UNIVERSIDAD AUTÓNOMA METROPOLITANA
UNIDAD IZTAPALAPA División de Ciencias Básicas e Ingeniería
Casa abierta al tiempo

"Desarrollo de una Suite BPM para el modelado, ejecución y monitoreo de los procesos de un Modelo de Mejora de Procesos de Desarrollo de Software"

Tesis que presenta:
C.P. y Lic. Silvia Nagheli Márquez Solís

Para obtener el grado de:
MAESTRA EN CIENCIAS
DEL
POSGRADO EN CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

Asesores:

Dr. Humberto Cervantes Maceda
Dr. Carlos Montes de Oca Vázquez

Jurado calificador:

Presidente: M. en C. Alfonso Martínez Martínez
Secretario: Dr. Humberto Cervantes Maceda
Vocal: M. en C. Mery Helen Pesantes Espinoza

MEXICO, D.F. JULIO 2011

Resumen

En este trabajo se presenta una de investigación enfocada a dar solución a problemas que surgen al trabajar con procesos de los Modelos de Mejora de Procesos de Desarrollo de Software. Estos problemas son: modelar la definición del proceso, mantener actualizada la documentación relacionada a esta definición, apoyar la ejecución, monitorear y medir los procesos. La propuesta de solución consiste en aplicar el enfoque de Administración de Procesos de Negocios (BPM, por sus siglas en inglés).

El objetivo de este trabajo es determinar si el enfoque de BPM es útil a las organizaciones que implementan procesos de un Modelo de Mejora de Procesos de Desarrollo de Software y que además cuentan con escasos recursos.

Este trabajo tuvo como resultado el desarrollo de una herramienta que soporta el enfoque BPM. Existen herramientas que soportan este enfoque, sin embargo, la herramienta construida, además de soportar las funciones usuales de las Suites BPM: modelado, ejecución y monitoreo de los procesos, soporta algunas características particulares de los Modelos de Mejora de Procesos de Desarrollo de Software. Adicionalmente, se presentan los resultados de utilizar la herramienta construida.

Agradecimientos

Agradezco a mi Esposo su apoyo y comprensión que me brindo durante la realización de esta investigación.

Agradezco a mi Asesor su guía para realizar mi trabajo con calidad y veracidad. También le agradezco a mi Asesor su propuesta de realizar este proyecto que me llevo a investigar exactamente los temas que son de mi completo interés.

Agradezco a mis Padres y a todos mis Profesores que durante la vida he tenido la oportunidad de ser su Alumna, porque con sus enseñanzas forjaron mi carácter, adquirí conocimiento y los altos niveles de calidad con los que ahora me conduzco y que me han permitido lograr mis objetivos.

Pero sobre todo, agradezco a Dios por la oportunidad de vivir.

Contenido

Lista de Figuras

Lista de Tablas

Capítulo
1 Introducción

Un proceso es un conjunto de actividades que se llevan a cabo para lograr un propósito. En el desarrollo del software se siguen frecuentemente procesos informales, es decir, procesos que surgen en el momento de crear el producto de software y que no son documentados. Al no ser documentados los procesos, rara vez son repetidos de la misma forma, la experiencia vivida de los procesos exitosos es desperdiciada, dejando al desarrollador la responsabilidad de guardar el conocimiento adquirido en su memoria. Esta forma de hacer software no ha resultado útil a medida que las necesidades de la sociedad actual demandan la creación de software más complejo y confiable. Por esta razón, han surgido metodologías y enfoques que buscan mejorar el proceso de desarrollo de software, como por ejemplo, el enfoque del modelo en cascada. Si bien estas metodologías y enfoques siguen todavía en la etapa de adaptarse a las características de la creación de software, han servido para satisfacer a los usuarios de los productos de software y a los creadores que han decidido utilizarlas. Un enfoque basado en procesos retoma principios provenientes de procesos de manufactura y los adapta al proceso de desarrollo de software. Estos principios señalan que las características del producto final, como por ejemplo: durabilidad, resistencia mecánica, son incorporadas durante el proceso de elaboración. Si estas características son erróneas, al final del proceso, el producto no dejará satisfecho al usuario. Estas características erróneas llamadas "Defectos" deben ser eliminadas durante el proceso o de ser posible evitar su introducción.

La calidad es el logro del conjunto de características específicas permisibles en el producto, también llamadas requerimientos, y los defectos, como ya se mencionó, son las características erróneas. Para tener mayor claridad y diferenciar las características permisibles de las que son erróneas, se plantean las permisibles en términos cuantificables, de esta manera si una no alcanza su valor específico, entonces se sabe que se ha introducido un defecto. Por tal razón, durante la ejecución de un proceso es necesario tomar los valores de las

características que permitan saber si estas alcanzarán su valor específico en el producto final, esto se conoce como “*monitorear el proceso”*. Si no alcanzarán su valor específico, entonces se hace una corrección al proceso, esto es conocido como “*mejorar el proceso”*, que también puede realizarse antes de iniciar un nuevo ciclo.

Al evidenciarse en la manufactura la importancia del proceso en la creación de un producto, algunos desarrolladores de software han dado importancia al proceso de desarrollo de software y han documentado los procesos exitosos y con estos procesos se han creado modelos a seguir. Estos modelos conocidos como "Modelos de Mejora de Procesos de Desarrollo de Software", que en las siguientes secciones sólo se nombran como Modelos de Mejora de Procesos, son modelos que integran un conjunto de procesos que guían a las organizaciones a implementar y adaptar los procesos que reúnen las mejores prácticas en el desarrollo de software. Ejemplos de ellos son: CMMI (Modelo de Capacidad y Madurez Integrado) *[16]* y MoProSoft (Modelo de Procesos para la Industria del Software) *[17]*. Estos procesos son procesos formales, es decir, son procesos que se documentan permitiendo su revisión y estudio para identificar y prevenir la introducción de defectos. Estos procesos tienen como objetivo producir software de calidad, o dicho de otra manera, deben producir productos de software libres de defectos que cumplen con sus requerimientos *[1, pág. 4]*.

1.1 Problemática

Los procesos para desarrollar productos de software “complejos”, frecuentemente lo son también; debido a esta complejidad, su modelado, documentación y monitoreo se dificulta. En particular, estas dificultades se les presentan a las organizaciones que trabajan de acuerdo a los Modelos de Mejora de Procesos. También, como los procesos son continuamente estudiados y rectificados, mantener la documentación actualizada representa un reto adicional. En general, las dificultades a las que se enfrentan las organizaciones con respecto a los procesos son: modelar, documentar, ejecutar

y monitorear los procesos; así mismo, mantener actualizada la documentación donde se encuentren descritos.

Existen organizaciones en México con menos de cincuenta participantes, conocidas como PyMES, que tienen que enfrentar las dificultades antes mencionadas con escasos recursos económicos y de personal. Dar solución a estas dificultades con bajo costo, permitiría a las PyMES dedicadas al desarrollo de software, implementar, sin estas dificultades específicas, los Modelos de Mejora de Procesos. Al lograr implementar estos modelos, trabajarían con procesos que ayudarían a producir software de calidad.

1.2 El enfoque de Administración de Procesos de Negocios

Las organizaciones que se dedican a la compra y venta de mercancías o que se dedican a dar servicios y que además estructuran sus actividades en procesos formales, enfrentan también las dificultades antes mencionadas. La manera en que estas organizaciones han buscado dar solución a estas dificultades, es con el uso del enfoque de Administración de Procesos de Negocios (BPM, por sus siglas en inglés). Este enfoque consiste en definir etapas para administrar los procesos. Las etapas son: Diseño, Modelado, Ejecución, Monitoreo y Optimización. Cada proceso pasa por estas etapas. La principal característica del enfoque BPM consiste en que las etapas: Modelado, Ejecución y Monitoreo del proceso, son automatizadas, es decir, existen herramientas que apoyan la realización de estas etapas. Estas herramientas son conocidas como Suites BPM.

1.3 Propuesta

En este trabajo, considerando las limitaciones económicas de las PyMES que se dedican al desarrollo de software, se expone una propuesta para hacer frente a las dificultades anteriormente citadas respecto a los procesos,

haciendo uso del enfoque BPM. También se expone la automatización de los procesos de un Modelo de Mejora de Procesos, en las etapas de Modelado, Ejecución y Monitoreo, etapas definidas de acuerdo al enfoque BPM. La automatización de los procesos se apoya en una Suite BPM que fue construida como parte de este proyecto. La construcción de la Suite BPM fue a partir de los requerimientos identificados de las características de los procesos de los Modelos de Mejora de Procesos. Cabe señalar que los principios identificados en la automatización de las etapas de Modelado, Ejecución y Monitoreo del proceso, se podrán aplicar a todos los procesos del modelo.

En los resultados de este trabajo se presenta el desarrollo de la Suite BPM genérica y extensible, que apoya a una PyME, en el Modelado, Ejecución y Monitoreo de un proceso de un Modelo de Mejora de Procesos.

También se presentan conclusiones sobre la posibilidad de que una Suite BPM pueda ser desarrollada para soportar el ciclo de vida de los procesos de un Modelo de Mejora de Procesos en las etapas de Modelado, Ejecución y Monitoreo. Esta posibilidad se explora por el modelado de los procesos de un Modelo de Mejora de Procesos. También se presentan las observaciones sobre la utilidad de la Suite BPM, que una representante del laboratorio de cómputo del Centro de Investigación de Matemáticas (CIMAT) realizó. La representante tiene experiencia laboral en asesoría a PyMES en implantación de procesos de desarrollo de software y está certificada en MoProSoft.

1.4 Objetivos, Hipótesis y Metodología

En este trabajo se plantea la siguiente hipótesis.

Hipótesis

Una Suite BPM puede ser adaptada para soportar el ciclo de vida de los procesos de un Modelo de Mejora de Procesos en las etapas de Modelado, Ejecución y Monitoreo.

En base a esta hipótesis, se plantean el objetivo general y los objetivos específicos que guiaron la investigación.

Objetivo general

Estudiar la factibilidad de construir una Suite BPM de bajo costo que permita soportar el ciclo de vida de los procesos de un Modelo de Mejora de Procesos de acuerdo a las etapas del enfoque BPM de Modelado, Ejecución y Monitoreo.

Objetivos específicos

Objetivo 1. Soportar dentro de la Suite BPM el modelado de un número variable de procesos de un Modelo de Mejora de Procesos.

Objetivo 2. Soportar la etapa de Ejecución de procesos operativos, en particular en lo que se refiere al intercambio de artefactos y el control de versiones de los mismos.

Objetivo 3. Soportar mecanismos flexibles de colecta de métricas que permitan el monitoreo durante la etapa de Ejecución del proceso.

Objetivo 4. Proveer información a los participantes del proceso sobre la etapa en la cual se encuentra la instancia del proceso durante la etapa de Ejecución del proceso y sobre las tareas que deben realizar.

Para alcanzar estos objetivos, en este trabajo de investigación se siguió la metodología que se presenta a continuación (ver Figura 1).

Metodología

1. **Investigación y redacción del Estado del arte.**
2. **Definición de los requerimientos de la Suite BPM**
   2.1 Identificación de Requerimientos Funcionales.
   2.2 Identificación de Requerimientos No Funcionales.

3. Diseño y Construcción de la Suite BPM.

2.3 Diseño de la arquitectura.

2.4 Desarrollo de componentes.

4. Evaluación de los objetivos de la investigación

4.1 Selección de procesos a modelar.

4.2 Modelado de procesos seleccionados.

4.3 Revisión del cumplimiento de los Requerimientos Funcionales.

5. Análisis e interpretación de resultados y redacción de conclusiones.

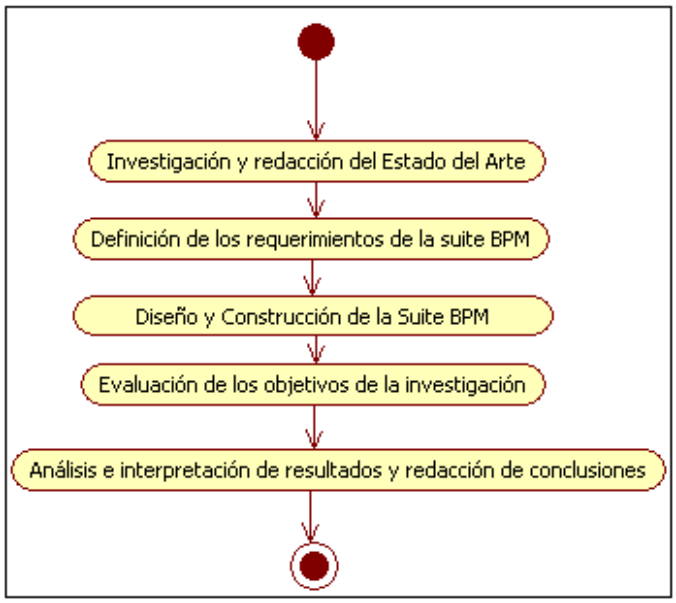


Figura 1: Metodología de la investigación

1.5 Estructura

El presente trabajo está organizado de la siguiente manera: en el capítulo 2 se presentan los principales conceptos sobre procesos y el estado del arte, en el capítulo 3, el desarrollo de la Suite BPM que se construyó, en el capítulo 4 el modelado con la Suite BPM construida, en el capítulo 5 las pruebas realizadas, en el capítulo 6 la discusión crítica y recomendaciones para el trabajo futuro, y para finalizar, en el capítulo 7 las conclusiones.

