low'" />
			<source linkName="assess-to-approval"

	transitionCondition="bpws:getVariableData('risk','level')!='low'" />
		</invoke>
		<assign>
			<target linkName="assess-to-setMessage" />
			<source linkName="setMessage-to-reply" />
			<copy>
				<from expression="'yes'" />
				<to variable="approval" part="accept" />
			</copy>
```

```
            </assign>
            <invoke partnerLink="approver" portType="lns:loanApprovalPT"
                  operation="approve" inputVariable="request"
                  outputVariable="approval">
                  <target linkName="receive-to-approval" />
                  <target linkName="assess-to-approval" />
                  <source linkName="approval-to-reply" />
            </invoke>
            <reply partnerLink="customer" portType="lns:loanServicePT"
                  operation="request" variable="approval">
                  <target linkName="setMessage-to-reply" />
                  <target linkName="approval-to-reply" />
            </reply>
      </flow>
</process>
```

A continuación en esta sección se presentan los resultados de las evaluaciones, discutiendo los aspectos más relevantes. Los mismos se presentaran en 3 categorías, Evaluaciones generales (engloban aspectos más generales del motor como instalación, deploy de procesos, usuarios, entre otros), Integración con logs en ProM e Interoperabilidad con eClarus. Para obtener información detallada de la evaluación de cada motor, ver el Anexo 2 – Evaluaciones.

4.2. Evaluaciones

4.2.1. Evaluación del motor jBPM BPEL

4.2.1.1. Evaluaciones generales

jBPM BPEL presenta una forma de instalación basada en la herramienta Ant, para la cual es necesario modificar el archivo *build.xml*, haciendo que la misma no presente el automatismo que tendría un wizard. Esto lo que hace es requerir un conocimiento amplio de la herramienta Ant, además de Java y jBoss, imprescindibles para el funcionamiento de dicho motor. El deploy de los procesos es asistido mediante una interfaz gráfica amigable, en la cual solamente es necesario indicar el archivo con la descripción del proceso y los servicios que éste usa (archivos BPEL y WSDL respectivamente). La misma también se puede realizar mediante la herramienta Ant. En la Figura 6 se presenta la forma de realizar el deploy asistido en el entorno presentado por el motor.

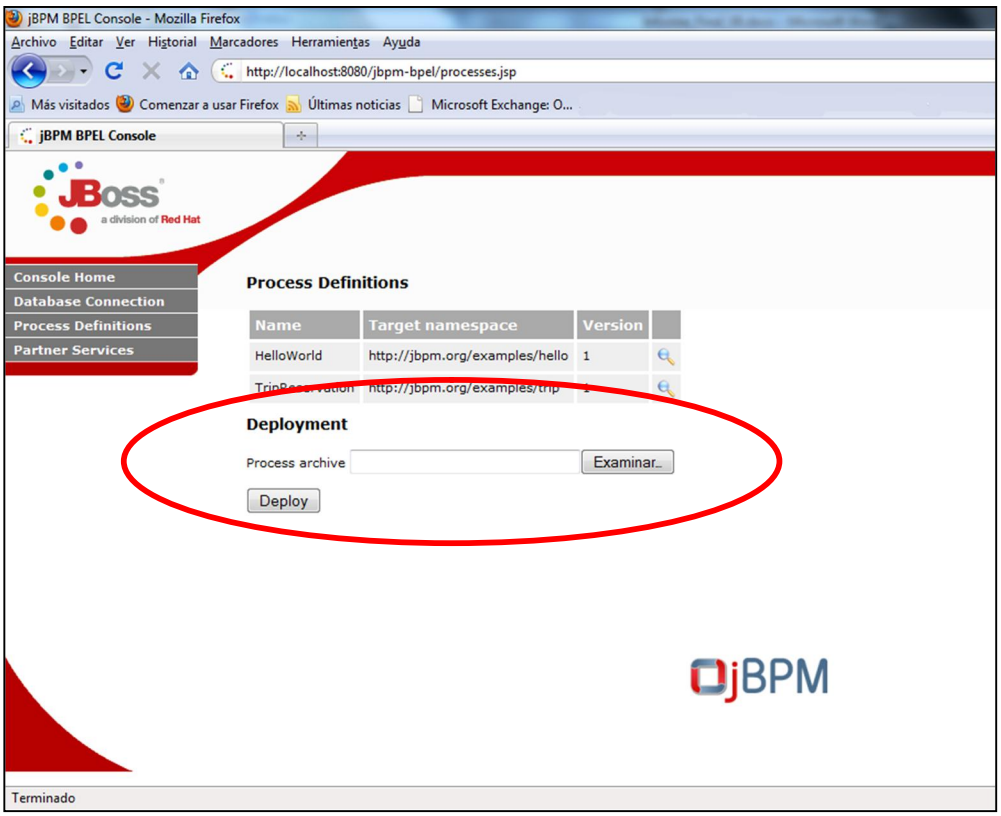

Figura 6 – Entorno de trabajo y deploy de un proceso en jBPM BPEL

Otra característica con la que se cuenta en jBPM BPEL es la ejecución de múltiples versiones de un mismo proceso, de forma simultánea. Esto se debe al control de versiones que posee el motor. Respecto a la Tolerancia a fallas, jBPM BPEL hereda de jBoss los mismos mecanismos que éste último tenga configurado, pero no posee una característica propia del motor que

implemente dicho criterio. Lo mismo ocurre con la Auditoría y la generación de reportes, son 2 características que el motor propiamente no posee.

La cantidad de procesos ejecutándose simultáneamente, así como la cantidad de instancias de un proceso ejecutándose, son 2 características que no están limitadas por el motor, sin embargo, están limitados por la configuración que posea el servidor jBoss donde se encuentre instalado. La cantidad de usuario que soporta el motor, tampoco tiene una limitante. Otra característica que es heredada de jBoss (servidor donde reside el motor), es la integración con distintas bases de datos. Esto se debe a que la misma se configura utilizando un archivo de datasource en el deploy del motor en el servidor. Esta tecnología de jBoss y Java permite que no haya limitantes en cuanto al manejador de bases de datos que se vaya a utilizar.

4.2.1.2. Integración con logs en ProM

La integración con la herramienta ProM se logra de manera sencilla, con la información de log que guarda el motor y la ayuda del framework ProM Import Framework, con el cual se realiza la traducción a MXML, formato de log que ProM reconoce.

4.2.1.3. Interoperabilidad con eClarus

Finalmente la interoperabilidad con la herramienta de modelado eClarus, no se pudo lograr debido a los errores que se generan en el motor cuando se realiza el deploy del proceso. Se encontraron diferencias en cuanto a formato de los archivos BPEL de jBPM BPEL con los generados por eClarus. Una vez detectadas esas diferencias, se realizaron cambios en los procesos para igualarlos, pero se mantuvieron los errores. Los mismos tienen una generalidad tal que no se pudieron detectar el origen de dichos errores, ni utilizando la documentación del motor, así como también en la comunidad de jBPM BPEL. Sin embargo, se logró sin problemas, la interoperabilidad entre procesos de distintos motores, en el caso de jBPM BPEL, se logró con Intalio y Orchestra.

4.2.2. Evaluación del motor Apache ODE

4.2.2.1. Evaluaciones generales

Apache ODE si bien no provee un asistente para su instalación, el mecanismo del mismo es sencillo. Basta con realizar el deploy del motor en el servidor de aplicaciones que se use, en este caso Tomcat. El deploy de los procesos sin embargo es asistido, el motor posee un cuadro de dialogo mediante el cual se indica dicho proceso. En la Figura 7 se puede observar el entorno brindado por Apache ODE.

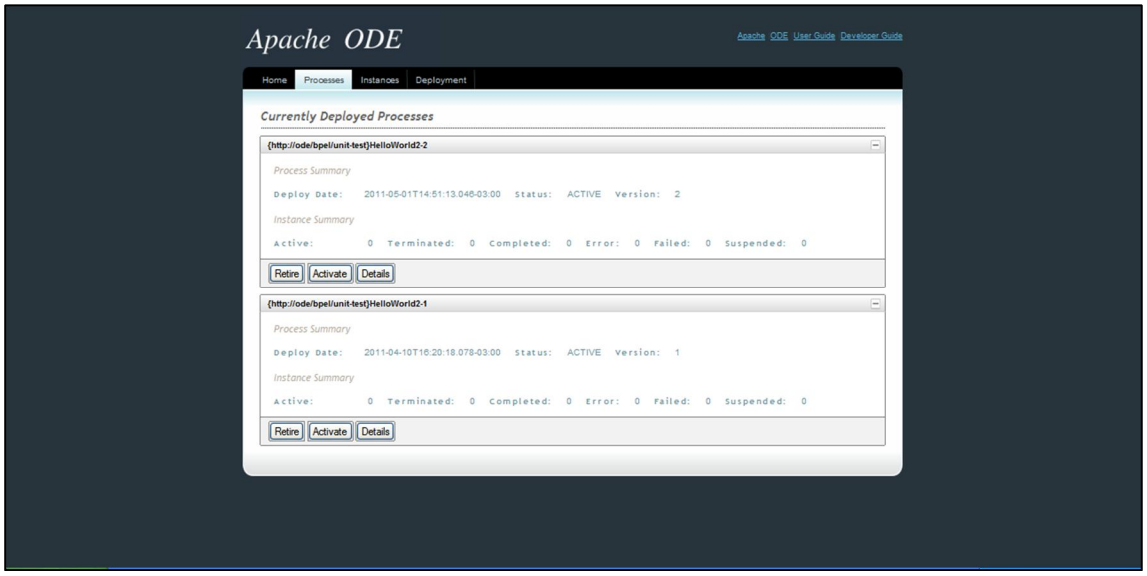

Figura 7 – Entorno de trabajo

Otra característica que posee Apache ODE es que permite la ejecución de múltiples versiones de un proceso, sin detener la ejecución de las anteriores. Al igual que en jBPM BPEL, Apache ODE no provee un mecanismo de tolerancia a fallas nativo del motor, sino que el mismo depende del servidor que se esté usando, o si se implementa a nivel del propio proceso. Tampoco se puede hacer el rollback de un proceso, aunque si se permite suspenderlo y retomarlo más tarde.

La cantidad de procesos ejecutándose simultáneamente, así como también la cantidad de instancias de un mismo proceso ejecutándose, no tienen limitantes respecto al motor, pero si pueden tener según el servidor de aplicaciones que se utilice. Una característica poco deseable, es que no presenta un mecanismo para administrar usuarios, con lo cual no existe login para acceder al administrador del motor, ni tampoco a nivel de procesos.

La información que brinda sobre auditoría es bastante escasa. Se puede saber para cada proceso las versiones y fechas de deploy pero no a nivel de instancias, ni tampoco más datos sobre ellos. Tampoco brinda un mecanismo para la generación de reportes sobre el estado del motor.

La integración con distintas bases de datos se logra mediante mecanismos provistos por el servidor de aplicaciones, realizando las configuraciones necesarias.

4.2.2.2. Integración con logs en ProM

En cuanto a la obtención de indicadores, se puede obtener información sobre cantidades de procesos e instancias de los mismos, pero no información relevante desglosándola por actividad de cada proceso. Esto hace que si bien se logre integrar con la herramienta ProM, no se brinde la información más importante.

4.2.2.3. Interoperabilidad con eClarus

Para finalizar la evaluación, se debe mencionar que la interoperabilidad con eClarus no se vio lograda, debido a errores que se generan al no reconocer la importación de los servicios importados (servicios a través de WSDL). Cabe acotar que la generalidad del error, así como también la documentación escasa al respecto, fueron factores fundamentales para no lograr dicha interoperabilidad.

4.2.3. Evaluación del motor Intalio

4.2.3.1. Evaluaciones generales

La instalación en Intalio no presenta dificultades, aunque no presenta un wizard para ello. Basta con descomprimir el motor y está listo para ser utilizado, sin la necesidad tampoco de realizar configuraciones extras. Esto hace que no sea necesario demasiado conocimiento en tecnologías como Java o Ant para poder utilizarlo. El deploy de los procesos también es sencillo, basta con indicar en un cuadro de dialogo al proceso en cuestión y se realiza el deploy. Esta acción también se puede realizar desde su diseñador. A continuación se presenta la Figura 8 en la cual se puede ver el cuadro de dialogo para el deploy de un proceso, y el entorno de trabajo de Intalio.

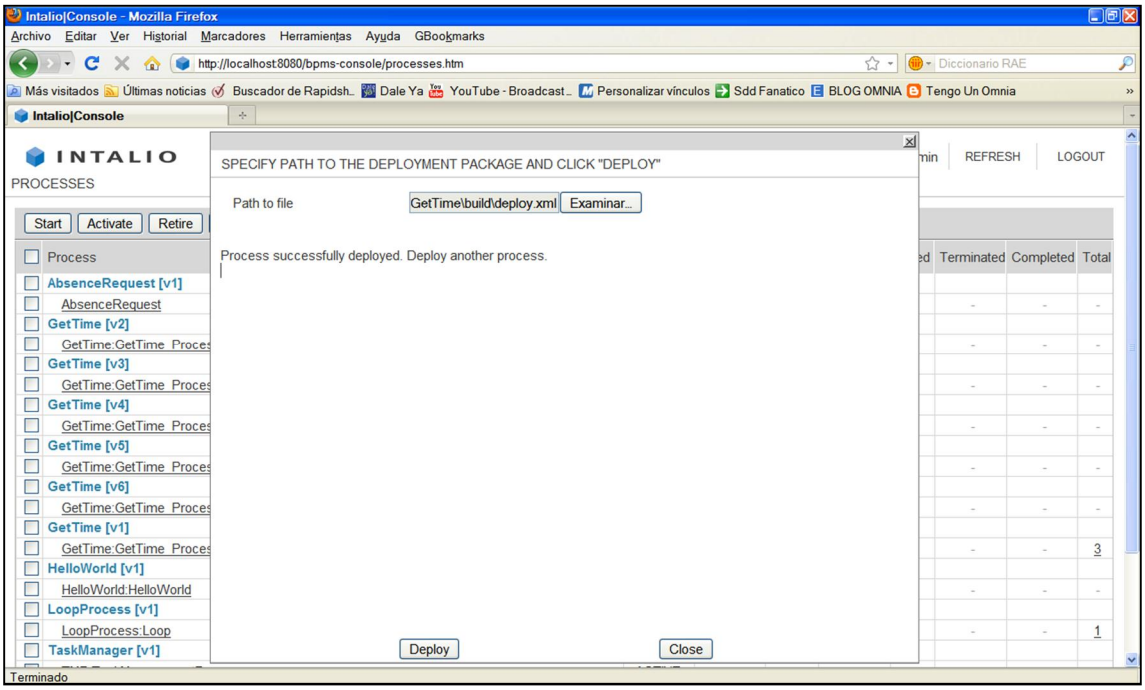

Figura 8 – Entorno de trabajo y deploy de un proceso en Intalio

Otra característica destacable de Intalio es que maneja el versionado de procesos de forma automática, permitiendo la ejecución de múltiples versiones de procesos simultáneamente.

En contrapartida, Intalio no presenta un mecanismo de tolerancia a fallas nativo, sino que el mismo debe ser implementado como una extensión del motor, o utilizando en el proceso los elementos adecuados presentados por el estándar de BPEL. Tampoco se permite de forma nativa del motor, el rollback de un proceso, aunque si la suspensión y retoma de uno. Al igual que en el criterio de tolerancia a fallas, es necesario implementar dicho mecanismo como extensión al motor.

Otro punto a destacar de Intalio es que no está limitado el número de procesos ejecutándose en el motor, así como tampoco la cantidad de instancias de un mismo proceso en ejecución.

Esto permite que sea un motor adecuado a una organización de gran porte que utilice un número importante de procesos. Tampoco está limitada la cantidad de usuarios, debido a la posibilidad de Intalio de integrarse con tecnologías específicas para el manejo de usuarios, como un servidor LDAP.

La auditoría es otra característica destacable del motor, brindando información relevante de los procesos como hora de comienzo, fin y estado (éxito o no), pudiéndose también observar el proceso en cuestión en esta misma funcionalidad. Sin embargo, Intalio no provee la funcionalidad para generar reportes sobre el estado del motor ni tampoco de los procesos, siendo esta una característica deseable.

También es destacable mencionar que permite la integración con cualquier base de datos, debido a un mecanismo propio del motor, para lo cual se necesita la librería de la base e indicar en un archivo de configuración, la información necesaria.

4.2.3.2. Integración con logs en ProM

Respecto a la obtención de indicadores, Intalio almacena información relevante sobre su estado, indicando información de procesos e instancias, desglosando por estado de las mismas. Esto permite que luego sea utilizada por la herramienta de ProM y presentar información de relevancia.

4.2.3.3. Interoperabilidad con eClarus

Con Intalio, se obtuvo un resultado similar al de Apache ODE en cuanto a la interoperabilidad con eClarus. Esto se debe a que el motor Intalio, se basa fundamentalmente en Apache ODE. Los errores que se generaron son similares entre sí, y no se pudieron solucionar por los mismos inconvenientes, aunque la documentación de Intalio sea más completa que la de Apache ODE.

4.2.4. Evaluación de motor Riftsaw

4.2.4.1. Evaluaciones generales

Riftsaw no presenta una forma amigable de instalación, sino que requiere conocimientos relacionados a la herramienta Ant, debido a que se necesita actualizarlo con información relacionada al servidor de aplicaciones donde se instalará el motor. Otro aspecto poco amigable de Riftsaw es que no presenta un cuadro de dialogo que permita el deploy de los procesos, sino que el mismo debe realizarse también a través de la tecnología Ant, lo cual implica generar un archivo *build.xml*. A continuación se presenta en la Figura 9 el entorno de trabajo que presenta el motor.

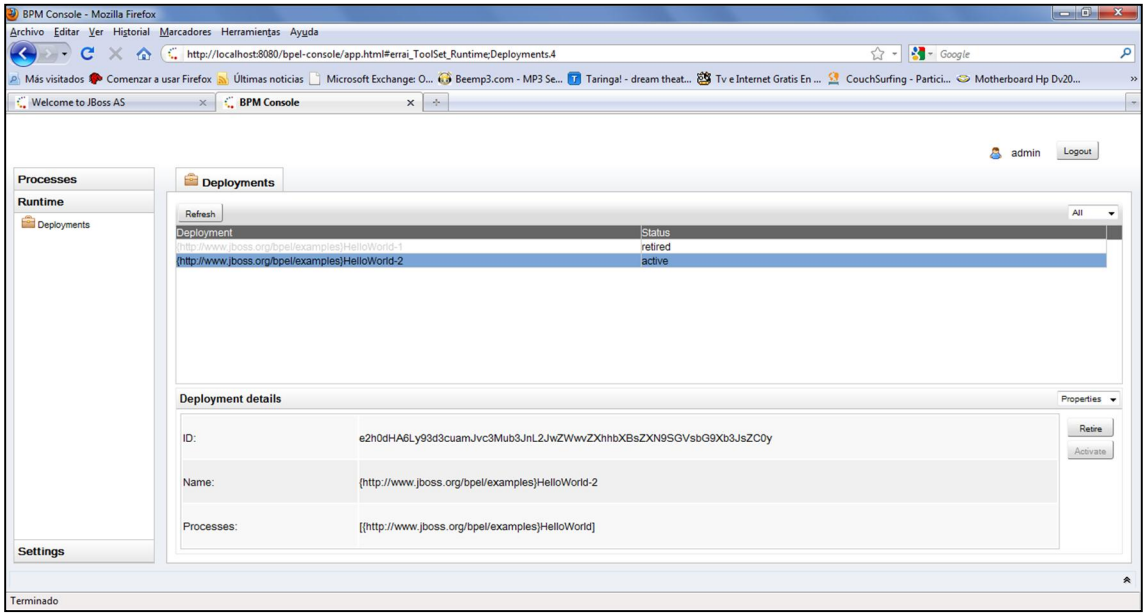

Figura 9 - Entorno de trabajo de Riftsaw

Riftsaw tampoco permite ejecutar múltiples versiones de un proceso, excepto cuando se realiza el deploy de una nueva versión, y se estaba ejecutando una anterior. De todas formas, una vez finalizada la anterior se la detiene y solamente se puede acceder a la nueva. Un aspecto positivo de este motor, es que si bien no provee un mecanismo propio de tolerancia a fallas, utiliza los mecanismos heredados por jBossESB, con lo cual se tiene a disposición buenas herramientas propias de dicho servidor para manejar fallas y errores. Otro aspecto negativo es que no presenta ningún mecanismo para realizar rollback ni suspensión de un proceso.

La cantidad de procesos, o de instancias de un proceso ejecutándose no poseen limitantes propias del motor, sino que, al igual que en los demás motores, pueden ser configurados a nivel del servidor de aplicaciones, aunque en este caso, dicha configuración se aplica para todos los servicios que se encuentren en el mismo. Tampoco es una limitante la cantidad de usuarios que accedan al motor.

Tampoco provee un mecanismo relacionado a obtener información de auditoría, y su implementación sería bastante compleja, debido a la escasa información que presenta el motor. Tampoco presenta una funcionalidad relacionada a la generación de reportes sobre el estado del motor o mismo de cada proceso o sus instancias.

Como aspecto positivo se puede mencionar la integración con distintas bases de datos. Esta configuración es bastante sencilla y se encuentra bien documentada, aunque es necesario tener conocimientos sobre tecnologías de persistencia en Java.

4.2.4.2. Integración con logs en ProM

Respecto a la obtención de indicadores, si bien el motor provee una funcionalidad con esa información, la misma no es a nivel de actividades o eventos de los procesos, sino que a nivel de instancias de procesos. Algo similar se puede mencionar de la integración con la herramienta ProM. Si bien la integración se logra sin problemas, la información que almacena Riftsaw no es tan relevante como la necesaria.

4.2.4.3. Interoperabilidad con eClarus

En cuanto a la interoperabilidad con la herramienta eClarus, no se ha logrado realizar el deploy sin errores de un proceso. Los errores que se generaron fueron solucionados a lo largo de la prueba, sin embargo, no se puede acceder al mismo.

4.2.5. Evaluación del motor Petals

4.2.5.1. Evaluaciones generales

Petals no provee un mecanismo al estilo wizard para su instalación, sin embargo, la misma se realiza de forma sencilla, descomprimiendo el motor en una carpeta del File System. Con esto se tienen las características más básicas del motor, como son deploy y ejecución de los procesos. También se descomprime un diseñador para los mismos. Sin embargo, si se necesitan otros componentes del motor, como deploy asistido por interfaz grafica, herramientas para auditorías o reportes, es necesario instalar componentes extra. Esto último no se encuentra debidamente documentado, con lo cual es necesario acceder a la comunidad para obtener información pertinente. Cabe aclarar, que la instalación de muchos componentes, requiere otros como previos. El deploy también es asistido a través de un cuadro de dialogo. El único aspecto complejo es la generación de un archivo JBI, requerido por el motor para el deploy, pero el mismo es generado automáticamente por el diseñador del motor. A continuación en la Figura 10 se muestra el entorno de trabajo del motor.

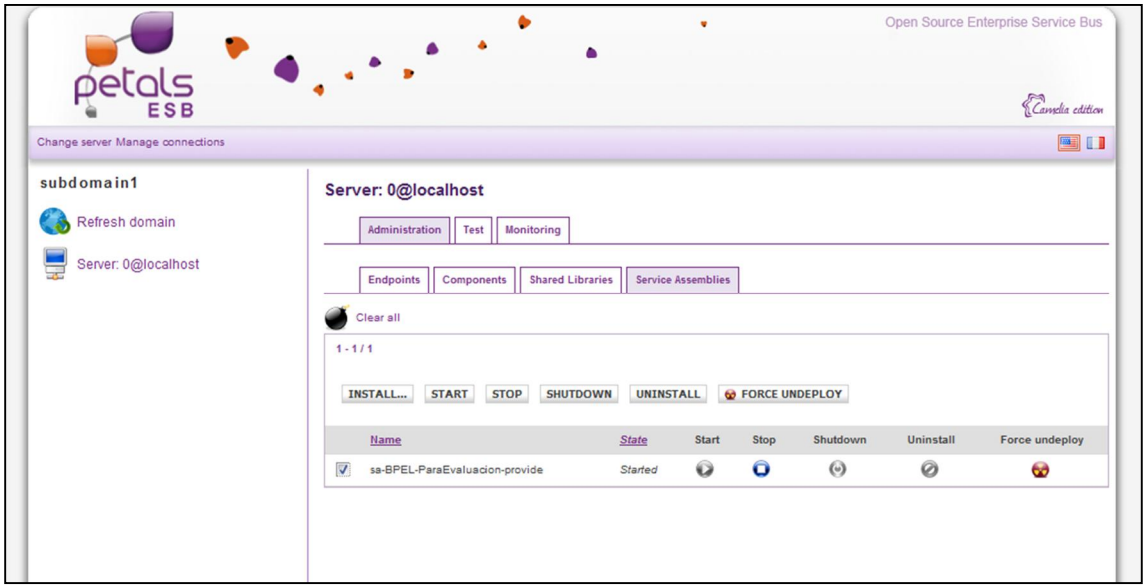

Figura 10 - Entorno de trabajo de Petals

Una desventaja de este motor, es que no posibilita la ejecución de múltiples versiones de un proceso, debido a que no presenta un control de versiones adecuado. Para solucionar este inconveniente, se puede realizar el deploy del mismo proceso utilizando el mismo nombre agregándole identificador de la versión. Otra característica que Petals no posee, es la tolerancia a fallas. Los errores que se generan debido a procesos (ya sean debido a los Web Services o errores en el diseño de los procesos), repercuten directamente en el motor de Petals, muchas veces siendo necesario su reinicio para retomar su normal funcionamiento. Tampoco se puede realizar rollback en un proceso utilizando Petals, aunque si se tiene la posibilidad de suspender uno y retomarlo en otro momento.

Petals no presenta limitante respecto a la cantidad de procesos ejecutándose simultáneamente, así como tampoco en cuanto a la cantidad de instancias de un mismo proceso. Cabe aclarar que Petals, utiliza como servidor de aplicaciones a Tomcat (se descomprime junto con el motor), por lo tanto, si se realiza alguna configuración al respecto, la misma repercutirá en el motor de Petals. Tampoco se detectaron limitantes respecto a la cantidad de usuarios del motor.

Petals no presenta una funcionalidad que presente información sobre auditoría del motor, ni tampoco la misma se almacena en logs del motor, haciendo difícil incluso, implementar tal funcionalidad. Sin embargo, es uno de los pocos motores que presenta una funcionalidad para generar reportes, en este caso exportando dicha información a formato CSV. De todas formas, la información que se exporta es solamente sobre cantidad de procesos y sus estados. La integración con distintas bases de datos no presenta ninguna dificultad para el motor.

4.2.5.2. Integración con logs en ProM

La obtención de indicadores se realiza a nivel de mensajes intercambiados en el motor, pero no exclusivamente para los procesos que se le hicieron deploy. De todas formas se presenta la cantidad de mensajes así como también la hora de comienzo y fin de esas transacciones. Si bien ProM se integra sin problemas con Petals, la información que el motor le brinda, no es de relevancia para un administrador que lo pueda usar.

4.2.5.3. Interoperabilidad con eClarus

Finalmente, la interoperabilidad con el diseñador eClarus, no se ha visto lograda, debido a la generalidad y poca información que se posee del motor con el diseñador en cuestión.

4.2.6. Evaluación del motor Orchestra

4.2.6.1. Evaluaciones generales

Orchestra no presenta una instalador al estilo wizard, sino que el proceso se realiza a través de la herramienta Ant, sin ninguna configuración en el archivo *build.xml*. El deploy de los procesos se realiza de forma sencilla y amigable, mediante un cuadro de dialogo en el que se indica el proceso y los servicios que este utiliza. En la Figura 11 se puede observar el entorno de trabajo de Orchestra.

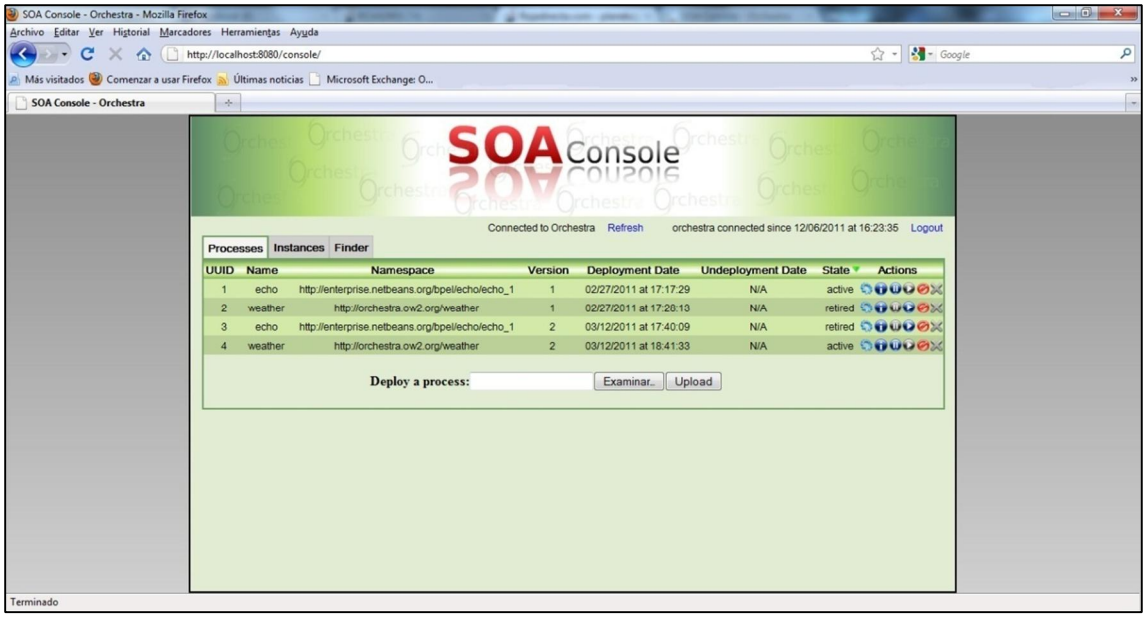

Figura 11 - Entorno de trabajo de Orchestra

Orchestra si bien permite el deploy de múltiples versiones de un proceso, no permite la ejecución simultanea de las mismas. Siempre se activa la nueva, y se desactivan las demás, en caso que existan instancias de las versiones anteriores ejecutándose, el comportamiento es desactivar la nueva. El administrador del motor debe activar la nueva manualmente luego que finalicen las demás anteriores. Otro aspecto que no maneja Orchestra es la tolerancia a fallas, cualquier error o excepción que surja, repercute directamente en el motor. Tampoco se tiene una funcionalidad para realizar rollback de un proceso, aunque si se puede suspender uno y retomarlo en otro momento.

Respecto a la cantidad de procesos ejecutándose, así como también la cantidad de instancias de un proceso ejecutándose simultáneamente, el motor no presenta ninguna limitante propia. Tampoco se ha detectado ninguna limitante respecto a la cantidad de usuarios utilizando el motor. Sin embargo, Orchestra no provee funcionalidades para el manejo de auditoría, ni tampoco respecto a la generación de reportes sobre el estado del motor ni de los procesos.

La integración con distintas bases de datos se logra mediante una configuración sencilla.

4.2.6.2. Integración con logs en ProM

En cuanto a la obtención de indicadores sobre la ejecución de procesos en el motor, el mismo no provee la información suficientemente relevante (a nivel de eventos y actividades de los procesos), sino que simplemente almacena información de instancias ejecutadas y sus estados.

4.2.6.3. Interoperabilidad con eClarus

La interoperabilidad con la herramienta eClarus, también presenta dificultades y no se ha podido lograr. Una vez realizadas todas las modificaciones necesarias en los archivos BPEL y WSDL generados por eClarus, el motor presentaba un error respecto a los roles del proceso, el cual no pudo ser subsanado debido a la poca información y generalidad del error.

4.2.7. Evaluación del motor jBPM5

4.2.7.1. Evaluaciones generales

La instalación de jBPM5 se realiza de forma sencilla, se descomprime el instalador y luego utilizando la herramienta Ant, se descargan los componentes necesarios para el motor, servidor, diseñador y componentes que proveen otras funcionalidades. El deploy de un proceso también se realiza de forma sencilla, con la intervención del designer del motor y su consola web. El mecanismo de deploy es completamente asistido, y no es necesaria la intervención de ningún script o herramienta Ant para realizarlo. Cabe aclarar también que esta funcionalidad se encuentra documentada de forma adecuada.

A continuación se muestra en la Figura 12 el entorno de trabajo del motor, en su consola web.

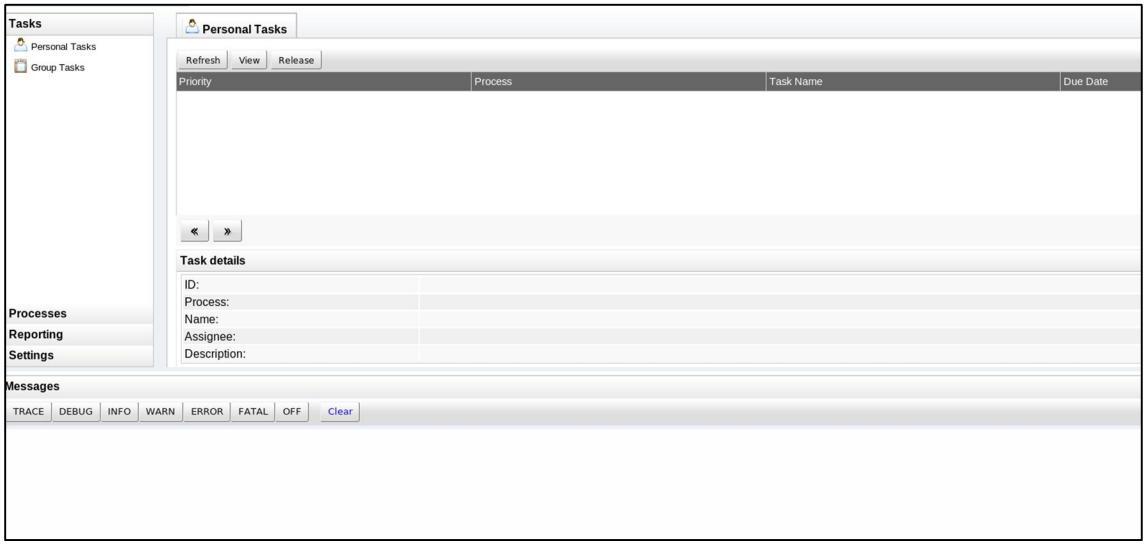

Figura 12 - Entorno de trabajo de jBPM5

jBPM5 si bien permite el deploy de múltiples versiones de un proceso, la ejecución siempre es de la versión más actual, excepto que manualmente se indique una versión anterior. Otro aspecto a tener en cuenta, es que no presenta un mecanismo nativo sobre la tolerancia a fallas en el motor. Tampoco en jBPM5 se tiene una funcionalidad que permita realizar el rollback de un proceso o suspenderlo. Si bien se puede llegar a implementar a nivel de proceso, modificándolos incluyendo timers, lo cual hace poco factible debido a la complejidad que requiere.

Al igual que los demás motores, jBPM5, no presenta limitantes respecto a la cantidad de procesos e instancias de un proceso ejecutándose simultáneamente. Tampoco presenta limitantes en cuanto a la cantidad de usuarios del motor. Como se ha aclarado para otros motores, la limitante puede darse a nivel del servidor donde se esté ejecutando el motor (en este caso jBoss).

En cuanto a la auditoría, jBPM5 no provee una funcionalidad para manejar dicha información, así como tampoco se la almacena. Sin embargo, sí provee una funcionalidad para la generación de reportes sobre el estado del motor, aspecto no menor ya que pocos motores lo disponen.

Otro aspecto positivo es que debido a la utilización de la tecnología JPA de Java, permite la integración con distintas bases de datos.

4.2.7.2. Integración con logs en ProM

El motor jBPM5 no provee una funcionalidad para obtener indicadores sobre ejecución de los procesos, ni tampoco la almacena para ser utilizada por otra herramienta como ProM.

4.2.7.3. Interoperabilidad con eClarus