Capítulo

2 Conceptos y Estado del Arte

El presente trabajo de investigación se enfoca en los procesos dentro de una organización, por lo que es necesario definir que es un proceso, que beneficios existen al trabajar con un enfoque basado en procesos y la relación que existe entre el proceso y la calidad del producto. También es necesario definir cuáles son las técnicas utilizadas para el modelado de procesos, en qué consisten los Modelos de Mejora de Procesos, a que se refiere el enfoque BPM y cuáles son las herramientas que existen para apoyar este enfoque. En las siguientes secciones se detallan estos temas.

2.1 Procesos

2.1.1 Definición de proceso

En este trabajo el proceso se refiere a un conjunto de fases sucesivas de una operación artificial y no de un fenómeno natural. Así un proceso es definido como:

> *Un conjunto de actividades que se ejecutan para lograr un propósito* [2, pág. 11].

Un proceso se formaliza cuando su descripción es plasmada en algún medio que permita que el proceso sea comunicado y almacenado. En la descripción del proceso se detalla lo que se hace en el proceso, quien lo hace, los materiales que se necesitan y que es lo que se produce. La descripción del proceso es llamada ***"Definición del Proceso"*** [3, pág. 18].

Los elementos típicos en la definición de un proceso son los siguientes (estos elementos fueron identificados en el estudio de los Modelos de Mejora de Procesos revisados en este trabajo; de los que se habla más adelante en la sección 2.2):

Entradas: *materiales, datos, Información o salidas de otros procesos.*

Salidas: *información o resultado del proceso. Dentro de las salidas se pueden considerar los "artefactos". Los artefactos son piezas tangibles que resultan de la realización de alguna tarea. Ejemplo: Reporte de Actividades.*

Rol: *función que alguien o algo cumple, también un rol describe un conjunto de habilidades, competencias y responsabilidades, y puede ser cumplido por una persona o varias personas, así mismo una persona puede cumplir varios roles.*

Actividad: *conjunto de tareas propias de un rol.*

Ejemplo: realizar una lluvia de ideas para identificar riesgos posibles.

Tarea: *es la acción concreta que hay que realizar para obtener un resultado deseado, expresado en un producto o subproducto final. Ejemplo: redactar los riesgos posibles de un proyecto determinado.*

Objetivo: *enunciado de un estado deseado hacia el cual está dirigido un proceso.*

Indicadores: *métrica utilizada para medir o comparar los resultados efectivamente obtenidos. Ejemplo: número de riesgos resueltos en el mes que se identifican.*

Metas: *es la cuantificación del objetivo que se pretende alcanzar en un tiempo señalado, con los recursos necesarios. Ejemplo: resolver el 100% de los riesgos identificados en el mismo mes.*

Cuando se llevan a cabo las actividades y las tareas del proceso, involucrando todos los elementos de la Definición del Proceso, se dice que se ha creado una **"Instancia del Proceso"**.

En la siguiente sección se define lo que es un Proceso de Negocio.

2.1.2 Proceso de Negocio

Las organizaciones que identifican y documentan los elementos para la definición de sus procesos, asignando recursos humanos, materiales y financieros, son organizaciones que trabajan con un enfoque basado en procesos. En el contexto de este trabajo, la palabra proceso hace referencia a Proceso de Negocio, un Proceso de Negocio se define como:

Un Proceso de Negocio es un conjunto de actividades o tareas ordenadas para producir un determinado producto o servicio de utilidad para un cliente interno o externo de la organización *[4, pág. 5]*.

De acuerdo a esta definición, los Procesos de Negocio son cualquier proceso que ocurre dentro de la organización.

Los procesos que ocurren dentro de la organización, según Pérez *[5, págs. 83-87]* y su división en categorías de acuerdo a la misión de los procesos, son los siguientes:

> ***Procesos Operativos:*** *combinan y transforman recursos para obtener el producto o proporcionar el servicio conforme a los requisitos del cliente externo, aportando en consecuencia un alto valor añadido.*
>
> ***Procesos de Apoyo:*** *proporcionan las personas y los recursos físicos para el resto de los procesos y conforme a los requisitos de sus clientes internos.*
>
> ***Procesos de Gestión:*** *mediante actividades de evaluación, control, seguimiento y medición aseguran el funcionamiento controlado del resto de los procesos, además proporcionan la información necesaria para la toma de decisiones. Son transversales a toda la empresa y se tiene que identificar su interacción con los procesos operativos y de apoyo, en la toma de datos y la entrega de información.*
>
> ***Procesos de Dirección:*** *son los procesos de determinación, despliegue, seguimiento y evaluación de objetivos.*

Los beneficios de una organización al trabajar con un enfoque basado en procesos, se presentan a continuación *[6, sección 2.5]*.

> ***Estabilidad operacional.*** *Al definir y establecer procesos es posible hacer predicciones en cuanto a tiempo, costo y logro de objetivos.*
> ***Identidad cultural (Integración).*** *Se transmite de manera formal a los participantes del proceso las prácticas de "éxito". Las personas que saben de*

manera formal las tareas que les corresponden y los roles que asumen en la organización, empiezan a tener identidad y sentido de pertenencia.

Mayor rendimiento. *Los esfuerzos organizados se dirigen a objetivos bien definidos, no hay pérdida de esfuerzos.*

Reducción de costos. *Por ejemplo, se reducen costos cuando los defectos son encontrados en fases tempranas.*

Los procesos tienen un ciclo de vida y en este trabajo este término es utilizado para designar las etapas por las que pasa un proceso desde que nace hasta que es desechado, sin entrar en detalle de cuantas y cuáles son estas etapas para no perder generalidad.

2.1.3 Procesos de Desarrollo de Software

Dentro de los procesos operativos, un proceso que es de particular importancia en el contexto de este trabajo es el "Proceso de Desarrollo de Software", que se define como:

> *Los procesos de desarrollo de software son un conjunto de actividades, métodos, prácticas y modificaciones que las personas realizan para desarrollar y mantener productos de software* *[3, pág. 15]*.

La definición de los procesos de desarrollo de software se establece de acuerdo a un plan respecto a las herramientas (hardware o software), métodos y personas involucradas en las tareas del desarrollo *[3, pág. 18]*. Un ejemplo de procesos de desarrollo de software es el RUP (Rational Unified Process) *[7]*.

2.1.3.1 Los procesos de desarrollo de software y la calidad.

Inicialmente los primeros procesos que fueron tomados como objeto de estudio fueron los procesos de manufactura. Estos procesos fueron estudiados con el fin de fabricar productos con la menor cantidad de defectos. La evolución de la aplicación de los principios descubiertos en estos estudios, hacia su utilización

en el desarrollo de productos de software, se explica brevemente a continuación.

La calidad se define como el cumplimiento de los requerimientos del cliente *[8, pág. 17]*. En 1931, Walter A. Shewhart comenzó a trabajar en mejorar los procesos de manufactura utilizando los Principios de Control Estadístico de la Calidad *[8, págs. 107, 153]*. Más tarde, estos principios fueron refinados por W. Edwards Deming, Phillip Crosby y Joseph Juran. Deming demostró que estos principios no sólo podían ser aplicados a procesos de manufactura *[8, pág. 14]*.

En 1968, durante la reunión que llevo a cabo la Organización del Tratado del Atlántico Norte (OTAN) con el fin de resolver la falta de calidad de los productos de software, se acuñó el término “Crisis del Software” *[9, pág. 1]*. En los años ochenta el término “Crisis del software” seguía vigente. Fue entonces cuando Watts Humphrey, Ron Radice y otros, extendieron los Principios de Control Estadístico de la Calidad y empezaron a aplicarlos a los procesos de desarrollo del software *[10, págs. 2, 3]*. Humphrey dijo: “Un primer paso para solucionar los problemas del software es tratar a todas las tareas del desarrollo del software como un proceso que puede ser controlado, medido y mejorado” *[3, pág. 15]*.

A principios de los noventa surgieron varias publicaciones de mejora de procesos de software, concepto conocido internacionalmente como SPI (Software Process Improvement). Fue entonces en este contexto que nacieron los Modelos de Mejora de Procesos *[11, pág. 7]* que se describen en la sección siguiente.

2.2 Modelos de Mejora de Procesos para el desarrollo de software.

Los resultados obtenidos al ejecutar el proceso se miden y se comparan con las metas previamente fijadas, la distancia entre la situación actual y la deseada se puede entender como un problema a resolver, como una situación a mejorar

[12, pág. 44]. Mejorar un proceso significa estudiarlo, documentarlo (definirlo y modelarlo), medir sus resultados y encontrar soluciones más eficientes y eficaces *[13, pág. 170]*. En la siguiente sección se define con más detalle en qué consiste la mejora de procesos y se definen también los Modelos de Mejora de Procesos.

2.2.1 Mejora continua o iterativa de procesos

Como lo indica Sommerville *[14, págs. 608, 609]*, la mejora de procesos es una actividad cíclica (ver Figura 2) que tiene tres estados principales: 1) Medición de los atributos del proyecto actual o del producto, 2) Análisis y 3) Introducción de los cambios al proceso identificados en el análisis. En este trabajo, cuando se hable de mejora de manera continua o iterativa, se hará referencia a este ciclo.

Un **modelo**, de acuerdo a la definición del Diccionario de la Lengua Española *[15]*, es un arquetipo, o punto de referencia para imitarlo o reproducirlo.

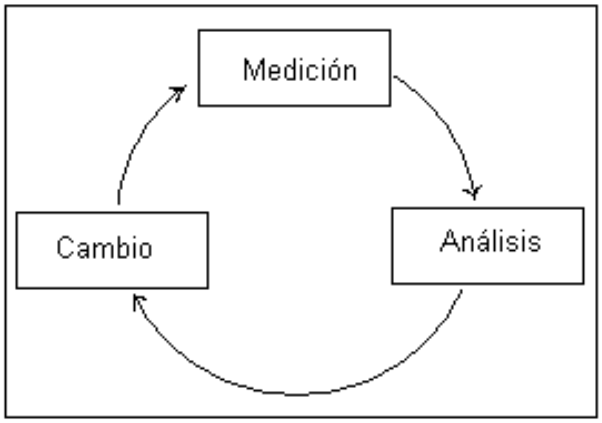


Figura 2: Ciclo de Mejora de Procesos.

Con respecto a los Modelos de Mejora de Procesos, en la revisión de la literatura no se encontró una definición específica, por lo que en este trabajo, se definen a los Modelos de Mejora de Procesos como*: "Modelos que presentan las mejores prácticas que le permiten a una organización implementar procesos que pueden ser controlados, medidos y mejorados de acuerdo al Ciclo de Mejora de Procesos"*.

De acuerdo a la definición de Modelos de Mejora de Procesos presentada, existen varios Modelos de Mejora de Procesos de Desarrollo de Software, entre los cuales se encuentran CMMI *[16]*, MoProSoft *[17]* y Tritón *[18]*. En las siguientes secciones se resumen estos tres modelos.

2.2.2 Modelo de capacidad y madurez integrado (CMMI).

CMMI es un modelo que guía a las organizaciones en el establecimiento de objetivos y prioridades en la mejora de sus procesos. Guía también en mejorar los procesos y en la forma de asegurar que se han creado procesos duraderos, con capacidad y madurez *[16, pág. 1]*.

La *capacidad del proceso* describe el alcance de los resultados que pueden lograrse siguiendo el proceso de software. La *capacidad del proceso del software* de una organización proporciona una manera de predecir, para el siguiente proyecto de la organización, la probabilidad de un determinado resultado. CMMI guía en la valoración del *nivel de madurez* que la organización ha alcanzado. Un *nivel de madurez* es un estado de evolución hacia el logro del proceso de software maduro.

CMMI establece cinco niveles de madurez, estos son: Inicial, Administrado, Definido, Administrado Cuantitativamente y Optimizado *[16, págs. 11-13]*. CMMI define las características que corresponden a cada nivel. El *nivel de madurez* de una organización se define de acuerdo al cumplimiento de las características de un nivel determinado.

2.2.2.1 Categorías y procesos de CMMI

CMMI es un modelo de referencia con “Áreas de Proceso”. Un Área de Proceso es un grupo de prácticas que se relacionan entre sí y que al llevarse a cabo, satisfacen los objetivos de la Área de Proceso. Las Áreas de Proceso están divididas en cuatro categorías que se presentan a continuación.

Gestión de Procesos. En esta categoría se agrupan las Áreas de Procesos que contienen las actividades que son transversales al proyecto, relacionadas con la definición, planeación, asignación de recursos, implementación, ejecución, supervisión, control, evaluación, medición y mejora de los procesos.

Gestión de proyectos. En esta categoría se agrupan las Áreas de Procesos que cubren las actividades de administración del proyecto relacionadas con la planeación, monitoreo y control del proyecto.

Ingeniería. En esta categoría se agrupan las Áreas de Procesos que cubren las actividades de desarrollo y mantenimiento del producto de software.

Soporte. En esta categoría se agrupan las Áreas de Procesos que cubren las actividades de soporte para el desarrollo y mantenimiento del producto de software.

El detalle de las Áreas de Procesos se presenta en la Tabla 1.

Tabla 1: Categorías y Áreas de procesos de CMMI.

Categoría	Área de Proceso	Descripción
Gestión de Procesos	**Enfoque de proceso organizacional**	Planifica y establece la mejora de procesos para la organización, toma en cuenta los puntos fuertes y débiles de los procesos de la organización.
	Entrenamiento organizacional	Desarrolla las habilidades y proporciona al personal el conocimiento para que pueda realizar sus funciones de manera eficaz y eficiente.
	Definición de proceso organizacional	Establece y actualiza los bienes de los procesos organizacionales.
	Rendimiento de los procesos de la organización	Establece y mantiene una interpretación cuantitativa de los resultados de los procesos estándar de la organización.
	Innovación organizacional y desarrollo	Selecciona e implementa las mejoras increméntales e innovadoras para los procesos de la organización.

Gestión de Proyectos	**Planificación del proyecto**	Establece y corrige los planes que definen las actividades del proyecto.
	Monitoreo y control del proyecto	Proporciona una visión de los avances del proyecto con la finalidad de llevar a cabo las acciones correctivas a los procesos cuando el avance del proyecto se desvía del plan de manera significativa.
	Administración de acuerdos con proveedores	Gestiona la adquisición de productos con los proveedores con los que existe un acuerdo formal.
	Integración del equipo de trabajo	Forma y mantiene un equipo integrado para la elaboración de productos.
	Administración de la integración de proveedores	Identifica donde se pueden adquirir los productos que pueden ser utilizados para satisfacer los requerimientos del proyecto.
	Administración de la integración del proyecto	Adapta los procesos organizacionales al proyecto y establece la visión del proyecto.
	Administración de riesgos	Identifica los problemas potenciales antes de que ocurran, para planificar y llevar a cabo las actividades necesarias en el desarrollo del producto o proyecto y así disminuir los impactos negativos.
	Administración cuantitativa del proyecto	Define el proyecto cuantitativamente para establecer la calidad y los objetivos de desempeño del proyecto.
Ingeniería	**Administración de requerimientos**	Produce y analiza los requerimientos del cliente, del producto, y los componentes del producto.
	Solución técnica	Diseña, desarrolla e implementa soluciones a los requerimientos.
	Integración de producto	Arma el producto a partir de sus componentes asegurándose de que el producto funciona correctamente, también se encarga de entregar el producto.
	Verificación	Garantiza que los productos cumplen con los requerimientos especificados.
	Desarrollo de requerimientos	Presenta y analiza los requerimientos de los clientes, del producto y de los componentes de los productos.

	Validación	Demuestra que un producto o un componente del producto, cumple con su uso específico al colocársele en el entorno para el cual fue construido.
Soporte	**Análisis causal y resolución**	Identifica las causas de los defectos y otros problemas y toma medidas para evitar que ocurran en el futuro.
	Aseguramiento de la calidad de procesos y productos	Proporciona al personal y a la administración una visión objetiva de los productos y de los procesos adaptados a estos productos.
	Entorno organizacional para la integración	Proporciona la infraestructura para el proceso de desarrollo de un producto integrado.
	Análisis y decisión de soluciones	Analiza posibles decisiones mediante un proceso de evaluación formal que evalúa alternativas a partir de criterios establecidos.
	Administración de la configuración	Establece y mantiene la integridad de los productos mediante la identificación y control de la configuración, lo que representa el estado de configuración y los resultados de las revisiones de la configuración.
	Medición y análisis	Desarrolla y mantiene la capacidad para tomar mediciones que se utilizan en la administración de las necesidades de información.

2.2.3 MoProSoft

La industria mexicana de software es en su mayoría pequeña y mediana. Este tipo de organizaciones, frecuentemente no pueden adoptar modelos tales como CMMI, por su alto costo. Por lo anterior, en el año 2005 se creó MoProSoft, que es un modelo de calidad de software para la industria mexicana de software, basado en las mejores prácticas internacionales. Sus características son: fácil de entender, fácil de aplicar, no es costoso en su adaptación y es la base para alcanzar evaluaciones exitosas con otros modelos o normas, tales como ISO 9001:2000 o CMM V1.1 *[17, pág. 3]*.

En MoProSoft se identifican categorías y procesos los cuales se describen en la siguiente sección.

2.2.3.1 Categorías y procesos de MoProSoft.

En la Figura 3 se muestra la jerarquía de las categorías y los procesos de MoProSoft. A continuación se describen las categorías *[17, pág. 10]* y en la Tabla 2 se describen los procesos *[17, pág. 12]*.

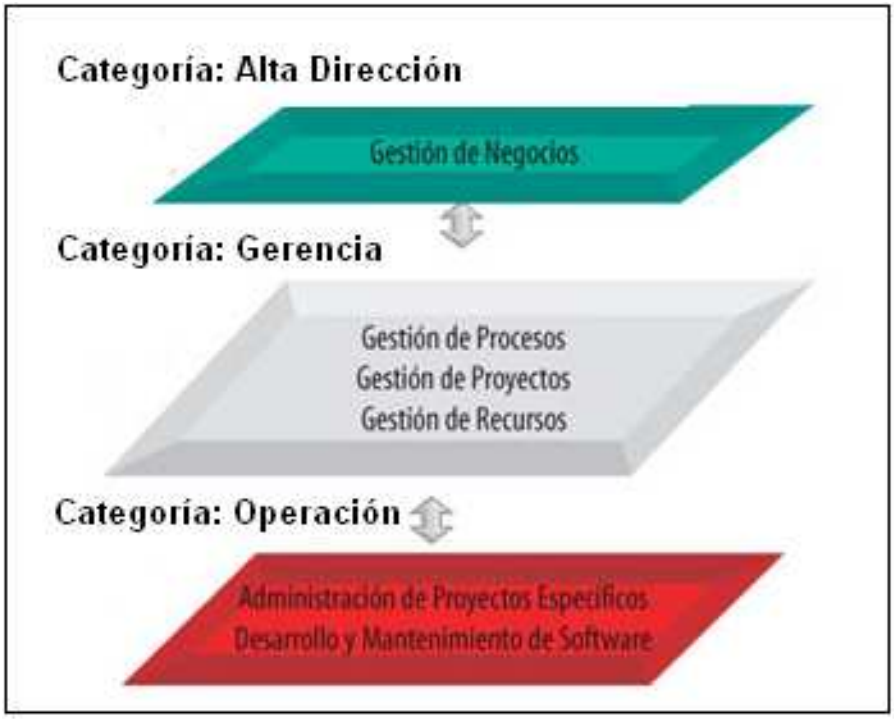


Figura 3: Categorías y procesos de MoProSoft.

Categoría de Alta Dirección. En esta categoría se agrupan los procesos relacionados con la gestión de negocio, es decir, todo lo relacionado a las directrices que sigue la organización como entidad.

Categoría de Gerencia. En esta categoría se agrupan los procesos que proporcionan los elementos para el funcionamiento de los procesos de la categoría de operación.

Categoría de Operación. En esta categoría se agrupan los procesos que abordan las prácticas de los proyectos de desarrollo y mantenimiento de software.

Tabla 2: Categorías y procesos de MoProSoft

Categoría	Proceso	Descripción
Alta Dirección	**Gestión de Negocio**	Establece la razón de ser de la organización, sus objetivos y las condiciones para lograrlos.
Gerencia.	**Gestión de Procesos**	Identifican y documenta los procesos que requiere la organización.
	Gestión de Proyectos	Identifica y hace cumplir los proyectos internos y de clientes.
	Gestión de Recursos	Consigue y proporciona a la organización recursos humanos, infraestructura, proveedores, ambiente de trabajo adecuado, crea y mantiene la base de conocimiento de la organización.
Operación	**Administración de Proyectos Específicos**	Establece y lleva a cabo sistemáticamente las actividades que permitan cumplir con los objetivos de un proyecto.
	Desarrollo y Mantenimiento de Software	Realización sistemática de las actividades de análisis, diseño, construcción, integración y pruebas de productos de software.

2.2.4 Tritón

Tritón, es un Modelo de Mejora de Procesos desarrollado por el CIMAT (Centro de Investigación en Matemáticas, A.C.) en Zacatecas. Tritón nace para mejorar los procesos de las organizaciones dedicadas al desarrollo de software de la región. Combina conceptos de MoProSoft, Administración de Proyectos, CMMI V1.2, la norma NMX-I-045-NYCE-2005, PSP (Personal Software Process) y TSP (Team Software Process) *[18]*. Tritón también está diseñado para pequeñas organizaciones como MoProSoft, pero a diferencia de este, provee un mayor detalle sobre la manera de realizar los procesos.

2.2.4.1 Categorías y procesos de Tritón.

En la Tabla 3 se presentan las categorías y procesos de Tritón.

Tabla 3: Categorías y procesos de Tritón

Categoría	Proceso	Descripción
Nivel Organizacional.	**Gestión de Negocio**	Establece la razón de ser de la organización, sus objetivos y las condiciones para lograrlos.
	Recursos Humanos y Ambiente de Trabajo	Proporciona los recursos humanos adecuados para cumplir las responsabilidades asignadas a los roles.
	Bienes, Servicios e Infraestructura	Proporciona proveedores de bienes, servicios e infraestructura que satisfagan los requisitos de adquisición de los procesos y proyectos.
Administración de Procesos.	**Conocimiento de la Organización**	Mantiene disponible y administra la base de conocimiento que contiene la información y los productos generados.
	Gestión de Procesos	Establece y mantiene utilizable un conjunto de bienes de procesos.
	Entrenamiento Organizacional	Desarrolla habilidades y conocimiento de personas, a fin de que puedan realizar sus roles de una manera efectiva y eficientemente.
Administración de Proyectos.	**Gestión de Proyectos**	Asegura que los proyectos contribuyan al cumplimiento de los objetivos y estrategia de la organización.
	Lanzamiento	Inicia el desarrollo de software, se realizan como actividades principales: lanzamiento, estrategia, planeación y postmortem.
	Relanzamiento	Se ejecuta cuando se empieza un nuevo ciclo en el desarrollo de un proyecto.
	Postmortem	Se realiza al finalizar un ciclo o un proyecto
	Iteración	Permite ejecutar en un ciclo uno o todos los procesos según la estrategia seleccionada, con la finalidad de obtener un release del producto.
	Desarrollo de Software	Muestra la ejecución de los “n” ciclos planeados para realizar el producto.
	Monitoreo y Control de Proyectos	Proporciona un entendimiento del progreso del proyecto y las acciones correctivas apropiadas.

Ingeniería	**Administración de Riesgos**	Identifica problemas potenciales antes de que ocurran y planear las actividades de manejo de riesgos.
	Pre-análisis	Determina los componentes y sus relaciones de un sistema basándose en los requerimientos de alto nivel.
	Requerimientos	Desarrollo y administración de requerimientos.
	Diseño de Alto Nivel	Define la estructura general del producto a desarrollar.
	Implementación	Implementa los elementos del diseño en términos de elementos de implementación.
	Pruebas	Descubre fallas en el producto que se está probando.
	Liberación	Entrega formalmente el sistema y su documentación al usuario, con la capacitación correspondiente.
	Mantenimiento	Modifica una aplicación o un componente después de liberado.
	Verificación	Asegura que el producto de trabajo seleccionado reúne sus requerimientos especificados.
	Validación	Demuestra que un producto de trabajo cumple su uso deseado cuando se encuentra en un entorno deseado.
Soporte	**Inspección**	Proceso que sirve de base para revisar documentos, código, entre otros.
	Aseguramiento de la Calidad del Producto y del Proceso	Define las actividades para asegurar de una manera objetiva que los productos de software y los procesos son conformes a sus requisitos especificados y se ajustan a sus planes establecidos.
	Medición y Análisis	Desarrolla y mantiene la capacidad de medición que es usada para soportar la administración de las necesidades de información
	Administración de la Configuración	Administra la evolución de un proyecto de software, desde su desarrollo hasta su mantenimiento.

En la siguiente sección se hace una comparación de los modelos CMMI, MoProSoft y Tritón.

2.2.5 Comparación de CMMI, MoProSoft y Tritón

En la Tabla 4 se muestra la comparación de los modelos CMMI, MoProSoft y Tritón. En esta tabla se observan en la columna izquierda, los elementos de cada modelo y, en la fila superior, el nombre del modelo. Se observa también que CMMI y MoProSoft indican que actividades llevar a cabo pero no como se llevan a cabo a diferencia del modelo Tritón cuando para el desarrollo de sus procesos toman como base TSP.