Finalmente la interoperabilidad con otras herramientas de diseño de procesos presenta dificultades que no se lograron subsanar. Se realizaron los cambios necesarios en los archivos BPMN generados pero la interoperabilidad no se vio lograda. Los errores que se generaron presentaban una generalidad tal que no se pudo determinar la causa.

4.3. Comparaciones y conclusiones de la evaluación

Los resultados de las evaluaciones de los motores, se brindaran a continuación en forma comparativa entre cada uno, en función de los distintos criterios empleados.

Respecto a la **Facilidad de instalación**, se pudo observar que las formas más empleadas para hacerlo son mediante un wizard, mediante ejecución de algún script utilizando Ant o realizando el deploy del motor en un servidor de aplicaciones (jBoss o Tomcat). En este aspecto, los motores que más dificultades presentaron fueron Petals y Orchestra. En el caso de Petals debido a la gran cantidad de componentes que requiere su instalación (aunque es asistida y presenta interfaz grafica para ello), mientras que Orchestra requiere modificación previa del script de instalación. Los demás motores requerían conocimientos previos de Java, pero no presentan grandes dificultades.

Respecto a la **Asistencia en el deploy**, el único motor que presenta mayores dificultades es Riftsaw, debido a que no presenta una interfaz grafica para ello, y requiere la creación de un script a ser ejecutado por Ant. En contrapartida, los otros motores necesitan nada más que el archivo BPEL y los WSDL que se usen.

La **Ejecución de múltiples versiones de un proceso**, es soportada nada más que por 3 motores, jBPM BPEL, Apache ODE e Intalio. Los motores Riftsaw, Orchestra y jBPM5, requieren que la versión anterior del proceso sea detenida y se ejecute la nueva para todas las instancias. El único motor que no soporta múltiples versiones simultáneas es Petals, debido a características propias de su funcionamiento interno.

La **Tolerancia a fallas** no es una característica común en los motores evaluados. Si bien en la mayoría se puede implementar algún mecanismo que lo soporte, el mismo es mas a nivel de servidor de aplicaciones donde se haya realizado el deploy del motor (en el marco de este trabajo seria jBoss o Tomcat). Cabe destacar que para realizar este desarrollo es necesario contar con documentación propia del motor, aspecto que Petals y Orchestra son los únicos que no poseen.

De los motores evaluados, los únicos que no presentan las características de **Suspender** ni hacer **Rollback** de un proceso son Riftsaw y jBPM5, pudiendo estos nada más que iniciarlos y detenerlos. Los demás motores, permiten la Suspensión de los procesos, pero no el Rollback.

Respecto a características de **Cantidad de procesos**, **Instancias de un proceso** o **Cantidad de usuarios** accediendo a los mismos, no se han detectado inconvenientes. La mayoría de los motores no presenta ninguna limitante propia sobre estas características, pero si queda a cargo de la configuración del servidor de aplicaciones muchas de estos aspectos.

Ninguno de los motores provee herramientas que asistan la **Auditoría** de los procesos ni el motor en sí, aunque en mayor o menor medida todos proveen información acerca de cantidad

de procesos e instancias y sus respectivos estados. Intalio en particular permite ver los eventos ejecutados en cada una de las instancias de cada proceso.

Se ha detectado también, que la generación de **Reportes** sobre el estado del motor y sus procesos, no son una característica común. Únicamente Petals y jBPM5 la poseen, con la salvedad que Petals lo hace exportando a un formato especifico (archivo CSV de Excel), con poca información relevante.

Los motores evaluados presentan todos una buena **Integración con distintas bases de datos**, en la mayoría de los casos debido a la herencia con tecnologías de persistencia propias de Java. También presentan distintas características respecto a la Obtención de indicadores. Excepto por Orchestra, los demás motores proveen algún mecanismo para obtener indicadores de la ejecución de los procesos, o almacena la información necesaria para **Integrarse con la herramienta ProM**.

Respecto a la **Interoperabilidad con herramienta eClarus**, se ha detectado que los motores generan distintos errores al realizar el deploy de dichos procesos. Esto se debe a que eClarus provee soporte para determinados motores (de carácter comercial como Oracle BPEL, SAP e IBM). Se ha realizado un estudio de los archivos BPEL de eClarus y de cada motor, y modificándolos, se ha detectado que los errores persisten, impidiendo esto el acceso al proceso. Cabe acotar, que la documentación y la participación en la Comunidad de cada motor (Foros, blogs, etc), no provee información relevante. En cuanto a la Interoperabilidad de procesos entre motores, se ha logrado mejores resultados, logrando dicha característica entre algunos motores.

En lo que refiere a seguridad algunos motores presentan características más fuertes que otros. Por un lado Apache ODE no soporta la definición de **Usuarios** por lo que no tiene un mecanismo de log-in. Otros motores como Orchestra o jBPM5 permiten la definición de usuarios (y sus respectivas contraseñas) mediante archivos de configuración. Por otro lado Petals presenta una interface GUI para el manejo de usuarios, mientras que Intalio maneja dos mecanismos para este fin. Se pueden definir usuarios mediante archivos como Orchestra pero también se lo puede conectar con un servidor LDAP.

Finalmente, se ha detectado que ningún motor provee una herramienta para la **Simulación de procesos.**

A modo de resumen general de la evaluación, se presenta a continuación la Tabla 8 con los resultados de las mismas.

Criterio	jBPM BPEL	Apache ODE	Intalio	Riftsaw	Petals	Orchestra	jBPM5
Facilidad de instalación	Soportado parcial	Soportado parcial	Soportado	Soportado parcial	Soportado	Soportado parcial	Soportado
Asistencia en el deploy	Soportado	Soportado	Soportado	No Soportado	Soportado	Soportado	Soportado
Ejecución de múltiples versiones de un	Soportado	Soportado	Soportado	Soportado parcial	No Soportado	Soportado parcial	Soportado parcial

proceso							
Tolerancia a fallas	Soportado parcial	Soportado parcial	Soportado parcial	Soportado parcial	No Soportado	No Soportado	Soportado parcial
Suspender o hacer rollback de un proceso	No soportado	No soportado	Soportado parcial	No Soportado	Soportado parcial	Soportado parcial	No Soportado
Cantidad de procesos ejecutándose	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado
Cantidad de instancias de un proceso ejecutándose	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado
Soporte para grandes cantidades de usuarios	Soportado	No Soportado	Soportado	Soportado	Soportado	Soportado	Soportado
Auditoría	Soportado parcial	No Soportado	Soportado	Soportado parcial	No Soportado	No Soportado	No Soportado
Reportes	No soportado	No soportado	No soportado	No Soportado	Soportado parcial	No Soportado	Soportado
Integración con distintas bases de datos	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado	Soportado
Obtención de indicadores de ejecución	Soportado	Soportado parcial	Soportado	Soportado parcial	Soportado parcial	No Soportado	Soportado parcial
Interoperabilidad	No Soportado	No Soportado	No soportado	No Soportado	No Soportado	No Soportado	No Soportado
Importación de archivos BPEL	Si	No	Si	Si	No	Si	N/A
Exportación de archivos BPEL	Si	No	No	No	No	No	N/A
Importación desde herramientas de modelado	Si	No	Si	No	No	Si	N/A
Usuarios	No	No	Si	No	Si	No	Si
Perfiles	No	No	No	No	No	No	No
Roles	No	No	Si	No	Si	No	Si
Simulación de procesos	No	No	No	No	No	No	No

Tabla 8 - Resumen de las evaluaciones

Teniendo en cuenta la Tabla anterior, y las escalas definidas, se presenta a continuación, en la Figura 13, una grafica de carácter comparativo entre motores. Para la concreción de la misma se sumaron los valores de las escalas de cada criterio (0 – No soportado, 1 – Soportado parcial y 2 - Soportado), para cada motor. Graficando los resultados, se puede tener una visión más general de las evaluaciones.

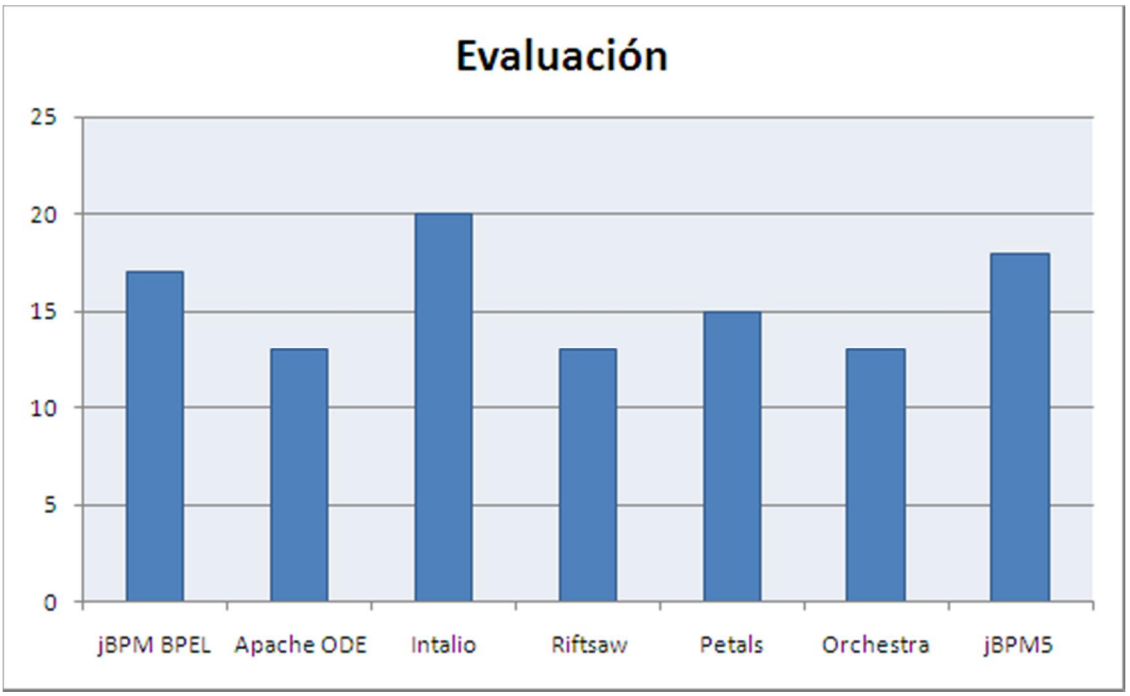

Figura 13 - Gráfica comparativa de los motores

En la misma se puede observar que Intalio, es el motor que posee más características deseables y se ha comportado de mejor forma a lo largo de la evaluación, presentando un ambiente amigable, un comportamiento estable y una comunidad activa (aunque la tendencia de esta última es enfocarse en el ambiente comercial de Intalio).

En una segunda instancia podemos mencionar motores como jBPM BPEL, jBPM5 y Petals, como otros de los más completos, aunque con algunas características menos que Intalio, como la ejecución de múltiples versiones de un proceso, o Auditoria.

Finalmente, contando con las características más básicas deseables en un motor de procesos, podemos mencionar a Apache ODE, Riftsaw y Orchestra.

5. Caso de Estudio

5.1. Introducción

Luego de realizada la evaluación de los motores seleccionados, es de especial interés realizar un caso de estudio, utilizando un motor de los evaluados, con un proceso de una organización dada. Con esto se podrá observar todos los pasos relacionados al ciclo de vida de un proceso, haciendo especial énfasis en la etapa de ejecución, la más importante para este trabajo.

Este capítulo se organiza de la siguiente manera, a continuación en la sección 5.2, se presenta una Descripción de las tareas en las que consiste este caso de estudio, luego en la sección 5.3 se presentan los detalles de la Implementación del mismo y finalmente en la sección 5.4 las Conclusiones pertinentes al mismo.

5.2. Descripción

El caso de estudio consiste en diseñar el proceso que se presenta en la Figura 14 en notación BPMN, utilizando la herramienta eClarus, luego con la misma exportarlo a BPEL y ejecutarlo en el motor seleccionado. Este proceso forma parte de un proceso más general de Cirugía Mayor Ambulatoria o CMA definido para el Hospital General de Ciudad Real, España y fue proporcionado por uno de los tutores del proyecto (Andrea Delgado). El proceso de Pre-Intervención modela el flujo de trabajo desde que el paciente llega al bloque quirúrgico hasta que es colocado en la mesa de operaciones. Durante el flujo de trabajo se realizan tareas como revisar la historia clínica del paciente, preparar los instrumentos para la cirugía, registrar signos vitales del paciente antes de la operación, etc.

El motor seleccionado para este caso de estudio es Intalio|Server, debido a las características que se pudieron comprobar a lo largo de la evaluación de los motores. Otro aspecto no menor es la exportación a BPEL. Como se ha podido comprobar en el capitulo anterior (Evaluación de motores BPEL seleccionados), los motores no logran la interoperabilidad con los procesos exportados a BPEL por la herramienta eClarus. Ello conlleva a que el proceso en cuestión, sea diseñado en la propia herramienta de diseño del motor, en este caso Intalio|Designer.

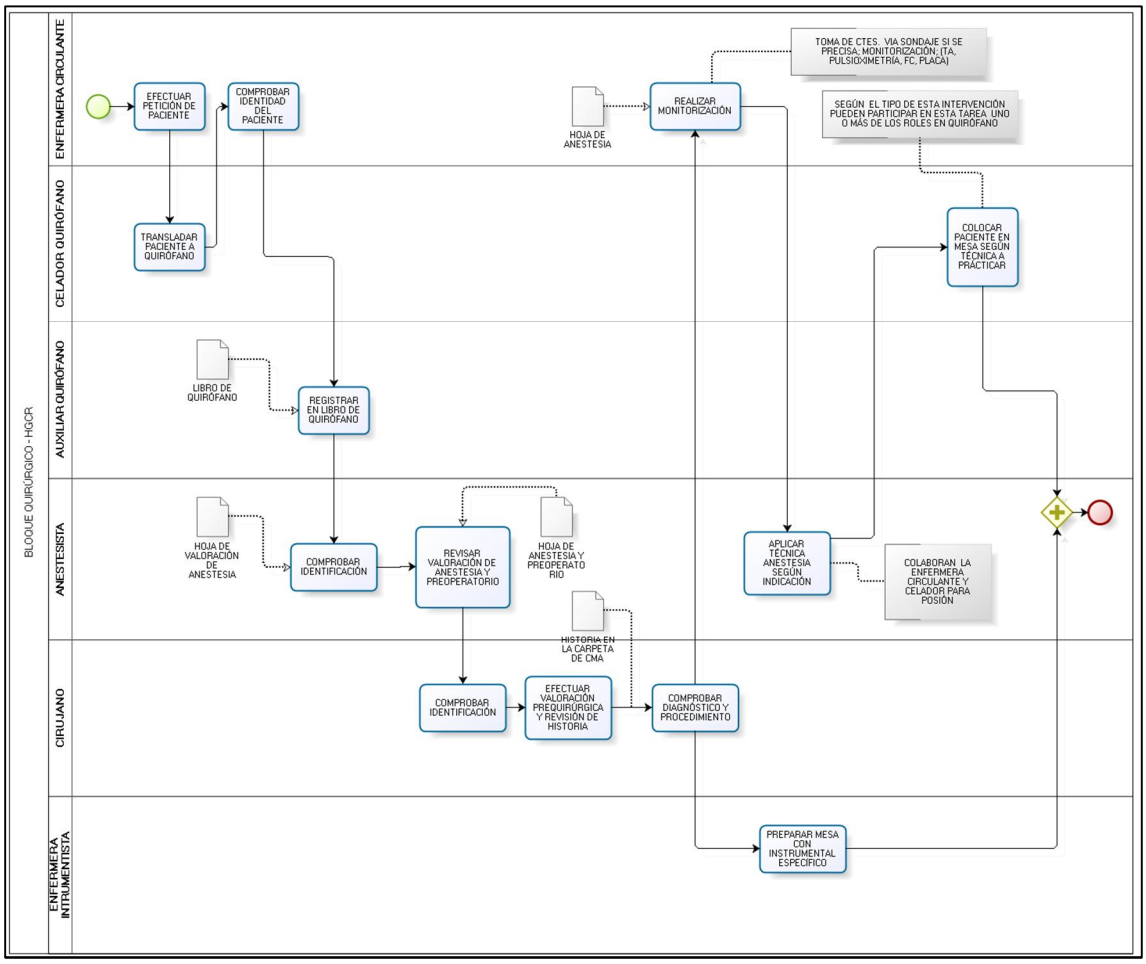

Figura 14 - Proceso utilizado para el Caso de Estudio (Pre-Intervención)

5.3. Aspectos de Implementación

Si tomáramos en cuenta la realidad completa de un proceso de Pre-Intervención de un paciente existen una serie de aspectos que deberían ser tenidos en cuenta, cómo controlar la identidad del paciente o asegurarse que el mismo tenga una cirugía programada entre otros. A los efectos de cubrir el objetivo del caso de estudio se realizaran una serie de simplificaciones sobre el mismo para facilitar su implementación. No se realizarán controles en cuanto a la identidad del paciente, programación de la cirugía ni chequeo de errores. Tampoco se implementara una historia clínica ni el sistema de registro del quirófano.

Para este proceso se crearán dos formularios que formaran parte del flujo de trabajo y sus campos deberán ser completados por los usuarios a los que se les asignará cada tarea. Para este fin se crearon también dos usuarios (jmasa y jdoe) y dos roles (anestesista y enfermera) en el servidor. Al usuario jdoe se le asoció el rol de enfermera y se le asignará la tarea de completar el formulario inicial representado en la **Error! Reference source not found.**. Al usuario jmasa se le asoció el rol de anestesista por lo que se le asignará la tarea de completar el formulario correspondiente a la valoración inicial representado en la Figura 17.

En la Figura 15 podemos observar el proceso de la Figura 14 modelado con Intalio|Designer 6.0.3.050. La primera diferencia con el modelo BPMN y el de Intalio que podemos encontrar es que en el primero el proceso está modelado con un solo pool (BLOQUE QUIRURGICO - HGCR) mientras que el segundo tiene tres pools (BLOQUE QUIRURGICO – HGCR, USER y SERVICIOS). Esto es así ya que en Intalio se deben modelar en un pool aparte las interacciones del proceso con agentes externos al mismo como pueden ser Web-Services y personas a través de formularios. Los pools que contienen elementos externos y que no forman parte del proceso propiamente son marcados como no ejecutables lo que en el diagrama se denota con un fondo gris oscuro en el área correspondiente al nombre del pool.

El proceso es iniciado por un usuario a través del formulario (Figura 16) representado por la tarea "PedirPaciente-init request" del pool USER. Esta tarea envía un mensaje hacia la tarea "EFECTUAR PETICION DE PACIENTE" la cual responde a su vez con otro mensaje hacia el formulario y luego continúa la ejecución del proceso con la siguiente tarea. La interacción entre el formulario y la tarea del proceso es bidireccional ya que esto le provee a Intalio|BPMS Workflow la capacidad de saber si la tarea fue creada con éxito o si hubo algún problema y así informarle al usuario al respecto.

Las tareas dentro de un pool pueden ser asignadas todas al mismo rol/usuario indicando esto en las propiedades del pool o puede hacerse de forma individual. En este caso dado que existe más de un lane y por consiguiente más de un rol que es desempeñado por diferentes personas se asignó el rol/usuario directamente en las tareas del pool USER.

Otra diferencia que podemos observar con el modelo original es la tarea "REVISAR VALORACION DE ANTESTESIA Y PREOPERATORIO". Debido a que se utiliza un formulario (Figura 17) para esta tarea la misma debió ser dividida en dos. Intalio requiere el uso de dos tareas para manejar formularios en tareas intermedias. La primera tarea es la encargada de crear el formulario mientras que la segunda es para recibir la información ingresada por el usuario en el formulario y volcarla en el proceso.

Si bien se simplifican algunos aspectos del proceso como lo es el libro de registro del quirófano, se desarrollará un Web-Service el cual será invocado durante la ejecución de la tarea que involucra dicho registro. Este Web-Service consta de un solo método el cual recibirá 4 parámetros de entrada (número de historia clínica, piso, habitación y cama), grabara estos valores en disco y luego retornará un string indicando el resultado de la ejecución. Los parámetros de entrada serán asignados con los valores que el usuario ingrese en formulario inicial del proceso. Para modelar la invocación a un WS se agrega un pool con el nombre "SERVICIOS" y se agrega a dicho pool el método que se desea invocar (en este caso será el método *registro*). Luego se conecta la tarea "REGISTRAR EN LIBRO DE QUIROFANO" con el método *registro*. Por último mediante la interface del diseñador se le indica al proceso que mapee los campos: Nro. Historia Clínica, Piso, Habitación y Cama, con los correspondientes parámetros del método (nroHistoria, nroPiso, nroHabitacion y nroCama).

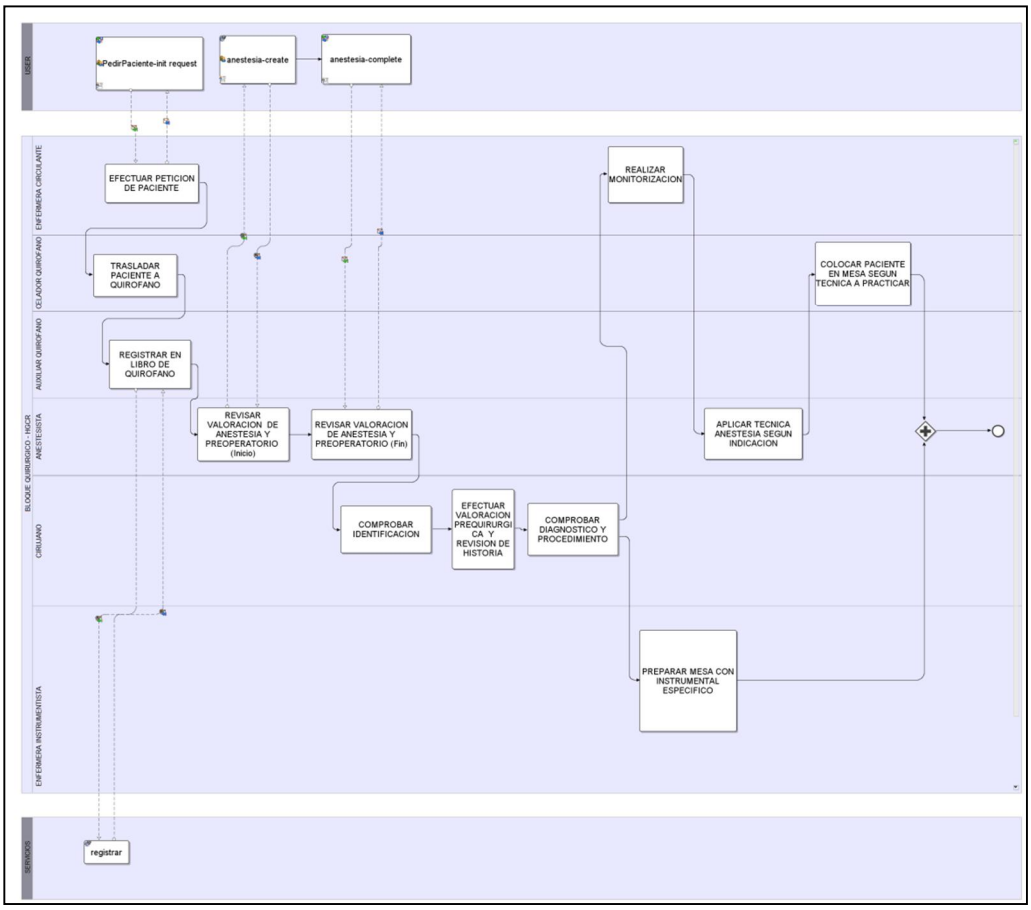

Figura 15 - Modelo del proceso de negocio de Pre-Intervención en Intalio|Designer

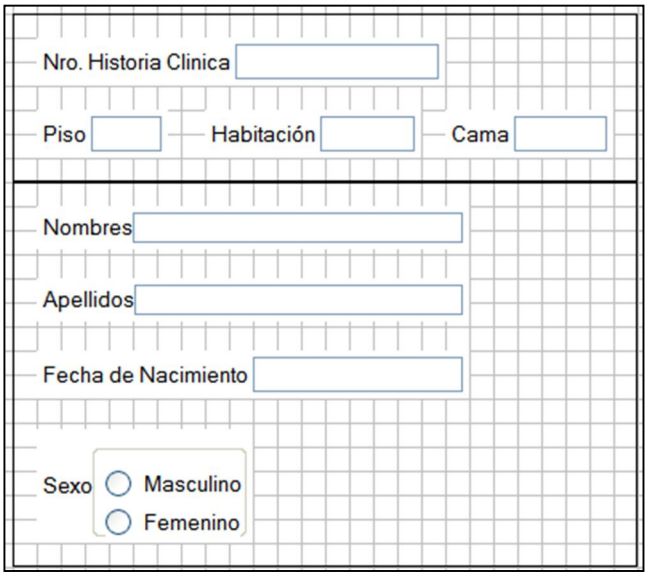

Figura 16 - Formulario Inicial

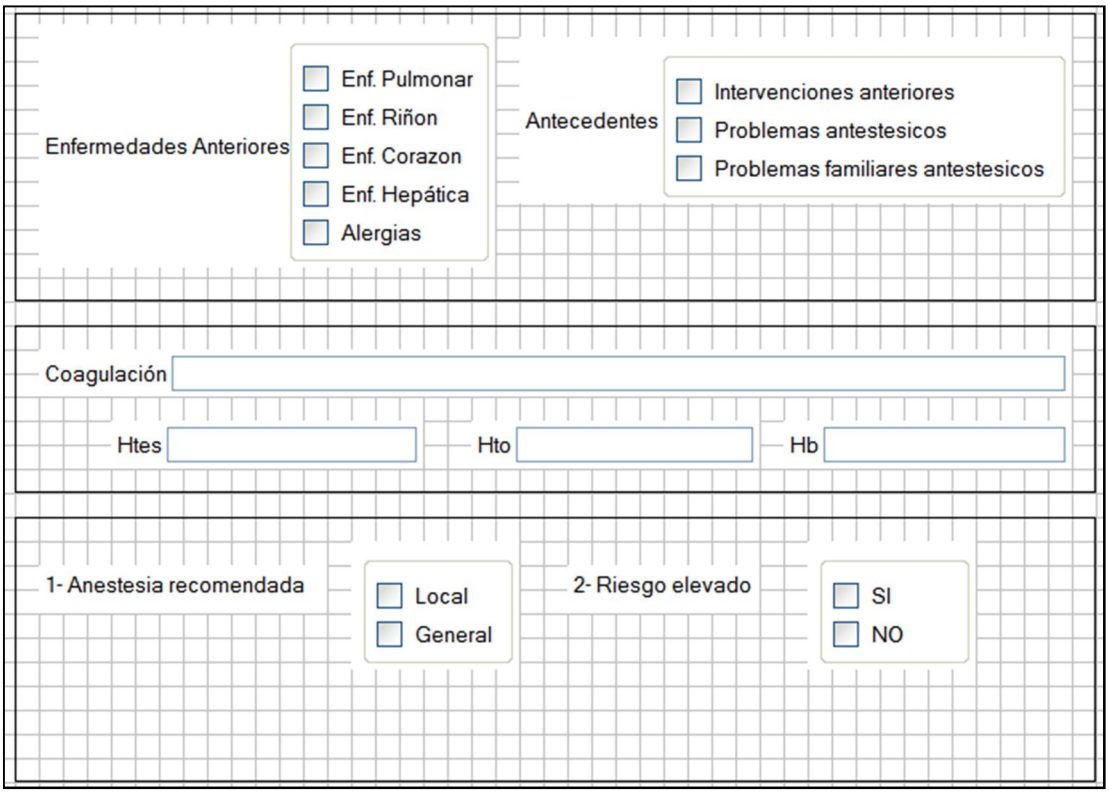

Figura 17 - Valoración preoperatoria

5.4. Realización del Caso

Para ejecutar el proceso nos logueamos en la consola web de Intalio|Workflow utilizando las credenciales del usuario jadoe. Al inicio, como podemos observar en la Figura 18, vemos que el usuario jadoe no tiene asignada ninguna tarea ya que es este usuario quien debe iniciar el proceso.

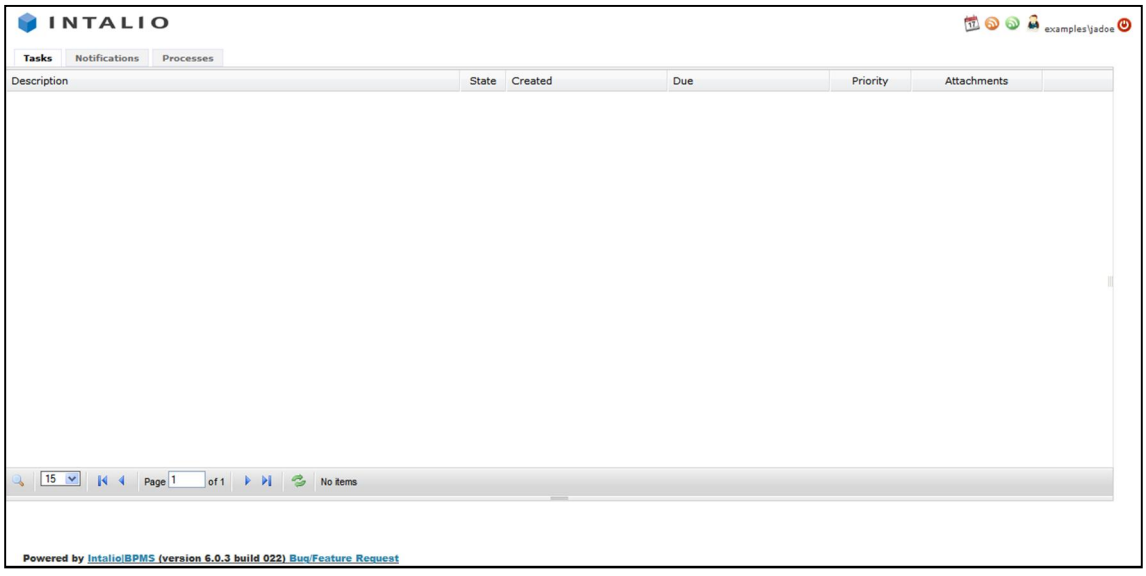

Figura 18 - Lista de tareas para el usuario

Seleccionando la pestaña “Processes” encontramos el proceso Pre-Intervención (Figura 19), haciendo click sobre él el servidor crea una instancia del formulario presentado en la Figura 16 y lo carga en pantalla.

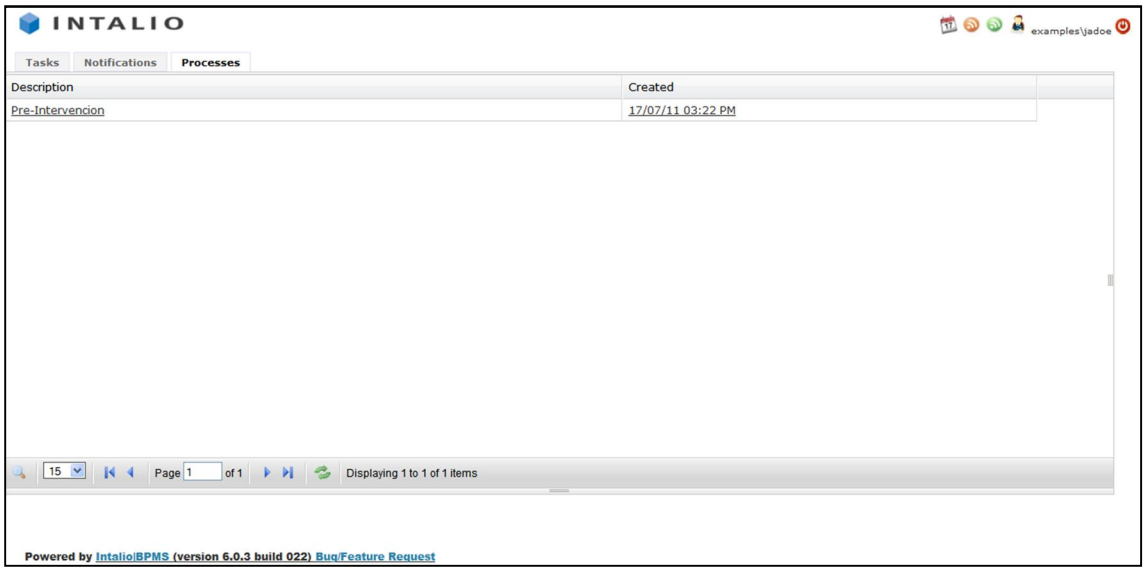

Figura 19 - Lista de procesos disponibles para el usuario

Este formulario es completado con los datos que se muestran en la Figura 20. Luego el flujo continua la ejecución mediante el botón “Start” lo que transfiere el control al servidor y al pool “BLOQUE QUIRURGICO – HGCR” a través de la tarea “EFECTUAR PETICION DE PACIENTE” dentro del lane “ENFERMERA CIRCULANTE”. Una vez que el flujo de control llegue a la tarea “REVISAR VALORACION DE ANESTESIA Y PREOPERATORIO (Inicio)” se crea una tarea y se asigna al usuario jmasa.

Figura 20 - Formulario inicial para ingresar datos del paciente

Al ingresar a Intalio|Workflow con las credenciales del usuario jmasa podemos observar, como se muestra en la Figura 21, asignada al usuario una tarea con el nombre “Realizar valoración preoperatoria” (descripción asignada a dicha tarea en Intalio|Designer).

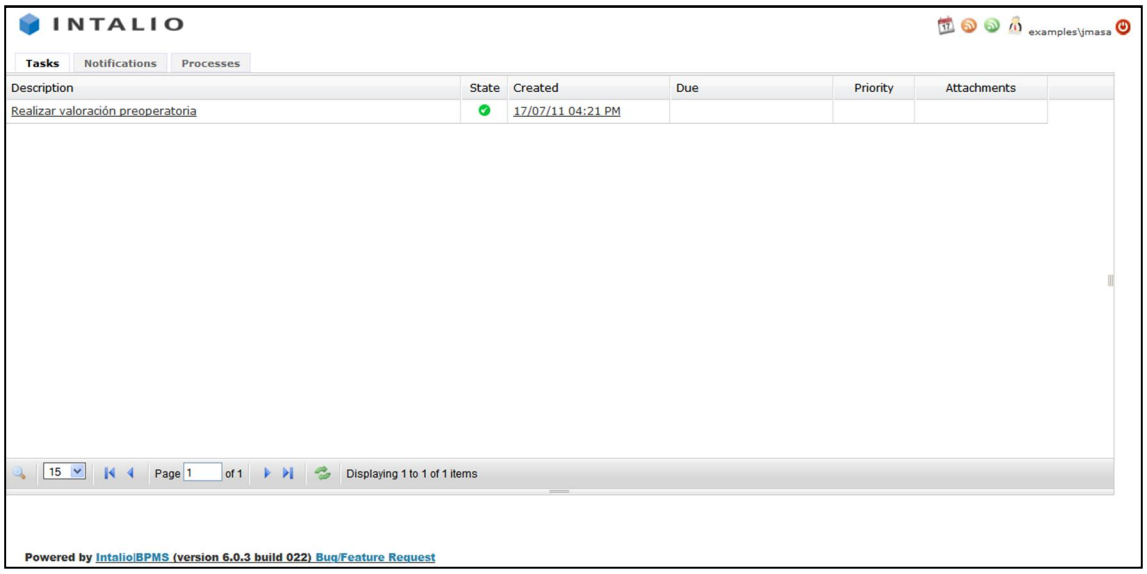

Figura 21 - Lista de las tareas disponibles

Al igual que con el formulario anterior al seleccionar la tarea se crea una instancia del formulario asignado a este usuario y se carga en la pantalla como se observa en la Figura 22. En este caso se tienen 3 opciones: reclamar la tarea para uno mediante el botón “Claim” (cuando una tarea esta asignada a más de un usuario a través de un rol mediante este botón un usuario se auto-asigna la tarea y ya no queda disponible para los demás), guardar el trabajo realizado (botón “Save”) sobre el formulario para continuar luego y terminar la tarea mediante el botón “Complete”.

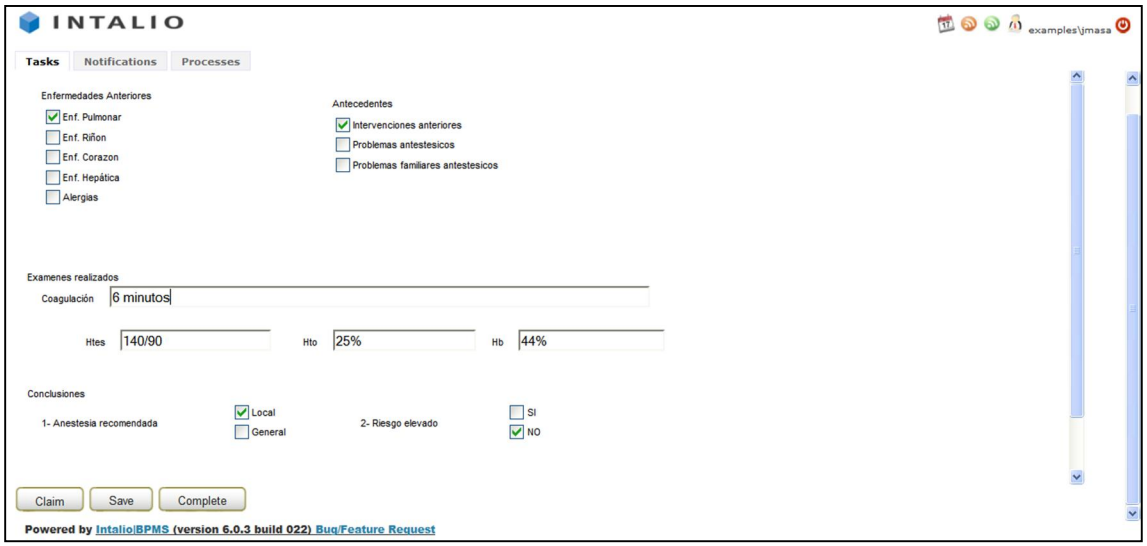

Figura 22 - Formulario de valoración preoperatoria

Antes de completar la tarea y continuar con la ejecución del proceso podemos ver en el administrador web de Intalio|Server el estado de la instancia como se muestra en la Figura 23. Como se esperaba la instancia se encuentra en progreso y se observa la lista de eventos ejecutados hasta este punto. Como podemos observar en la figura la tarea "REVISAR VALORACION DE ANESTESIA Y PREOPERATORIO (fin)" fue activada y se encuentra en ejecución. Esta tarea es la encargada de recibir la confirmación de recibir la información ingresada en el formulario y propagarla al resto del proceso.

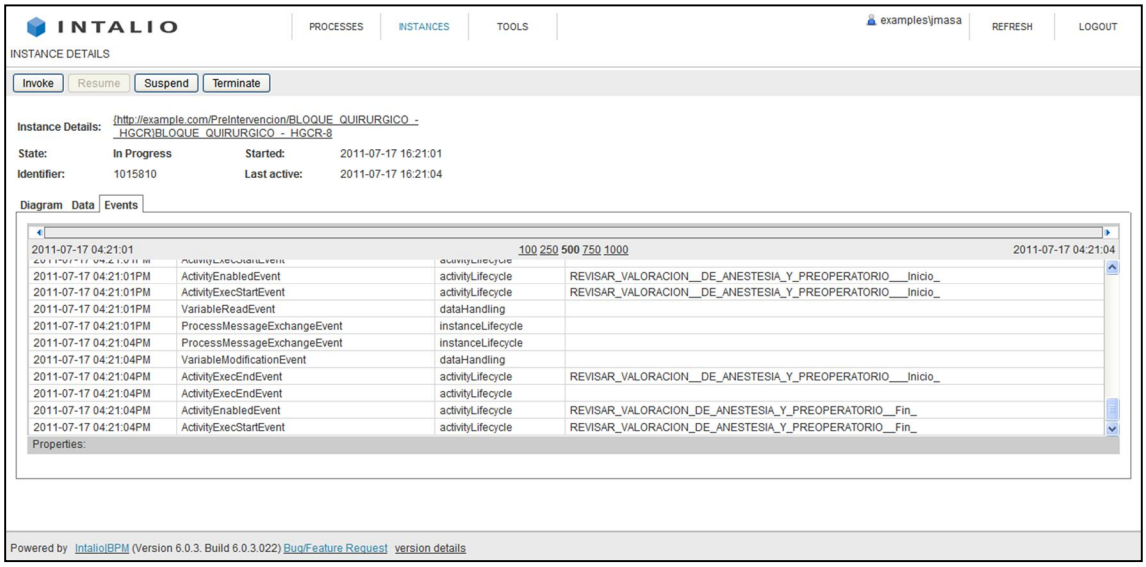

Figura 23 - Eventos ejecutados (ejecución parcial)

En este punto el web-service ya fue invocado y cómo podemos observar en la Figura 24 al consultar un evento de tipo *dataHandling* vemos el mensaje que el servidor envió al WS usando como parámetros los valores ingresados en el formulario inicial. En la Figura 25 podemos observar el mensaje de respuesta enviado por el WS.

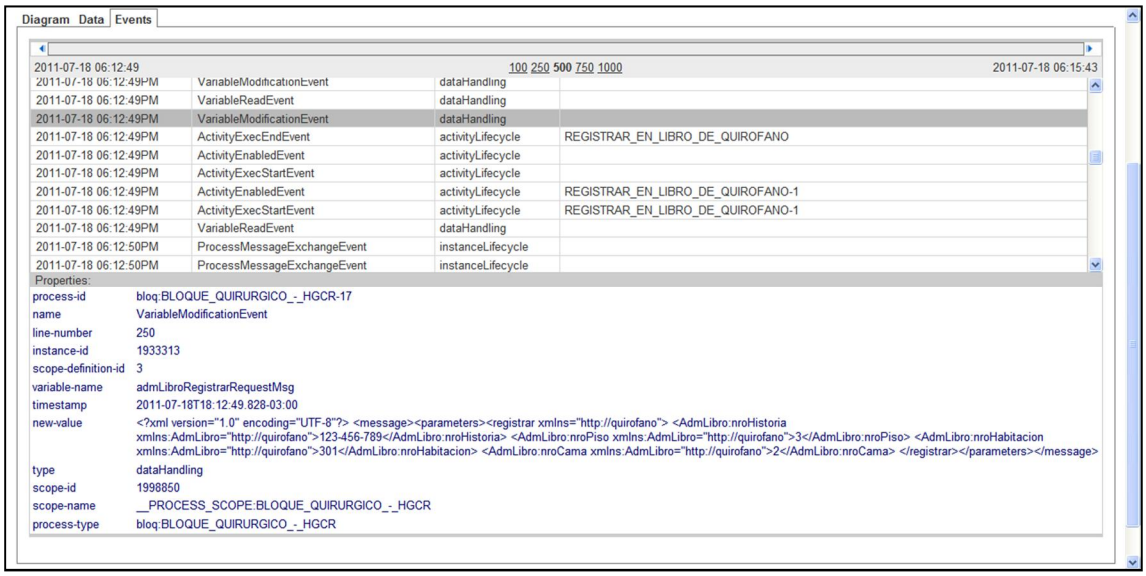

Figura 24 - Mensaje para enviar al web service

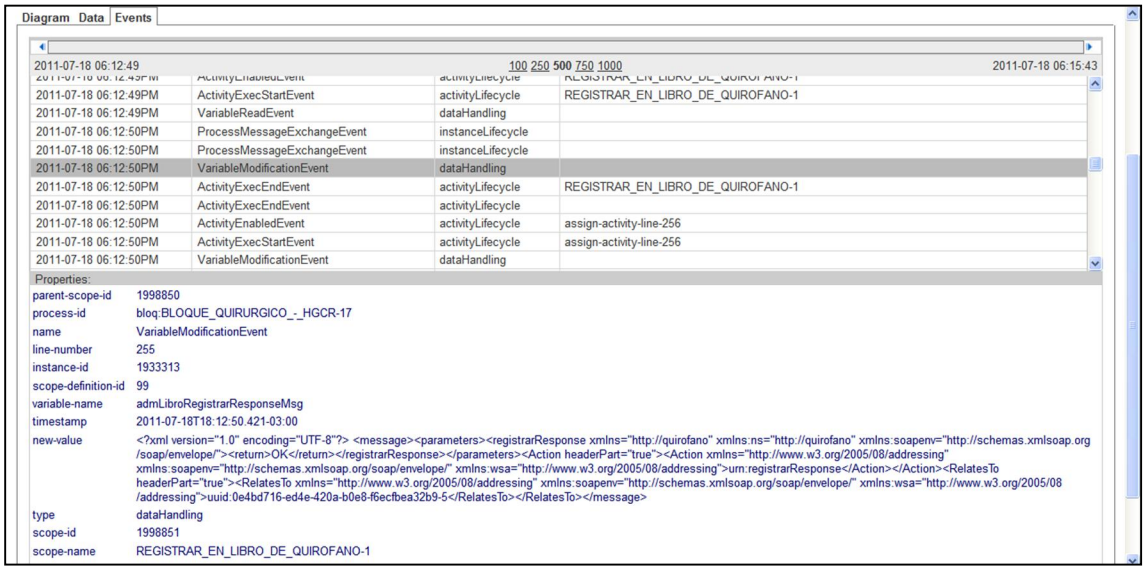

Figura 25 - Mensaje de respuesta del WS

Por último al oprimir el botón “Complete” se envía la información ingresada a la tarea “REVISAR VALORACION DE ANESTESIA Y PREOPERATORIO (fin)” y continúa la ejecución logueandose cada evento del proceso hasta su fin como se observa en la Figura 26.

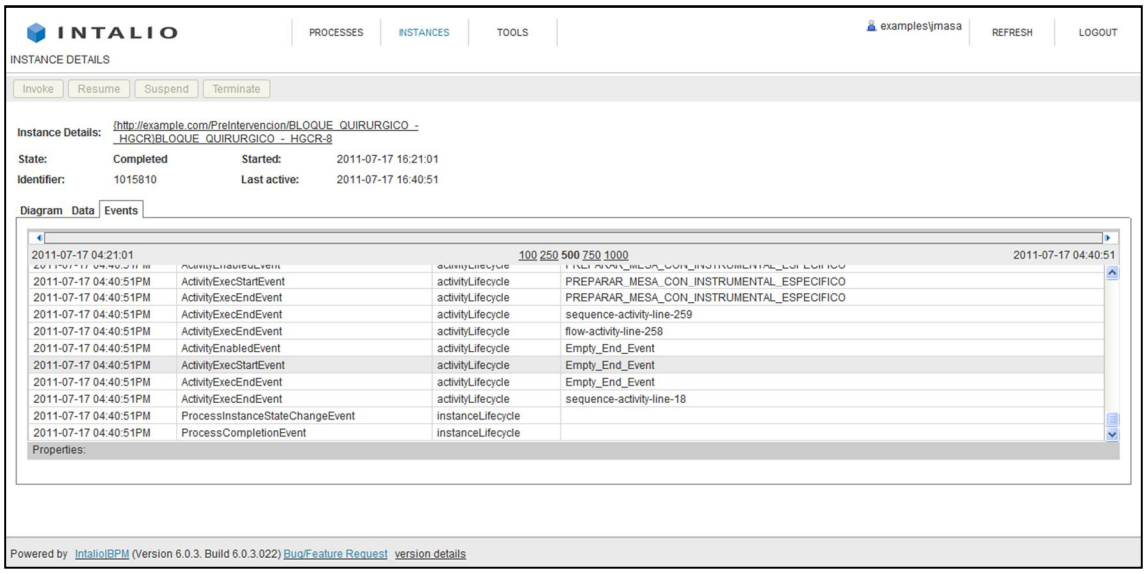

Figura 26 - Eventos ejecutados en el final del proceso

Finalmente con los eventos ejecutados y logueados, se procede a su importación para la herramienta ProM. Debido a que Intalio almacena en tablas la información de logueo de sus ejecuciones, para realizar la importación se debe extraer dicha información desde esa fuente. Para ello, se extrae de la tabla bpel_event, la siguiente información devuelta por la consulta indicada a continuación:

select * from bpel_event where PID= <id_proceso>;

La misma retorna la información de <ID,TSTAMP,TYPE,DETAIL>, donde ID es el identificador de la instancia ejecutada, TSTAMP es el momento de ejecución, TYPE es el tipo de evento logueado (inicio de una ejecución, cambio en una variable, entre otras), y DETAIL es un String conteniendo información extra, como ser la clase ejecutada. Una vez obtenida esta información, se deberá exportarla al formato CSV (Comma-Separated Values), que luego será utilizada como fuente de datos para ProM Import Framework. Como ejemplo de archivo CSV generado, se muestra a continuación en la Figura 27 como seria el formato:

ID	TSTAMP	TYPE	DETAIL
1048964	2011-07-17 16:21:01.109	NewProcessInstanceEvent	"NewProcessInstanceEvent:
			Type = instanceLifecycle
			ScopeDeclarationId = 0
			Operation = initProcess
			PortType = {http://example.com/PedirPaciente/xform}Process
			MessageExchangeId = 917513
			Aspect = 0
			ProcessInstanceId = 1015810
			ProcessId = {http://example.com/PreIntervencion/BLOQUE_QUIRURGICO_-_HGCR}BLOQUE_QUIRURGICO_-_HGCR-8
			ProcessName = {http://example.com/PreIntervencion/BLOQUE_QUIRURGICO_-_HGCR}BLOQUE_QUIRURGICO_-_HGCR
			Timestamp = Sun Jul 17 16:21:01 UYT 2011
			LineNo = -1
			Class = class org.apache.ode.bpel.evt.NewProcessInstanceEvent"
1048965	2011-07-17 16:21:01.109	ProcessInstanceStartedEver	"ProcessInstanceStartedEvent:
			RootScopeId = 1081353
			ScopeDeclarationId = 3
			Type = instanceLifecycle
			ProcessInstanceId = 1015810
			ProcessId = {http://example.com/PreIntervencion/BLOQUE_QUIRURGICO_-_HGCR}BLOQUE_QUIRURGICO_-_HGCR-8
			ProcessName = {http://example.com/PreIntervencion/BLOQUE_QUIRURGICO_-_HGCR}BLOQUE_QUIRURGICO_-_HGCR
			Timestamp = Sun Jul 17 16:21:01 UYT 2011
			LineNo = -1
			Class = class org.apache.ode.bpel.evt.ProcessInstanceStartedEvent"

Figura 27 - Ejemplo de archivo CSV importación el ProM

Luego, con el archivo CSV creado, se lo importa en el ambiente ProM Import Framework, indicando que se desea exportar a formato MXML y la fuente en formato CSV. Presionando Start, se ejecuta el proceso que una vez finalizado se muestra como en la Figura 28 :

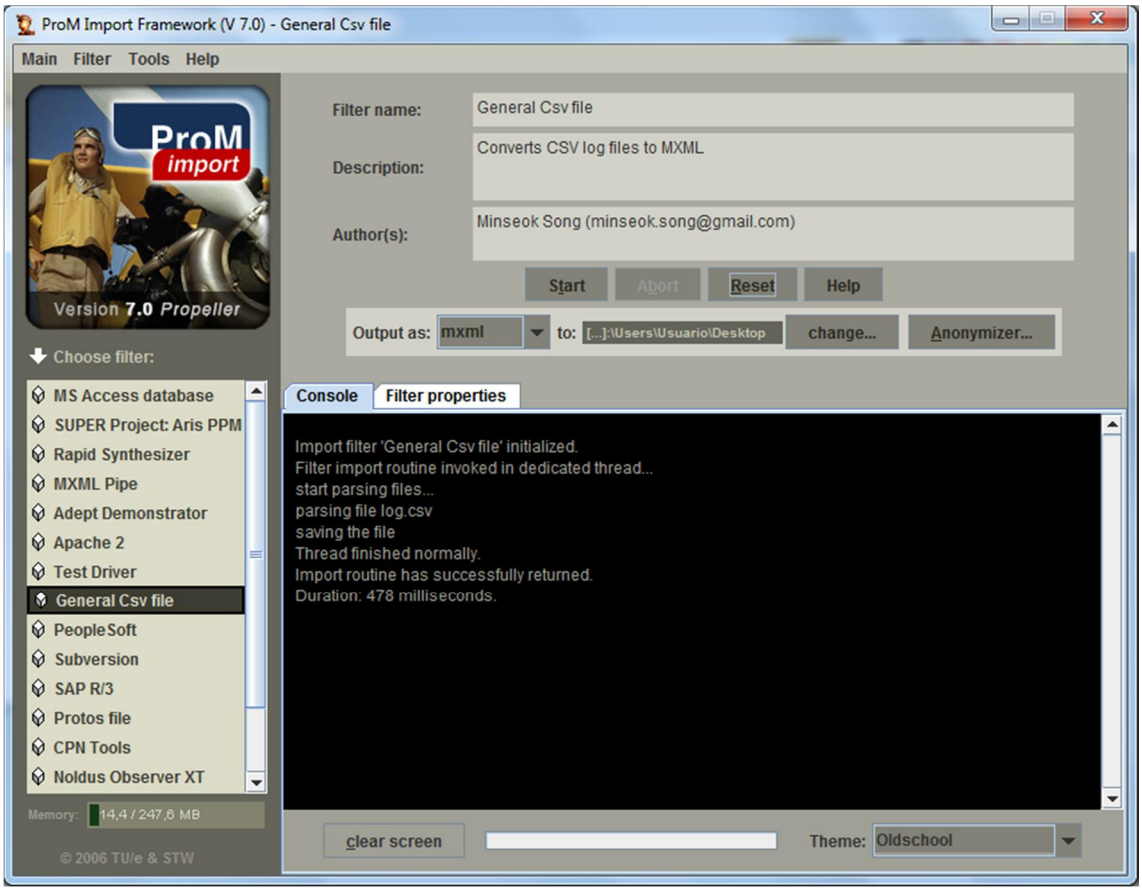

Figura 28 - Importación a MXML finalizada

A continuación se muestra un fragmento del archivo MXML generado:


```
<?xml version="1.0" encoding="UTF-8" ?>
<WorkflowLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://is.tm.tue.nl/research/processmining/WorkflowLog.xsd"
description="Unified single process">
	<Data>
		<Attribute name="app.name">ProM Import Framework</Attribute>
		<Attribute name="app.version">7.0 (Propeller)</Attribute>
		<Attribute name="java.vendor">Sun Microsystems Inc.</Attribute>
		<Attribute name="java.version">1.6.0_22</Attribute>
		<Attribute name="mxml.creator">MXMLib (http://promimport.sf.net/)</Attribute>
		<Attribute name="mxml.version">1.1</Attribute>
		<Attribute name="os.arch">x86</Attribute>
		<Attribute name="os.name">Windows 7</Attribute>
		<Attribute name="os.version">6.1</Attribute>
		<Attribute name="user.name">Usuario</Attribute>
	</Data>
	<Source program="CSV files"/>
	<Process id="UNIFIED" description="Unified single process">
		<ProcessInstance id="1048973,2011-07-17 16:21:01.593,ActivityEnabledEvent,">
			<AuditTrailEntry>
				<WorkflowModelElement>1048973,2011-07-17
16:21:01.593,ActivityEnabledEvent,</WorkflowModelElement>
				<EventType>complete</EventType>
				<Timestamp>1969-12-31T21:00:00.000-03:00</Timestamp>
			</AuditTrailEntry>
		</ProcessInstance>
		<ProcessInstance id="1049283,2011-07-17 16:40:51.062,ActivityEnabledEvent,">
			<AuditTrailEntry>
				<WorkflowModelElement>1049283,2011-07-17
16:40:51.062,ActivityEnabledEvent,</WorkflowModelElement>
				<EventType>complete</EventType>
				<Timestamp>1969-12-31T21:00:00.000-03:00</Timestamp>
			</AuditTrailEntry>
		</ProcessInstance>
		<ProcessInstance id="LineNo = 258">
			<AuditTrailEntry>
				<WorkflowModelElement>LineNo = 258</WorkflowModelElement>
				<EventType>complete</EventType>
				<Timestamp>1969-12-31T21:00:00.000-03:00</Timestamp>
			</AuditTrailEntry>
			<AuditTrailEntry>
				<WorkflowModelElement>LineNo = 258</WorkflowModelElement>
				<EventType>complete</EventType>
				<Timestamp>1969-12-31T21:00:00.000-03:00</Timestamp>
			</AuditTrailEntry>
			<AuditTrailEntry>
				<WorkflowModelElement>LineNo = 258</WorkflowModelElement>
				<EventType>complete</EventType>
				<Timestamp>1969-12-31T21:00:00.000-03:00</Timestamp>
			</AuditTrailEntry>
		</ProcessInstance>
		…
		…
```

Para finalizar el proceso, se procede a importar el archivo MXML generado en la herramienta ProM. En la Figura 29 se puede observar el resultado de dicha importación:

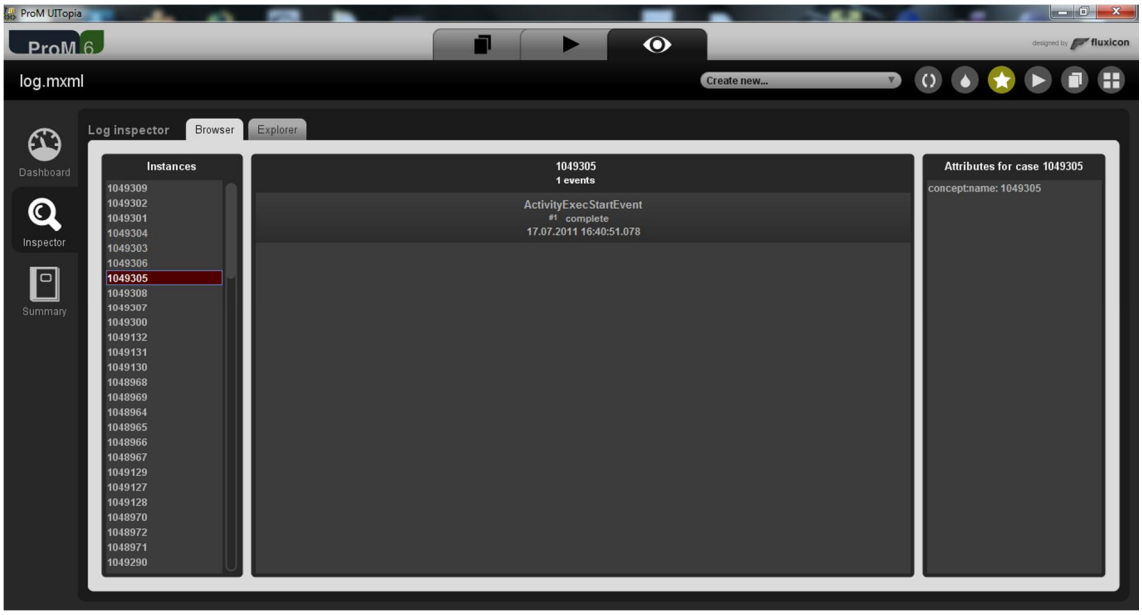

Figura 29 - Vista de resumen en ProM

5.5. Conclusiones

Intalio ha demostrado a lo largo de la ejecución del caso de estudio un comportamiento estable y un entorno amigable. Los espacios de trabajo de cada usuario presentan una interface sencilla y la asignación de tareas es intuitiva para la ejecución de flujos de trabajo. Si bien el diseño del proceso en Intalio|Designer presenta una notación similar a BPMN se deben introducir ciertas modificaciones al mismo para poder incluir formularios y web-services en los procesos. Desde el administrador del servidor se puede ver un log de todos los eventos involucrados en la ejecución de una instancia incluyendo un timestamp. Se incluyen los eventos de activación, inicio y fin de ejecución de cada tarea. A pesar de tener un nivel interesante de detalle en el log de ejecución no aparece información del usuario involucrado en el evento por lo que no se puede saber quien ejecutó cada tarea.

6. Gestión de Proyecto

En este Capítulo se detalla la planificación, coordinación y los tiempos que se utilizaron en el transcurso de este proyecto, indicando además los productos finales de cada etapa.

El grupo de trabajo fue integrado por 2 estudiantes y coordinado por 2 tutores, utilizando como métodos de comunicación el envío de mails y reuniones vía Skype cuando necesario. La forma de trabajo consistía en la división de tareas entre los miembros del grupo y planificación se cada entrega. Cabe acotar que el proyecto presentaba un cronograma tentativo inicial, para todo su transcurso. De todas formas se realizaron planificaciones intermedias para los entregables.

A continuación en este Capítulo, en la sección 6.1 se presenta el Cronograma inicial para el proyecto, en la sección 6.2 el Cronograma realizado con las descripciones de los entregables y el tiempo empleado en cada uno. Finalmente en la sección 6.3 las Conclusiones de la Gestión.

6.1. Cronograma Inicial

Estudio del estado del arte en procesos de negocio, en particular en lo que refiere a la fase de ejecución, motores de procesos y características **(2 meses)**

- Producto: documento del estado del arte

Definición de la evaluación de motores de procesos: **(1 mes)**

- Definición de características a evaluar en motores de procesos, incluyendo revisión de metodologías, técnicas y trabajos existentes al respecto (por ej. implementación de patrones de procesos o Workflow)
- Selección de motores de procesos a evaluar (mínimo 6 motores que implementan WS-BPEL)
- Productos: documento de definición de características y selección de motores de procesos a evaluar

Evaluación de los motores de procesos seleccionados (6 herramientas) incluyendo instalación, pruebas, implementación y ejecución de prototipos **(3 meses)**

- Producto: documento de evaluación de motores de procesos según características definidas

Caso de estudio de aplicación con herramientas seleccionadas (de las evaluadas previamente) que podrá ser de laboratorio o real en alguna organización afín **(2 meses)**

- Producto: desarrollo del caso de estudio, lecciones aprendidas, conclusiones

Informe del proyecto de grado durante todo el proyecto por capítulos, final, correcciones y defensa **(1 mes)**

- Producto: informe final del proyecto de grado por capítulos y defensa

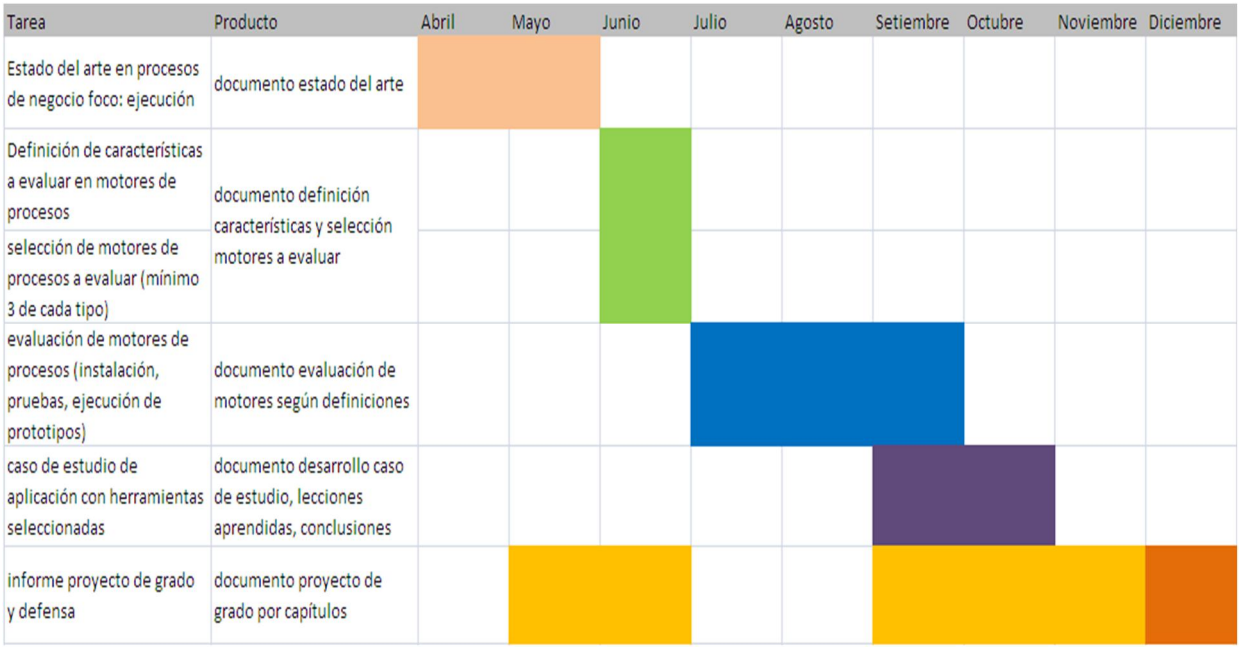

Figura 30 - Cronograma inicial

6.2. Cronograma Realizado

6.2.1. Estado del Arte

Consistió en una tarea para elaborar el marco teórico necesario para realizar las demás tareas relacionadas con el trabajo. Para realizarlo se llevaron a cabo 2 etapas, una primera etapa de investigación para relevar la información pertinente al tema, y una segunda etapa de documentación de dicha investigación. A continuación se muestran fechas de comienzo y fin de cada etapa, indicando además la cantidad de horas que fueron utilizadas.

Investigación

- Periodo: 30/Marzo20100 - 10/Mayo/2010
- Cantidad de horas: 248 hs. (ambos integrantes sumados)

Documentación

- Periodo: 10/Mayo/2010 - 08/Agosto/2010 (se excluye el mes de Julio/2010)
- Cantidad de horas: 612 hs. (ambos integrantes sumados)

6.2.2. Criterios

En esta etapa se realizó una investigación sobre evaluaciones existentes hasta el momento sobre las tecnologías de Procesos de Negocio, luego se definieron criterios, métricas y procedimientos para la evaluación de los motores que se seleccionaron en la siguiente etapa, la de selección de motores. A continuación se muestran fechas de comienzo y fin de cada etapa, indicando además la cantidad de horas que fueron utilizadas.

Investigación

- Periodo: 10/Agosto/2010 – 17/Agosto/2010
- Cantidad de horas: 52 hs. (ambos integrantes sumados)

Definición de criterios, métricas y procedimientos

- Periodo: 17/Agosto/2010 - 29/Agosto/2010
- Cantidad de horas: 132 hs. (ambos integrantes sumados)

Selección de motores

- Periodo: 29/Agosto/2010 - 16/Setiembre/2010
- Cantidad de horas: 180 hs. (ambos integrantes sumados)

6.2.3. Evaluaciones

En esta etapa se realizaron las distintas evaluaciones de los motores seleccionados (7 motores en total), según criterios y procedimientos definidos en la etapa anterior. A continuación se muestran fechas de comienzo y fin de cada evaluación, indicando además la cantidad de horas que fueron utilizadas.

jBPM BPEL

- Periodo: 25/Setiembre/2010 - 25/Octubre/2010
- Cantidad de horas: 167 hs.

Apache ODE

- Periodo: 25/Setiembre/2010 - 25/Octubre/2010
- Cantidad de horas: 117 hs.

Intalio

- Periodo: 25/Octubre/2010 - 20/Diciembre/2010
- Cantidad de horas: 283 hs.

Riftsaw

- Periodo: 25/Octubre/2010 - 20/Diciembre/2010
- Cantidad de horas: 400 hs.

Petals

- Periodo: 17/Enero/2011 - 20/Marzo/2011
- Cantidad de horas: 376 hs.

Orchestra

- Periodo: 17/Enero/2011 - 20/Marzo/2011
- Cantidad de horas: 381 hs.

jBPM5

- Periodo: 26/Marzo/2011 – 22/Mayo/2011
- Cantidad de horas: 264 hs.

6.2.4. Caso de Estudio

En esta etapa se realizó la implementación de un caso real, utilizando como motor de ejecución a Intalio, el cual según las evaluaciones realizadas en la etapa anterior, brindaba mejores características que los demás. A continuación se muestran fechas de comienzo y fin de cada etapa, indicando además la cantidad de horas que fueron utilizadas.

- Periodo: 23/Mayo/2011 - 13/Junio/2011
- Cantidad de horas: 90 hs. (ambos integrantes sumados)

6.2.5. Informe Final

Finalmente en esta etapa se realizó un resumen de todo el trabajo realizado, presentando los resultados obtenidos y conclusiones del mismo. A continuación se muestran fechas de comienzo y fin de cada etapa, indicando además la cantidad de horas que fueron utilizadas.

- Periodo: 7/Mayo/2011 - 25/Julio/2011
- Cantidad de horas: 294 hs. (ambos integrantes sumados)

A modo de resumen de las etapas, en la Figura 31 se puede observar un diagrama de Gantt que ilustra el desarrollo del proyecto y en la Figura 32, una grafica ilustrando la cantidad de horas, por etapa, insumidas a lo largo del proyecto.

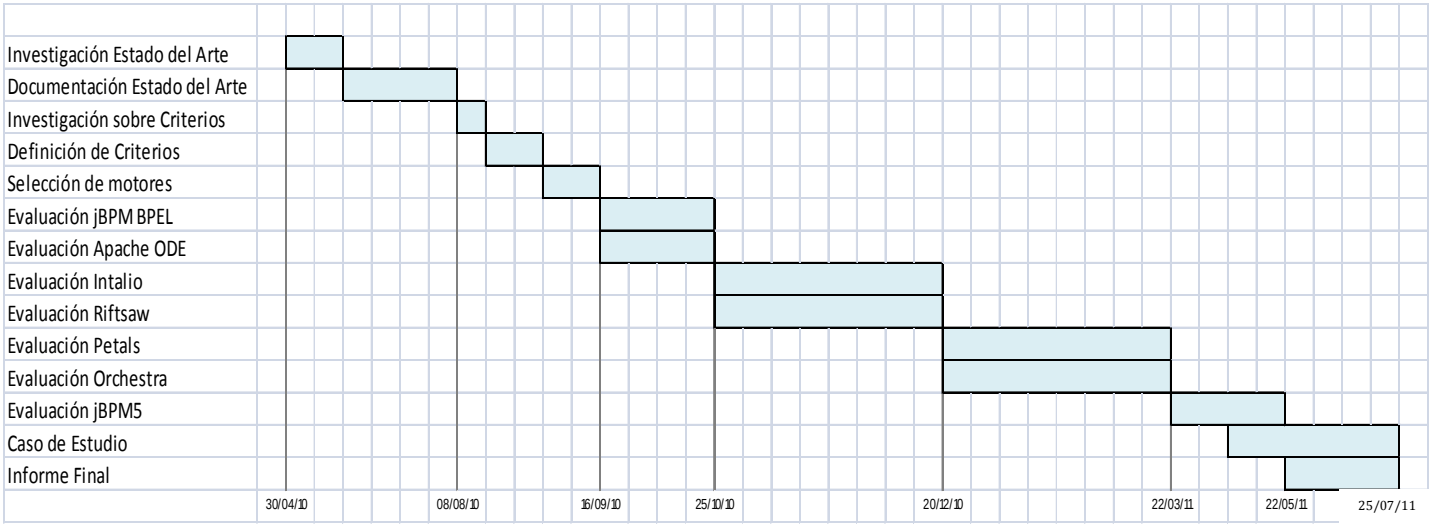

Figura 31 - Diagrama de Gantt de realización del proyecto

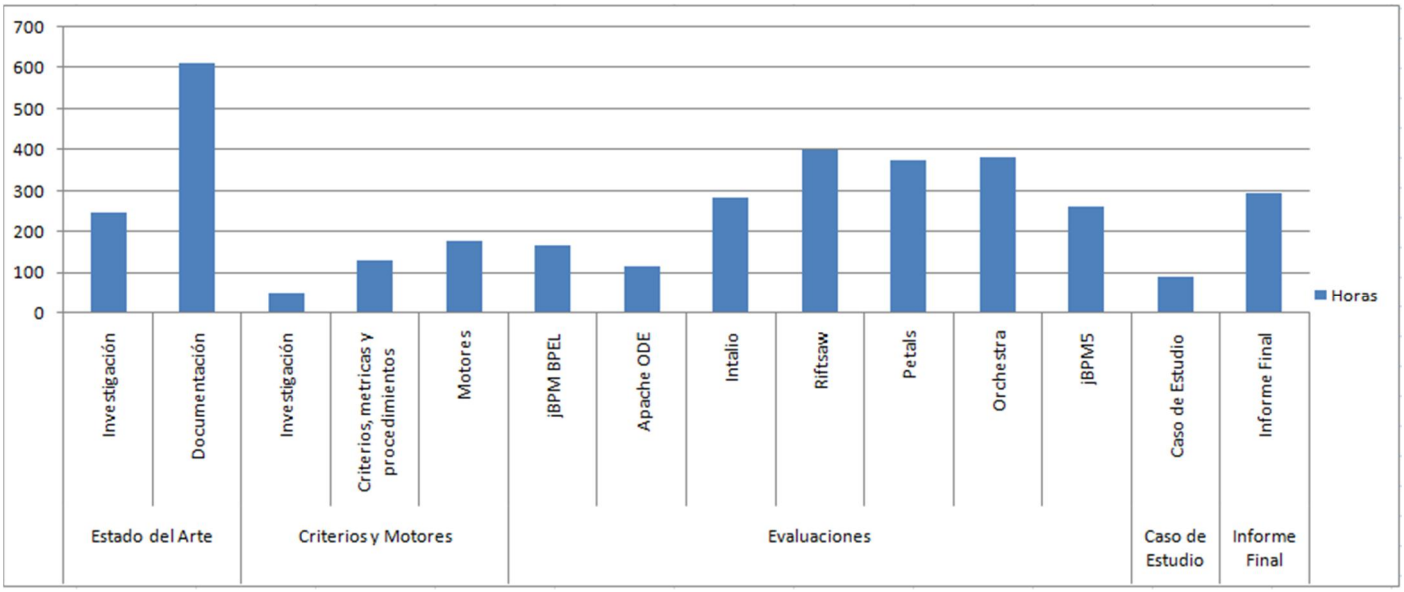

Figura 32 - Cantidad de horas por etapa del proyecto

6.3. Conclusiones

Las desviaciones de los tiempos estipulados en el Cronograma Inicial, respecto al Cronograma Final, se debieron a distintas razones que se detallaran a continuación.

En primera instancia la pérdida de un integrante del grupo (por previas) fue causante de demoras en la realización de las distintas tareas, Estado del Arte, Evaluaciones, etc. Esto conllevó a que el trabajo fuera realizado por 2 integrantes.

En Julio como se puede observar en la Figura 33, no se realizaron tareas pertinentes al Proyecto de Grado debido a exámenes de los integrantes del grupo, lo cual fue causante de un retraso de aproximadamente 1 mes, en el comienzo de la actividad de Definición de Criterios y Selección de Motores para la evaluación.

Figura 33 - Cronograma Final

Una vez realizadas las tareas antes mencionadas, se procedió a la Evaluación de los Motores, los cuales fueron 7 (en el Cronograma Inicial se estipularon 6), debido a la inclusión de jBPM5, como motor que implementa el nuevo estándar BPMN 2.0. Las distintas dificultades de instalación de los motores y en muchos de los casos, la escasa información de documentación de las mismas, causaron un retraso también en dicho tiempo de realización, así como también mayor profundidad en determinados aspectos de la evaluación. En el caso de Riftsaw, la cantidad de horas es mayor que la de los demás debido a que se incluyó el tiempo dedicado al motor Active BPEL, el cual se inició la evaluación, pero se tuvo que abandonar debido a problemas con la licencia del motor.

Finalmente, el Informe Final del Proyecto se fue realizando en paralelo con las demás actividades.

7. Conclusiones

El trabajo presentado se vio motivado por la necesidad de contar con información pertinente a motores de Procesos de Negocios que implementen el estándar BPEL, enfocándose en una evaluación en la cual se presentaran fortalezas y debilidades de cada una, teniendo como principal criterio comparativo, la interoperabilidad de éstas, con determinadas herramientas de modelado de procesos y su integración con logs de ejecución.

En primera instancia, fue necesario realizar un relevamiento de toda la información existente hasta el momento, que se relacionara con la temática de los Procesos de Negocios, y que pertenezca al mundo Open Source. El producto resultante de dicho relevamiento se encuentra en el Estado del Arte generado, el cual tiene como principal hilo conductor, el Ciclo de Vida de los Procesos de Negocio. Cabe destacar que el énfasis mayor se concentro en herramientas de ejecución de Procesos, que implementen el estándar BPEL.

Una vez generado el marco teórico adecuado al trabajo, se procedió a definir Criterios de Evaluación para los distintos motores que también fueron relevados en esta instancia, así como también de métricas para compararlos. Por tal motivo, se realizó una investigación de evaluaciones existentes, para crear nuevos criterios que abarcaran características interesantes de los motores que aun no hayan sido evaluadas. En esta instancia también se realizó un relevamiento exhaustivo de los motores que implementan BPEL y sean Open Source, existentes en el mercado.

Finalmente se realizó la evaluación de los motores con los criterios seleccionados, enfocándose principalmente en la interoperabilidad con la herramienta de modelado eClarus, teniendo en cuenta los problemas que surgen y las posibles soluciones. En este aspecto, se puede decir que la evaluación logró realizar una comparativa bastante amplia entre los motores, y detectando los problemas que surgen en dicha interoperabilidad de las distintas herramientas.

Como resultado de las evaluaciones, se puede destacar que Intalio posee más características deseables que los demás y brinda un ambiente más amigable. Esto se desprende de las conclusiones del Capítulo 4 (Evaluaciones), lo cual es indicado también en las graficas allí presentadas. De todas formas, podemos afirmar que también tiene muchos aspectos a mejorar. Los demás motores si bien se encuentran un escalón más abajo, cada uno tiene aspectos propios interesantes.

Se puede mencionar como conclusión final de los aspectos más técnicos del trabajo, que el mismo puede contribuir para la obtención de una comparativa detallada de características de los motores BPEL evaluados. El marco teórico generado, puede contribuir también como punta pie inicial, para otros trabajos relacionados con el mundo de los Procesos de Negocio y estándares asociados, con énfasis en estándar BPEL. También es posible utilizar este trabajo como modelo para otras evaluaciones de motores de Procesos de Negocio (BPEL o no), debido a la generalidad de muchos de los criterios definidos.

En el ámbito personal, se puede concluir que el trabajo ha otorgado un conocimiento bastante amplio en el mundo de los Procesos de Negocio, enfocándose más fuertemente en aspectos de ejecución de dichos procesos. La utilización de una organización de gran porte como ser un

hospital, y el trabajar con procesos reales (como en el Capitulo 5 – Caso de Estudio), brindaron un conocimiento más en profundidad aun de los usos que se puede dar a este tipo de tecnología, teniendo presente los beneficios y debilidades de la misma.

Respecto a la gestión y el desarrollo del proyecto, podemos concluir que fue satisfactorio, debido a que se cumplieron los objetivos del proyecto. De todas formas se pudo mejorar en algunos aspectos como la comunicación con los tutores y en la realización de cronogramas más efectivos. Sin embargo, la documentación en tiempo y forma, en paralelo con las tareas de investigación y evaluación, fue uno de los puntos más favorables de la gestión del trabajo, haciendo que el resumen final (en el Informe Final del Proyecto), pueda ser realizado de forma más rápida y clara.

Finalmente, se puede decir que este acercamiento con las tecnologías de Procesos de Negocio, ha sido ampliamente satisfactorio para el grupo de trabajo, generando un marco de conocimiento tal, que permite trabajar en otras instancias con el tema tratado en este proyecto.

8. Trabajos a futuro

- Realizar un estudio específico de cada motor, tratando de solucionar los problemas y errores detectados en las pruebas. Esto debería incluir un estudio del código fuente de los motores, ya que en la mayoría de los casos, no se cuenta con la información necesaria en la documentación de los mismos, ni tampoco con la ayuda requerida, por parte de la comunidad. Esto se debe a que muchos motores no tienen la popularidad necesaria para poseer una comunidad activa.

- Otra tarea a futuro seria realizar la evaluación de una herramienta comercial, para realizar una comparativa con las de carácter Open Source evaluadas en esta instancia. De esta forma, se podría tener una visión más general del estado de las herramientas Open Source de Procesos de Negocios, respecto a las comerciales.

- También se podría realizar una evaluación de la interoperabilidad de ProM, utilizando el nuevo estándar XES, con lo cual, se podría generar un análisis comparativo con el realizado en este trabajo (MXML). De esta forma se podría tener mayor información de las fortalezas y debilidades del nuevo estándar, entre otras diferencias que pueda tener.

- De gran interés también, es realizar un estudio comparativo con más herramientas que implementen el estándar BPMN 2.0, ya que en el marco de este trabajo, únicamente se realizó el estudio de uno, jBPM5.

Referencias

Aalst, Wil M. P. van der. 2009. *Pattern-based Analysis of Windows Workflow.* 2009.

Agrawal, Ashish, y otros. 2007. *WS-BPEL Extension for People(BPEL4People), Version 1.0.* 2007.

Allen, Rob. 2001. *Workflow: An Introduction.* 2001.

Amend, Mike, y otros. 2007. *Web Services Human Task(WS-HumanTask), Version 1.0.* 2007.

Baeyens, Tom y Valdes Faura, Miguel. 2007. The Process Virtual Machine. [En línea] 8 de May de 2007. http://docs.jboss.com/jbpm/pvm/article/.

Beginner's Guide to Windows Workflow Foundation. *msdn.microsoft.com.* [En línea] Microsoft. http://msdn.microsoft.com/en-us/netframework/first-steps-with-wf.aspx.

eClarus. 2011. eClarus. *eClarus.* [En línea] 14 de 04 de 2011. [Citado el: 14 de 04 de 2011.] http://www.eclarus.com/index.html.

Eswaran, Sharanya, y otros. 2006. *Adapting and Evaluating Commercial Workflow Engines for e-Science.* Department of Computer Sience, University of Virginia. Charlottesville, VA USA : IEEE Computer Society, 2006.

Garcês, Ricardo, y otros. 2008. *Open Source Workflow Management Systems: A Concise Survey.* University of Madeira, Portugal. 2008.

Havey, Mike. 2005. *Essential Business Process Modeling.* 2005.

Hollingsworth, David. 1995. *Workflow Management Coalition. The Workflow Reference Model.* 1995.

—. 1995. *Workflow Management Coalition. The Workflow Reference Model.* 1995.

Holmes, Ta'id, Vasko, Martin y Dustdar, Schahram. 2007. *VieBOP: Extending BPEL Engines with BPEL4People.* 2007.

IBM and SAP. 2005. *WS-BPEL Extension for People – BPEL4People.* 2005.

jBPM. 2011. jBPM. *jBPM.* [En línea] 2011. http://www.jboss.org/jbpm/.

Keller, G, Nüttgens, M y Scheer, A -W. 1992. *Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK)".* 1992.

Microsoft BizTalk Server Home Page. *www.microsoft.com.* [En línea] Microsoft. http://www.microsoft.com/biztalk/en/us/default.aspx.

OASIS. 2006. eClarus BPMN to BPEL. *eClarus BPMN to BPEL.* [En línea] 2006. http://bpel.xml.org/resource/bpmn-bpel-transformation-and-round-trip-engineering.

—. 2006. SOA. [En línea] 12 de Octubre de 2006. http://docs.oasis-open.org/soa-rm/v1.0/.

—. 2007. Standard WS-BPEL Version 2.0. *OASIS Standard.* [En línea] 2007. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf.

—. UDDI. [En línea] http://uddi.xml.org/.

ODE. Apache ODE. [En línea] [Citado el: 04 de Abril de 2011.] http://ode.apache.org/creating-a-process.html.

OpenWFE. 2011. [En línea] 2011. http://sourceforge.net/projects/openwfe/.

Oracle BPEL Process Manager. *www.oracle.com.* [En línea] Oracle. http://www.oracle.com/technetwork/middleware/bpel/overview/index.html.

ProM. 2011. Process Mininig. [En línea] 14 de 04 de 2011. [Citado el: 14 de 04 de 2011.] http://www.win.tue.nl/processmining/prom/downloads.

ProMImportFramework. 2011. ProMImportFramework. *ProMImportFramework.* [En línea] 14 de 04 de 2011. [Citado el: 14 de 04 de 2011.] http://www.promtools.org/promimport/.

Quon, Gerald. 2008. *Evaluating Workflow Management Systems for fMRI.* Deparment of Computer Science, University of Toronto. Toronto, Canada : s.n., 2008.

Russell, Nick, ter Hofstede, Arthur H.M. y Edmond, David. 2005. *Workflow Resource Patterns.* 2005.

Russell, Nick, van der Aalst, Wil y ter Hofstede, Arthur. 2006. *Workflow Exception Patterns.* 2006.

Russell, Nick, y otros. 2007. *Workflow Control-Flow Patterns.* 2007.

Russell, Nick, y otros. 2005. *Workflow Data Patterns: Identification, Representation and Tool Support.* 2005.

Shark, Ehyndra. 2011. [En línea] 2011. http://www.together.at/prod/workflow/tws.

Terminology & Glossary. **Workflow Management Coalition. 1999.** 3.0, 1999.

Tsalgatidou, Aphrodite. 2007. *Selection Criteria for Tools Supporting Business Process Transformation for Electronic Commerce.* Athens : University of Athens, Dept. of Informatics, 2007.

van der Aalst, Wil M.P. 2000. *Formalization and Verification of Event-driven Process Chains.* 2000.

van der Aalst, Wil M.P., y otros. *Design and implementation of the YAWL system.*

van der Aalst, Wil M.P., y otros. 2002. *Workflow Patterns.* 2002.

W3C. 2007. SOAP Specifications. [En línea] 28 de Abril de 2007. [Citado el: 06 de Agosto de 2010.] http://www.w3.org/TR/soap12-part1/.

—. 2001. WSDL. [En línea] 2001. http://www.w3.org/TR/wsdl.

Weske, Mathias. 2007. *Business Process Management.* 2007.

WfMC. 2008. XPDL. [En línea] 10 de Octubre de 2008. http://www.wfmc.org/.

Wohed, Petia, y otros. 2003. *Analysis of Web Services Composition Languages: The case of BPEL4WS.* 2003.

Wohed, Petia, y otros. 2007. *Pattern-based Evaluation of Open Source BPM Systems.* 2007.

Glosario

A

Auditoria: Proceso que permite recoger, agrupar y evaluar evidencias para determinar si un sistema de información mantiene la integridad de los datos, utiliza eficientemente los recursos, y cumple con las leyes y regulaciones establecidas

B

BPEL (Business Process Execution Language): Lenguaje estándar para ejecución de Procesos de Negocio

Business Process Management System: Software encargado de automatizar la ejecución de procesos y sus actividades

BPMN (Business Process Modelling Notation): Conjunto de construcciones gráficas que permiten modelar las diferentes partes de un Proceso de Negocio

C

CRM (Costumer Relationship Management): Tecnología para organizar, automatizar y sincronizar Procesos de Negocio

CSV (Comma Separated Values): Formato de archivo que almacena datos de forma tabular utilizando el carácter coma como separador de valores

D

DataHandling: Forma de mantener los datos en una base de datos

Deploy: Actividades que permiten que un servicio, quede disponible para ser usado

Diagrama de Gantt: Diagrama utilizado para mostrar tiempo de dedicación

E

EPC (Event-driven Process Chain): Diagrama de flujo utilizado en el modelado de Procesos de Negocio, para configuración de un ERP

ERP (Enterprise Resource Planning): Sistema utilizado para el manejo y coordinación de recursos, información y funciones de un negocio

Escalabilidad: Capacidad de extenderse sin perder la calidad

I

IDEF (Integration DEFinition): Lenguajes de modelado en el campo de la Ingeniería de Software, utilizado para modelado funcional, simulación y análisis

Interoperabilidad: Capacidad de intercambiar procesos y datos, por parte de sistemas heterogéneos

L

Logs: Forma de registrar una secuencia de ejecuciones

M

Model Driven Development (MDD): Metodología para desarrollo de Software, que se enfoca en el modelado a través de representaciones abstractas como un modelo de dominio

MXML: Formato extensible basado en XML que permite almacenar informacion de logs de procesos

O

OMG (Object Managment Group): Consorcio sin fines de lucro dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos

Open Source: Son prácticas en producción y desarrollo que promueven el acceso al código fuente de un producto

P

Procesos de Negocio: Conjunto de actividades que son realizadas en coordinación en entorno organizacional y técnico

R

Redes de Petri: Representación matemática de un sistema distribuido discreto, incluyendo, al igual que BPMN, UML, etc., una representación grafica

Rollback: Operación que retorna un sistema, proceso o base de datos, a un estado previo

S

Service Oriented Computing (SOC): Paradigma que utiliza servicios para desarrollo de rápido y de bajo costo

SOA (Service Oriented Architecture): Paradigma de organización y distribución de capacidades y características (a través de servicios) en distintos dominios

Stakeholders: Termino que refiere a las partes interesadas en una actividad

T

Timestamp: Tipo de datos que incluye fecha y hora

W

WMS (Workflow Managment System): Sistema que permite el manejo de un Workflow, pudiendo definir tareas, conexiones, etc.

Workflow: Flujo de trabajo, consiste en una secuencia de pasos conectados

WSDL (Web Services Description Language): Lenguaje basado en el estándar XML, utilizado para describir un Web-service

X

XES (eXtensible Event Stream): Nuevo estándar de la herramienta ProM, con similar finalidad que MXML

XML (eXtensible Markup Language): Estándar que permiten codificar documentos de forma tal que sean legibles por una maquina.

Y

YAWL (Yet Another Workflow Language): Lenguaje de modelado de procesos basado en los patrones de Workflow

Evaluación de Tecnologías de Procesos de Negocio centrado en motores BPEL

Anexo 1 – Estado del Arte

Año 2010

Autores: Daniel Bonini, Federico Parins

Tutor: Andrea Delgado
Co-Tutor: Pablo Miranda

1. Estado del Arte

1.1. BPMN

En la Error! Reference source not found. de la sección **Error! Reference source not found.** del Informe Final, se presentó un BPD (Business Process Diagram) sobre un proceso de solicitud de crédito. A continuación utilizaremos este proceso para presentar algunas de las construcciones empleadas en un BPD.

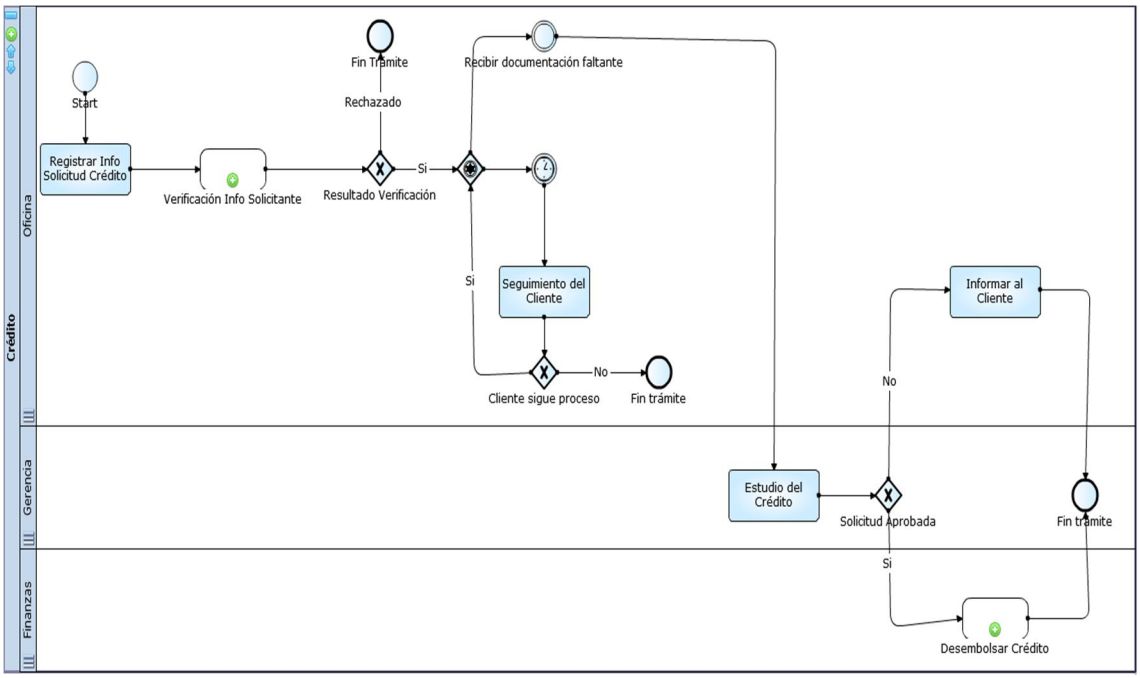

Figura 1 - Ejemplo de proceso en BPMN

Algunos elementos que forman parte de un BPD (Business Process Diagram) son los que se emplean para representar eventos como el de inicio del proceso, el que finaliza el mismo y algunos intermedios como eventos simples empleados para la recepción de documentación faltante y un evento de timer empleado en este caso para darle tiempo al cliente a que presente la documentación faltante.

Figura 2 - Eventos

También podemos encontrar tareas y sub-procesos. La diferencia entre ambas es que las tareas implican trabajo simple en el sentido de que no se descompone en otras actividades mientras que los sub-procesos si se descomponen en otras actividades. Un sub-proceso gráficamente se diferencia de una tarea agregando un signo de *"+"* en el borde inferior del símbolo de Tarea, y puede encerrar en sí mismo un BPD.

Figura 3 - Actividades

Para modelar las decisiones en el flujo del proceso se utilizan los Gateway que permiten bifurcar el flujo de control en función de eventos o datos del proceso. En la Figura 4 podemos observar dos Gateway XOR que permiten tomar uno y solo un camino en el flujo de control. Por ejemplo, luego de la actividad de Estudio del Crédito (Figura 1) se utiliza un XOR que toma un camino u otro basado en si la solicitud fue aprobada o no, este es un XOR que bifurca en función de los datos del proceso. Por otro lado, luego de que la información del solicitante ha sido verificada en forma positiva, en caso que el cliente no haya presentado toda la documentación se le da un cierto plazo de tiempo para que lo haga. En caso que el cliente presente la documentación antes de que venza dicho plazo se envía la solicitud a la Gerencia para su estudio y aprobación/rechazo. Si el plazo para presentar la documentación vence se le realiza un seguimiento al cliente para determinar si desea continuar con los trámites, en caso de ser así se le da otro plazo para la presentación de la documentación. Para modelar este comportamiento se empleó un XOR basado en eventos, donde se tomará un camino u otro basado en el evento que ocurra primero, el vencimiento del plazo (evento Timer) o la presentación de la documentación antes del vencimiento (evento Recibir documentación faltante)

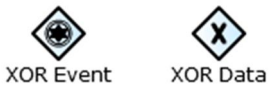

Figura 4 – Gateways

1.2. Patrones de Procesos

Los patrones están divididos en cuatro categorías o perspectivas desde las cuales se puede ver un Proceso de Negocio; estas perspectivas son, Control de flujo, Datos, Recursos y Manejo de excepciones.

Control de flujo

Esta perspectiva describe las actividades y su orden de ejecución a través de diferentes construcciones, las cuales permiten representar el flujo de ejecución, como por ejemplo, secuencias, decisiones, combinaciones de flujos, sincronización, ciclos, etc. (van der Aalst, y otros, 2002) (Russell, y otros, 2007).

A continuación se presentan a modo de ejemplo algunos patrones de control de flujo junto con un diagrama del mismo expresado con el uso de Redes de Petri. Los patrones seleccionados muestran la estructura de algunas construcciones básicas (como ejecución concurrente de tareas, sincronización, ciclos y acceso a recursos compartidos).

- **WPC-2 Parallel Split:** Este patrón permite que luego de completada una tarea se puedan ejecutar dos o más flujos de tareas en forma concurrente. Este patrón se conoce también con el nombre AND-split dado que deben ejecutarse todas las ramas de la bifurcación.

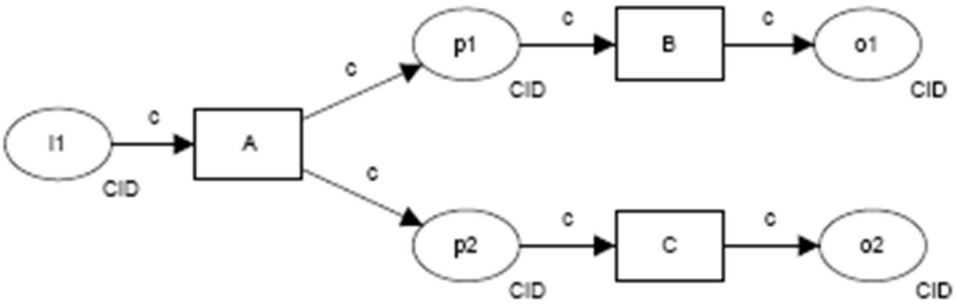

Figura 5 - Diagrama del patrón Parallel Split (Russell, y otros, 2007)

- **WPC-3 Synchronization:** Este patrón permite unir dos o más flujos de ejecución paralelos, los cuales por lo general son creados con el patrón *Parallel Split*. Este patrón se conoce también con el nombre AND-join, debido a que la tarea a continuación de la construcción que sincroniza los diversos flujos debe esperar a que todas las ramas precedentes finalicen para poder ejecutarse.

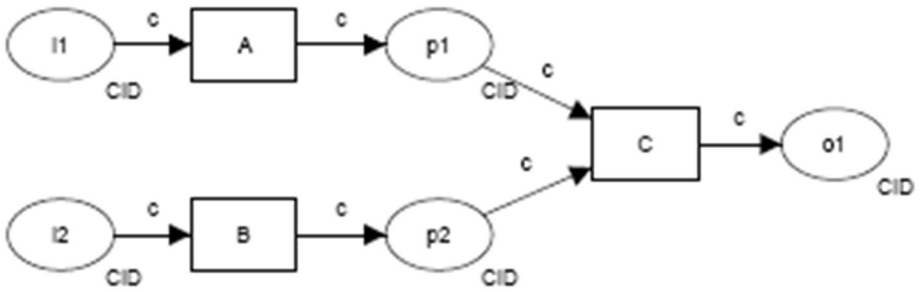

Figura 6 - Diagrama del patrón Synchronization (Russell, y otros, 2007)

- **WPC-21 Structured Loop:** Este patrón permite la ejecución repetida de una tarea o sub-proceso en función del resultado de la evaluación de condición. Hay dos formas posibles para este patrón, el ***while-do*** y el ***repeat-until.*** Básicamente estas dos formas se comportan de forma análoga a las construcciones similares de cualquier lenguaje de programación. La forma *while-do* repetirá las tareas cero o más veces mientras la condición sea verdadera y la forma *repeat-until* ejecutará una o más veces la tarea hasta que la condición sea verdadera.

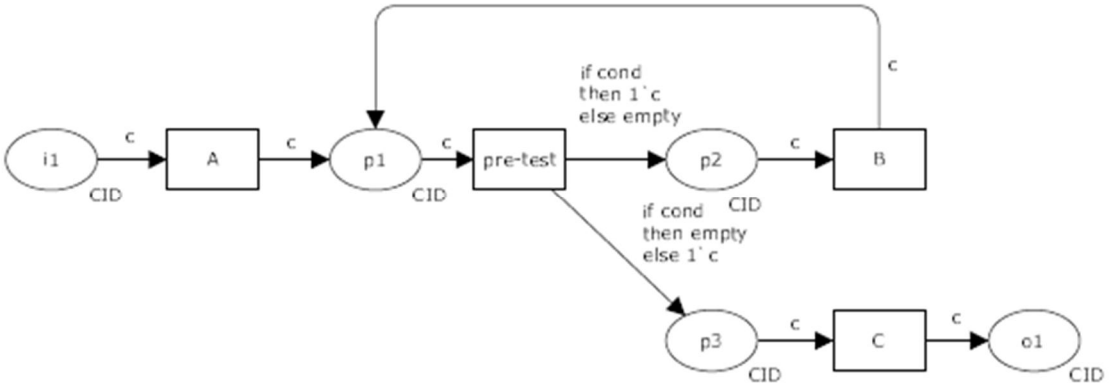

Figura 7 - Diagrama del patrón Structured Loop (while-do) (Russell, y otros, 2007)

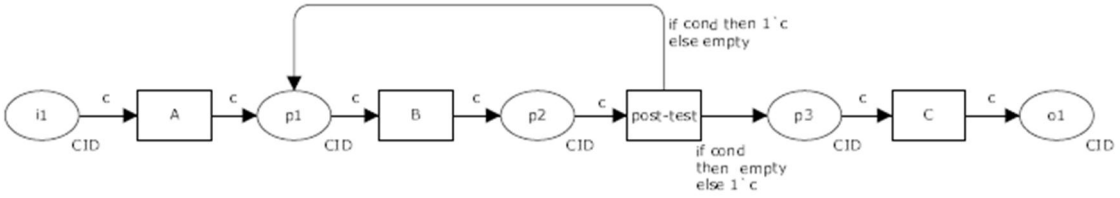

Figura 8 - Diagrama del patrón Structured Loop (repeat-until) (Russell, y otros, 2007)

- ***WPC-39*** **Critical Section:** El patrón *Critical Section* provee los mecanismos para evitar que dos o más flujos de ejecución se ejecuten en forma concurrente. Esto por lo general es necesario si las tareas dentro de la región crítica necesitan acceder algún recurso compartido (información o recursos físicos) para poder completar su trabajo. Esto es análogo a un semáforo en un sistema operativo.

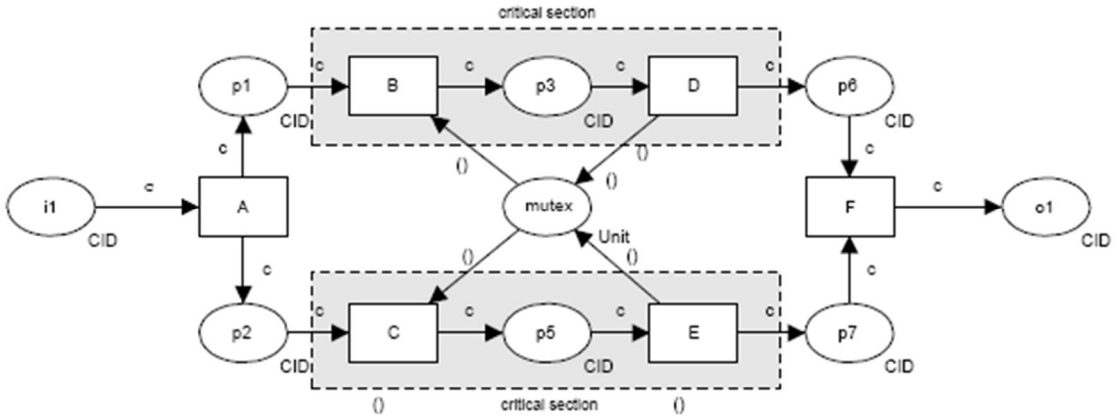

Figura 9 - Diagrama del patrón Critical Section (Russell, y otros, 2007)

Datos

Esta perspectiva se centra en las diferentes formas en que la información es representada y utilizada por los workflows. Existen diferentes características atribuibles a la información y que pueden ser divididas en cuatro grupos, los cuales son Visibilidad (refiriéndose a la forma en que la información es vista por los componentes de un proceso de workflow), Interacción (enfocándose en la manera en que la información es comunicada a través de los elementos activos dentro de un workflow), Transferencia (la forma en que la información se transfiere entre los componentes de un workflow) y Enrutamiento (el cual caracteriza la forma en que la información puede tener cierta influencia sobre otros aspectos del workflow, en particular el flujo de control) (Russell, y otros, 2005).

Recursos

Estos patrones pretenden caracterizar las diferentes formas en que los recursos son representados y utilizados dentro del workflow. En este caso se considera un recurso a una entidad capaz de realizar algún tipo de trabajo y se clasifican en humanos y no humanos (Russell, y otros, 2005).

Manejo de Excepciones

Esta perspectiva se centra en los eventos anormales y las diferentes formas en que estos pueden generarse en un punto determinado del tiempo durante la ejecución de un proceso y que se relacionan con un único elemento. La forma en que las excepciones son manejadas dependerá del tipo de la excepción pero la estrategia se centra en tres consideraciones principales: cómo será manejado el elemento al cual está relacionada la excepción, cómo serán manejados los otros elementos y cuál será la acción de recuperación que será tomada para resolver los efectos causados por la excepción (Russell, y otros, 2006).

1.3. Elementos de WS-BPEL

Proceso

Un Proceso en BPEL es un contendor de otros objetos que hacen al proceso, contiene las variables, los manejadores y por supuesto las actividades. En el mismo se define en primera instancia si el Proceso será ejecutable o un proceso abstracto, donde un proceso abstracto significa que el mismo no cubre toda la definición del proceso, sino que solamente una parte del mismo.

Un ejemplo se puede ver a continuación,