Tabla 4: Comparación de los modelos CMMI, MoProSoft y Tritón

Modelo / Elemento	CMMI	MoProSoft	Tritón
Propósito	√	√	√
Tarea y Actividades	√	√	√
Criterios de entrada	√	√	√
Criterios de salida	√	√	√
Diagrama de flujo del proceso	x	√	√ (cuando se usa de base MoProSoft.)
Roles por actividad	√	√	√ (cuando se usa de base MoProSoft.)
Responsables del proceso	x	√	√ (cuando se usa de base MoProSoft.)
Objetivos medibles	√	√	√ (cuando se usa de base MoProSoft.)
Métricas	√	√	√ (cuando se usa de base MoProSoft.)
Actividades de comunicación	√	√	√ (cuando se usa de base MoProSoft.)
Descripción de cómo llevar a cabo tareas	x	x	√ (cuando para el desarrollo del proceso se usa de base TSP)
Verificación	√	√	√
Validación	√	√	√
Tamaño de organización al que está dirigido	Grandes	De pequeñas a grandes	De pequeñas a grandes

2.2.6 Características en común de los procesos de CMMI, MoProSoft y Tritón

El estudio de los tres Modelos de Mejora de Procesos, CMMI, MoProSoft y Tritón, permite concluir que todo Modelo de Mejora de Procesos implementa procesos que presentan las siguientes características:

1. Los procesos están documentados.
2. Cada proceso está compuesto por tareas.
3. Cada tarea es asignada a un rol.
4. Cada proceso es instanciado.
5. Cada tarea tiene un momento de inicio.
6. Cada tarea tiene un momento de terminación.
7. Cada Instancia del Proceso tiene una o más entradas que son utilizadas por una o más tareas.
8. Cada Instancia del Proceso tiene una o más salidas producidas por una o más tareas de la misma Instancia del Proceso.
9. A cada Instancia del Proceso se le da seguimiento.
10. Cada proceso es continuamente mejorado.

De acuerdo a la lista de características anteriores, es necesario para toda organización:

- Tener documentados los procesos.
- Poder seguir las instancias de los procesos.
- Poder mejorar los procesos de manera continua.

Para tener documentados los procesos es necesario contar con técnicas para modelarlos, algunas técnicas se presentan en la siguiente sección.

2.3 Técnicas de modelado de procesos

Así como un programa de software, que está plasmado en un lenguaje, define un proceso que una computadora debe seguir para producir un resultado, un proceso debe estar descrito de alguna manera con el fin de que las personas que participan en él puedan ejecutarlo. Un “Modelo” de proceso se refiere justamente a esta descripción.

En el modelo de un proceso se representan los elementos típicos de un proceso vistos en la sección 2.1.1. Un proceso es modelado a través de una representación gráfica, textual o textual y gráfica. Estas representaciones se explican con mayor detalle en las siguientes secciones.

2.3.1 Representación gráfica.

Una gráfica es una descripción que se representa por medio de figuras o signos. En las técnicas de modelado de procesos, al conjunto de estas figuras o signos y sus nombres, se le llama “notación”. Cuando se modelan los procesos, la notación representa los elementos de la Definición del Proceso. También existen signos para expresar el orden secuencial de las tareas.

El modelado de procesos con representación gráfica puede llevarse a cabo, por ejemplo, con Diagramas de Actividades de UML. Estos diagramas tienen su propia notación. Existen esfuerzos para estandarizar la notación, uno de estos esfuerzos es la Notación de Modelado de Procesos de Negocios (BPMN por sus siglas en inglés) *[19, pág. 1]*. El modelado de procesos se puede representar también gráficamente con el Lenguaje de Definición de Procesos Java (JPDL por sus siglas en inglés) o de XPDL (Lenguaje de Definición de Procesos XML). A continuación se comentan estas técnicas de modelado de procesos.

Diagramas de Actividad UML. *El Lenguaje de Modelado Unificado (UML por sus siglas en inglés) es un lenguaje visual para modelar y comunicar sistemas utilizando diagramas y texto [20, pág. 4]. Entre estos diagramas*

existen los Diagramas de Actividad UML. Los diagramas de actividad UML muestran actividades y transiciones entre estas, su notación tiene los siguientes elementos: Estado de Inicio, Actividad, Transición, Bifurcación, División y Unión, Canal y Estado Final.

BPMN (Business Process Modeling Notation). *La Iniciativa de Administración de Procesos de Negocios (BPMI, por sus siglas en inglés) desarrolló una notación estándar llamada Business Process Modeling Notation (BPMN) o Notación de Modelado de Procesos de Negocios [19, pág. 1]. Varios autores coinciden que BPMN es el estándar con mayor aceptación en el modelado de procesos. Dentro de esta notación se identifican cuatro categorías básicas de elementos que son: Objetos de Flujo, Objetos Conectores, Artefactos y Canales de Nado [19, pág. 15].*

JPDL. *JPDL, a diferencia de los Diagramas de Actividades UML, es un lenguaje de ejecución de procesos, es decir, cada elemento de la notación de JPDL se describe en lenguaje XML que es leido por un Motor de Ejecución, del que se hablará mas adelante en la sección 2.4.2, este motor iniciará la indicación que se lleven a cabo las tareas de la Definición del Proceso modelado. JPDL se describirá con más detalle en la sección 3.3.3.5.*

XPDL. *También es un lenguaje de ejecución de procesos como JPDL, es un formato de archivo basado en XML que puede ser usado para intercambiar modelos de procesos de negocios entre distintas herramientas. Es un formato de archivo que representa la notación del modelo de la Definición del Proceso. Tiene el tamaño y las coordenadas X e Y del nodo. Tiene un concepto de líneas que señalan el camino a seguir. Los nodos y las líneas tienen atributos que pueden especificar información ejecutable tales como roles, descripción de actividades, temporizadores, llamadas a servicios Web. Desde la versión XPDL 2.0, contiene extensiones para ser capaz de representar todos los aspectos de BPMN, es decir, partiendo de un proceso modelado con la notación BPMN es posible obtener una implementación XPDL y viceversa [21].*

BPEL (Lenguaje de Ejecución de Procesos de Negocios) no tiene notación de modelado propia como JPDL por lo que usualmente es acompañado por BPMN, es decir, a partir de un proceso modelado con notación BPMN es posible pasar a un proceso en BPEL[1].

2.3.2 Representación Textual.

El modelado textual de un proceso puede llevarse a cabo a través de guías (scripts) como por ejemplo aquellas que plantean el PSP (Personal Software Process) ó TSP (Team Software Process).

El PSP consiste en un conjunto de métodos, formas, y guías (scripts) que muestran a los desarrolladores de software como planificar, medir y administrar su trabajo *[23, pág. ix]*. Los desarrolladores usan los scripts TSP para aplicar conceptos de integración de equipos, tales como la definición de las funciones del equipo.

Scripts PSP / TSP. Los scripts PSP y los scripts TSP son guiones que definen los pasos para cada parte del proceso. Están diseñados con instrucciones cortas y precisas, de acuerdo a *[23, pág. 9]*, los scripts están diseñados para no ser utilizados como una guía detallada, porque a diferencia de los scripts que sirven como tutoriales, los scripts PSP/TSP son utilizados para guiar a los ingenieros de software en un proceso que ellos ya entienden y verificar que se han completado todas las tareas.

2.3.3 Representación Gráfica y Textual.

Existen también técnicas que apoyan el modelado de procesos con dos representaciones, gráfica y textual, como lo es en el caso de EPF (Eclipse Process Framework).

[1] BPEL consiste en un lenguaje de orquestación basado en XML *[22]*.

EPF (Eclipse Process Framework). *Eclipse Process Framework Composer (EPF Composer), es una herramienta para ingenieros de procesos y administradores de proyectos quienes son responsables de mantener e implementar procesos para la organización o para proyectos individuales [24, sitio Web]. EPF cuenta con la ventaja de soportar la posibilidad de exportar el modelo del proceso como una página Web.*

Las distintas notaciones de las técnicas de modelado vistas en esta sección tienen en común que en todas se representan: el inicio del proceso, las tareas a realizarse, el flujo que sigue el proceso, los roles y las salidas del proceso.

2.4 Administración de Procesos de Negocios (BPM).

El enfoque BPM es una evolución natural y convergente de varios campos: Metodologías de Desarrollo de Software, Tecnología de Aplicación Empresarial y Teoría de Administración. Estos campos se han fusionado en lo que ahora se conoce como Administración de Procesos de Negocios *[25, pág. 6]*.

2.4.1 Enfoque BPM

Frecuentemente, el desarrollo de software busca automatizar tareas correspondientes a procesos dentro de una organización. Por ejemplo, en una biblioteca, un software administrativo puede ayudar a realizar los pasos del proceso de préstamo de libros tales como la consulta de catálogos y la realización del préstamo en sí. Este enfoque tiene ciertas limitantes, en particular, es difícil adaptar las herramientas a cambios en los procesos. BPM busca resolver esta problemática a través del uso de herramientas, llamadas *Suites BPM*, que permiten modelar y automatizar la ejecución de los procesos. Estas herramientas permiten realizar modificaciones al proceso sin necesidad de modificar el software.

Por otro lado, BPM cubre aspectos adicionales que incluyen el diseño de los procesos con el fin de que éstos estén alineados a los objetivos de negocio. En

ese sentido, BPM es distinto al enfoque de desarrollo clásico que asume que los procesos son adecuados y simplemente se limita a automatizar parte de ellos. Adicionalmente, BPM considera el Ciclo de Mejora de Procesos (ver sección 2.2.1) de forma intrínseca a través de la colecta de datos de la ejecución del proceso y el soporte a la modificación de los modelos de proceso dentro de la Suite.

El enfoque BPM plantea que el ciclo de vida de los procesos debe consistir en una serie de etapas, estas son: Diseño, Modelado, Ejecución, Monitoreo y Optimización, el detalle de estas etapas se presenta en la sección 2.4.3.

2.4.2 Suites BPM

El enfoque BPM se apoya en un software llamado Suite BPM donde el usuario puede crear el modelo gráfico de un proceso, que es usado para controlar la ejecución de instancias del proceso, tomando en cuenta su interacción con humanos y otras aplicaciones. Adicionalmente, durante la ejecución de la Instancia del Proceso se colectan datos, con el fin de permitir el análisis del progreso y la mejora continua o iterativa *[25, pág. 12]*. Las Suites BPM son entonces, productos de software que pueden ser utilizados para modelar, ejecutar y monitorear los procesos de distintas organizaciones.

Las Suites BPM, están compuestas por una serie de componentes que forman una estructura o arquitectura que les permiten apoyar el enfoque BPM *[25, pág. 16]* (ver Figura 4) estos son los siguientes.

1. **Modelador Gráfico de Procesos.** El Modelador Gráfico de Procesos es un componente que permite crear y editar el modelo gráfico de la definición de los procesos a partir de elementos disponibles en el catálogo de la notación (ver sección 2.3.1). El Modelador Gráfico de Procesos es usado por el Administrador del Proceso (Administrador del Proceso es una categoría de roles que se describe más adelante).

2. **Motor de Ejecución de Procesos.** El Motor de Ejecución de Procesos es un componente de software que en tiempo de ejecución crea una Instancia del Proceso a partir de su definición, cuando el Administrador del Proceso le indica crear una instancia, y tiene interacción con los participantes de la Instancia del Proceso *[26, pág. 273]*.

3. **Interfaz de Usuario.** La Interfaz de Usuario es un componente que permite la interacción entre los participantes del proceso y la suite. La interfaz de usuario provee información a los participantes del proceso sobre las tareas que tienen que realizar y recibe retroalimentación (interacción) por parte de ellos con el fin de poder seguir avanzando la ejecución de la Instancia del Proceso.

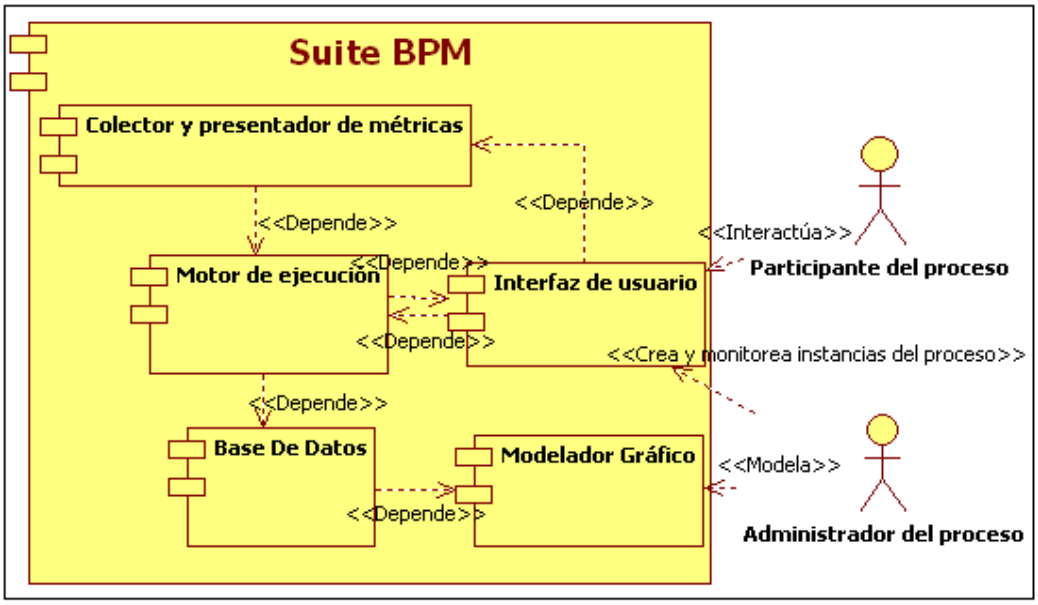


Figura 4: Componentes básicos de una Suite BPM

4. **Colector y Presentador de Métricas.** Este componente colecta datos de la ejecución de la Instancia del Proceso y los presenta para facilitar su análisis y apoyar en el monitoreo.

5. **Base de Datos.** Este componente se encarga de almacenar las definiciones de los procesos y el estado de sus instancias.

La *Interfaz de Usuario* es usada por roles divididos en dos categorías: el Participante del Proceso y el Administrador del Proceso (estas categorías se describen en la siguiente sección).

En la actualidad existen Suites BPM comerciales y open source, que presentan distintas características, aunque todas presentan las características antes mencionadas.

2.4.3 Ciclo de vida BPM

Como se mencionó en la sección 2.4.1 el enfoque BPM, también contempla el Ciclo de Mejora de Procesos. El Ciclo de Mejora de Procesos hace que las etapas que plantea el enfoque BPM se consideren como un ciclo al que se nombra como el Ciclo de Vida BPM. En este ciclo intervienen dos categorías de roles, el Administrador del Proceso y el Participante del Proceso. En la primera categoría están agrupados el o los roles autorizados para crear instancias de la Definición del Proceso y para modificar definiciones del proceso. En la segunda categoría se agrupan los distintos roles que intervienen en el proceso. En la Figura 5 se muestran las etapas del ciclo de vida BPM, estas etapas son las siguientes *[27, pág. 92]*.

> ***Diseño.*** *En esta etapa se crea o modifica conceptualmente la Definición del Proceso. (identificando los elementos vistos en la sección 2.1.1).*
>
> ***Modelado.*** *En esta etapa se hace un modelo gráfico de la Definición del Proceso. Cuando se hace uso de una Suite BPM, ésta provee el Modelador Gráfico de Procesos para apoyar en el modelado la Definición del Proceso.*
>
> ***Ejecución.*** *En esta etapa el Administrador del Proceso crea una Instancia del Proceso y los participantes del proceso intervienen. Esta etapa se lleva a cabo apoyada en la Suite BPM, cuando el Administrador del Proceso indica crear una Instancia del Proceso, el Motor de Ejecución de Procesos*

crea la instancia y los participantes del proceso reciben a través de la interfaz de usuario las tareas a realizar.

Monitoreo. *En esta etapa se da seguimiento a la Instancia del Proceso por medio de mediciones llamadas métricas. Esta etapa se lleva a cabo apoyada en la Suite BPM con el Colector y Presentador de Métricas.*

Optimización. *En esta etapa los datos de las métricas tomadas durante el seguimiento de la Instancia del Proceso, son analizados e interpretados para identificar oportunidades de mejora del proceso y plantear modificaciones a la Definición del Proceso.*

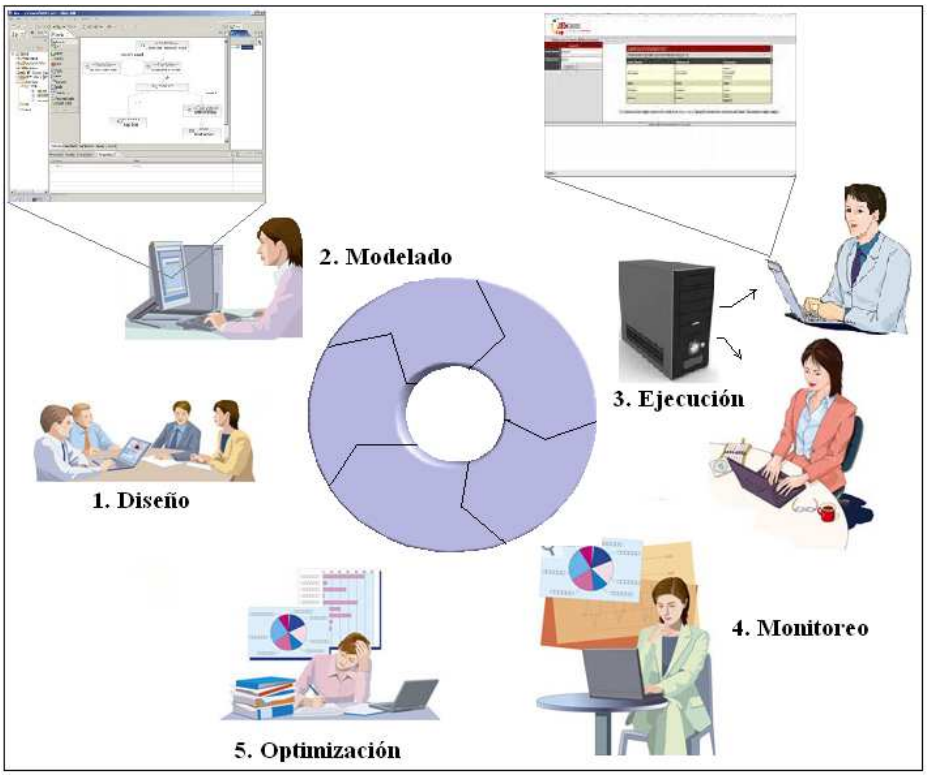


Figura 5: Ciclo de vida BPM

Como se puede observar, la Suite BPM juega un papel importante en las etapas de Modelado, Ejecución y Monitoreo. Esto tiene los siguientes beneficios *[25, págs. 12, 13]*.

Incremento en la productividad. Una Suite BPM presenta a los participantes del proceso una lista de tareas que asegura que se está trabajando sobre el ítem de mayor prioridad, mejorando la velocidad del proceso.

Incremento en la adherencia al proceso. Una Suite BPM va presentando a los participantes del proceso cada tarea en un orden estricto y cada tarea es presentada hasta que es concluida la tarea antecesora, de esta manera, se asegura que el participante se apegue o adhiera al proceso.

Las organizaciones pueden adaptarse a los cambios de su entorno con más agilidad porque pueden cambiar más rápidamente sus procesos. Una Suite BPM apoya en el modelado del proceso, los cambios afectan sólo al modelo, no al código del sistema esto hace generalmente más fácil cambiar el orden en que los participantes hacen las tareas.

La organización tiene la capacidad de soportar el aumento de participantes en el proceso. Las organizaciones que intentan escalar, por ejemplo, de 10 a 100 participantes en sus procesos, a menudo pierden el control sobre los procesos porque no cuentan con un software que las apoye. Con el uso de una Suite BPM es posible soportar cambiar la Definición del Proceso incorporando un mayor número de participantes en la Instancia del Proceso sin perder el control sobre la instancia.

Mejora la comunicación, cooperación, coordinación. Las Suites BPM permiten presentar en las interfaces de usuario de un equipo el resultado alcanzado por otro equipo, reduciendo la necesidad de que los equipos sean expertos en comunicación y cooperación.

2.4.4 Comparación de Suites BPM.

En este trabajo se investigaron tres herramientas open source que apoyan al Ciclo de vida BPM: Intalio BPMS, Bonita y Jboss JBPM.

Intalio Business Process Management System (Intalio BPMS). Es una Suite BPM open source creada por la Compañía Intalio, la notación de modelado es BPMN, es dirigida al usuario sin conocimientos de programación, no requiere codificación y cuenta con actividades de monitoreo de procesos. La edición para la comunidad (open source), cuenta con los siguientes modulos *[28, sitio Web]*.

Diseñador BPMN (BPMN Designer). Es el Modelador Gráfico de Procesos cuya función es apoyar el modelado de los procesos BPM usando la notación BPMN.

BPEL Servidor (BPEL Server). Es un motor de procesos que implementa el estándar BPEL basado en la arquitectura J2EE.

Servicio de tareas (Taskservice). Son dos interfaces de usuario, una expone el servicio ofrecido por la tarea como por ejemplo traducción, y otra es para consultar las tareas que se tienen que realizar.

Bonita. Es una Suite BPM open source creada por BonitaSoft, la notación de modelado es BPMN, es dirigida al usuario sin conocimientos de programación, no requiere codificación. Cuenta con los siguientes módulos *[29, sitio Web]*.

Bonita Motor de ejecución (Bonita ExecutionEngine). Se encarga de la conexión de los procesos que existen en el sistema así como la ejecución de los procesos.

Bonita Estudio (Bonita studio). Es el modelador gráfico cuya función es apoyar el modelado de los procesos BPM usando la notación BPMN.

Bonita constructor de formas (Bonita form builder). Es la aplicación encargada de mostrar los formularios a los usuarios de la aplicación. Recordar que muchos de los pasos que se producen en un proceso BPM requieren de la entrada de datos por parte del Participante del Proceso.

Bonita experiencia de usuario (Bonita user experience). Es la herramienta que despliega la interfaz de usuario y el monitoreo del proceso.

Jboss JBPM. Es una Suite BPM open source creada por Jboss, que utiliza su propia notación JPDL y se diferencia de otras Suites BPM porque no se enfoca

solamente al usuario sin experiencia en programación, también al desarrollador de software, el detalle de la arquitectura de Suite Jboss JBPM se presenta en el Apéndice A *[30, sitio Web]*.

En la Tabla 5 se comparan las Suites Intalio BPMS, Bonita y Jboss JBPM, de acuerdo a las características de las Suites BPM vistas en la sección 2.4.2. La marca "√" indica que la Suite BPM tiene la característica presentada en la columna. Cabe señalar que de forma nativa, JBPM no soporta la recolección de métricas, sin embargo, se le pueden agregar componentes para recolectarlas.

En la Tabla 6 se comparan las Suites Intalio BPMS, Bonita y Jboss JBPM, de acuerdo a: el tipo de usuario que utilizará la suite, el tipo de notación que utiliza el modelo gráfico (ver sección 2.3.1), la intervención del usuario en el código de la suite para modelar el proceso y el lenguaje al que se mapea el modelo gráfico. Cabe señalar que un *Analista de Proceso* se encarga del diseño, modelado y optimización del proceso mientras que un *Analista Técnico* es un desarrollador de software.

Tabla 5: Comparación de Intalio BPMS, Bonita y Jboss JBPM.

CONCEPTO	Intalio BPMS	Jboss JBPM	BONITA
Modelador Gráfico de Procesos.	√	√	√
Herramienta que despliega una interfaz de usuario.	√	√	√
Motor de Ejecución de Procesos	√	√	√
Herramienta que recolecte y reporte las métricas del proceso.	√	Permite agregar componentes para recolectar métricas	√

Tabla 6: Cuadro comparativo de JBPM con Intalio BPMS y Bonita.

Concepto	Intalio BPMS	Jboss JBPM	Bonita
Usuario objetivo	Analista de proceso	Analista técnico	Analista de proceso
Notación de modelado	Estándar BPMN	Notación propia	Estándar BPMN
Requiere codificar	No	Sí	No
Lenguaje de definición de procesos	Estándar BPEL	JPDL	XPDL

2.5 Síntesis del capítulo

Los principales puntos de este capítulo son los siguientes.

- Un proceso es un conjunto de actividades que se ejecutan para lograr un propósito.
- Un proceso se formaliza cuando su descripción es plasmada en algún medio que permita que sea comunicado y almacenado. En la descripción del proceso se detalla lo que se hace en el proceso, quien lo hace, los materiales que se necesitan y que es lo que se produce. Esta descripción es llamada ***"Definición del Proceso"***.
- Cuando se llevan a cabo las actividades y las tareas del proceso, involucrando todos los elementos de la Definición del Proceso, se dice que se ha creado una **"*Instancia del Proceso*"**.
- Un Proceso de Negocio es un conjunto de actividades o tareas ordenadas para producir un determinado producto o servicio de utilidad para un cliente interno o externo de la organización. De acuerdo a esta definición, los Procesos de Negocio son cualquier proceso que ocurre dentro de la organización.
- Los procesos de desarrollo de software son un conjunto de actividades, métodos, prácticas y modificaciones que las personas realizan para desarrollar y mantener productos de software.
- El ciclo de mejora continua o iterativa tiene tres estados principales: 1) Medición de los atributos del proyecto actual o del producto, 2) Análisis y 3) Introducción de los cambios al proceso identificados en el análisis.