```
<process name="PrimerProcess"
targetNamespace="http://oasis-open.org/WSBPEL/Primer/"
xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable" />
```

Scope

Se utiliza para separar secciones dentro de un proceso, donde los objetos que se declaren dentro de éste (actividades, variables, manejadores, entre otros), serán visibles únicamente en el mismo y no desde otro scope.

```
<scope name="ScopeEjemplo" />
```

Business Partner

Define las declaraciones de los Web services que usa, así como también que funciones cumplen y con sus roles. La interacción entre Web services se ve reflejada a través de los Business Partners. A continuación se muestra un ejemplo de su declaración,

```
<partnerLinks>
<partnerLink name="ClientStartUpLink"
partnerLinkType="wsdl:ClientStartUpPLT" myRole="Client" />
</partnerLinks>
```

Variables

Las variables son las que almacenan el estado del proceso, las mismas pueden ser declaradas a nivel del proceso global o en un scope, siendo en este caso, visible solamente en el alcance de este scope.

Las mismas soportan 3 tipos, datos desde un mensaje proveniente de un WSDL (intercambiado por un Business Partner), un tipo simple XML (para intercambiar información dentro del proceso) o un tipo XML complejo (también llamado XML Element). Un ejemplo se muestra a continuación,

```
<variables>
<variable name="myVar1" messageType="myNS:myWSDLMessageDataType" />
<variable name="myVar2" element="myNS:myXMLElement" />
<variable name="myVar3" type="xsd:string" />
<variable name="myVar4" type="myNS:myComplexType" />
</variables>
```

Receive

Es una actividad que se encarga de recibir un mensaje desde un Web service que se indica y guardarlo en una variable. En caso que sea necesaria una respuesta al mensaje se debe asociar a esta actividad una respuesta a través de un Reply. A continuación se presenta un ejemplo,

```
<receive name="ReceiveRequestFromPartner"
createInstance="yes"
partnerLink="ClientStartUpPLT"
operation="StartProcess" ... />
```

Reply

Es la actividad encargada de manejar la respuesta a un pedido de un Web service, y de forma sincrónica. También maneja las posibles Excepciones, las cuales se indican con la propiedad faultName, en caso de ser necesario.

```
<reply name="ReplyResponseToPartner"
partnerLink="ClientStartUpPLT"
operation="StartProcess" ... />
```

Invoke

Es la actividad utilizada cuando se necesita invocar un método de un determinado Web service. Para ello, es necesario especificar el Business Partner asociado a dicho Web service. La invocación puede ser de 2 tipos, de ida y vuelta, con lo cual es necesario también inputVariable y outputVariable; o puede ser solamente de ida, con lo cual se necesita nada más que la especificación de la inputVariable. A continuación se muestra un ejemplo,

```
<invoke name="RequestResponseInvoke"
partnerLink="BusinessPartnerServiceLink"
operation="RequestResponseOperation"
inputVariable="Input"
outputVariable="Output" />
```

Sequence

Es una estructura que indica un conjunto de actividades que se deben ejecutar secuencialmente en el orden indicado.

```
<sequence name="InvertMessageOrder">
<receive name="receiveOrder" ... />
<invoke name="checkPayment" ... />
<invoke name="shippingService" ... />
<reply name="sendConfirmation" ... />
</sequence>
```

If-else, while, repeatUntil y forEach

Debido a su similitud con las estructuras más utilizadas en los lenguajes de programación más comunes, se procede a explicarlas simplemente con un ejemplo de cada una.

Ejemplo para estructura **if-else**,

```
<if name="isOrderBiggerThan5000Dollars">
<condition>
$order &gt; 5000
</condition>
<invoke name="calculateTenPercentDiscount" ... />
<elseif>
```

```
<condition>
$order &gt; 2500
</condition>
<invoke name="calculateFivePercentDiscount" ... />
</elseif>
<else>
<reply name="sendNoDiscountInformation" ... />
</else>
</if>
```

Ejemplo para estructura **while**,

```
<while>
<condition>
$iterations &gt; 3
</condition>
<invoke name="increaseIterationCounter" ... />
</while>
```

Ejemplo para estructura **repeatUntil**,

```
<repeatUntil>
<invoke name="increaseIterationCounter" ... />
<condition>
$iterations &gt; 3
</condition>
</repeatUntil>
```

Ejemplo para estructura **forEach**,

```
<forEach parallel="no" counterName="N" ...>
<startCounterValue>1</startCounterValue>
<finalCounterValue>5</finalCounterValue>
<scope>
<documentation>check availability of each item ordered</documentation>
<invoke name="checkAvailability" ... />
</scope>
</forEach>
```

Flow

Cuando determinadas tareas pueden ser ejecutadas en paralelo, las mismas son declaradas dentro de la actividad Flow. Dentro de una actividad de este tipo, se pueden declarar cualquier otro tipo de actividades y estructuras, ya sean Receive, Invoke, o cualquiera de las ya mencionadas anteriormente. También con una actividad de este tipo, se puede definir transacciones para todas las actividades pertenecientes al mismo. Un pequeño ejemplo se muestra a continuación,

```
<flow ...>
<links> ... </links>
<documentation>
check availability of a flight, hotel and rental car concurrently
</documentation>
<invoke name="checkFlight" ... />
<invoke name="checkHotel" ... />
<invoke name="checkRentalCar" ... />
</flow>
```

Assign

Esta operación se utiliza cuando se desea copiar el valor de una o más variables a otras. Para cada sentencia de copiado se debe indicar un origen y destino, from y to respectivamente.

A continuación se muestra un ejemplo,

```
<assign>
<copy>
<from variable="TimesheetSubmissionFailedMessage" />
<to variable="EmployeeNotificationMessage" />
</copy>
</assign>
```

También se puede copiar partes de variables, como se muestra a continuación,

```
<assign>
<copy>
<from variable="Input" part="operation1Parameter">
<query>
creditCardInformation
</query>
</from>
<to variable="CreditCardServiceInput" />
</copy>
</assign>
```

Al igual que una variable (o una parte de una variable), también se pueden copiar otras expresiones como se muestra a continuación,

Variable Property:

```
<from> bpel:getVariableProperty("Input", "a:orderNo")
</from>
```

Partner Link:

```
<from partnerLink="Supplier"/>
```

Expression:

```
<from>count($po/poline)</from>
```

FaultHandlers

Se utiliza para manejar excepciones dentro de un proceso (Process), Scope o mismo dentro de una instrucción Invoke.

Un ejemplo de su uso se muestra a continuación,

```
<faultHandlers>
<catch faultName="BookOutOfStockException"
faultVariable="BookOutOfStockVariable">
...
</catch>
<catchAll>...</catchAll>
</faultHandlers>
```

1.4. Elementos de XPDL

Package

La estructura de un XPDL, está formada primeramente por una entidad llamada Package, es quien define la aplicación y consiste en uno o más definiciones de procesos en workflows.

Activities

Las actividades en este lenguaje se categorizan en 3 tipos, Route, Block e Implementation. Los de tipo Route no realizan operaciones, sino que simplemente manejan el control del flujo, indicando la siguiente transición a tomar. Los de tipo Block tienen en si un conjunto de actividades y un alcance de ejecución, muy similares a los Scopes en WS-BPEL (sección 4.1.2). El siguiente es implementation, el cual consiste en 3 tipos, Null, Tool y Subflow. El de tipo Null es la actividad asignada a un humano, Tool es la encargada de realizar invocaciones a las aplicaciones y Subflow es quien invoca un proceso workflow (sincrónico o asincrónico).

Variables

Al igual que en WS-BPEL, en XPDL se puede crear variables y realizar distintas operaciones con ellas (copias, asignaciones, entre otras).

Split and Join

Para implementar operaciones de Split and Join, se cuenta con AND y XOR's, además de actividades de tipo Route. Esto permite a XPDL ser un estándar más poderoso y lograr una mayor expresividad a la hora de interpretar los procesos.

Loops

Los loops se pueden construir mediante la clase ConformanceClass con la propiedad GraphConformance="NON_BLOCKED", con el mismo comportamiento de un goto. Es importante destacar que XPDL no soporta loops del estilo while o foreach.

A continuación se muestra un ejemplo donde se puede observar las actividades en cuestión y las condiciones de parada del loop.

```
<Package>
   <ConformanceClass GraphConformance="NON_BLOCKED"/>   . . .
   <WorkflowProcesses>
      <WorkflowProcess>
         <Activities>
            <Activity id="A"> . . . </Activity>
            <Activity id="B"> . . . </Activity>
            <Activity id="C"> . . . </Activity>
         </Activities>
         <Transitions>
            <Transition id="A2B" from="A" to="B"/>
            <Transition id="B2C" from="B" to="C"/>
               <Condition
Type="CONDITION">okToProceed="yes"</Condition>
            </Transition>
            <Transition id="B2A" from="B" to="A"/>
               <Condition Type="OTHERWISE"/>
            </Transition>
         </Transitions>
     </WorkflowProcess>
   </WorkflowProcesses>
</Package>
```

Applications

Un Package en XPDL puede definir Applications que pueden ser definidas como operaciones de un WSDL, pudiendo comunicarse con otros actores externos.

Extensions

A la mayoría de los elementos en XPDL se les puede extender agregando más información. En el ejemplo general que se muestra más adelante se puede ver su uso.

Ejemplo general de XPDL

A continuación se muestra un ejemplo simple de proceso en XPDL,

```
<WorkflowProcess>
   <Activities>
      <Activity id="A"> . . . </Activity>
      <Activity id="B"> . . . </Activity>
      <Activity id="C"> . . . </Activity>
   </Activities>
   <Transitions>
      <Transition id="A2B" from="A" to="B"/>
      <Transition id="B2C" from="B" to="C"/>
   </Transitions>
   <ExtendedAttributes>
      <ExtendedAttribute Name="StartOfWorkflow" Value="A"/>
      <ExtendedAttribute Name="EndOfWorkflow" Value="C"/>
   </ExtendedAttributes>
</WorkflowProcess>
```

1.5. Patrones de Procesos y Lenguajes de Ejecución/Interpretación

Resulta de interés conocer qué Patrones de Workflow (incluyendo algunos no presentados), son soportados por los estándares de interpretación de modelos que se mencionaron en la sección 2.3.

A continuación, se presenta tanto para WS-BPEL como para XPDL, algunos de los patrones son soportados y porque actividades de los respectivos lenguajes. (Havey, 2005)

PATRON	WS-BPEL	XPDL
Sequence	Actividad de tipo **Sequence**	Con una transición entre 2 actividades
Parallel Split	Actividad de tipo **Flow**	AND Split
Synchronization	Actividad de tipo **Flow** seguida de una **Activity**	AND Join
Exclusive Choice	Actividad de tipo **Switch**	XOR Split
Simple Merge	Actividad de tipo **Switch** seguida de una **Activity**	XOR Join
Multi-Choice	Actividad de tipo **Flow** seguida de varios de tipo **Activity**	AND Join y Split con condiciones en las transiciones
Sync Merge	Similar a Multi-Chioce	AND Join
Multi Merge	No soportado	No soportado
Discriminator	No soportado	No soportado

Arbitrary cycles	Actividad de tipo **While-Repeat**	Con transiciones
Implicit Termination	Actividad de tipo rlow	Actividad sin transiciones de salida
Multiple Instances (MI) Without Synchronization	Actividad de tipo **Invoke** en un loop **while**	Con Arbitrary cycles se lo puede implementar.
MI With Design Time Knowledge	Ejecutar cada instancia en una actividad separada en una actividad de tipo **flow**	Replicar la actividad tantas veces como sea necesaria su ejecución. En paralelo.
MI With Runtime Knowledge	No soportado	No soportado
MI Without Runtime Knowledge	No soportado	No soportado
Deferred Choice	Actividad de tipo **pick**	No soportado
Interleaved Parallel Routing	Múltiples **scopes** sin un **flow** que compite por una variable compartida.	No soportado
Milestone	Un pool para hitos en un loop **while**	No soportado
Cancel Activity	Actividad de tipo F**ault**	No soportado
Cancel Case	Actividad de tipo **Terminate**	No soportado

Tabla 1 - Implementación de Patrones de Procesos

Evaluación de Tecnologías de Procesos de Negocio centrado en motores BPEL

Anexo 2 - Evaluaciones

Año 2010

Autores: Daniel Bonini, Federico Parins

Tutor: Andrea Delgado
Co-Tutor: Pablo Miranda

1. Evaluaciones

1.1. jBPM BPEL

1.1.1. Criterios técnicos

Software

- jBoss 4.2.3 GA
- JDK 1.5 o superior
- Ant 1.8.1 (para instalación)
- Eclipse Helios (opcional)
- Eclipse BPEL Designer (opcional)

Hardware

- Procesador 1GHz, single-core (mínimo)
- Memoria 1 GB
- Disco 500 MB libres

1.1.2. Criterios funcionales (escala 0 – No soportado, 1 – Soportado parcial y 2 – Soportado)

Facilidad de instalación: *1 – Soportado parcial*

No presenta un wizard para su instalación, pero se tiene la ayuda de archivos Ant que permiten automatizar parte de la misma. También se deben hacer cambios en el archivo build.xml que utiliza el Ant indicando ruta de instalación del jBoss. Cabe aclarar que la documentación sobre el aspecto de la instalación es poco clara y requiere mucho conocimiento sobre las tecnologías Java, jBoss y Ant. Este último también es necesario para automatización de deploy de procesos.

Asistencia en el deploy: *2 – Soportado*

jBoss BPEL posee una vista de administración en la cual se puede hacer el deploy de un proceso. La misma se encuentra en el entorno de la administración de jBoss que se tenga instalado y ejecutando. La url es: http://<ruta_jboss:puerto>/jbpm-bpel/processes.jsp. También es posible automatizar el deploy mediante la herramienta Ant, pero en este caso es necesario generar los archivos build.xml y configuraciones necesarias para su ejecución.

A continuación, en la Figura 1, se muestra la pantalla de deploy para jBPM BPEL:

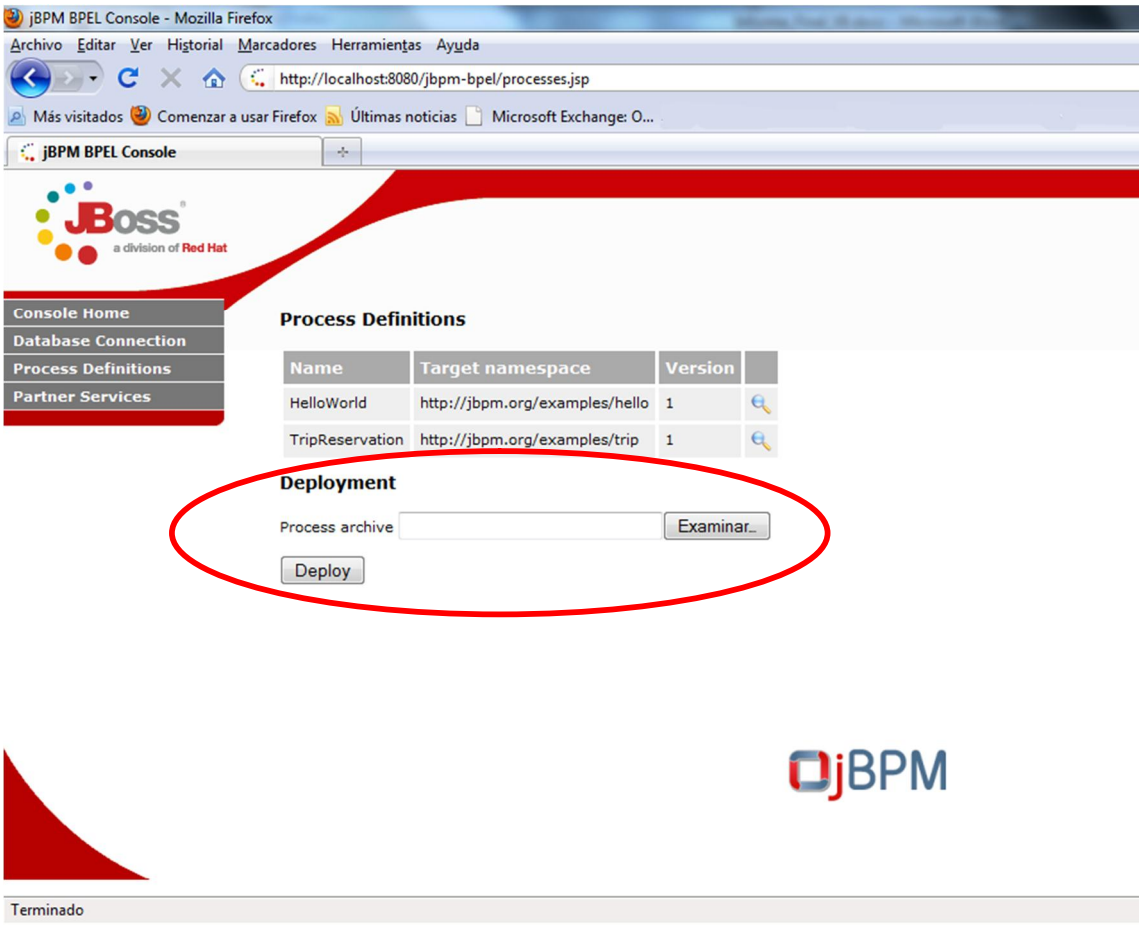

Figura 1 - Deploy de un proceso en jBPM BPEL

Ejecución de múltiples versiones de un proceso: *2 – Soportado*

Permite la ejecución de múltiples versiones de un proceso, siempre y cuando se haga un nuevo deploy del mismo. Es decir, deben coexistir las distintas versiones en la carpeta de deploy del jBoss. La baja de una versión debe hacerse manualmente.

La versión de cada proceso de puede observar en la vista de administración de los procesos, en la url que se indica en la evaluación del criterio anterior (Asistencia en el deploy).

Tolerancia a fallas: *1 – Soportado parcial*

No presenta ningún sistema de tolerancia a fallas nativo. Sin embargo, se puede integrar con herramientas de manejo de errores que existen para jBoss. La otra opción es el manejo de errores dentro de los proceso a través de los FaultHandlers. El log de errores es el mismo que utiliza el jBoss.

Suspender o hacer rollback de un proceso: *0 – No soportado*

No presenta ningún mecanismo automático o amigable, propio de jBPM BPEL, para realizar ninguna de esas 2 funciones, sino que el proceso debe ser modificado para permitir dicha funcionalidad (o parte de ella).

La suspensión se puede llegar a implementar mediante la inclusión de timers en el proceso, mientras que el rollback con transiciones hacia atrás. Esto hace que dichas funcionalidades sean muy complejas de implementar, y en el caso del rollback, implementar parcialmente debido a que si se ejecuta algún método que no permita su vuelta hacia atrás, el mismo se efectuará de forma incompleta.

Cantidad de procesos ejecutándose: *2 – Soportado*

No existe una limitante en la cantidad de procesos ejecutándose. Lo que se debe tener en cuenta es que debido a que jBPM BPEL se ejecuta en el contexto de jBoss, la cantidad de accesos puede estar limitada según se haya configurado en el propio jBoss. Cambiando este parámetro se cambia también la cantidad de accesos que puede tener jBPM BPEL.

Si bien se pueden tener todos los procesos que se deseen ejecutando, no necesariamente se pueden acceder a todos, esto es lo que limita la configuración de jBoss.

Cantidad de instancias de un proceso ejecutándose: *2 – Soportado*

Similar comentario que para el criterio anterior, si bien se puede acceder a la cantidad que se desee de procesos (o en este caso, la cantidad de instancias de un mismo proceso), se está limitado por la configuración de jBoss. Cabe destacar que esta configuración es global a todo el servidor.

Soporte para grandes cantidades de usuarios: *2 – Soportado*

No se ha detectado ninguna limitante propia del motor al respecto. Sin embargo, jBPM BPEL se ejecuta en el entorno de jBoss, lo cual hace que herede la configuración del servidor. De esta manera podría estar sujeto a limitantes del servidor. Esta configuración se puede realizar en el archivo *login-config.xml* del servidor de jBoss.

Auditoria: *1 – Soportado parcial*

No posee una herramienta integrada para la auditoria de los procesos donde se pueda visualizar dicha información. Sin embargo se pueden obtener dichos datos desde la base de datos donde se persisten los procesos o mismo del log del jBoss e implementar alguna herramienta de auditoría. La obtención de los datos provenientes del log es tarea compleja, debido a que éste es compartido por todo el servidor del jBoss.

Reportes: *0 – No soportado*

No posee herramientas para generar reportes de procesos ni del estado general del servidor.

Integración con distintas bases de datos: *2 – Soportado*

Brinda soporte a integración con diferentes bases de datos. Esto se debe a que hereda el sistema del servidor del jBoss, utilizando los archivos datasources. Para ello es necesario modificar el archivo *jbpm-ds.xml* (que se encuentra en la carpeta deploy del servidor de jBoss), indicando en él los datos necesarios para la conexión a la base que se desee (connection string, usuario y contraseña de la base, etc.). También se debe disponer del driver de jBoss para esa base, en la carpeta *lib* del servidor.

Obtención de indicadores de ejecución como cantidad de procesos finalizados por unidad de tiempo, etc.: *2 – Soportado*

Si bien jBPM BPEL no brinda una herramienta propia para obtención de indicadores, presenta la información necesaria de toda la ejecución de los procesos almacenada a través de la información de log de su base de datos.

El motor dispone de la información de cada proceso del sistema que haya sido ejecutada discriminando por actividad ejecutada, indicando el momento de ejecución, finalización (también brinda la duración de la misma), además de un campo dedicado para excepciones en caso de que ocurran.

La misma se encuentra disponible en la tabla JBPM_LOG, donde, además de la información anteriormente descripta, se almacena el mensaje intercambiado, los nodos de origen y destino de la transición, la instancia de la tarea ejecutada, los actores anterior y nuevo de la tarea y el pool.

JBPM_LOG: Los campos antes mencionados y otros de interés, se muestran a continuación en la Tabla 1:

Nombre	Descripción
ID_	Identificador numérico de la instancia
CLASS_	Nombre de la clase que fue ejecutada
DATE_	Timestamp del momento de la ejecución
MESSAGE_	String conteniendo el mensaje intercambiado en la ejecución
EXCEPTION	String conteniendo el stacktrace completo en caso de excepción
NODE_	Identificador del nodo ejecutado
ENTER_	Timestamp del comienzo de la ejecución
LEAVE_	Timestamp de la finalización de la ejecución
DURATION_	Tiempo de ejecución
TASKINSTANCE_	Identificador de la tarea ejecutada
TASKACTORID_	String con el nombre del actor actual
TASKOLDACTORID_	String con el nombre del actor que ejecutó la tarea
SWIMLANEINSTANCE_	Identificador del pool

Tabla 1 - Tabla de log en jBPM BPEL

Interoperabilidad: *0 – No Soportado*

La interoperabilidad entre jBPM BPEL y los procesos generados por la herramienta eClarus no se ve lograda debido a que jBPM BPEL le agrega a las tags del estándar de BPEL el prefijo *bpws:*, y los archivos .bpel generados en eClarus no lo poseen, impidiendo esto, directamente el deploy de los procesos. Algunos ejemplos de las tags que requiere jBPM BPEL, comparadas con las que genera eClarus son:

<bpws:process></bpws:process> mientras que en eClarus seria *<process></process>*

<bpws:partnerLink/> mientras que en eClarus *<partnerLink/>*

Otra particularidad que posee jBPM BPEL, es que acepta procesos con otros elementos además de los definidos en el estándar. Estos elementos se deben encontrar en un archivo con extensión .bpelex y son propios del motor.

Para solucionar dicho esta diferencia de formatos de los archivos, se agregaron los prefijos a cada tag del proceso BPEL, y también las siguientes referencias a los mismos:

xmlns:bpws="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
xmlns:jbpm="urn:jbpm.org:bpel-1.1"

Cabe aclarar que no se detectaron cambios necesarios en los archivos WSDL de los servicios que usan los procesos y que el efecto de deploy fue el mismo tanto realizándolo vía interfaz grafica (ver Figura 1), como vía Ant, creando un archivo *build.xml*.

Los errores que se generan son los siguientes:
ERROR [[deploymentServlet]] Servlet.service() para servlet deploymentServlet lanz¾ excepci¾n
at org.jbpm.bpel.web.DeploymentServlet.parseProcessArchive

1.1.3. Criterios funcionales (escala Si/No)

Importación de archivos BPEL: *Si*
Permite la importación de los motores que se muestran en la Tabla 2. Cabe aclarar que si bien no provee la funcionalidad específica de Importación, la misma se puede hacer realizando el deploy del proceso proveniente de otro motor.

Motor	Detalle
jBPM BPEL	Permite la importación de sus propios archivos
Apache ODE	No permite. Apache ODE agrega características propias a los archivos .bpel
Intalio	Permite la importación sin problemas
Riftsaw	No permite. Riftsaw agrega características propias a los archivos .bpel
Petals	No permite. Petals agrega características propias a los archivos .bpel
Orchestra	Permite la importación siempre que sea en formato .zip

Tabla 2 - Importación de archivos BPEL para jBPM BPEL

Exportación de archivos BPEL: *Si*
A través de Eclipse BPEL Designer, se puede exportar el proceso y los archivos wsdl para que sean importados. El formato que utiliza para la importación es el .zip, .jar o .tar.gz. La exportación también puede ser realizada manualmente.

Importación desde herramientas de modelado: *Si*
Utilizando la herramienta Eclipse BPEL Designer, se puede importar procesos y sus respectivos archivos wsdl en los formatos .zip., .jar o .tar.gz. Los motores soportados son los mismos que se indicaron en la Tabla 2.

Usuarios: ***No***
No se dispone de una herramienta o sistema específico para el manejo de usuarios. Se puede utilizar los mecanismos de autenticación/autorización que brinda jBoss en general para todo el servidor, pero no específico para jBPM BPEL.

Perfiles: ***No***
No se dispone de una herramienta o sistema específico para el manejo de Perfiles. Similar al criterio de Usuarios.

Roles: ***No***
No se dispone de una herramienta o sistema específico para el manejo de Roles Similar al criterio de Usuarios o Perfiles.

Simulación de procesos: ***No***
No se dispone de una herramienta de simulación para procesos.

1.1.4. Resumen

A modo de resumen, se puede decir que jBPM BPEL, si bien posee un desarrollo bastante sostenido, no provee de forma amigable y sencilla, muchas características importantes que se requiere en un motor de procesos. Información de Auditoria o Indicadores, si bien son factibles de implementarse como extensión del motor, no son para nada sencillas de realizar. Esto se debe a que la información necesaria para ello, se encuentra en el log del servidor jBoss, el cual puede ser compartido con otros sistemas que tenga deployados.

Cabe también acotar, que la documentación del mismo, no abarca en detalle las funcionalidades, siendo necesaria, mucha experiencia por parte de los usuarios sobre las tecnologías que usa, ya sean Java, jBoss, etc. Esta característica negativa, se ve compensada con la comunidad que tiene (foros, blogs, etc.), ya que se encuentra dentro del mundo jBoss, siendo el mismo bastante activo.

Por otro lado, las características relacionadas a la importación/exportación de procesos, son bastante aceptables, aunque presenta dificultades de integración con algunos de los motores que se probaron. Respecto a la interoperabilidad no se ha logrado el deploy de un procesos BPEL diseñado en la herramienta eClarus, debiéndose esto a diferencias que posee jBPM BPEL respecto a lo generado por la herramienta de diseño. A continuación se presenta un resumen de la evaluación en la Tabla 3.

Criterio	Resultado de la Evaluación
Facilidad de instalación	Soportado parcial
Asistencia en el deploy	Soportado
Ejecución de múltiples versiones de un proceso	Soportado
Tolerancia a fallas	Soportado parcial
Suspender o hacer rollback de un proceso	No soportado
Cantidad de procesos ejecutándose	Soportado
Cantidad de instancias de un proceso	Soportado

ejecutándose	
Soporte para grandes cantidades de usuarios	Soportado
Auditoria	Soportado parcial
Reportes	No soportado
Integración con distintas bases de datos	Soportado
Obtención de indicadores de ejecución	Soportado
Interoperabilidad	No Soportado
Importación de archivos BPEL	Si
Exportación de archivos BPEL	Si
Importación desde herramientas de modelado	Si
Usuarios	No
Perfiles	No
Roles	No
Simulación de procesos	No

Tabla 3 - Resumen de la evaluación de jBPM BPEL

1.2. Apache ODE

1.2.1. Criterios técnicos

Software

Ode 1.3.4 se distribuye en dos formatos diferentes, una distribución WAR que puede correr en cualquier contenedor J2EE como Apache Tomcat o JBoss AS y una distribución JBI que de acuerdo a la documentación disponible puede correr sobre cualquier contenedor JBI como Open ESB o Petals ESB aunque solo ha sido testeado sobre Apache ServiceMix 3.2.1.
Cualquiera de las distribuciones requiere además Java 1.5.x o superior.

Esta evaluación ha sido realizada sobre el servidor Tomcat 6.0 con una maquina virtual Java 1.6.0.12.

Hardware

No se encontraron requerimientos de hardware documentados. Las pruebas fueron realizadas sobre un procesador Intel T1350 de 1.86 GHz y 1.5 Gb de RAM.

1.2.2. Criterios funcionales (escala 0 – No soportado, 1 – Soportado parcial y 2 – Soportado)

Facilidad de instalación: *1 – Soportado parcial*

Si bien Ode no provee un instalador gráfico la instalación del motor resulta sencilla. Para poder instalar el motor en su distribución WAR se debe copiar el archivo *ode.war* contenido dentro del archivo *apache-ode-war-1.3.4.zip* (disponible en http://ode.apache.org/index.html) al directorio *webapp* de instalación de Tomcat. Una vez que se inicia el servicio Tomcat se puede comenzar a utilizar ODE.

Asistencia en el deploy: *2 – Soportado*

El deploy de los procesos puede realizarse en forma manual copiando la carpeta con los archivos necesarios. También se puede realizar a través del administrador web desde el cual se puede realizar el deploy en forma automática para lo que se utiliza un archivo con extensión zip, jar o tar.gz y que contiene los archivos necesarios para el deploy. La consola de administración se encuentra en la URL http://<servidor-de-instalacion>:<puerto-de-instlacion>/ode/deployment.html.

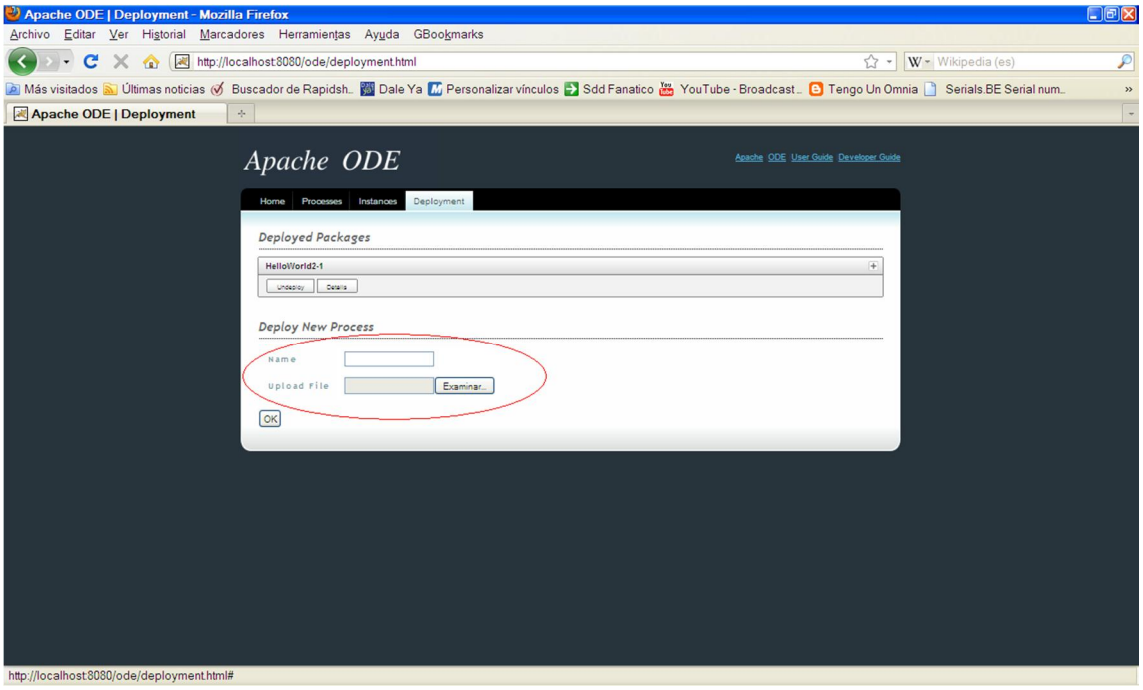

Figura 2 - Deploy de un proceso en ApacheODE

Ejecución de múltiples versiones de un proceso: *2 – Soportado*

ODE permite la ejecución de múltiples versiones de un proceso. Cada vez que se realiza un deploy de un proceso existente en el servidor, ODE incrementa la versión del proceso, marca la versión anterior como retirada y realiza el deploy con la nueva versión. Las nuevas instancias que se inicien lo harán con la nueva versión mientras que las instancias que se encontraban ejecutando previo al deploy lo harán con la versión correspondiente.

Tolerancia a fallas: *1 – Soportado parcial*

ODE no provee ningún mecanismo de tolerancia a fallas nativo. El mismo podría ser incluido mediante extensión del motor o en la definición del proceso BPEL mediante la utilización del FaultHandlers.

Suspender o hacer rollback de un proceso: *2 – Soportado*

ODE permite suspender y retomar la ejecución de las instancias de un proceso.

Cantidad de procesos ejecutándose: *2 – Soportado*

No existen limitantes impuestas por el motor en cuanto a la cantidad de procesos en ejecución. Dicha limitante surge por la memoria del sistema en la cual se encuentra instalado el servidor.

Cantidad de instancias de un proceso ejecutándose: *2 – Soportado*

No existen limitantes impuestas por el motor en cuanto a la cantidad de instancias de un proceso en ejecución. Dicha limitante surge por la memoria del sistema en la cual se encuentra instalado el servidor.

Soporte para grandes cantidades de usuarios: *0 – No Soportado*

Ode no provee un mecanismo de login ya que no existe la definición de usuarios a nivel del motor.

Auditoria: *1 – Soportado Parcial*

No posee mecanismos de auditoría más allá de los indicadores de ejecución como cantidad de procesos e instancias por proceso. También permite ver para cada proceso las versiones del mismo, fecha en que se realizó el deploy de cada una de ellas, estado de cada versión del proceso (Activa, Retirada) y para cada versión la cantidad de instancias Activas, Completadas Terminadas, con Error, Fallidas y Suspendidas.

Reportes: *0 – No soportado*

No posee herramientas para generar reportes de procesos ni del estado general del servidor.

Integración con distintas bases de datos: *2 – Soportado*

ODE se puede integrar con diferentes bases de datos relacionales como MySql y Oracle. Para ello se debe copiar el controlador jdbc correspondiente a la carpeta de librerías de Tomcat y modificar el archivo *conf/server.xml* indicando la información del driver jdbc y los parámetros de conexión a la base (string de conexión, usuario y contraseña). Por último se debe agregar un archivo de nombre *ode-axis2.properties* en la carpeta *webapps/ode/WEB-INF/conf* indicando que se debe usar una base de datos externa.

Obtención de indicadores de ejecución como cantidad de procesos finalizados por unidad de tiempo, etc.: *1 – Soportado parcial*

La consola de administración de ODE provee la siguiente información respecto al estado de ejecución de los procesos e instancias:

- Cantidad total de procesos en el sistema
- Cantidad total de instancias en el sistema
- Cantidad y % de instancias Activas, Completadas, Terminadas, con Error, Fallidas y Suspendidas en el sistema.

Respecto a la informacion de logueo de Apache ODE, se tiene que para cada instancia ejecutada por el motor, se almacena la información de su identificador, el tiempo de inicio de ejecución de la misma, el tiempo de inicio de la misma y de su finalización. En caso que la instancia haya sido suspendida y luego retomada, la información del tiempo de suspensión se almacena como el tiempo de finalización de la ejecución de la instancia. Una vez retomada, la misma se almacena como tiempo de inicio.

Apache ODE loguea información de ejecución en su base de datos en las tablas mostradas a continuación. Para ninguna de estas tablas se definen codigueras de valores.

BPEL_INSTANCE: Almacena información de cada instancia.

Nombre	Descripción
ID	Identificador numérico de la instancia
PREVIOUS_STATE	Estado anterior de la instancia (valor numérico)
PROCESS_ID	Identificador numérico del proceso al cual corresponde la instancia. Hace referencia a la tabla BPEL_PROCESS en la cual está definido el proceso al que corresponde la instancia.
STATE	Estado actual de la instancia (valor numérico)
LAST_ACTIVE_DT	Timestamp de la última vez que estuvo activa la instancia
FAILURE_COUNT	Cantidad de veces que falló la instancia
FAILURE_DT	Timestamp del último fallo
FAULT	Identificador numérico de la falla. Hace referencia a la tabla BPEL_FAULT

Tabla 4 - Tabla de instancias

BPEL_EVENT*:* Almacena información de los eventos ocurridos durante la ejecución de una instancia de un proceso.

Nombre	Descripción
ID	Identificador numérico del evento.
IID	Identificador numérico de la instancia a la que pertenece el evento. Hace referencia a la tabla BPEL_INTANCE
PID	Identificador numérico del proceso. Hace referencia a la tabla BPEL_PROCESS
TSTAMP	Timestamp en que ocurrió el evento
TYPE	Si bien este campo se llama “Tipo”, no es exactamente el tipo del evento sino el nombre del mismo. Algunos valores posibles para este campo son: ActivityEnabledEvent, ActivityExecStartEvent, ActivityFailureEvent
DETAIL	Este campo de tipo CLOB contiene información (en forma de texto) sobre: nombre del evento, tipo del evento (activityLifecycle, dataHandling, instanceLifecycle, etc.), nombre del proceso, identificador de la instancia, etc.
DATA	Campo de tipo blob el cual se utiliza para almacenar información adicional asociada al evento como puede ser datos en formato xml.

Tabla 5 - Tabla de eventos

BPEL_FAULT*:* Almacena información de las fallas que ocurren durante la ejecución de una instancia.

Nombre	Descripción
ID	Identificador numérico de la falla
FAULTNAME	Nombre de la falla
EXPLANATION	Descripción de la falla
DATA	Campo del tipo blob en que almacena información más detallada de la falla.
LINE_NUM	Indica el número de línea en que se produjo la falla.

Tabla 6 - Tabla de fallas

BPEL_MESSAGE*:* Almacena información estática de los mensajes enviados.

Nombre	Descripción
ID	Identificador numérico del mensaje
MEX	Identificador numérico que hace referencia a la tabla BPEL_MESSAGE_EXCHANGE.
TYPE	Tipo del mensaje. Formato: (<namespace>)<tipo>. Algunos ejemplos de tipo son: initProcessRequest, createTaskRequest, createTaskResponse, notifyTaskCompletionResponse, etc.
MESSAGE_DATA	Campo del tipo blob que contiene el cuerpo del mensaje.
MESSAGE_HEADER	Campo del tipo blob que contiene el cabezal del mensaje.

Tabla 7 - Tabla de mensajes

BPEL_MESSAGE_EXCHANGE*:* Almacena información de invocación sobre los mensajes intercambiados.

Nombre	Descripción
ID	Identificador numérico.
PORT_TYPE	Nombre del PortType empleado para enviar el mensaje. Formato: {<namespace>}<nombre>
OPERATION	Método invocada
STATE	Estado de la invocación del mensaje. (REQUEST, RESPONSE)
PROCESS	Identificador numérico del proceso. Hace referencia a la tabla BPEL_PROCESS
PIID	Identificador numérico de la instancia a la que corresponde el mensaje.
REQUEST	Identificador numérico que hace referencia a la tabla BPEL_MESSAGE. Indicando el mensaje de request enviado.
RESPONSE	Identificador numérico que hace referencia a la tabla BPEL_MESSAGE indicando la respuesta al mensaje {REQUEST}

Tabla 8 - Tabla de intercambio de mensajes

Interoperabilidad: *0 – No Soportado*

No fue posible realizar un deploy sobre ODE de un proceso generado por el modelador eClarus. Para probar la interoperabilidad entre el modelador y ODE se modeló en eClarus un proceso deployeado con éxito en ODE utilizando el mismo WSDL. Este proceso es un simple "HelloWorld" con un evento inicial, una tarea y un evento final como se observa en la Figura 3.

Figura 3 - HolaMundo

Para el deploy se utilizó el archivo BPEL generado por eClarus sin modificar, el WSDL y el deploy.xml del proceso ya deployeado en ODE obteniendo como resultado el error de la Figura 4.

Al revisar el archivo generado por eClarus se observaron algunas diferencias respecto de los procesos en ODE. Todas las tags tenían el prefijo *"bpel"*, el encabezado es igual salvo el prefijo en el tag *process* y la inclusión del namespace XML "*xmlns:bpel="http://schemas.xmlsoap.org/ws/2003/03/business-process/".* Además faltaba importar el WSDL del servicio como lo hacen todos los archivos BPEL en ODE.

En primer lugar se intentó agregar el import del WSDL (como muestra el siguiente fragmento) sin modificar el cabezal ni los tags del BPEL.