- Los Modelos de Mejora de Procesos son modelos que presentan las mejores prácticas que le permiten a una organización implementar procesos que pueden ser controlados, medidos y mejorados de acuerdo al Ciclo de Mejora de Procesos. Como por ejemplo CMMI, MoProSoft y Tritón.
- En el modelo de un proceso se representan los elementos típicos de un proceso. Un proceso es modelado a través de una representación gráfica, textual o textual y gráfica.
- El modelado de procesos con representación gráfica puede llevarse a cabo, por ejemplo, con Diagramas de Actividades de UML, con el Lenguaje de Definición de Procesos Java (JPDL por sus siglas en inglés) y con BPMN (Notación de Modelado de Procesos de Negocios), el modelado textual puede llevarse a cabo con guías (scripts), como por ejemplo, aquellas que plantea el PSP (Personal Software Process) ó TSP (Team Software Process). Existen también herramientas que apoyan el modelado con sus dos representaciones, gráfica y textual como lo es en el caso de EPF (Eclipse Process Framework).
- El enfoque BPM plantea administrar los procesos de acuerdo a una serie de etapas que son: Diseño, Modelado, Ejecución, Monitoreo y Optimización.
- Las Suites BPM que apoyan el enfoque BPM están compuestas por una serie de componentes, estos son: Modelador Gráfico de Procesos, Motor de Ejecución de Procesos, Interfaz de Usuario y Base de Datos.
- Las Suites BPM estudiadas son Intalio BPMS, Bonita y Jboss JBPM.

Capítulo

3 Propuesta: Desarrollo de la Suite BPM.

En este capítulo se hace el planteamiento del problema acotándolo en un entorno específico. Se plantea la propuesta de solución basándose en el estado del arte, que consiste en el desarrollo de una Suite BPM adaptada a las características de los procesos de los Modelos de Mejora de Procesos. El desarrollo de esta Suite BPM se presenta en este capítulo, presentando los requerimientos, el diseño de la arquitectura y la implementación.

3.1 Discusión del problema

Como ya se ha mencionado anteriormente (sección 2.1.3.1), existe una relación entre calidad del producto, los procesos y la mejora continua o iterativa de los procesos. En ese sentido es deseable para toda organización seguir un enfoque asociado a un Modelo de Mejora de Procesos. Esto es particularmente válido para pequeñas organizaciones en las cuáles se tiene un enfoque generalmente reactivo (se va de problema en problema). Una de las dificultades que enfrentan las pequeñas organizaciones es la falta de recursos, sin embargo, la adopción de Modelos de Mejora de Procesos es complicada por otras razones:

1. Porque hay que tener documentados los procesos.
2. Porque aún en épocas de estrés hay que llevar a cabo las tareas de los procesos, es decir, hay que adherirse a los procesos.
3. Porque hay que llevar a cabo el ciclo de mejora continua, y esto implica, colectar datos, analizar los datos, encontrar oportunidades de mejora, redefinir el proceso y modificar la documentación.

Frecuentemente las pequeñas organizaciones dedicadas al desarrollo de software, no trabajan con un enfoque basado en procesos (ver sección 2.1.2) por esta razón no cuentan con los beneficios de: estabilidad operacional,

integración, mayor rendimiento y reducción de costos (ver sección 2.1.2). En este caso, la problemática que se plantea es de cómo lograr que las organizaciones pequeñas con pocos recursos:

1. Definan sus procesos y los tengan documentados.
2. Se adhieran a los procesos.
3. Recolecten datos.
4. Mejoren los procesos continuamente.

De acuerdo con el capítulo 2, una posible solución a la problemática podría ser el uso de una Suite BPM existente. Sin embargo, las Suites BPM existentes tienen limitaciones porque son genéricas y no están adecuadas para apoyar los procesos de los Modelos de Mejora de Procesos. Las Suites BPM estudiadas (Intalio BPMS, Bonita y Jboss JBPM), por ejemplo, se limitan a mostrar y recibir datos muy puntuales, así, para crear los campos de entrada y salida de datos en las interfaces de usuario, sólo proveen herramientas para diseñar formularios o crear variables. Como ilustración de estos formularios tenemos a las facturas, las notas de remisión, requisiciones de compras, vales de gastos menores que se usan en procesos operativos de ventas y compras. Una organización de desarrollo de software que quiere adoptar un Modelo de Mejora de Procesos necesita, por ejemplo, soportar los procesos operativos para crear software y estos procesos involucran intercambios de artefactos entre los roles del proceso tales como el documento de visión o de requerimientos. También, estos artefactos sufren cambios a lo largo del proceso operativo, cada vez que se realiza un cambio se crea una nueva versión del artefacto. En ocasiones, estos cambios se realizan de manera concurrente, es decir, un mismo artefacto es cambiado de manera simultánea por dos roles distintos. Esto hace necesario introducir un sistema de control de versiones para administrar las distintas versiones de cada artefacto. Por lo tanto, es necesario que la Suite BPM que apoye a los Modelos de Mejora de Procesos, soporte el intercambio y control de versiones de artefactos y las Suites BPM estudiadas no brindan este soporte. Otra limitante es el número reducido de métricas que manejan, como los procesos de los Modelos de Mejora de Procesos están orientados a la mejora iterativa o continua que se basa en el análisis de métricas, es necesario

tener un esquema que permita flexibilidad en los tipos y número de métricas a recolectar.

Además, en los Modelos de Mejora de Procesos, hay un número variable de procesos que adicionalmente sufren modificaciones a lo largo del tiempo, y esto es una característica adicional que se debe poder soportar. En este sentido, este trabajo propone crear una Suite BPM de bajo costo que soporte agregar, modificar o quitar procesos de manera flexible, que también soporte el intercambio y control de versiones de artefactos y que provea mecanismos flexibles para soportar la introducción de componentes para la recolección de métricas. Esto con el fin de ayudar a que los procesos de las organizaciones pequeñas tengan el ciclo de mejora continua o iterativa, cuando estos procesos, son modificados por la adopción de algún Modelo de Mejora de Procesos para organizaciones pequeñas.

Una solución tentativa a la problemática expuesta parte de la siguiente hipótesis.

Hipótesis

Una Suite BPM puede ser adaptada para soportar el ciclo de vida de los procesos de un Modelo de Mejora de Procesos en las etapas de Modelado, Ejecución y Monitoreo.

Partiendo de esta hipótesis se desarrollo una Suite BPM, la Figura 6 muestra la metodología de desarrollo; los resultados de las actividades de esta metodología se presentan en las siguientes secciones en tres partes, estas son:

- Requerimientos de la Suite BPM para procesos de desarrollo de software.
- Diseño y documentación de la Arquitectura de la Suite BPM.
- Implementación de la Suite BPM.

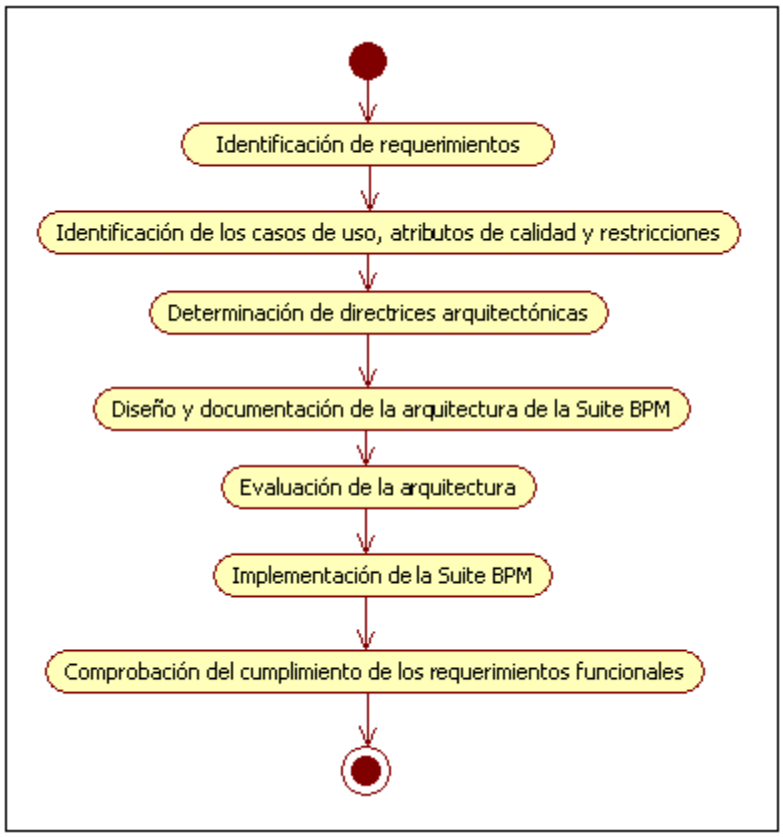


Figura 6: Metodología de desarrollo de la Suite BPM.

El resultado de la actividad *comprobación del cumplimiento de los requerimientos funcionales* que se muestra en la Figura 6, se presenta más adelante en la sección 5.2.

3.2 Requerimientos de la Suite BPM para procesos de desarrollo de software

En esta sección se presentan los requerimientos de la Suite BPM que se construyó adaptándose a las características de los procesos de los Modelos de Mejora de Procesos. Esta adaptación se realizó debido a las limitaciones de la Suites BPM expuestas en la sección anterior.

Los requerimientos de la Suite BPM y su arquitectura se presentan en las siguientes secciones.

3.2.1 Requerimientos Funcionales y No Funcionales de la Suite BPM

Los requerimientos de la Suite BPM se dividieron en Requerimientos Funcionales y Requerimientos No Funcionales. Los Requerimientos Funcionales se refieren a las declaraciones de los servicios que debe proporcionar un sistema (en este caso la Suite BPM), de la manera en que el sistema debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones específicas *[14, pág. 109]*. Los Requerimientos No Funcionales, como su nombre sugiere, son aquellos requerimientos que no se refieren a determinadas funciones que debe proporcionar el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento *[14, pág. 111]*. En la siguiente sección se presentan los Requerimientos Funcionales.

3.2.1.1 Requerimientos Funcionales

Con la finalidad de crear una Suite BPM genérica, se identificaron los Requerimientos Funcionales para la suite a partir de los aspectos en común de los procesos de los Modelos de Mejora de Procesos vistos en la sección 2.2.6. A partir de estos Requerimientos Funcionales se establecieron los Casos de Uso que se presentan en la Tabla 7. También en esta tabla, se muestra la prioridad de los Casos de Uso donde E, A y B significan: esencial, alta y baja, respectivamente. Esta prioridad define la importancia y con ello el orden del desarrollo de los Casos de Uso. El detalle de los Casos de Uso se encuentra en el Apéndice B.

Tabla 7: Elementos en común de los procesos y Casos de Uso

Elementos en común de los procesos de los modelos CMMI, MoProSoft y Tritón.	Caso de uso	Descripción	Identificador	
			Prioridad	Número
Cada proceso es instanciado	1. Instanciar definición del proceso	Este caso de uso permite soportar crear nuevas instancias de una Definición del Proceso.	E	1
Cada tarea tiene un rol	2. Ingresar al sistema por rol	Este caso de uso permite soportar el ingreso restringido al sistema.	A	2
Cada tarea tiene un momento de inicio	3. Iniciar tarea	Este caso de uso permite que se registre en el sistema la fecha y hora de inicio de una tarea.	A	3
Cada tarea tiene un momento de terminación	4. Terminar tarea	Este caso de uso permite que se registre en el sistema la fecha y hora de terminación de una tarea.	A	4
Cada Instancia del Proceso tiene una o más entradas (artefactos) que son utilizadas por una o más tareas	5. Bajar artefactos	Este caso de uso permite soportar la descarga de archivos (artefactos) del sistema.	A	5
Cada Instancia del Proceso tiene una o más salidas (artefactos) producidas por una o más tareas de la misma Instancia del Proceso	6. Subir artefactos	Este caso de uso permite cargar archivos (artefactos) al sistema.	A	6
A cada Instancia del Proceso se le da seguimiento	7. Monitorear el avance de la instancia del proceso.	Este caso de uso permite desplegar un listado de las tareas que han iniciado.	B	7
Cada proceso es continuamente mejorado	8. Actualizar e instalar nueva versión de la	Este caso de uso permite soportar agregar, quitar o cambiar el modelo gráfico de	E	8

	definición del proceso	la Definición del Proceso en el sistema, y reiniciar el sistema para que reconozca la nueva versión de la Definición del Proceso.		

En los Casos de Uso del número 2 al número 6 presentados en la Tabla 7, interviene el *Participante del Proceso*. Como ya se mencionó en la sección 2.4.3, el *Participante del Proceso* es un actor genérico para representar los distintos roles que participan en el proceso. El *Administrador del Proceso* interviene en los casos del 1 al 8, como se muestra en la Figura 7.