```
<import location="HelloWorld2.wsdl"
     namespace="http://ode/bpel/unit-test.wsdl"
     importType="http://schemas.xmlsoap.org/wsdl/" />
```

A pesar de importar el archivo .wsdl en el archivo .bpel el motor continuaba dando el mismo error al momento de realizar el deploy.

Luego se intentó modificar el archivo BPEL quitando los prefijos a las tags y quitando la inclusión del xmlns:bpel pero aún así el motor continuó dando el mismo error.

Dado que eClarus permite generar un modelo a partir de un archivo .bpel, se intentó crear un modelo a partir del proceso deployeado en ODE y provisto como parte de los ejemplos del motor. Una vez creado se transformó el modelo BPMN a BPEL a través de la funcionalidad provista por el modelador. Se intentó realizar el deploy de este proceso pero se obtuvo el mismo error que el de la Figura 4.

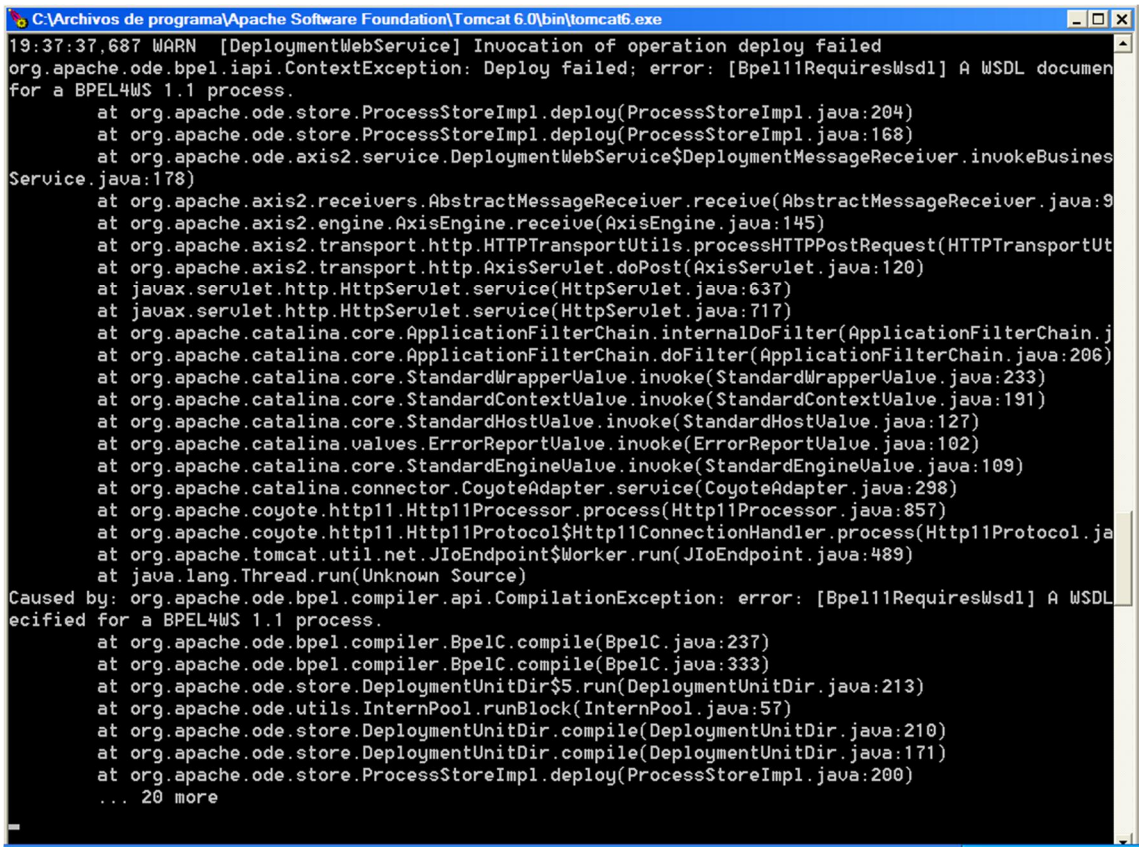

Figura 4 - Error deploy ODE

1.2.3. Criterios funcionales (escala Si/No)

Importación de archivos BPEL: *No*
No es posible importar archivos BPEL desde otros motores.

Exportación de archivos BPEL: *No*
ODE no provee ningún mecanismo para exportar archivos BPEL.

Importación desde herramientas de modelado: *No*
No es posible importar procesos BPEL desde herramientas de modelado.

Usuarios: *No*
No provee mecanismos para la administración de usuarios.

Perfiles: *No*
No provee mecanismos para la administración de perfiles.

Roles: *No*
No provee mecanismos para la administración de roles.

Simulación de procesos: *No*
No se dispone de una herramienta de simulación para procesos.

1.2.4. Resumen

A modo de resumen, se puede decir que si bien Apache ODE posee algunas características buenas, como la obtención de algunos indicadores o la asistencia en el deploy, no brinda interoperabilidad con otros motores, cualidad de vital importancia. A continuación se muestra en la Tabla 9, el resumen de la evaluación.

Criterio	Resultado de la Evaluación
Facilidad de instalación	Soportado parcial
Asistencia en el deploy	Soportado
Ejecución de múltiples versiones de un proceso	Soportado
Tolerancia a fallas	Soportado parcial
Suspender o hacer rollback de un proceso	No soportado
Cantidad de procesos ejecutándose	Soportado
Cantidad de instancias de un proceso ejecutándose	Soportado
Soporte para grandes cantidades de usuarios	No Soportado
Auditoria	No Soportado
Reportes	No soportado
Integración con distintas bases de datos	Soportado
Obtención de indicadores de ejecución	Soportado parcial
Interoperabilidad	
Importación de archivos BPEL	No
Exportación de archivos BPEL	No
Importación desde herramientas de modelado	No
Usuarios	No
Perfiles	No
Roles	No
Simulación de procesos	No

Tabla 9 - Resumen de la evaluación de Apache ODE

1.3. Intalio | Community Edition

1.3.1. Criterios técnicos

Software

- Intalio|Server
- JDK 1.5 o superior
- Intalio|Designer

Hardware

- Memoria: 512 Mb mínimo, 1 Gb recomendado
- Disco: 300 Mb de espacio mínimo

1.3.2. Criterios funcionales (escala 0 – No soportado, 1 – Soportado parcial y 2 – Soportado)

Facilidad de instalación: *2 – Soportado*

La instalación de Intalio|Server si bien no posee un wizard es realmente sencilla de realizar ya que solo requiere descomprimir un archivo. El modelador Intalio|Designer presenta un wizard para su instalación. Tanto el servidor como el modelador al ser instalados pueden comenzar a ser usados sin necesidad de configuraciones adicionales. En caso de querer utilizar el servidor en un puerto diferente al 8080 o utilizar https en lugar de http se debe ejecutar un script ubicado en *<ruta_servidor>/extras.*

Asistencia en el deploy: *2 – Soportado*

El deploy de los procesos puede ser realizado desde la consola web del servidor disponible en la URL http://<nombre-servidor>:<puerto>/bpms-console así como también desde el modelador. Como se observa en la Figura 5, para realizar el deploy desde el servidor solo es necesario proveer la ruta en la que se encuentra el archivo *deploy.xml* (generado automáticamente por el diseñador) el cual contiene la información necesaria para realizar el deploy. En la Figura 6 se muestra el asistente para el deploy provisto por el diseñador. En este podemos comprobar la conexión con el servidor e incluso modificar la URL del mismo. También podemos seleccionar los archivos que queremos hacer el deploy.

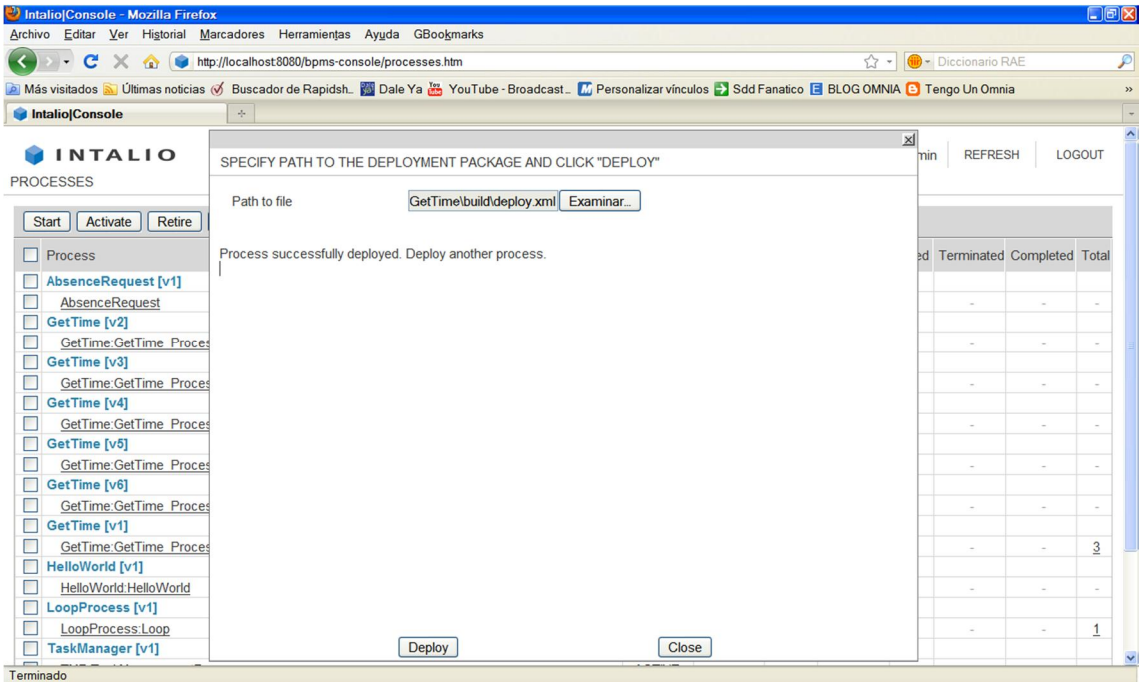

Figura 5 - Deploy de procesos desde Intalio|Server

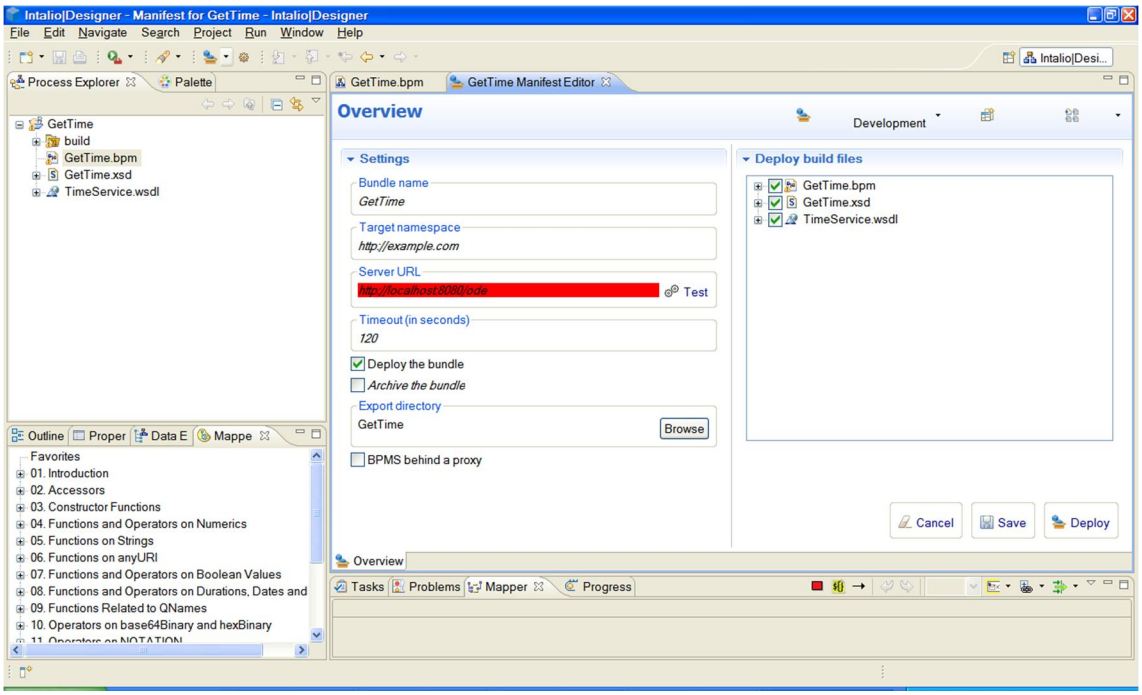

Figura 6 - Deploy de procesos desde Intalio|Designer

Ejecución de múltiples versiones de un proceso: *2 – Soportado*

Intalio|Server soporta el versionado (automático) de procesos a medida que los mismos son deployeados en el servidor. Las nuevas instancias serán ejecutadas con la nueva versión del proceso mientras que las instancias que se encontraban en ejecución continuaran ejecutando la versión anterior. Como podemos observar en la Figura 7 la versión anterior del proceso se marca como *RETIRED* mientras que la nueva es marcada como *ACTIVE*.

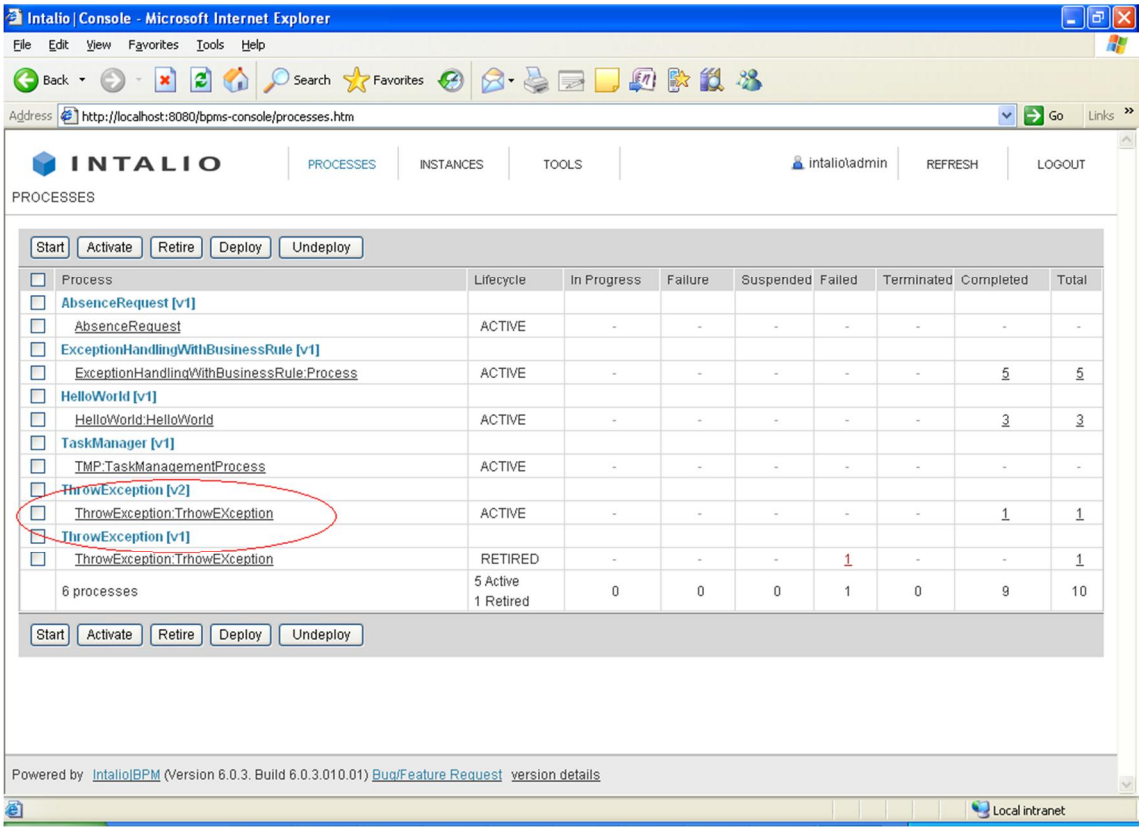

Figura 7 - Versionado de procesos

Tolerancia a fallas: *1 – Soportado parcial*

Intalio no provee ningún mecanismo de tolerancia a fallas nativo. Para incorporar un mecanismo de tolerancia a fallas se debe extender el motor o incluirlo en la definición del proceso bpel mediante la utilización de FaultHandlers.

Suspender o hacer rollback de un proceso: *1 – Soportado parcial*

Desde la consola de administración se puede suspender la ejecución de una instancia que se encuentra en ejecución, retomar la ejecución de una instancia suspendida, terminar o eliminar una instancia, pero no hacer rollback de la misma. Ver Figura 8

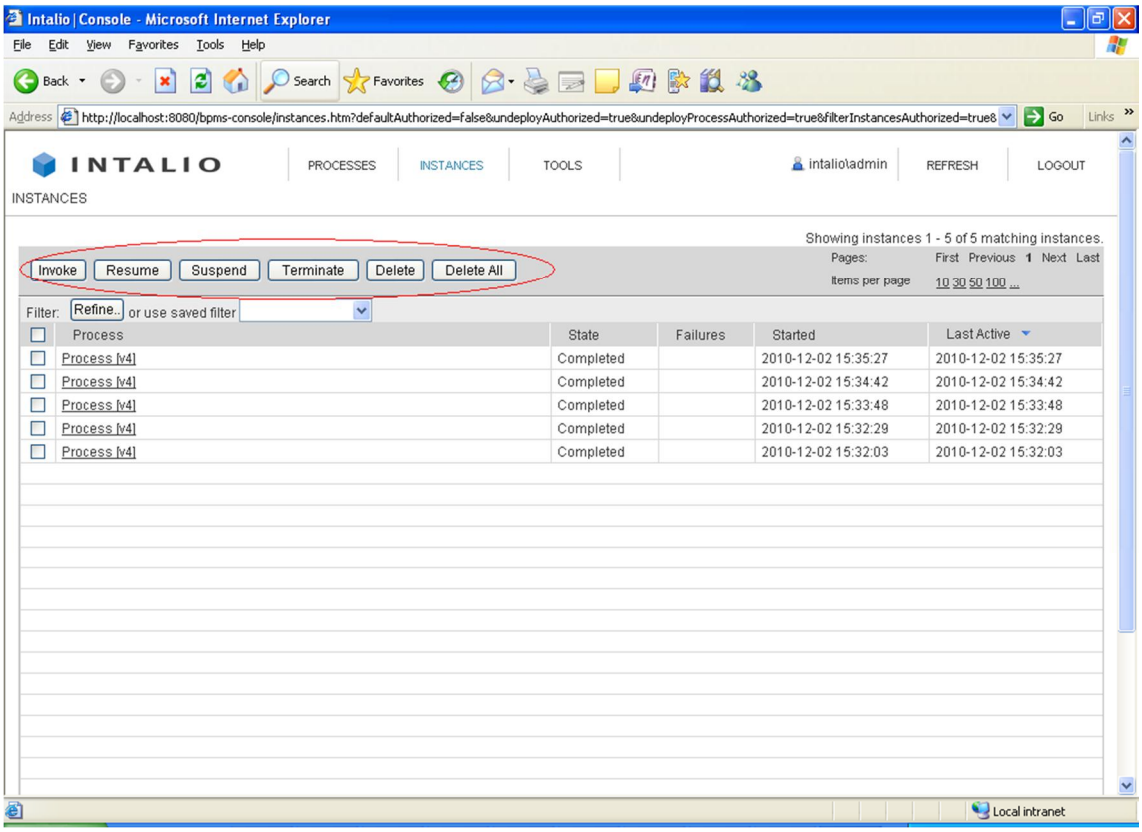

Figura 8 - Suspender procesos

Cantidad de procesos ejecutándose: *2 – Soportado*

No existe una limitante (más allá de la configuración de Hardware del sistema) en cuanto a la cantidad de procesos en ejecución en el servidor.

Cantidad de instancias de un proceso ejecutándose: *2 – Soportado*

Al igual que en la cantidad de procesos no existe una limitante impuesta por el motor respecto a la cantidad de instancias.

Soporte para grandes cantidades de usuarios: *2 – Soportado*

Debido a que Intalio puede ser configurado para trabajar con un servidor LDAP no existen limitantes en cuanto a la cantidad de usuarios configurables en el sistema.

Auditoria: *2 – Soportado*

Intalio permite obtener información de Auditoria respecto a cada instancia de un proceso. En el mismo se puede observar el evento desencadenado y la hora que fue realizado, así como también si tuvo éxito o no. En la Figura 9 se muestra como el administrador indica la información. Cabe acotar que también se puede visualizar el diagrama del proceso en esta funcionalidad.

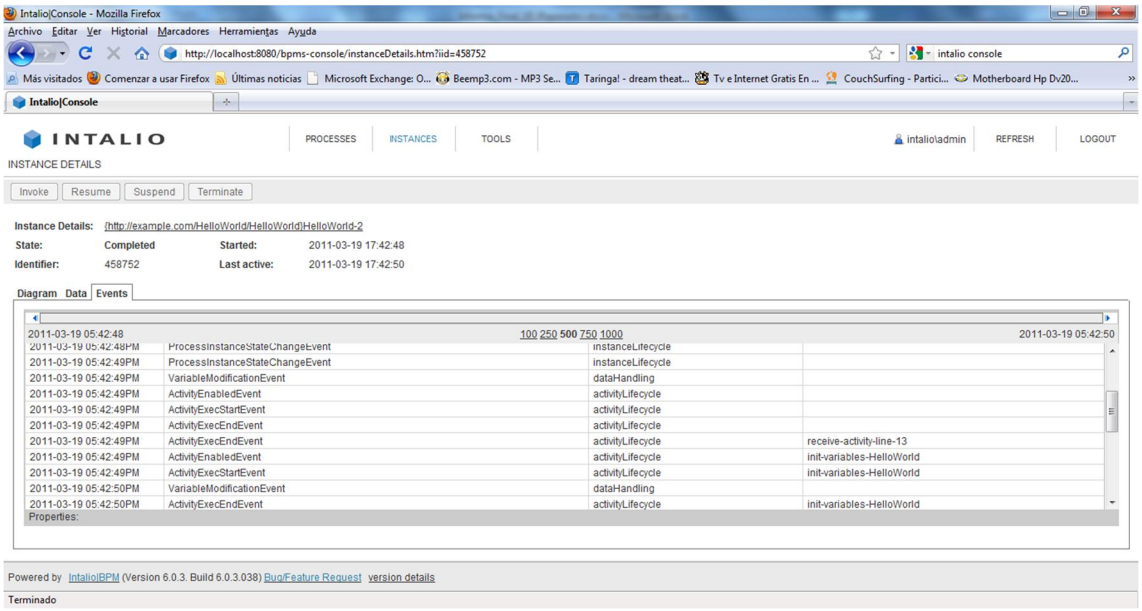

Figura 9 - Auditoria en Intalio

Reportes: *0 – No soportado*

Intalio no posee herramientas para generar reportes de procesos ni del estado general del servidor.

Integración con distintas bases de datos: *2 – Soportado*

Intalio soporta el uso de diferentes bases de datos como Derby, MySQL, SQL Server, Oracle y PostgreSQL. Para ello se debe copiar el controlador JDBC correspondiente a la carpeta *<ruta_servidor >/common/lib*. Luego se debe modificar el archivo *<ruta_servidor >/conf/resources.properties* para configurar el driver y los parámetros de conexión (url de la base, usuario, contraseña, etc.)

Obtención de indicadores de ejecución como cantidad de procesos finalizados por unidad de tiempo, etc.: *2 – Soportado*

Como podemos observar en la Figura 7 la consola de administración del servidor provee información sobre la cantidad de procesos e instancias en el sistema: procesos activos y retirados e instancias por cada proceso (desglosada por estado – *In-Progress, Failure, Suspended, Failed, Terminated* y *Completed*). Esta información también se encuentra totalizada a nivel del servidor.

La información de logueo de las ejecuciones, se almacenan a nivel de base de datos. En este caso, la información que se tiene a disposición es el identificador de cada instancia del proceso que fue ejecutada, el tiempo de ejecución de la misma, el tipo (si se inicio el proceso, o actividad, o si fue una finalización) y un String con detalles de la misma. En el mismo, se dispone de información como el nombre de la clase ejecutada, el proceso al cual está asociado, entre otra información interna del motor.

Debido a que Intalio|BPMS Server está basado en el motor Apache ODE el esquema de base de datos que utiliza es similar. Las tablas que utiliza para loguear información de ejecución de las instancias son las mismas que las presentadas en las tablas 4 a 8. La diferencia es que Intalio incorpora una nueva tabla denominada LARGE_DATA con los campos ID (identificador numérico del registro) y BIN_DATA (campo blob para almacenar información de cualquier tamaño y tipo en formato binario). Luego sustituye en las tablas correspondientes los campos BLOB por valores numérico como claves foranes a esta tabla. De esta forma la tabla de mensajes mostrada en la Tabla 7, los campos correspondientes MESSAGE_DATA y MESSAGE_HEADER de tipo BLOB se sustituyen por dos campos numéricos y la información correspondiente se almacena en la tabla LARGE_DATA.

Interoperabilidad: *0 – No soportado*

No se consiguió realizar con éxito el deploy en Intalio de un proceso modelado con eClarus. Para la prueba de interoperabilidad se utilizó el mismo proceso *HelloWorld* empleado con ODE, deployeando para ello el archivo zip conteniendo: el archivo BPEL, el WSDL del servicio utilizado y el deploy.xml. En este caso el motor indica que se realizó el deploy con éxito y no se reportan errores ni en el administrador web ni en el log del servidor. Sin embargo el proceso no aparece en la lista de procesos deployeados del administrador. Al revisar en la estructura de archivos del servidor bajo la ruta "*%INTALIO_SERVER%**var**deploy"* se encuentra una carpeta *"HelloWorld"* con los archivos correspondientes al proceso.

Al consultar la base de datos del motor se detectó que el deploy no había sido completado ya que si bien se encontraba un registro en la tabla *"deploy_assemblies"* no se habían grabado el resto de las tablas para completar el deploy como por ejemplo: *deploy_components, deploy_resources, store_du, store_process, etc.*

Luego del primer intento se modificó la estructura del archivo zip y se colocaron los archivos dentro de una carpeta con el nombre "*processess.ode*" y se intentó nuevamente realizar el deploy. Esta vez se obtuvo el mismo error que en ODE como se muestra en la Figura 10.

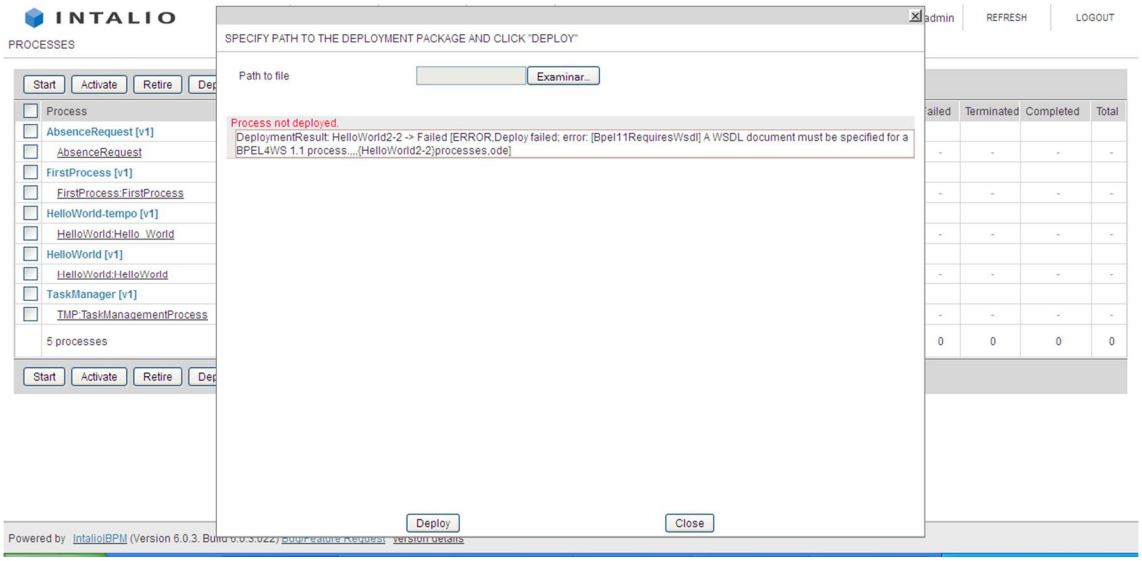

Figura 10 - Deploy HelloWorld en Intalio

En este caso el archivo BPEL contenía el siguiente tag import para incluir una referencia al WSDL del Web-Service.

```
<import location="HelloWorld2.wsdl"
        namespace="http://ode/bpel/unit-test.wsdl"
        importType="http://schemas.xmlsoap.org/wsdl/" />
```

1.3.3. Criterios funcionales (escala Si/No)

Importación de archivos BPEL: *Si*

Permite la importación de procesos de los motores que se muestran en la Tabla 10. Cabe aclarar que si bien no provee la funcionalidad especifica de Importación, la misma se puede hacer realizando el deploy del proceso proveniente de otro motor.

Motor	Detalle
jBPM BPEL	Permite la importación sin problemas
Apache ODE	No permite
Intalio	Permite la importación de sus propios archivos
Riftsaw	Permite la importación sin problemas
Petals	No permite. Petals agrega características propias a los archivos .bpel
Orchestra	Permite la importación siempre que sea en formato .zip

Tabla 10 - Importación de archivos BPEL para Intalio

Exportación de archivos BPEL: *No*

Intalio no presenta una herramienta o funcionalidad que permita la exportación de sus procesos. Esto debe hacerse manualmente con los archivos BPEL y WSDL que los demás motores requieran.

Importación desde herramientas de modelado: *Si*
La herramienta de modelado que provee Intalio, permite realizar la importación de los mismos motores presentados en la Tabla 10.

Usuarios: *Si*
En Intalio los usuarios se pueden manejar de dos formas diferentes, mediante un archivo de configuración, o mediante LDAP. Para definir mediante archivo los usuarios y sus contraseñas (sin encriptar) se debe modificar el archivo *<ruta_servidor >/var/config/security.xml*. Esta opción puede ser útil para ambientes de desarrollo pero no así para ambientes de producción debido a su falta de seguridad. Se puede configurar Intalio para trabajar con un servidor LDAP modificando el archivo *<ruta_servidor >/var/config/securityConfig.xml*. Mediante el archivo *<ruta_servidor >/var/config/ldap.properties* se configura la dirección y otros parámetros del servidor LDAP.

Perfiles: *No*
No se dispone de una herramienta o sistema específico para el manejo de Perfiles.

Roles: *Si*
Los roles pueden definirse en el archivo de configuración *<server_home>/var/config/security.xml* y asignarse a los usuarios. Se puede definir jerarquías de roles permitiendo así por ejemplo que el rol A descienda del rol B.

Simulación de procesos: *No*
No se dispone de una herramienta de simulación para procesos.

1.3.4. Resumen
Sin duda alguna, se puede decir que Intalio es uno de los motores de ejecución de procesos, más completos. Soporta la mayoría de las características deseables e importantes de motores como auditoria, obtención de indicadores y lo referente a la interoperabilidad con otros motores.

Otras características positivas, son las relativas a la documentación y la comunidad, siendo ambos, de gran utilidad debido a su gran actividad y rápida respuesta.

A continuación se presenta en la Tabla 11, el resumen de la evaluación.

Criterio	Resultado de la Evaluación
Facilidad de instalación	Soportado
Asistencia en el deploy	Soportado
Ejecución de múltiples versiones de un proceso	Soportado
Tolerancia a fallas	Soportado parcial
Suspender o hacer rollback de un proceso	Soportado parcial
Cantidad de procesos ejecutándose	Soportado
Cantidad de instancias de un proceso ejecutándose	Soportado
Soporte para grandes cantidades de usuarios	Soportado

Auditoria	Soportado
Reportes	No soportado
Integración con distintas bases de datos	Soportado
Obtención de indicadores de ejecución	Soportado
Interoperabilidad	No Soportado
Importación de archivos BPEL	Si
Exportación de archivos BPEL	No
Importación desde herramientas de modelado	Si
Usuarios	Si
Perfiles	No
Roles	Si
Simulación de procesos	No

Tabla 11 - Resumen de la evaluación de Intalio

1.4. Riftsaw

1.4.1. Criterios técnicos

Software

- jBoss AS 5.0 o superior
- jBoss ESB 4.8 o superior
- JDK 1.6 o superior
- Ant 1.8.1 (para instalación)
- Eclipse Helios (opcional)
- Eclipse BPEL Designer (opcional)

Hardware

- Procesador 1.8 GHz
- Memoria 1.5 GB

1.4.2. Criterios funcionales (escala 0 – No soportado, 1 – Soportado parcial y 2 – Soportado)

Facilidad de instalación: *1 – Soportado parcial*

No presenta un wizard para su instalación, pero se tiene la ayuda de archivos Ant que permiten automatizar parte de la misma. También se deben hacer cambios en el archivo deployment.properties que utiliza el Ant indicando ruta de instalación del jBoss AS y también del jBoss ESB.

Asistencia en el deploy: *0 – No Soportado*

Riftsaw no presenta un componente que permita realizar el deploy de un proceso de forma asistida. El deploy puede realizarse vía Ant, generando el archivo *build.xml* con la información necesaria, o también con la ayuda del Eclipse BPEL Designer (aunque este último requiere tener el Eclipse instalado).

Ejecución de múltiples versiones de un proceso: 1 – *Soportado parcial*

El motor permite el deploy de varias versiones de un mismo proceso. El comportamiento por defecto es detener la ejecución de las versiones anteriores y activar la nueva, esto siempre que no existan instancias de ese proceso ejecutándose. En dicho caso, se espera la finalización de las mismas, para luego detenerlas. Cabe aclarar que manualmente se puede cambiar el proceso activo nuevamente, si es necesario. En la Figura 11 se muestra un ejemplo de este comportamiento.

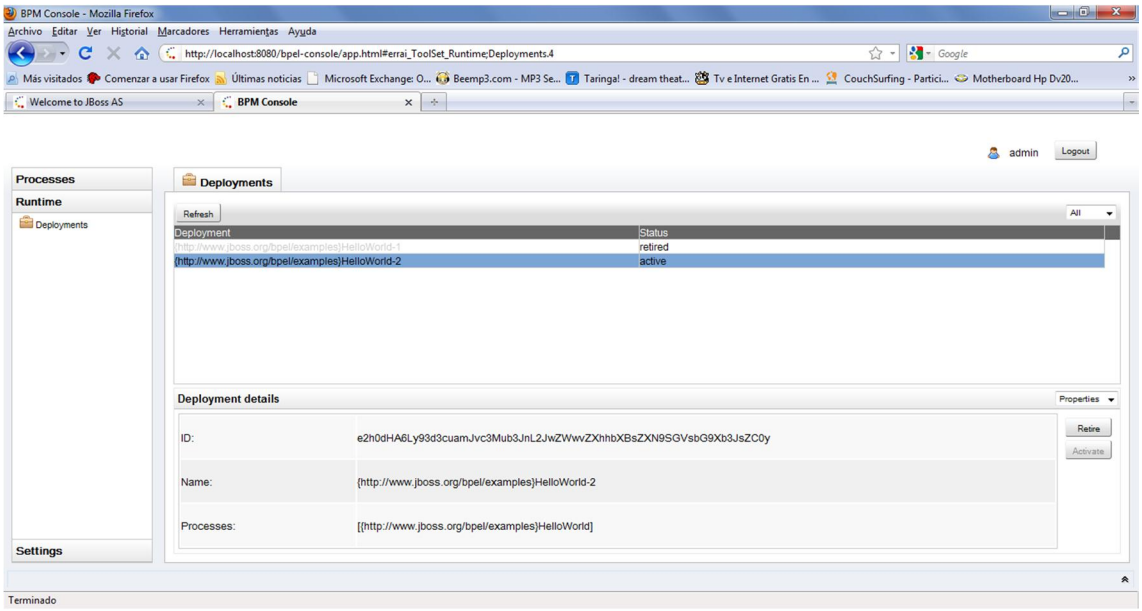

Figura 11 - Versionado de procesos

Como se mencionaba en el criterio anterior (Asistencia en el deploy), el deploy de los procesos debe hacerse mediante Ant, con lo cual, para indicar la nueva versión del proceso se debe ejecutar como mínimo la siguiente instrucción: *ant –Dversion=<nro_version> deploy* dentro de la carpeta donde se encuentra el proyecto con el proceso y el archivo *build.xml*.

Tolerancia a fallas: *1 – Soportado parcial*

Riftsaw utiliza como soporte a fallas, un sistema heredado por jBossESB, en el cual a través del archivo *jboss-esb.xml*, se configura una acción en caso de alguna falla en la invocación de los servicios. Configurando adecuadamente dicho archivo, se puede lograr un mejor control de las fallas, con implementaciones propias. A continuación se muestra un ejemplo de una acción:

```
<action name="<nombre_accion>" class="org.jboss.soa.esb.actions.bpel.BPELInvoke">
   <property name="service" value="{http://example.com/wsdl/}Service"/>
   <property name="operation" value="request" />
   <property name="abortOnFault" value="true" />
</action>
```

Suspender o hacer rollback de un proceso: *0 – No soportado*

No presenta ningún mecanismo automático o amigable para realizar ninguna de esas 2 funciones, sino que el proceso debe ser modificado para permitir dicha funcionalidad (o parte de ella). Las únicas opciones que se tienen son las de detener (Retire) y activar los procesos (Active).

Al igual que en jBPM BPEL, la suspensión se puede llegar a implementar mediante la inclusión de timers en el proceso, mientras que el rollback con transiciones hacia atrás. Esto hace que dichas funcionalidades sean muy complejas de implementar, y en el caso del rollback, implementar parcialmente, debido a que si se ejecuta algún método que no permita su vuelta hacia atrás, el mismo se efectuará de forma incompleta.

Cantidad de procesos ejecutándose: *2 – Soportado*
No existe una limitante en la cantidad de procesos ejecutándose.

Cantidad de instancias de un proceso ejecutándose: *2 – Soportado*
No existe una limitante a la cantidad de instancias de un proceso ejecutándose provista por el motor. Sin embargo, y debido a que Riftsaw se ejecuta en el entorno de jBoss, se puede estar limitado por la configuración del mismo, o también por memoria del Servidor.

Esta configuración se realiza en el archivo *jboss-service.xml*, de la carpeta *conf* del servidor de jBoss, donde se puede indicar el máximo de sesiones que soporta ejecutándose. Esta variable limitaría también la cantidad de instancias ejecutándose.

Soporte para grandes cantidades de usuarios: *2 – Soportado*
No se ha detectado ninguna limitante propia del motor al respecto. Sin embargo, Riftsaw se ejecuta en el entorno de jBoss, lo cual hace que herede la configuración del servidor. De esta manera podría estar sujeto a limitantes del servidor. Esta configuración se puede realizar en el archivo *login-config.xml* del servidor de jBoss.

Auditoria: *1 – Soportado parcial*
No posee una herramienta o componente incorporado que permita realizar la tarea de auditoría. También en este caso se podría llegar a implementarlo mediante la utilización del log que generan los procesos. Cabe aclarar que dicho log es compartido por todo el servidor jBoss.

Reportes: *0 – No soportado*
No posee herramientas para generar reportes de procesos ni del estado general del servidor.

Integración con distintas bases de datos: *2 – Soportado*
Brinda soporte a integración con los siguientes manejadores de bases de datos:

- IBM DB2
- Postgres
- SQL Server
- Oracle
- MySQL
- Apache Derby

En la carpeta */datasources*, que se encuentra en la raíz del paquete de Riftsaw se encuentran los archivos datasources con las configuraciones para cada uno de los manejadores. Para cada uno es necesario indicar los valores que se solicitan para la conexión a la base de datos.

Luego, en la carpeta */install*, que también se encuentra en la raíz del paquete de Riftsaw, se indica en el archivo *deployment.properties*, cuál de éstas se va a utilizar.

Obtención de indicadores de ejecución como cantidad de procesos finalizados por unidad de tiempo, etc.: *1 – Soportado parcial*

Como indicadores de ejecución, permite a través de la opción Execution History, ver un historial de un determinado proceso, indicando por día la cantidad de instancias que se ejecutaron.

La información de se presenta en forma de grafica similar a la que se presenta en la Figura 12.

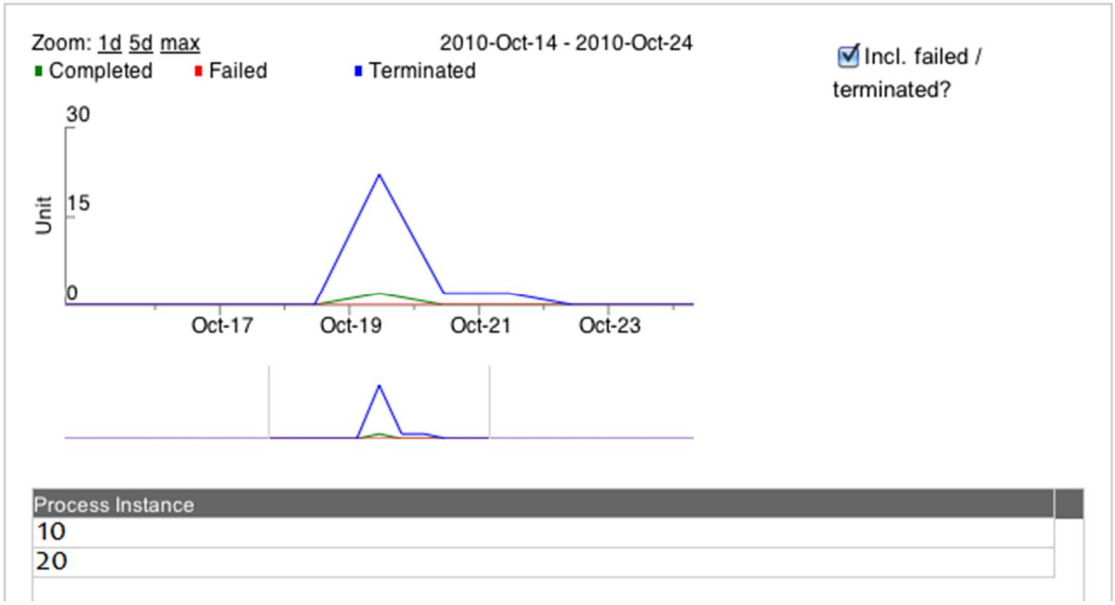

Figura 12 - Grafica con el historial de ejecuciones de un proceso

La información de logueo de ejecuciones que se almacena sobre el historial y estado de Riftsaw, son las mismas que se pueden encontrar para el motor Apache ODE, en sus tablas 4 a 8. La única diferencia detectada respecto a la implementación original de Apache ODE, es que la tabla llamada BPEL_INSTANCE, en Riftsaw pasa a llamarse BPEL_PROCESS_INSTANCE, manteniendo el mismo esquema. Esta similitud se debe a que Riftsaw ha sido desarrollado en base a ODE.

Además de lo anterior se cuenta con la tabla BPEL_ACTIVITY_RECOVERY, conteniendo la misma los identificadores de la instancia y actividad que fueron ejecutadas, así también como la acción realizada (en forma de un String) y el tiempo en el que se realizó. De esta forma se tiene el resumen de las actividades realizadas en el motor.

BPEL_ACTIVITY_RECOVERY: A continuación se muestra la tabla anteriormente descripta, con sus campos y la semántica de los mismos:

Nombre	Descripción
ID	Identificador numérico de la tupla
PROCESS_ID	Identificador numérico del proceso
ACTIVITY_ID	Identificador numérico de la actividad ejecutada del proceso
INSTANCE_ID	Identificador de la instancia ejecutada del proceso

DATE_TIME	Timestamp de la ejecución de la instancia
ACTIONS	String conteniendo el mensaje intercambiado en la ejecución. También se almacena de forma concatenada, el estado de la instancia, separado por el carácter ";" además de otra información interna del motor
DETAILS	String que contiene información sobre el nombre del evento, tipo del evento, nombre del proceso, identificador de la instancia, etc.

Tabla 12 - Tabla de resumen de ejecuciones en Riftsaw

Interoperabilidad: *0 – No Soportado*

Se ha detectado que Riftsaw no permite el deploy de procesos diseñados con la herramienta eClarus. Para detectar dicho comportamiento, se ha procedido a crear un archivo *build.xml* para el deploy vía herramienta Ant, respetando el siguiente formato:

```
<project name="<nombre_proceso>" default="deploy" basedir=".">
        <description>
                ${ant.project.name}
                ${line.separator}
        </description>

        <property file="<ruta_riftsaw>/install/deployment.properties" />
        <import file="<ruta_riftsaw>/common/base-build.xml"/>

        <property name="version" value="1" />

        <property name="sample.jar.name" value="${ant.project.name}-${version}.jar" />

        <target name="deploy">
                <echo>Deploy ${ant.project.name}</echo>
                <jar basedir="." destfile="${deploy.dir}/${sample.jar.name}" />
        </target>

        <target name="undeploy">
                <echo>Undeploy ${ant.project.name}</echo>
                <delete file="${deploy.dir}/${sample.jar.name}" />
        </target>

        <target name="<nombre_proceso>">
                <echo>Send test message to: ${ant.project.name}</echo>
                <java classname="org.apache.ode.tools.sendsoap.cline.HttpSoapSender">
                <arg value="http://<ruta_servidor>:<puerto_servidor>/<nombre_proceso>"/>
                <arg value="messages/math_${op}_request.xml"/>
                                <classpath refid="rs-exec-classpath"/>
            </java>
        </target>
</project>
```

Con el mismo el deploy del proceso no genera errores, pero el mismo no se puede observar ni acceder desde la consola web de Riftsaw.
Debido a este comportamiento, se ha detectado también que Riftsaw agrega a las tags propias del estándar BPEL, el prefijo *bpel:* Esto hace que las tags comunes del estándar no sean reconocidas por el motor y que la interoperabilidad no sea directa.

Algunos ejemplos de las tags que requiere Riftsaw, comparadas con las que genera eClarus son:

<bpel:process></bpel:process> mientras que en eClarus seria *<process></process>*

<bpel:partnerLink/> mientras que en eClarus *<partnerLink/>*

Para solucionar este comportamiento, se realizaron cambios en el archivo .bpel generado por eClarus, agregándole a las tags el prefijo anteriormente indicado, además de la siguiente referencia:

xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"

El resultado del deploy luego de este procedimiento fue el mismo que el anterior, no se generaron errores en el log del servidor, pero no se puede acceder al proceso, aunque el mismo se encontraba en la carpeta de deploy junto a los demás procesos. La única información de deploy que se cuenta a nivel de servidor es la siguiente:

Buildfile: C:\Java\BPMengines\riftsaw\interoperabilidad\build.xml
deploy:
[echo] Deploy prueba_inteorper
[jar] Building jar: C:\ Java\jboss-5.1.0.GA\server\default\deploy\ prueba_inteorper.jar
BUILD SUCCESSFUL
Total time: 0 seconds

1.4.3. Criterios funcionales (escala Si/No)

Importación de archivos BPEL: *Si*
Permite importar procesos a través de Ant (ya que no posee una interfaz grafica para ello, o mismo hacer el deploy de un proceso), de los motores que se muestran a continuación, en la Tabla 13.

Motor	Detalle
jBPM BPEL	No permite. jBPM BPEL agrega características propias a los archivos .bpel
Apache ODE	No permite
Intalio	Permite la importación sin problemas
Riftsaw	Permite la importación de sus propios archivos
Petals	No permite. Petals agrega características propias a los archivos .bpel
Orchestra	No permite. Orchestra agrega características propias a los archivos .bpel

Tabla 13 - Importación de archivos BPEL desde Riftsaw

Exportación de archivos BPEL: *No*
No es posible exportar archivos BPEL, esta acción debe ser realizada manualmente creando un archivo .zip conteniendo los archivos BPEL y WSDL necesarios.

Importación desde herramientas de modelado: *No*
No es posible importar archivos BPEL desde herramientas de modelado específicas para Riftsaw. La importación debe realizarse únicamente vía Ant, de los motores indicados en la Tabla 13, presentada anteriormente.

Usuarios: *No*
No se dispone de una herramienta o sistema específico para el manejo de usuarios. Se puede utilizar los mecanismos de autenticación/autorización que brinda jBoss en general para todo el servidor, pero no específico para Riftsaw.

Perfiles: *No*
No se dispone de una herramienta o sistema específico para el manejo de Perfiles. Similar al criterio de Usuarios.

Roles: *No*
No se dispone de una herramienta o sistema específico para el manejo de Roles Similar al criterio de Usuarios o Perfiles.

Simulación de procesos: *No*
No se dispone de una herramienta de simulación para procesos.

1.4.4. Resumen
Se puede decir que Riftsaw posee características bastante aceptables para un motor de ejecución. Sin embargo, algunas otras más deseables como auditoria u obtención de indicadores, son bastante complejas de implementar. Es de interés recalcar la complejidad que se requiere para hacer el deploy de un proceso, ya que no posee interfaz grafica amigable para ello y la misma debe hacerse a través de Ant.

Al igual que para jBPM BPEL, Riftsaw posee una comunidad bastante amplia y activa, siendo un punto importante a la hora de evacuar dudas o hacer comentarios, debido a que la documentación tampoco está del todo clara y completa.

Sin embargo, una característica importante a tener en cuenta, es que debido a que utiliza jBoss ESB, hereda los mecanismos de tolerancia a fallas que dicho servidor posee. Esta cualidad lo hace más robusto, dando mayor libertad al implementador de desarrollar algún mecanismo para el manejo de fallas.

En cuanto a la interoperabilidad, no se ha podido realizar dicha integración. Esto se debe a las características propias del motor, que impiden incluso el deploy de los procesos diseñados por la herramienta eClarus.

A continuación se muestra un resumen de la evaluación de Riftsaw, en la Tabla 14.

Criterio	Resultado de la Evaluación
Facilidad de instalación	Soportado parcial
Asistencia en el deploy	No Soportado
Ejecución de múltiples versiones de un proceso	Soportado parcial
Tolerancia a fallas	Soportado parcial
Suspender o hacer rollback de un proceso	No Soportado
Cantidad de procesos ejecutándose	Soportado
Cantidad de instancias de un proceso ejecutándose	Soportado
Soporte para grandes cantidades de usuarios	Soportado
Auditoria	Soportado parcial
Reportes	No Soportado
Integración con distintas bases de datos	Soportado
Obtención de indicadores de ejecución	Soportado parcial
Interoperabilidad	No Soportado
Importación de archivos BPEL	Si
Exportación de archivos BPEL	No
Importación desde herramientas de modelado	No
Usuarios	No
Perfiles	No
Roles	No
Simulación de procesos	No

Tabla 14 - Resumen de la evaluación de Riftsaw

1.5. Petals

1.5.1. Criterios técnicos

Software

- JDK 1.6 o superior
- Apache Tomcat 6.0 o superior
- Petals ESB
    - Petals-SE-BPEL
- Petals WebConsole
    - Petals-SE-RMI
- Petals Studio
- Petals View
    - Petals-SE-KPI
    - Petals-SE-Notification

Hardware

- Procesador 2GHz
- Memoria 512 MB
- Disco 100 MB

1.5.2. Criterios funcionales (escala 0 – No soportado, 1 – Soportado parcial y 2 – Soportado)

Facilidad de instalación: *2 – Soportado*

Si bien no presenta un instalador estilo wizard para todos los componentes, la instalación es sencilla, ya que consiste en descomprimir carpetas y ejecutar los aplicativos directamente. Estos son los casos de Petals ESB y Petals Studio.

Los componentes Petals WebConsole y Petals View, consisten en archivos .war, que se instalan copiándolos a la carpeta webapps, donde se encuentra instalado el Apache Tomcat, con el servicio Iniciado.

Luego, estos necesitan otros componentes específicos para su funcionamiento los cuales se instalan desde la ventana de administración del propio Petals WebConsole (http://<ruta_tomcat>:<puerto_tomcat>/petals-webconsole-ui-2.0.5/).

En la Figura 13 se puede ver la ruta que hay que seguir para instalar un componente, luego presionando Install, se redirecciona a una página como la que se muestra en la Figura 14, en la cual se debe ingresar la ruta del .zip con los archivos necesarios.

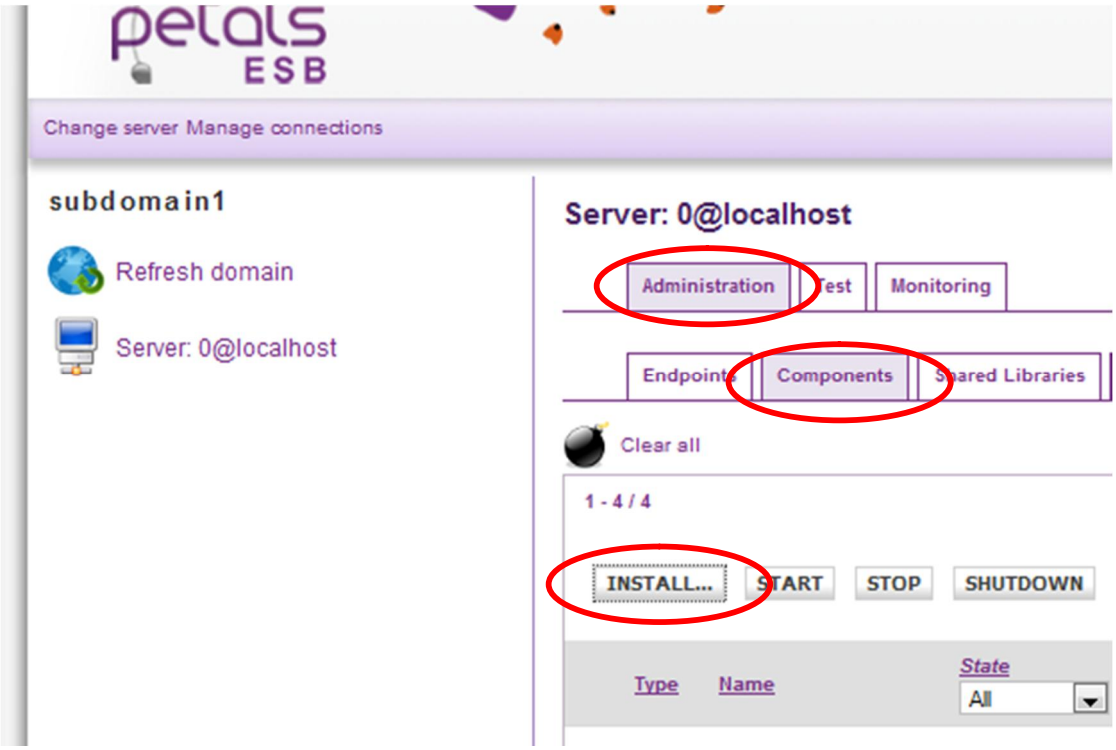

Figura 13 - Instalación de un componente

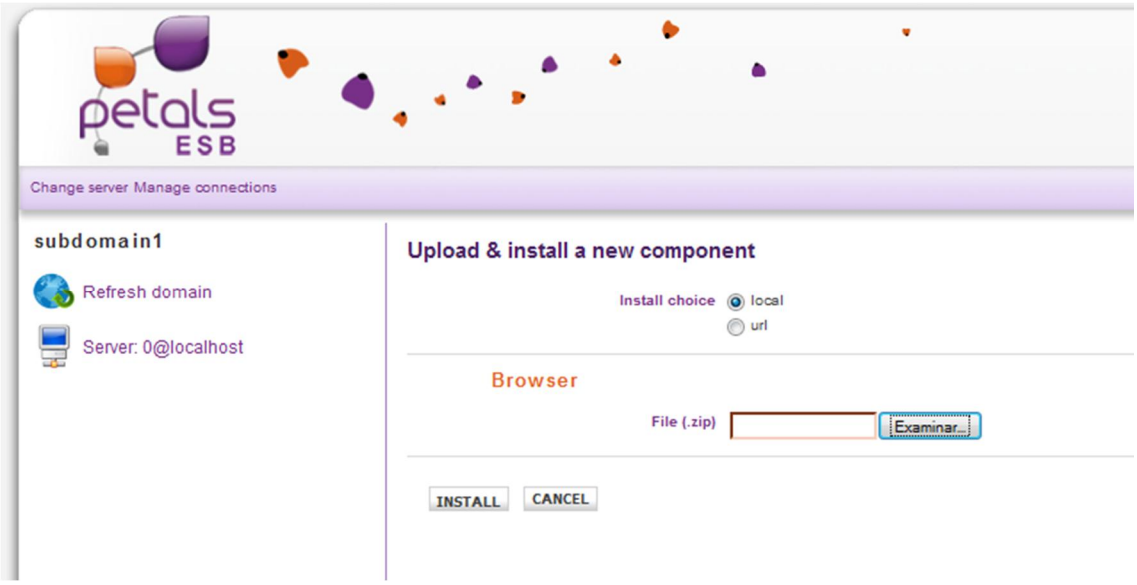

Figura 14 - Datos para la instalación

Luego de instalados los mismos se pueden Iniciar, Detener, Desinstalar, etc. En una consola como se muestra en la Figura 15.

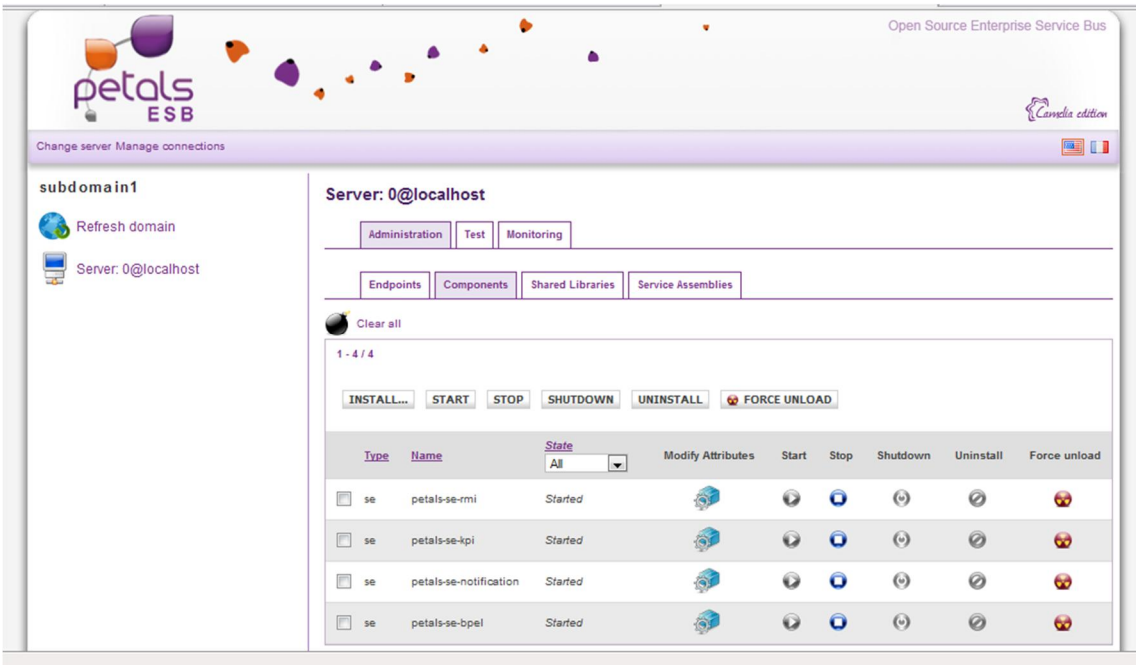

Figura 15 - Administración de los componentes

Asistencia en el deploy: *2 –Soportado*

Petals presenta un asistente para el deploy de los procesos, el cual se realiza de forma similar a la instalación de los componentes (criterio anterior). El proceso debe haber sido exportado en formato .zip, el cual debe contener el archivo .bpel con la especificación del proceso, los .wsdl de los servicios a los que hace referencia y un archivo JBI con la información para el deploy, el cual debe tener el siguiente formato:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- JBI descriptor for the Petals component petals-se-bpel  -->
<jbi:jbi version="1.0"
        xmlns:bpel="http://petals.ow2.org/components/petals-bpel-engine/version-1"
        xmlns:jbi="http://java.sun.com/xml/ns/jbi"
        xmlns:petalsCDK="http://petals.ow2.org/components/extensions/version-5"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

        <jbi:services binding-component="false">

                <jbi:provides
                        service-name="<nombre_servicio_en_petals>"
                        endpoint-name="<endpoint_servicio>"
                        xmlns:generatedNs="<namespace_servicio>">

                        <!-- CDK elements -->
                        <petalsCDK:validate-wsdl>true</petalsCDK:validate-wsdl>
                        <petalsCDK:wsdl>archivo.wsdl</petalsCDK:wsdl>

                        <!-- Component specific elements -->
                        <bpel:bpel>archivo.bpel</bpel:bpel>
```

```
                    <bpel:poolsize>1</bpel:poolsize>
                </jbi:provides>
            </jbi:services>
</jbi:jbi>
```

Luego, en la solapa Service Assembies, de la solapa de Administracion del Petals WebConsole, se tiene la opción para realizar el deploy. En la Figura 16 se muestra un ejemplo.

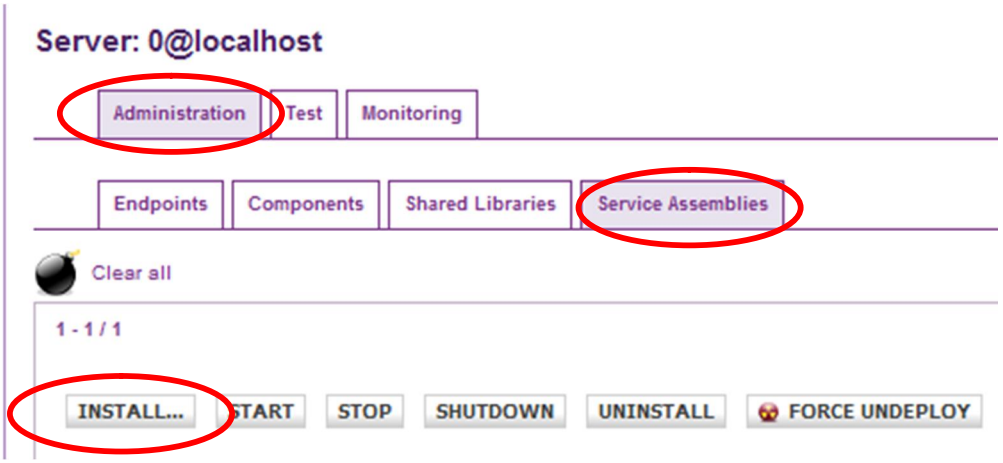

Figura 16 - Deploy de un proceso

Luego, se despliega una pantalla similar a la de la Figura 14, solicitando la ruta del .zip con el proceso que se desea instalar.

La vista de los procesos del sistema se despliega como se muestra a continuación, en la Figura 17:

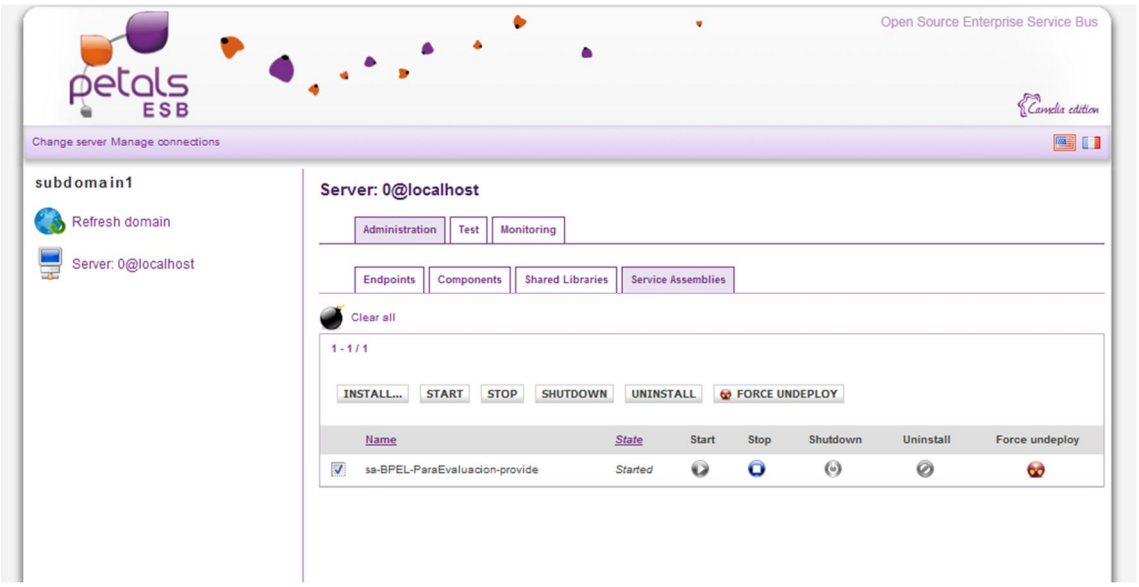

Figura 17 - Listado con los procesos

Ejecución de múltiples versiones de un proceso: *0 –No Soportado*

El motor no permite el deploy de varias versiones de un mismo proceso, esto se debe a que utiliza un identificador interno del proceso, el cual no cambia a través de las distintas versiones. Sin embargo, la ejecución de múltiples versiones se puede simular, realizando un deploy distinto por cada versión del proceso, como si fuese un proceso distinto, agregándole la versión del proceso al identificador. Esto resulta incómodo cuando se trabaja con varios procesos, ya que queda del lado del usuario el control de dichas versiones.

Tolerancia a fallas: *0 – No Soportado*

Petals no provee ningún mecanismo para el manejo o recuperación ante errores. Cualquier error que se produzca en el proceso, se muestra en pantalla y se detiene su ejecución.

Si bien se podría llegar a implementar algún mecanismo, el mismo debería ser a nivel de Apache Tomcat, ya que no se dispone de la suficiente información de Petals para su extensión o mejoras. De todas formas, aun no se ha desarrollado ningún mecanismo para el mismo.

Suspender o hacer rollback de un proceso: *1 – Soportado parcial*

Presenta únicamente la posibilidad de suspender los procesos mediante el botón Stop del listado de procesos. Sin embargo, no se puede hacer el Rollback de un proceso. Para ello, se debe implementar el proceso, y manualmente agregar transiciones a otras actividades del proceso. Cabe aclarar que sí posee un Shutdown, el cual actúa como un Reset del proceso, volviéndolo al estado inicial. El Reset actúa solamente sobre los estados del proceso, y no sobre los datos que fueron modificados. En la Figura 17 se pueden ver ambas funcionalidades, Stop y Shutdown.

Cantidad de procesos ejecutándose: *2 – Soportado*

No existe una limitante en la cantidad de procesos ejecutándose.

Cantidad de instancias de un proceso ejecutándose: *2 – Soportado*

No existe una limitante a la cantidad de instancias de un proceso ejecutándose provista por el motor. Sin embargo, y debido a que Petals se ejecuta en el entorno de Apache Tomcat, se puede estar limitado por la configuración del mismo, o también por memoria del Servidor.

Soporte para grandes cantidades de usuarios: *2 – Soportado*

No existe una limitante en la cantidad de usuarios ejecutando uno o varios procesos. Cabe hacer una aclaración similar a la del criterio anterior. El entorno de Apache Tomcat y algunos aspectos de hardware pueden llegar a limitar esto, pero no por una característica propia del motor.

Auditoria: *0 – No Soportado*

Petals no posee una herramienta de auditoría. Se podría llegar a implementar mediante el uso del log del servidor, pero el mismo no despliega información relevante. Si bien Petals maneja Usuarios y Roles, los mismos no se guardan en el log del servidor, para luego obtener la información. Cada proceso debería indicar con algún mecanismo propio dicha información, para que luego pueda ser utilizada por otro sistema.

Reportes: *1 –Soportado parcial*

Se puede exportar la lista de procesos deployados en formato .csv, pero únicamente conteniendo información de fechas de comienzo y fin de los mismos, y sus respectivos estados (Success, Error o In Progress).

Integración con distintas bases de datos: *2 – Soportado*

Brinda soporte a integración con los siguientes manejadores de bases de datos:

- MySQL (Recomendado)
- Postgres
- Oracle
- Apache Derby

Obtención de indicadores de ejecución como cantidad de procesos finalizados por unidad de tiempo, etc.: *1 – Soportado parcial*

Posee una herramienta para monitoreo para el intercambio de mensajes en el servidor, pero no de los procesos que se hicieron el deploy. A través de la misma se puede tener información respecto a errores en dichos intercambios en un determinado periodo de tiempo, mediante graficas.

En la Figura 18 se muestra un ejemplo de dicha información,

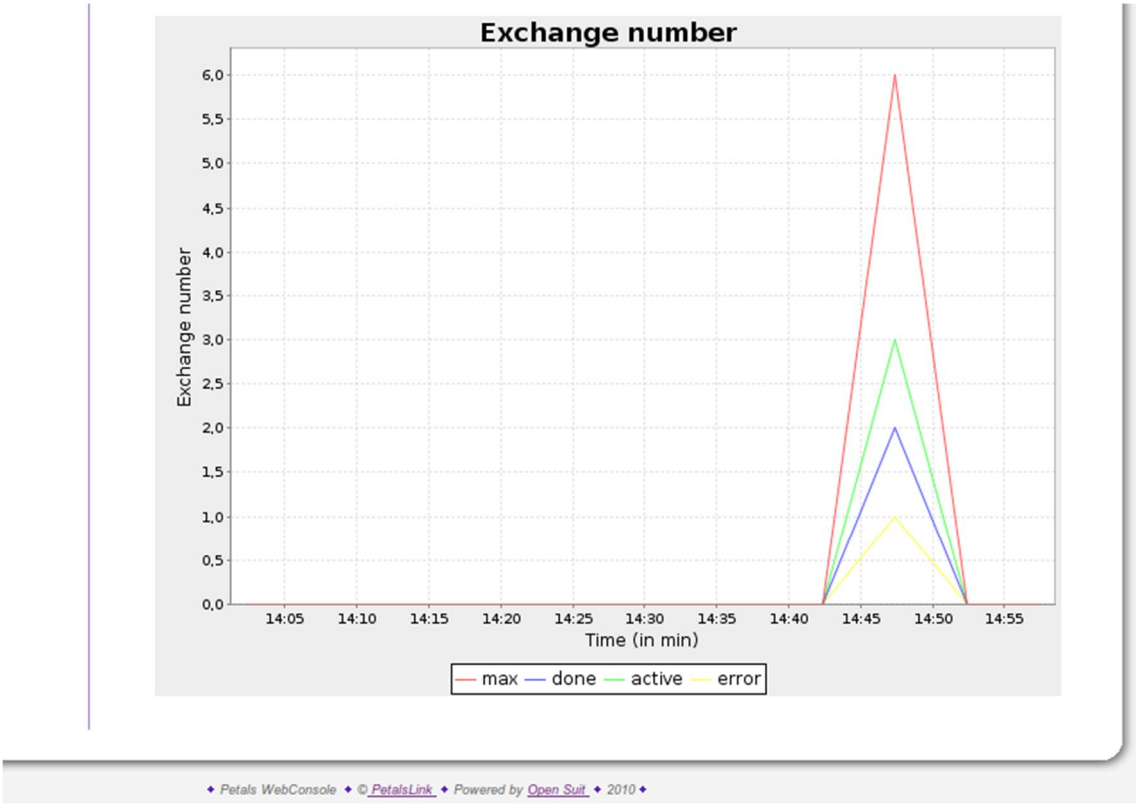

Figura 18 - Grafica sobre intercambio de mensajes

Respecto a la información de logue que maneja Petals, no se ha podido determinar otro tipo de datos que son almacenados. Esto se debió que la documentación y la comunidad alrededor de Petals no lograron brindar el conocimiento de dicha información.

Interoperabilidad: *0 –No Soportado*

Con Petals no se ha logrado la interoperabilidad con eClarus, debido a que a las tags del estándar de BPEL, agrega el prefijo *bpel:*. Un ejemplo de este comportamiento se puede ver a continuación en la siguiente tag:

<*bpel:variable/*>, mientras que la generada por eClarus seria <*variable/*>

Por tal motivo, al realizar el deploy de un proceso con estas características, el motor genera un error en el cual no reconoce la estructura del archivo BPEL.
El procedimiento para lograr la interoperabilidad, fue el de agregar dichos prefijos a los archivos .bpel generados por eClarus, además de agregar la siguiente referencia en la tag de <*process*>:

xmlns:bpel=http://docs.oasis-open.org/wsbpel/2.0/process/executable

Este es el único cambio que se debe hacer debido a que no se detectaron mas diferencias con los procesos BPEL originales de Petals. Tampoco se detectaron diferencias respecto a los archivos WSDL. El error que generado por el motor es el siguiente:

org.ow2.opensuit.xml.base.error.DefaultErrorHandler preRender

ADVERTENCIA: Technical Error Occured

org.ow2.opensuit.core.error.NonLocalizedError: [Petals service error]

Este error no se pudo solucionar debido a su generalidad y escasa información tanto de la documentación disponible, como la ayuda existente dentro de la comunidad del motor.

1.5.3. Criterios funcionales (escala Si/No)

Importación de archivos BPEL: *No*

No es posible importar archivos BPEL desde otros motores. Si bien el formato de importación/exportación es el mismo para casi todos los motores (un archivo .zip con la descripción del proceso y sus wsdl's), el archivo BPEL tiene características propias del motor.

Exportación de archivos BPEL: *Si*

A través de Petals Studio, se puede exportar el proceso y los archivos wsdl para que sean importados. El formato que utiliza para la importación es el .zip.

Importación desde herramientas de modelado: *No*

No es posible importar archivos BPEL desde herramientas de modelado Petals Studio, que no hayan sido exportados por el propio modelador.

Usuarios: *Si*

Dispone de una herramienta o sistema específico para el manejo de usuarios. En la misma se puede ingresar el nombre de usuario, contraseña y lista de roles que tendrá en el sistema. En la Figura 19 se muestra el menú para el manejo de usuarios y en la Figura 20 la pantalla para ingreso de datos.

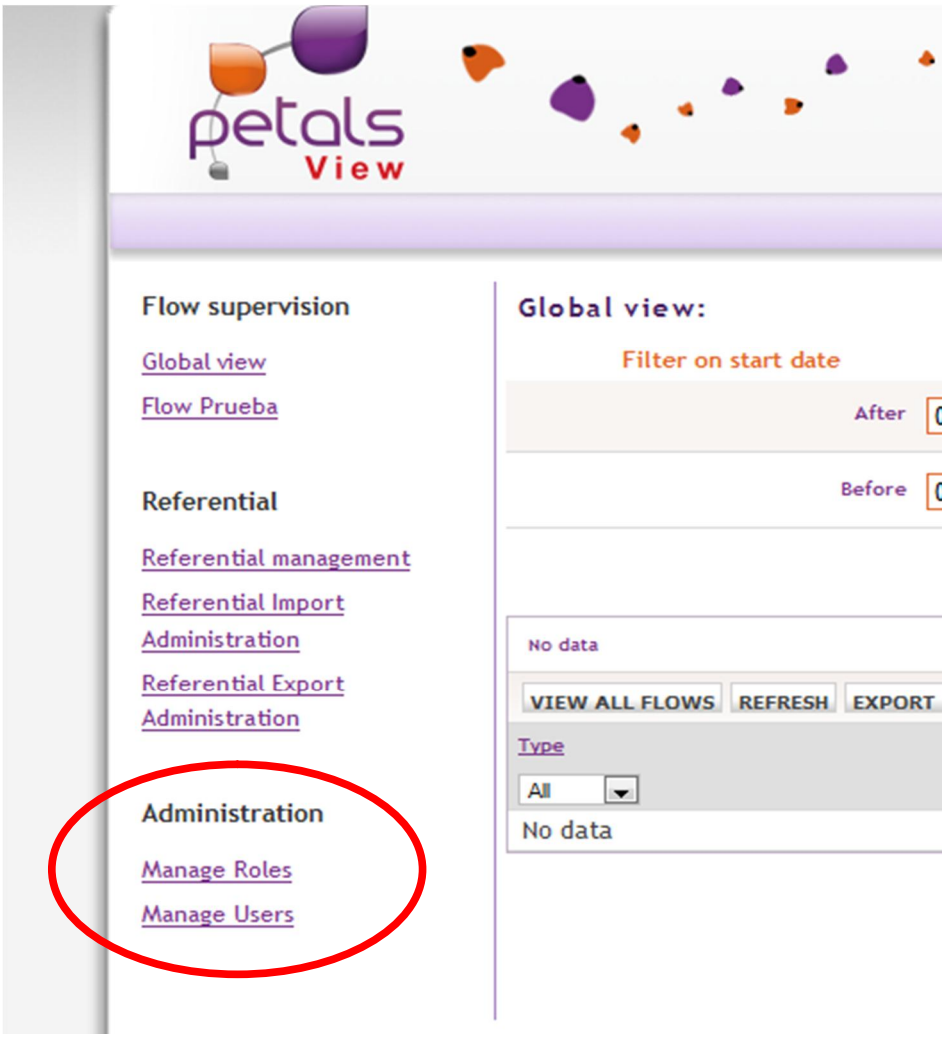

Figura 19 - Administración de Roles y Usuarios

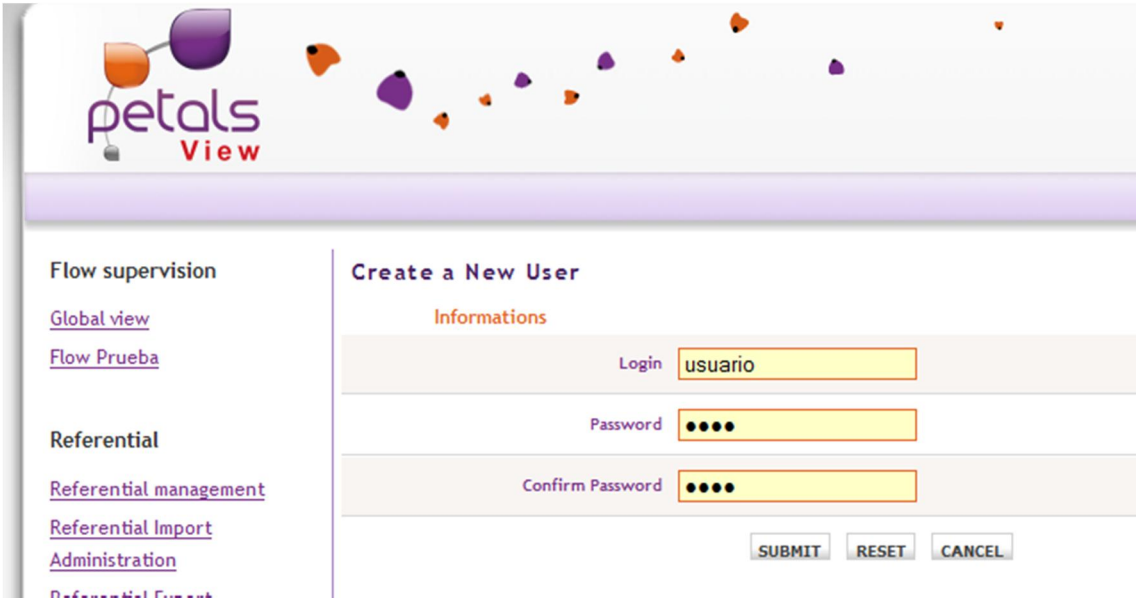

Figura 20 - Crear un usuario

Perfiles: *No*

No se dispone de una herramienta o sistema específico para el manejo de Perfiles.