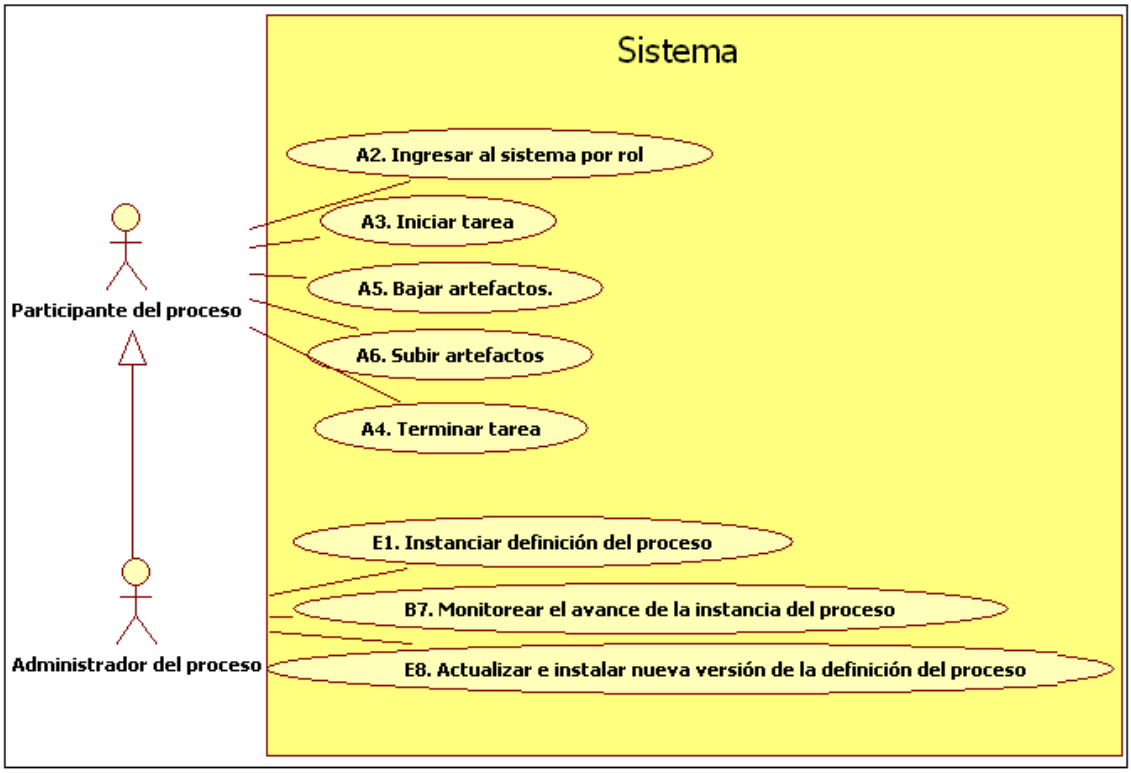


Figura 7: Casos de Uso para el diseño de la Suite BPM

3.2.1.2 Requerimientos No Funcionales

Los Requerimientos No Funcionales incluyen las restricciones y los Atributos de Calidad del sistema, estos atributos que se encuentran en categorías como por ejemplo: Rendimiento, Modificabilidad, Usabilidad y Seguridad. El diseño de la arquitectura de un sistema requiere que se tenga particular cuidado con

respecto a los Atributos de Calidad, y para documentarlos, se usa el concepto de Escenario que describe como la arquitectura debe de responder a cierto estímulo *[31, sección 4.3]*. Un escenario generalmente se compone de 6 partes: estímulo, fuente del estímulo, entorno, estado, respuesta y medida de la respuesta.

Debido a que se requiere una Suite BPM que permita el cambio constante de la definición de los procesos y que sin hacer cambios al código siga proporcionando la misma funcionalidad, uno de los Atributos de Calidad que guían la creación de la Arquitectura de la Suite BPM, pertenece a la categoría de Modificabilidad (ver Tabla 8, ID RNF-01). También se requiere que la Suite BPM permita agregar métricas para el monitoreo de los procesos, por esta razón, la arquitectura debe soportar la adición de componentes para la toma de estas métricas, por lo que se tiene otro atributo de calidad que pertenece a la categoría de Extensibilidad (ver Tabla 8, ID RNF-02). El acceso a la Suite BPM es restringido, por esta razón, se considera un atributo de calidad que pertenece a la categoría de Seguridad (ver Tabla 8, ID RNF-03).

Tabla 8: Atributos de Calidad del sistema

ID	**Escenario que documenta el Atributo de Calidad**	**Justificación de la clasificación en esta categoría**	**Categoría**
RNF-01	Un Administrador del Proceso modela e instala una nueva versión de la Definición del Proceso en el sistema. El sistema presenta pantallas que muestran la nueva versión del proceso sin necesidad de agregar código.	El sistema debe permitir cambios en las pantallas por cambios en el modelo de la Definición del Proceso sin realizar cambios en el código de estas.	**Modificabilidad**
RNF-02	Un desarrollador agrega un componente de análisis de métricas en tiempo de desarrollo. El componente es agregado exitosamente y no se modifica más de un elemento de la arquitectura del sistema.	El sistema debe permitir agregar componentes, para recolectar métricas que permitan la realización de distintos tipos de análisis, sin afectar su funcionalidad.	**Extensibilidad**
RNF-03	Un usuario no administrador	El sistema debe restringir el	**Seguridad**

	ingresa un rol y contraseña correctos en la página de Selección de Rol. El sistema le da acceso a la página de instancias de procesos y únicamente le permite ver las tareas que le corresponden a su rol.	acceso a sus servicios.	

Las restricciones en el diseño de la Suite BPM son las presentadas en la Tabla 9.

Tabla 9: Restricciones del sistema

ID	Descripción de la restricción.
RES-01	El sistema debe estar construido usando componentes open source (para soportar bajo costo)
RES-02	El sistema debe permitir a varios usuarios el acceso simultáneo vía Web (para soportar trabajo en equipo).
RES-03	El sistema debe estar integrado a un sistema de control de versiones (para soportar la evolución de artefactos).
RES-04	El sistema se debe ejecutar en ambientes Linux y Windows.

En la siguiente sección se muestra la arquitectura que se diseñó para soportar las Directrices Arquitectónicas, que serán descritas más adelante.

3.3 Diseño y documentación de la Arquitectura de la Suite BPM

La Arquitectura de Software de un programa o sistema informático es la estructura o estructuras del sistema, que incluyen los elementos de software, las propiedades externamente visibles de estos elementos, y las relaciones entre ellos *[31, pág. 3]*. Las estructuras que conforman la arquitectura se crean a partir de decisiones arquitecturales guiadas por las Directrices Arquitectónicas.

3.3.1 Directrices Arquitectónicas

Al momento de diseñar la arquitectura, se parte de un sub-conjunto de requerimientos llamados Directrices Arquitectónicas. Estas directrices incluyen ciertos Casos de Uso Primarios (los de mayor importancia para el negocio y/o alta complejidad), los Atributos de Calidad y las restricciones. En la Tabla 10 se muestran los Casos de Uso Primarios considerados Directrices Arquitectónicas, con sus respectivos escenarios y si se trata de un flujo principal o alternativo. En la Tabla 11, se muestran todas las Directrices Arquitectónicas utilizadas en el desarrollo de la Suite BPM.

Tabla 10: Casos de Uso Primarios como Directrices Arquitectónicas

ID Caso de uso	Caso de Uso Primario	Escenario	Tipo de flujo
E1	Instanciar Definición del proceso	Un Administrador del Proceso crea una Instancia del Proceso y le da nombre en la pantalla de procesos del sistema. El sistema (Motor JBPM) lee la Definición del Proceso en la base de datos y crea una instancia, lo nombra y le asigna un identificador único que muestra en pantalla de procesos.	Principal
A2	Ingresar al sistema por rol	Un Participante del Proceso ingresa su contraseña y rol en la pantalla de rol del sistema. El sistema verifica los datos y le da acceso a la pantalla de procesos.	Principal
A3	Iniciar tarea	Un Participante del Proceso inicia una tarea en la pantalla de tareas del sistema. El sistema registra la hora de inicio y le da acceso a la pantalla de artefactos donde despliega la lista de artefactos que corresponden a la tarea.	Principal
A4	Terminar tarea	Un Participante del Proceso termina una tarea en el sistema. El sistema registra la hora de terminación y la muestra en la pantalla de las tareas.	Principal
B7	Monitorear el avance de la instancia del	Un Administrador del Proceso solicita ver el avance de procesos en la Pantalla de procesos del sistema. El sistema muestra en la pantalla	Principal

	proceso	de monitoreo la lista de tareas iniciadas y terminadas, con sus respectivos registros de tiempos.	
E8	Actualizar e instalar nueva versión de la definición del proceso	Un Administrador del Proceso modifica un modelo de la definición del proceso en el sistema. El sistema crea nueva versión y la almacena en la Base de datos.	Principal

Tabla 11: Directrices Arquitectónicas

Casos de Uso Primarios	Atributos de Calidad	Restricciones
- E1 - A2 - A3 - A4 - B7 - E8	- RNF-01 - RNF-02 - RNF-03	- RES-01 - RES-02 - RES-03 - RES-04

Las Directrices Arquitectónicas de la Tabla 11 guiaron las decisiones que se tomaron para diseñar la Arquitectura de la Suite BPM. Estas decisiones de diseño se presentan en la siguiente sección.

3.3.2 Diseño de la Suite BPM

Las decisiones de diseño consisten principalmente en elegir los Patrones Arquitecturales que permiten satisfacer a las Directrices Arquitectónicas *[32, pág. 7]*. En las siguientes secciones se describen los Patrones Arquitecturales elegidos para la Suite BPM.

3.3.2.1 Patrón de Capas

Una Arquitectura de Capas define una partición lógica de la funcionalidad del software de acuerdo a una característica del sistema, de tal forma que un grupo

de funcionalidades este claramente encapsulada y pueda ser desarrollada de manera independiente *[33, libro electrónico no paginado]*.

El diseño que se propuso para la Suite BPM consta de las siguientes capas (ver Figura 8).

Capa de Presentación. *Es la capa que permite soportar la interacción con el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio* *[34, pág. 235]*.

Capa de Aplicación o Negocio. *Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados. También se comunica con la capa de acceso a la base de datos, para solicitar que almacene o recupere datos de la base de datos* *[34, pág. 235]*.

Capa de Dominio. *Es la capa que contiene el Modelo de Dominio del problema. Este modelo, es un modelo de clases, consiste en los objetos del dominio del problema, es decir, objetos que tienen correspondencia directa en el área de la aplicación* *[34, pág. 235]*.

Capa de Acceso a la Base de Datos. *Es la capa responsable de tener los métodos que la aplicación invoca y que contienen el formato para el acceso a las tablas de la base de datos. Así, esta capa desacopla la aplicación de los detalles de implementación de la base de datos y hace una traducción del modelo orientado a objetos al modelo relacional* *[33, libro electrónico no paginado]*.

Capa de Persistencia. *Se encarga de realizar la persistencia de los objetos del modelo de dominio.*

Capa de Definición de Procesos. *Es donde residen los componentes para crear y enviar a la capa de persistencia las definiciones de los procesos.*

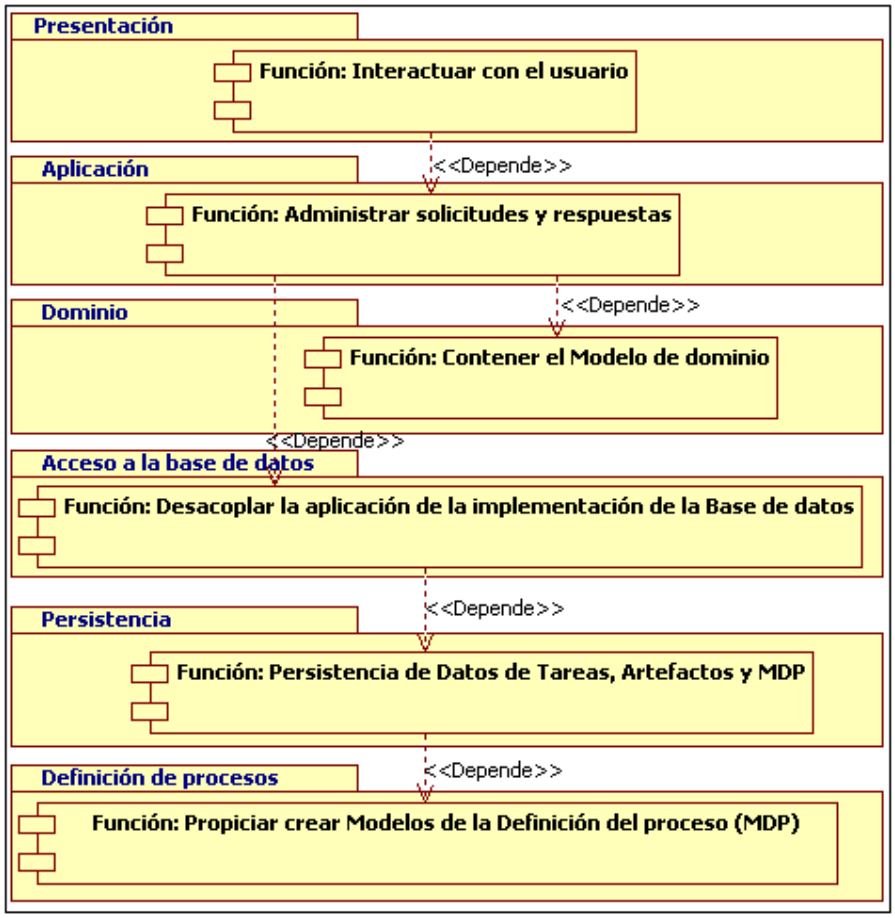


Figura 8: Patrón de Capas en el diseño de la Suite BPM

3.3.2.2 Patrón Modelo Vista Controlador

Un Patrón Modelo-Vista-Controlador (MVC) es una configuración basada en la división de responsabilidades, de una aplicación que tiende a cambiar a distintas velocidades, con el propósito de soportar su desarrollo de manera independiente *[33, libro electrónico no paginado]*.