Roles: *Si*

Dispone de una herramienta o sistema específico para el manejo de roles. En la misma se puede ingresar el nombre del rol y la lista permisos que tendrá en el sistema. En la Figura 19 se muestra el menú para el manejo de roles (junto con el de usuarios) y en la Figura 21 la pantalla para ingreso de datos.

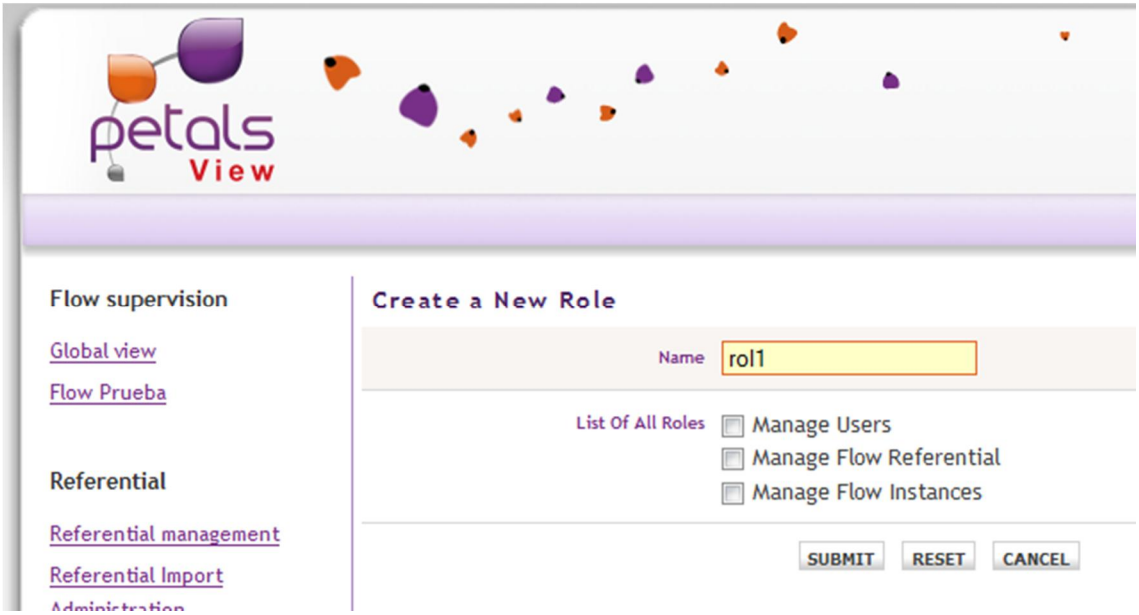

Figura 21 - Crear un rol

Simulación de procesos: *No*

No se dispone de una herramienta de simulación para procesos.

1.5.4. Resumen

Si bien Petals posee características deseables, la interoperabilidad no se logra totalmente debido a que agrega elementos propios a los archivos BPEL de los procesos. Esto impide que se realice el deploy de los mismos en otros motores y también viceversa, ya que el motor también los necesita para ejecutarlos. Cambiando los archivos BPEL generados por eClarus, no se puedo lograr la interoperabilidad, generándose otros errores. Lo mismo se ha detectado para la exportación/importación de procesos BPEL provenientes de otros motores.

Otra característica que cabe resaltar es la instalación. Si bien es sencilla y asistida, puede llegar a ser compleja por los complementos que el motor requiere. Si bien todos están disponibles en el sitio web de Petals, la documentación no es clara en este aspecto, sin mencionar la relación de dependencia que tienen entre sí, es decir, las necesidades que tiene cada componente entre sí. También es destacable mencionar, que gran parte de la documentación no está generada y no existe gran aporte por parte de la comunidad al respecto.

El no manejar el versionado de los procesos, también hace que Petals no sea tan funcional y práctico.

En la Tabla 15, se muestra un resumen de la evaluación del Petals.

Criterio	Resultado de la Evaluación
Facilidad de instalación	Soportado
Asistencia en el deploy	Soportado
Ejecución de múltiples versiones de un proceso	No Soportado
Tolerancia a fallas	No Soportado
Suspender o hacer rollback de un proceso	Soportado parcial
Cantidad de procesos ejecutándose	Soportado
Cantidad de instancias de un proceso ejecutándose	Soportado
Soporte para grandes cantidades de usuarios	Soportado
Auditoria	No Soportado
Reportes	Soportado parcial
Integración con distintas bases de datos	Soportado
Obtención de indicadores de ejecución	Soportado parcial
Interoperabilidad	No Soportado
Importación de archivos BPEL	No
Exportación de archivos BPEL	No
Importación desde herramientas de modelado	No
Usuarios	Si
Perfiles	No
Roles	Si
Simulación de procesos	No

Tabla 15 - Resumen de la evaluación de Petals

1.6. Orchestra

1.6.1. Criterios técnicos

Software

- JDK 1.5 o superior
- Ant 1.7.1 o superior (para instalación)

Hardware

- Procesador 1GHz (mínimo)
- Memoria 1GB (mínimo)

1.6.2. Criterios funcionales (escala 0 – No soportado, 1 – Soportado parcial y 2 – Soportado)

Facilidad de instalación: *1 – Soportado parcial*

No presenta un wizard para su instalación, pero se tiene la ayuda de archivos Ant que permiten automatizar parte de la misma. Para realizar la instalación se debe ejecutar la instrucción: *ant install*, en la raíz de la carpeta donde se encuentran los archivos de instalación, que se obtienen de la página web de Orchestra.

Asistencia en el deploy: *2 – Soportado*

Orchestra presenta una vista de administración que permite deployar procesos. Los procesos deben presentarse como un archivo .bar o .zip, el cual debe contener el .bpel con la descripción del proceso y los wsdl que utilice. Cabe destacar que la creación de un archivo con estas características queda a cargo del usuario que desea hacer el deploy del proceso. Orchestra no provee una herramienta para generar dicho archivo. En la Figura 22 se muestra la interface que brinda la herramienta para el deploy.

Figura 22 - Deploy de un proceso

Ejecución de múltiples versiones de un proceso: *1 – Soportado parcial*

Permite el deploy de múltiples versiones de un proceso, pero no la ejecución de las mismas simultáneamente. Por defecto, el comportamiento del Orchestra consiste en desactivar la versión anterior y activar la nueva. Si existen instancias del proceso anterior ejecutándose, la versión que permanece activada sigue siendo la anterior.

Tolerancia a fallas: *0 – No Soportado*

No presenta ningún sistema de tolerancia a fallas nativo. Si bien se podría llegar a desarrollar alguno, no se cuenta con la documentación necesaria del código como para realizar tal implementación. Los errores ocurridos, se despliegan directamente en la pantalla del administrador web y en la consola, así como también en el log del motor. En la Figura 23 se muestra el despliegue de un error ocurrido en un proceso, en tiempo de ejecución.

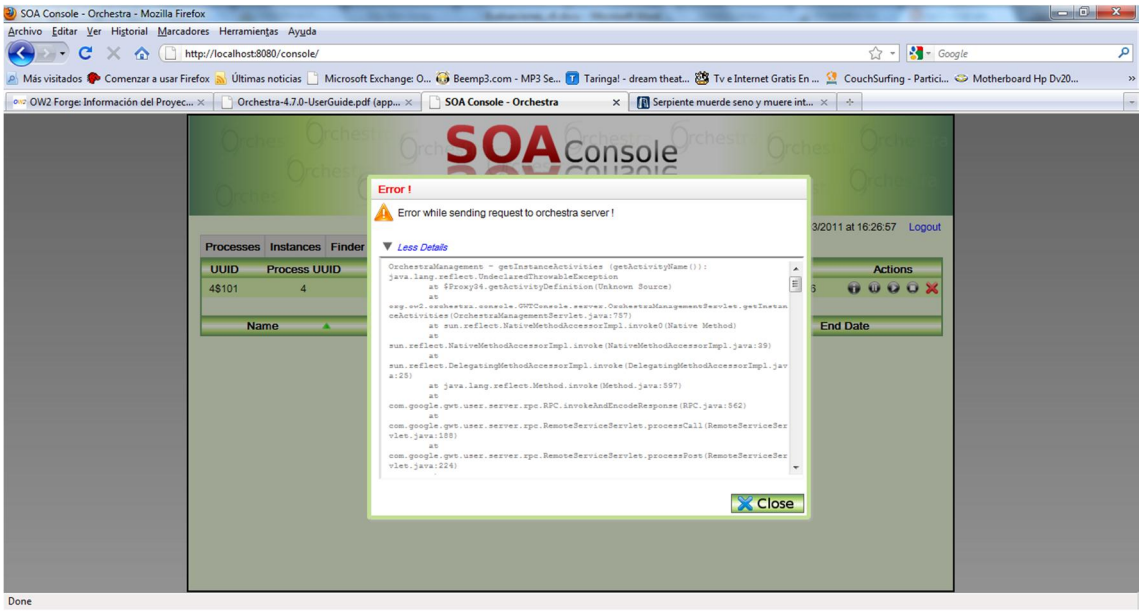

Figura 23 - Despliegue de error en un proceso

Suspender o hacer rollback de un proceso: *1 –Soportado parcial*

No presenta ningún mecanismo automático o amigable para realizar el rollback de un proceso, sino que el proceso debe ser modificado para permitir dicha funcionalidad (o parte de ella).

Sin embargo, sí se puede suspender un proceso, mediante el administrador web. El comportamiento del mismo consiste en suspender todas las instancias que se estén ejecutando. Cabe aclarar que también se puede realizar la suspensión específica de una instancia de un proceso, sin la necesidad de afectar a las demás. Para realizar dichas acciones, se dispone de botones como los que se muestra en la Figura 24.

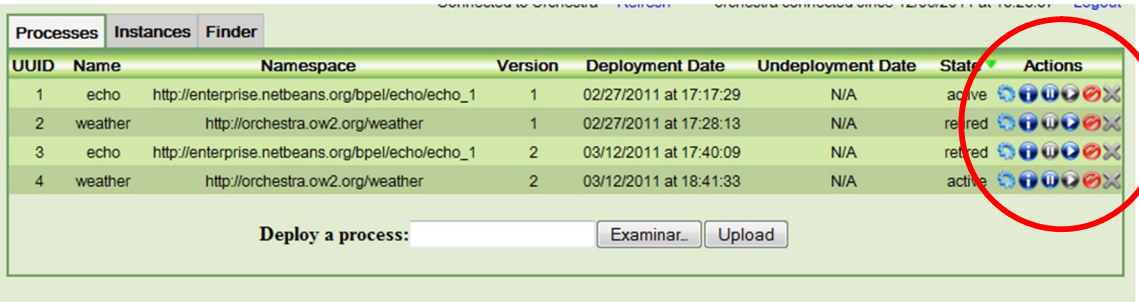

Figura 24 - Botones de manejo de acciones

Cantidad de procesos ejecutándose: *2 – Soportado*

No existe una limitante en la cantidad de procesos ejecutándose. Dicha configuración se obtiene modificando el parámetro <invoke-executor threads="<cantidad>"/>, el cual se encuentra en la carpeta *<Ruta_Orchestra>\conf\environment.xml*.

Cantidad de instancias de un proceso ejecutándose: *2 – Soportado*

Tampoco existe una limitante respecto a la cantidad de instancias de un proceso ejecutándose por proceso.

Soporte para grandes cantidades de usuarios: *2 – Soportado*

No se ha detectado ninguna limitante al respecto.

Auditoria: *0 – No Soportado*

No posee una herramienta integrada para la auditoría de los procesos donde se pueda visualizar dicha información, ni tampoco se cuenta con la información necesaria para desarrollar un sistema con estas características.

Reportes: *0 – No soportado*

No posee herramientas para generar reportes, ni de procesos ni del estado general del servidor.

Integración con distintas bases de datos: *2 – Soportado*

Brinda soporte a integración con diferentes bases de datos. Esta configuración se logra indicando en el archivo *hibernate.properties*, que se encuentra en la carpeta *<Ruta_Orchestra>\conf*, la base de datos que se desea utilizar. Las bases de datos que se pueden utilizar son las siguientes:

- Oracle
- MySQL
- Postgres

Obtención de indicadores de ejecución como cantidad de procesos finalizados por unidad de tiempo, etc.: *0 – No Soportado*

No posee una herramienta integrada para la obtención de dicha información. Únicamente se dispone, por instancia, de información de finalizados con o sin error pero sin indicadores ni tampoco algún tipo de agrupación de datos.

La única información de logueo de ejecuciones que se almacena es la relacionada a las instancias ejecutadas indicando el tiempo de ejecución (inicio y fin) y su estado actual (ejecutando, suspendido, finalizado, terminado), pero sin detalles más relevantes. Dicha información se presenta en la tabla ORCHESTRA.MON_RUNT_ACTIVITY.

ORCHESTRA. MON_RUNT_ACTIVITY: A continuación se muestra la información anteriormente descripta.

Nombre	Descripción
DBID	Identificador numérico.
CLASS	Nombre de la clases ejecutada
PROCESS_UUID	Identificador del proceso ejecutado
INST_UUID	Identificador de la instancia del proceso que fue ejecutado
STARTED_DATE	Timestamp del comienzo de la ejecución
ENDED_DATE	Timestamp de la finalización de la ejecución
ACTIVITY_STATE	Estado actual de la ejecución, la misma se guarda como una cadena de caracteres
OPERATION	Operación invocada en la ejecución de la instancia
MESSAGE_EXCHANGE	Mensaje intercambiado en la invocación de la operación
LAST_EXCEPTION	Identificador de la excepción que se haya generado, en caso que así haya ocurrido

Tabla 16 - Tabla de logueo para Orchestra

Interoperabilidad: *0 – No Soportado*

Orchestra no permite interoperabilidad con los procesos diseñados en la herramienta eClarus. Esto se debe a que Orchestra le agrega a las tags del estándar de BPEL el prefijo *bpws:*, y los archivos .bpel generados en eClarus no lo poseen, impidiendo esto, directamente el deploy de los procesos. Algunos ejemplos de las tags que requiere Orchestra, comparadas con las que genera eClarus son:

<bpws:process></bpws:process> mientras que en eClarus seria *<process></process>*

<bpws:partnerLink/> mientras que en eClarus *<partnerLink/>*

Al realizar el deploy de dicho proceso en Orchestra, con las características antes mencionadas, se generan errores.

Para subsanarlos es detectaron que se deberían realizar las siguientes modificaciones:

1) Agregar a cada tag del archivo BPEL generado por eClarus, el prefijo *bpws:* excepto en las tags de *<target>* y *<source>*, que se encuentra dentro de la tag de *<invoke>*. El estándar no las menciona dentro de la tag de *<invoke>*, siendo éstos agregados al estándar.
2) Dentro de la tag de *<process>*, se debe agregar la referencia a los prefijos anteriormente mencionados,

xmlns:bpws="http://docs.oasis-open.org/wsbpel/2.0/process/executable"

3) En los archivos WSDL para los servicios es necesario cambiar la referencia a *<plnk:partnerLinkType>*, dentro de la tag de *<definitions>*, por la que se muestra a continuación:

xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"

Realizando las modificaciones anteriores, se logra igualar los cabezales de los BPEL generados por eClarus y los archivos BPEL que generan los diseñadores de Orchestra. Sin embargo, el deploy de los procesos no se ha logrado debido a que Orchestra no reconoce los roles indicados en la tag de *<partnerLink>*.

<bpws:partnerLink myRole="<nombre_rol>" partnerRole="<nombre_rol2>" name=… />

El error generado es el siguiente:

error : SA00016: A partnerLink MUST specify the myRole or the partnerRole, or both.

Dicho error no pudo ser solucionado debido a la escasa información que se tiene tanto por parte de la documentación de Orchestra, así como también, por la Comunidad.

1.6.3. Criterios funcionales (escala Si/No)

Importación de archivos BPEL: *Si*

Orchestra permite la importación de procesos de motores provenientes de jBPM BPEL e Intalio, siempre y cuando las mismas sean presentadas en un archivo .zip (o .bar), conteniendo los archivos .bpel y wsdl que usan. Cabe aclarar que si bien no provee la funcionalidad especifica de Importación, la misma se puede hacer realizando el deploy del proceso proveniente de otro motor.

Motor	Detalle
jBPM BPEL	Permite la importación sin problemas
Apache ODE	No permite. Apache ODE agrega características propias a los archivos .bpel
Intalio	Permite la importación sin problemas
Riftsaw	No permite. Riftsaw agrega características propias a los archivos .bpel
Petals	No permite. Petals agrega características propias a los archivos .bpel
Orchestra	Permite la importación de sus propios archivos

Tabla 17 - Importación de archivos BPEL para Orchestra

Exportación de archivos BPEL: *No*

No se tiene una herramienta que permita esta funcionalidad, sino que el usuario si desea exportar sus procesos, debe hacerlo manualmente, generando un archivo .bar o .zip con los archivos .bpel y wsdl necesarios.

Importación desde herramientas de modelado: *Si*

Utilizando la herramienta Eclipse BPEL Designer o NetBeans BPEL Designer, se puede importar procesos y sus respectivos archivos wsdl en los formatos .zip o .bar.

Usuarios: ***No***
No se dispone de una herramienta o sistema específico para el manejo de usuarios.

Perfiles: ***No***
No se dispone de una herramienta o sistema específico para el manejo de Perfiles.

Roles: ***No***
No se dispone de una herramienta o sistema específico para el manejo de Roles Similar al criterio de Usuarios o Perfiles.

Simulación de procesos: ***No***
No se dispone de una herramienta de simulación para procesos.

1.6.4. Resumen

Si bien Orchestra soporta muchas características deseables, no ocurre lo mismo con las más interesantes, como la obtención de indicadores o auditoria de los procesos. También se puede mencionar que la documentación y la comunidad alrededor de Orchestra, no brindan demasiada información.

En cuanto a la interoperabilidad, la misma no se ha podido lograr, debido a que Orchestra posee tags específicas en los archivos BPEL que necesita para hacer el deploy de los procesos.

Sin embargo la importación de archivos BPEL (provenientes de los demás motores), se ha logrado sin demasiados problemas, más allá del que requiere un archivo específico para su deploy (.bar). Pero el mismo se genera de forma sencilla, similar a un .zip.

En la Tabla 18, se puede observar un resumen de la evaluación realizada.

Criterio	Resultado de la Evaluación
Facilidad de instalación	Soportado parcial
Asistencia en el deploy	Soportado
Ejecución de múltiples versiones de un proceso	Soportado parcial
Tolerancia a fallas	No Soportado
Suspender o hacer rollback de un proceso	Soportado parcial
Cantidad de procesos ejecutándose	Soportado
Cantidad de instancias de un proceso ejecutándose	Soportado
Soporte para grandes cantidades de usuarios	Soportado
Auditoria	No Soportado
Reportes	No Soportado
Integración con distintas bases de datos	Soportado
Obtención de indicadores de ejecución	No Soportado
Interoperabilidad	No Soportado
Importación de archivos BPEL	Si
Exportación de archivos BPEL	No
Importación desde herramientas de	Si

modelado	
Usuarios	No
Perfiles	No
Roles	No
Simulación de procesos	No

Tabla 18 - Resumen de la evaluación de Orchestra

1.7. jBPM5

1.7.1. Criterios técnicos

Software

- JDK 1.5 o superior
- Ant 1.7.1 o superior (para instalación)

Hardware

- Procesador 1GHz (mínimo)
- Memoria 1 GB
- Disco 500 MB libres

1.7.2. Criterios funcionales (escala 0 – No soportado, 1 – Soportado parcial y 2 – Soportado)

Facilidad de instalación: *2 – Soportado*

La instalación de jBPM5 es sumamente sencilla. Una vez descargado el archivo .zip conteniendo el instalador y habiendo extraído el mismo, se ejecuta la instrucción *ant install*, y el script se encarga de descargar todos los componentes necesarios para el funcionamiento del motor. Esto es Eclipse Helios conjuntamente con los plugins correspondientes, jBoss 5.0 como servidor de aplicaciones y le instala los componentes de jBPM para manejar la consola web y el designer para los procesos en BPMN2 (Drools).

Asistencia en el deploy: *2 – Soportado*

El deploy en jBPM5, se realiza mediante 2 componentes, el Eclipse para manejar el designer (en este caso, con un plugin) y la consola web de Drools (la cual se puede acceder en la URL que se construye como se muestra a continuación: http://<ruta_jboss>:<puerto_jboss>/drools-guvnor/).

En el Eclipse y con el proceso en BPMN seleccionado, se los agrega a Drools-Guvnor, (indicando posteriormente el repositorio en donde residirá el mismo). Luego en la consola de Drools-Guvnor, se ejecuta un Build del proceso. Si esta actividad se ejecuta sin errores, dicho proceso se puede observar en la consola de jBPM5, a la cual se ingresa mediante: http://<ruta_jboss>:<puerto_jboss>/jbpm-console/). La Figura 25 muestra como se observa finalmente el proceso luego de realizado su deploy.

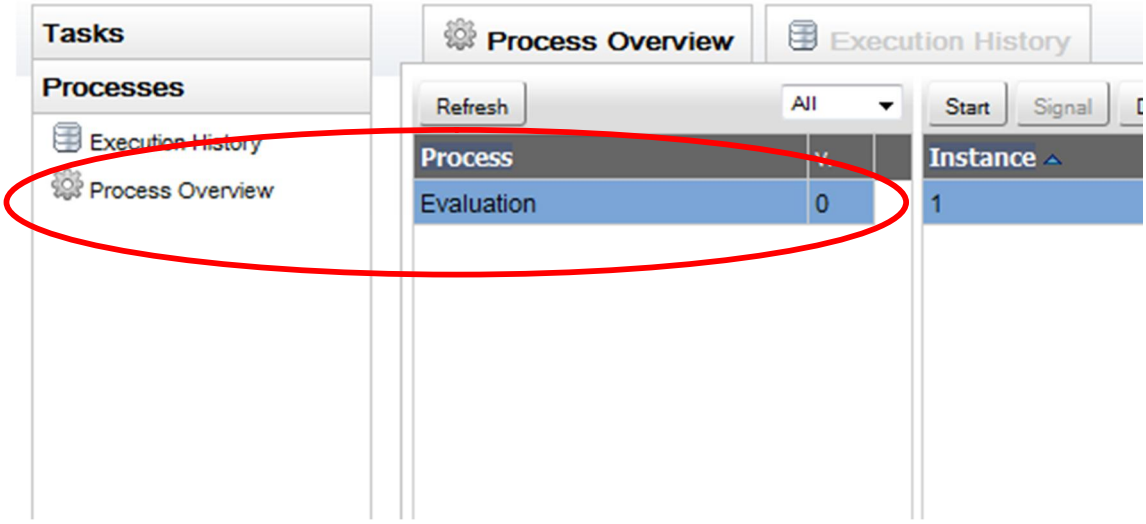

Figura 25 - Deploy de un proceso

Ejecución de múltiples versiones de un proceso: *1 – Soportado parcial*

El motor permite el deploy de varias versiones de un mismo proceso. El comportamiento por defecto es detener la ejecución de las versiones anteriores y activar la nueva, esto siempre que no existan instancias de ese proceso ejecutándose. En dicho caso se espera la finalización de las mismas para luego detenerlas. Cabe aclarar que manualmente se puede cambiar el proceso activo nuevamente, si es necesario.

Tolerancia a fallas: *1 – Soportado parcial*

No presenta ningún sistema de tolerancia a fallas nativo. Sin embargo, se puede integrar con herramientas de manejo de errores que existen para jBoss. La otra opción es el manejo de errores dentro de los proceso a través de los FaultHandlers.

De todas formas, cabe aclarar que ante cualquier error, el motor despliega toda la información pertinente en pantalla, como por ej.: un stacktrace.

Suspender o hacer rollback de un proceso: *0 –No Soportado*

No presenta ningún mecanismo automático o amigable para realizar ninguna de esas 2 funciones, sino que el proceso debe ser modificado para permitir dicha funcionalidad (o parte de ella).

La suspensión se puede llegar a implementar mediante la inclusión de timers en el proceso, mientras que el rollback con transiciones hacia atrás. Esto hace que dichas funcionalidades sean muy complejas de implementar, y en el caso del rollback, implementar parcialmente, debido a que si se ejecuta algún método que no permita su vuelta hacia atrás el mismo se efectuará de forma incompleta.

Cantidad de procesos ejecutándose: *2 – Soportado*

No existe una limitante en la cantidad de procesos ejecutándose. Lo que se debe tener en cuenta es que debido a que JBPM5 se ejecuta en el contexto de JBoss, está limitada la cantidad de accesos que éste, según se haya configurado en dicho servidor. Cambiando este parámetro se cambia también la cantidad de accesos que puede tener jBPM5.

Si bien se pueden tener todos los procesos que se deseen ejecutando, no necesariamente se pueden acceder a todos, esto es lo que limita la configuración de jBoss.

Cantidad de instancias de un proceso ejecutándose: *2 – Soportado*

Similar resultado que para el criterio anterior, si bien se puede acceder a la cantidad que se desee de procesos (o en este caso, la cantidad de instancias de un mismo proceso), se está limitado por la configuración de jBoss. Cabe destacar que esta configuración es global a todo el servidor.

Soporte para grandes cantidades de usuarios: *2 – Soportado*

No se ha detectado ninguna limitante al respecto, aunque es de especial interés tener en cuenta los comentarios de los últimos 2 criterios.

Auditoria: *0 – No Soportado*

No posee una herramienta integrada para la auditoria de los procesos donde se pueda visualizar dicha información, ni tampoco se cuenta con la información necesaria, para desarrollar un sistema con estas características.

Reportes: *2 –Soportado*

jBPM5 posee una funcionalidad para realizar reporte de todos los procesos ejecutados en el sistema, así como también, de un proceso en particular. Sin embargo no posee una funcionalidad para exportarlo a distintos formatos. En la Figura 26, se muestra un ejemplo de reporte de todos los procesos del sistema.

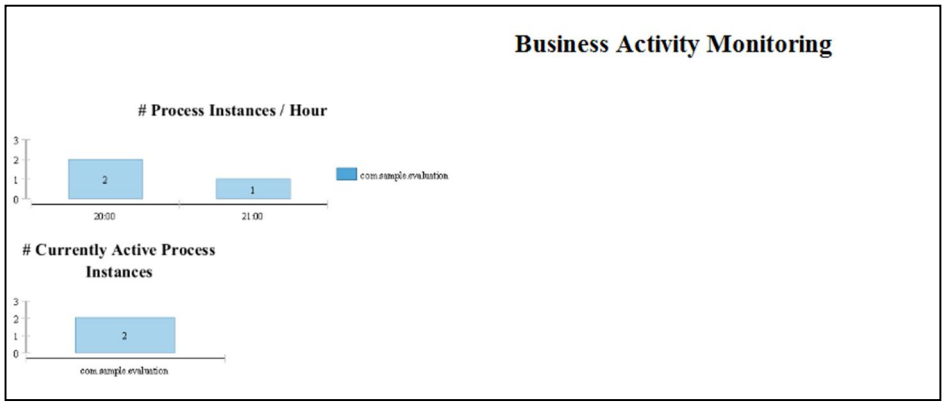

Figura 26 - Reporte de Estado General

Integración con distintas bases de datos: *2 – Soportado*

jBPM 5 utiliza como método de persistencia a JPA (Java Persistence API), con lo cual es compatible con cualquier sistema de bases de datos.

Para realizar la configuración se debe indicar la información requerida por el archivo *persistence.xml* y el datasource correspondiente, de la misma forma que para cualquier otro tipo de proyecto que utilice JPA. A continuación se muestra la estructura del archivo.

```
<persistence-unit name="org.jbpm.persistence.jpa">
        <provider>org.hibernate.ejb.HibernatePersistence</provider>
        <jta-data-source>jdbc/<nombre_data_source></jta-data-source>
        <class>org.drools.persistence.session.SessionInfo</class>
        <class>org.jbpm.persistence.processinstance.ProcessInstanceInfo</class>
        <class>org.jbpm.persistence.processinstance.WorkItemInfo</class>
        <properties>
                <property name="hibernate.dialect" value="libreria_base_de_datos"/>
                <property name="hibernate.max_fetch_depth" value="3"/>
                <property name="hibernate.hbm2ddl.auto" value="update"/>
                <property name="hibernate.show_sql" value="true"/>
                <property name="hibernate.transaction.manager_lookup_class"
                value="org.hibernate.transaction.BTMTransactionManagerLookup"/>
        </properties>
</persistence-unit>
```

También a continuación se muestra un archivo Datasource de ejemplo:

```
<datasources>
    <local-tx-datasource>
        <jndi-name>jdbc/<nombre_data_source></jndi-name>
        <connection-url><connection_string_base_de_datos></connection-url>
        <driver-class><driver_base_de_datos></driver-class>
        <user-name> </user-name>
        <password> </password>
    </local-tx-datasource>
</datasources>
```

Obtención de indicadores de ejecución como cantidad de procesos finalizados por unidad de tiempo, etc.: *0 – No Soportado*

No posee una herramienta integrada para la obtención de dicha información. Únicamente se dispone, por instancia, de información de finalizados con o sin error, pero sin indicadores, ni tampoco algún tipo de agrupación de datos.

De todas formas, jBPM5 almacena a nivel de base de datos, la información antes mencionada, la cual puede ser utilizada en la herramienta ProM. La información almacenada se presenta de manera similar a la almacenada en jBPM BPEL, en un esquema similar de tablas, indicándose por instancia, los tiempos de finalización de cada una y si las mismas generaron excepciones y cuales fueron. No se ha detectado otra forma de almacenamiento de logs de ejecución en este motor, que brinde mayor calidad en la información brindada.

De acuerdo a la documentación de jBPM5 el módulo de jbpm-bam almacena la información relacionada a los procesos en una base de datos con dos tablas: una para almacenar información de las instancias y otra para información de los nodos de cada instancia.

PROCESSINSTANCELOG: Esta tabla contiene el identificador del proceso (definición), el identificador de la instancia, el timestamp del inicio de la instancia y en caso de corresponder el timestamp de finalización de la misma.

NODEINSTANCELOG: Esta tabla contiene información sobre que nodos fueron efectivamente ejecutados en cada instancia de cada proceso. Cada vez que se alcanza o abandona un nodo por alguna de sus conexiones entrantes/salientes esa información se almacena en esta tabla. Se guarda información de: identificador de la instancia, identificador del proceso al que pertenece la instancia en ejecución, identificador de la instancia del nodo (identifica la instancia de un nodo dentro de una instancia de proceso), identificador del nodo al que corresponde la instancia del mismo (correspondencia similar a proceso-instancia), tipo de evento (0 = llegar, 1 = salir) y el timestamp en que ocurre el evento.

Interoperabilidad: *0 – No Soportado*

De forma similar que para los motores BPEL, para jBPM5 se realizaron pruebas de interoperabilidad con procesos diseñados en otras herramientas. En este caso, el proceso utilizado es el que se muestra en la Figura 27.

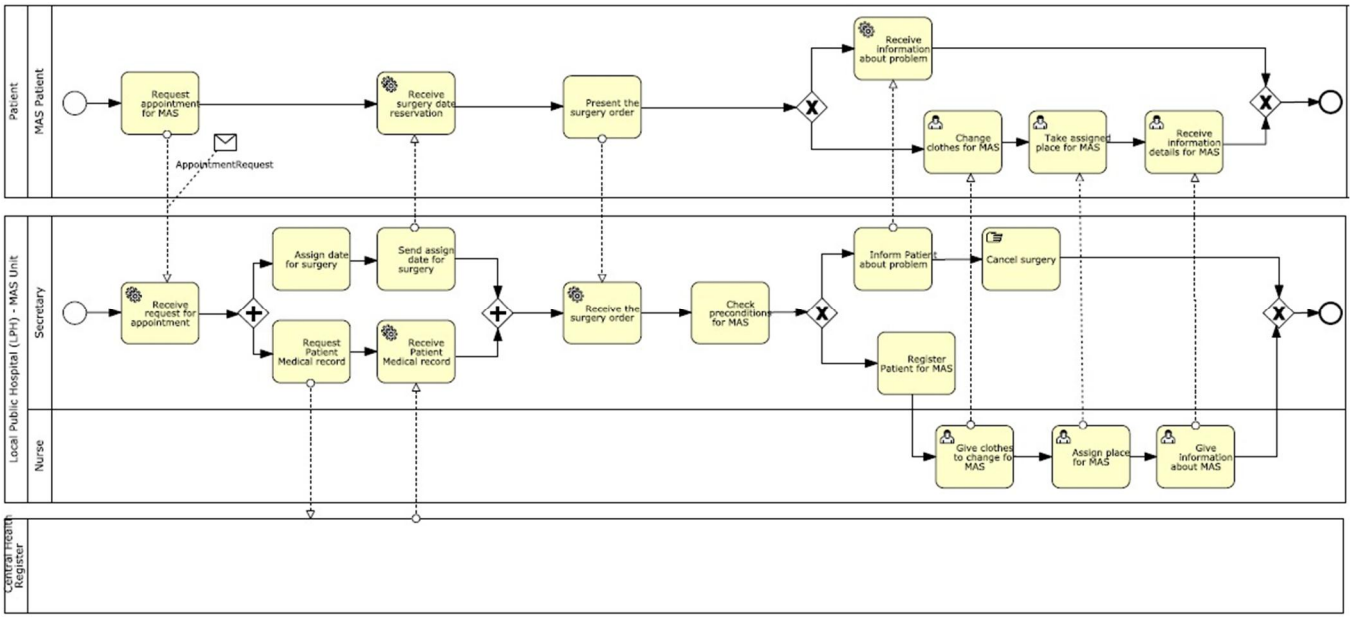

Figura 27 - Proceso utilizado en el motor jBPM5 para interoperabilidad

Para el caso de jBPM5, no se pudo lograr la interoperabilidad, debido a errores que genera el motor al intentar hacer el deploy del proceso en cuestión. Tampoco se ha podido solucionar los errores que ocasiona, debido a que la información que proporciona el log del servidor, no es suficiente. Cabe aclarar que tampoco se ha logrado acceder al proceso vía el Designer de jBPM5.

1.7.3. Criterios funcionales (escala Si/No)

Importación de archivos BPEL: *N/A*
Este criterio no aplica para un motor BPMN 2.0

Exportación de archivos BPEL: *N/A*
Este criterio no aplica para un motor BPMN 2.0

Importación desde herramientas de modelado: *N/A*
Este criterio no aplica para un motor BPMN 2.0

Usuarios: *Si*
Si bien no se dispone de una interfaz grafica para agregar/modificar/quitar usuarios, esta acción se puede realizar modificando el archivo *users.properties* que se encuentra en la carpeta *<ruta_jbpm5>/auth*.

Perfiles: *No*
No se dispone de una herramienta o sistema específico para el manejo de Perfiles.

Roles: *Si*
Similar comentario que para el criterio de Usuarios, en este caso el archivo que se debe modificar es *roles.properties.*

Simulación de procesos: *No*
No se dispone de una herramienta de simulación para procesos.

1.7.4. Resumen

Modo de conclusión, se puede decir que jBPM5, es un motor bastante completo y cumple con las características más deseadas.

La instalación es muy sencilla, aunque requiere muchos componentes para poder sacarle el mayor provecho y poder acceder a todas las funcionalidades existentes, como generación de reportes o mismo el Designer de BPMN 2.0.

El deploy, que no es muy sencillo e intuitivo, se puede lograr debido a que la documentación es bastante completa. Además, el hecho que soporte grandes cantidades de usuarios, lo hace ser más robusto y adecuado a grandes organismos.

En la Tabla 19, se puede observar un resumen de la evaluación realizada.

Criterio	Resultado de la Evaluación
Facilidad de instalación	Soportado
Asistencia en el deploy	Soportado
Ejecución de múltiples versiones de un proceso	Soportado parcial
Tolerancia a fallas	Soportado parcial
Suspender o hacer rollback de un proceso	No Soportado
Cantidad de procesos ejecutándose	Soportado

Cantidad de instancias de un proceso ejecutándose	Soportado
Soporte para grandes cantidades de usuarios	Soportado
Auditoria	No Soportado
Reportes	Soportado
Integración con distintas bases de datos	Soportado
Obtención de indicadores de ejecución	No Soportado
Interoperabilidad	No Soportado
Importación de archivos BPEL	N/A
Exportación de archivos BPEL	N/A
Importación desde herramientas de modelado	N/A
Usuarios	Si
Perfiles	No
Roles	Si
Simulación de procesos	No

Tabla 19 - Resumen de la evaluación de jBPM5