En este patrón, el controlador implementa la lógica para procesar la entrada del usuario. El modelo contiene los datos de la aplicación y la vista presenta los datos almacenados en el modelo *[35, pág. 910]*.

El Diagrama Secuencial de la Figura 9 muestra la interacción resumida entre los componentes Modelo, Vista y Controlador.

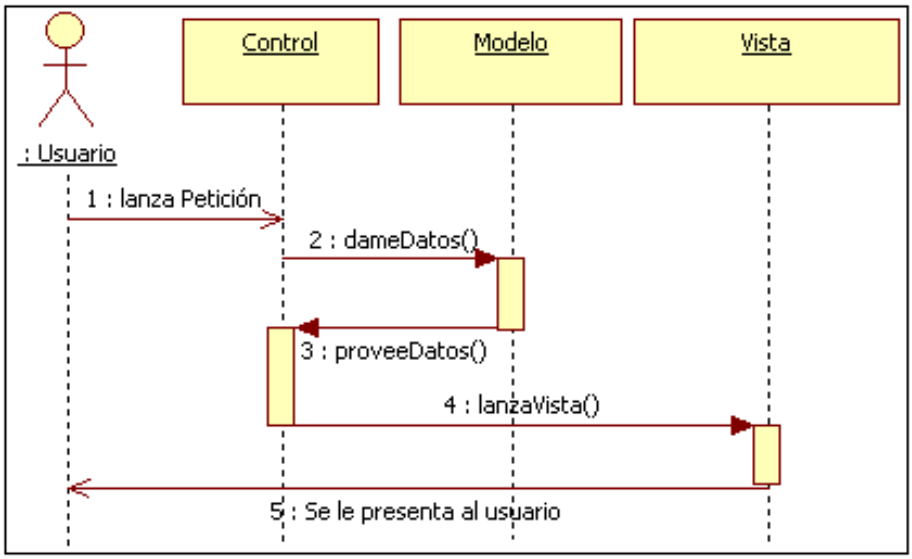


Figura 9: Diagrama Secuencial de la interacción resumida de los componentes del Patrón MVC

En la Figura 10 se muestran las capas que corresponden a cada componente del Patrón MVC para la Suite BPM. Las líneas punteadas representan "dependencia", así, la *Capa de Presentación* depende de la información que le provea la *Capa de Aplicación*. La *Capa de Aplicación* depende de los datos de que son recuperados por la *Capa de Acceso a la Base de Datos* que depende de la *Capa de Persistencia*. Los objetos de dominio son creados para almacenar temporalmente los datos que son traídos de la *Capa de Persistencia* y que la *Capa de Aplicación* requiere. La *Capa de Persistencia* almacena las definiciones de procesos que son creadas en la *Capa de Definición de Procesos.*

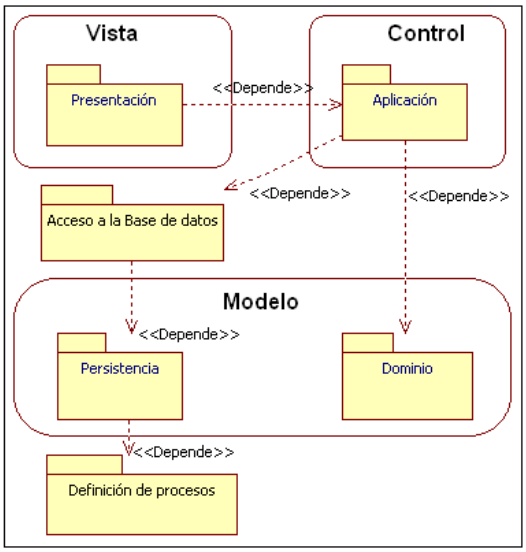


Figura 10: El Patrón MVC y las capas del diseño de la Suite BPM

3.3.2.3 Patrón Observador implementado para la Suite BPM

El Patrón Observador define un mecanismo de propagación en el que un proveedor conocido como “sujeto”, siempre que su estado cambia, notifica a los consumidores registrados conocidos como observadores. Los observadores al ser notificados realizan cualquier acción que consideren necesaria. Los observadores pueden ser registrados o eliminados de manera dinámica *[33, libro electrónico no paginado]*. En el contexto de la Suite BPM este patrón es útil porque cuando se está ejecutando la Instancia del Proceso se generan eventos que son utilizados para análisis específicos. El DespachadorDeEventos (sujeto) conoce el evento, como por ejemplo, la hora y fecha en que inicia una tarea, y lo comunica al ObservadorRegistroTiempo (observador).

3.3.2.4 Patron DAO (Data Access Object)

El Patrón DAO consiste en introducir una *Capa de Acceso a la Base de Datos* entre la aplicación y la base de datos relacional. La aplicación puede almacenar y recuperar datos persistentes invocando al método respectivo en la *Capa de Acceso a la Base de Datos* *[33, libro electrónico no paginado]* (ver Figura 8).

Usualmente el Patrón DAO se complementa con el Patrón Fábrica, (FabricaDAO en la Suite BPM construida) que le proporciona a la aplicación un objeto de acuerdo a la base de datos implementada.

Las decisiones de diseño se presentan de forma resumida en la Tabla 12.

Tabla 12: Decisiones de diseño

Decisión de diseño	Directriz arquitectónica que satisface	Justificación
Uso del Patrón de Capas	Todas las Directrices Arquitectónicas.	Las responsabilidades del sistema deben de identificarse por separado para permitir el mantenimiento independiente del sistema
Uso del Patrón Modelo	RNF-02	El sistema debe permitir el acceso vía Web y se

Vista Controlador	RES-02 E1, A2, A3, A4, B7, E8.	requiere la separación lógica entre la interfaz de usuario, la lógica de control y los datos. La separación se hace con el fin de estructurar el sistema a un alto nivel y favorecer la modificabilidad.
Uso del Patrón Observador	RNF-02 B7	Se requiere que un componente recolector de métricas, sea agregado y no se modifique más de un componente de la arquitectura del sistema. En este caso, el componente es un observador de las instancias del proceso.
Uso del Patrón de DAO	RES-03 B7	La implementación de la base de datos debe ser cambiada sin afectar el sistema.

En la siguiente sección se describen las tecnologías utilizadas para implementar el diseño de la Suite BPM.

3.3.3 Descripción de tecnologías utilizadas.

Se presentan a continuación las capas del diseño y las principales tecnologías elegidas para implementarlas (ver Figura 11), las tecnologías se explican brevemente más adelante.

- **Las capas de Presentación y Aplicación.** Se implementaron con Spring MVC.
- **Las capas de Persistencia y Dominio.** Forman parte del Patrón Modelo Vista Controlador y son requeridas en la implementación de Spring MVC.
- **La Capa de Acceso a la Base de Datos.** Se implementó con Hibernate.
- **La Capa de Definición de Procesos.** Se implementó para usar la notación del Lenguaje de Definición de Procesos Java (JPDL, por sus siglas en inglés).

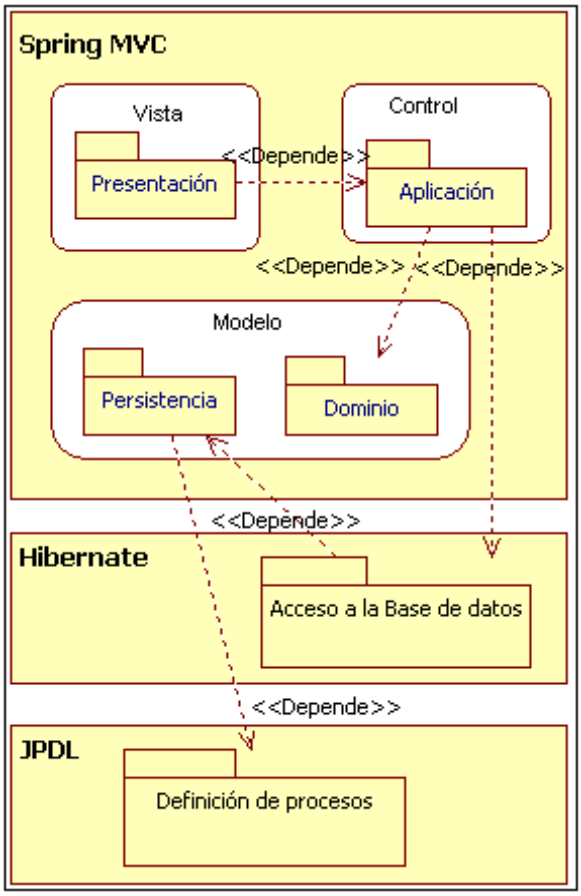


Figura 11: Capas de diseño y tecnologías utilizadas

Otras tecnologías utilizadas son:

- **Spring Framework.** Se utilizó para implementar la conexión de los componentes del sistema.
- **ANT.** Se utilizó para realizar la construcción.

En las siguientes secciones se describen las tecnologías utilizadas.

3.3.3.1 Spring Framework

Spring Framework *[42, sitio Web]* se basa en el patrón de inversión de control e inyección de dependencias *[39, pág. 49]*. Esto se refiere a que los objetos toman la iniciativa de conectarse, las conexiones son externas, descritas en un archivo independiente. En este archivo XML se define el contexto de la aplicación,

basado en JavaBeans. Spring Framework lee ese archivo y conecta los componentes, por eso se llama inversión de control porque en vez de que la aplicación tome la iniciativa de crear el objeto, el Framework toma el control y lo que inyecta a la aplicación son las dependencias, pero además de eso, provee módulos para soportar Atributos de Calidad, utilizando un enfoque de programación orientada a aspectos.

La programación orientada a aspectos *[41]* mueve a un módulo separado el código que no se relaciona directamente con la solución básica del problema y que muchas veces se encuentra repetido en módulos que tienen funciones principales diferentes. De esta forma, el código de la aplicación ya no contiene pedazos de funciones transversales dispersas en otros módulos *[40, pág. 34]*. Algunos ejemplos de aspectos son: los patrones de acceso a memoria, la sincronización de procesos concurrentes, el manejo de errores. En Spring Framework, se interpone un proxy (mediador) entre la clase y el aspecto. De esta forma podemos acoplar o desacoplar aspectos comunes sin tocar el código.

Spring Framework está formado por frameworks más pequeños, entre ellos Spring MVC Framework, que es ocupado en el contexto de este trabajo.

3.3.3.2 Spring MVC Framework

Spring MVC Framework *[38, sitio Web]* está estructurado siguiendo el patrón MVC: tiene controladores configurables, resolución de vistas y actualización de archivos. Esta diseñado en torno de un servlet (pequeña aplicación Java que se ejecuta en un Servidor Web) central llamado DispatcherServlet.

Al recibir una petición (request) el DispatcherServlet mapea dicho request a una clase que implementa la interfaz Controller. En este caso se utilizan los componentes mappings.xml, workflowControllers.xml y el controlador correspondiente, que se explican más adelante en la sección 3.3.5.2.1. El controlador puede ser una implementación de la interface “Controller”, que sólo

tiene un método el “ModelAndViewhandleRequest(request, response)”, pero es recomendable utilizar la jerarquía propuesta por Spring MVC Framework, por ejemplo, se puede usar SimpleFormController si se utiliza un formulario *[38, sitio Web]*. Los controladores se encargan de crear el modelo y agregar el jsp que es la vista que se presenta.

3.3.3.3 Hibernate

Al tener un modelo orientado a objetos y un modelo relacional, el problema que surge es persistir las entidades del modelo de dominio en una base de datos relacional. Al ser dos modelos diferentes, hay que hacer lo que se llama un mapeo de un modelo orientado a objetos a un modelo relacional. Existen frameworks encargados de hacer el mapeo del modelo orientado a objetos al modelo relacional y Hibernate *[43, sitio Web]* es uno de esos frameworks, donde básicamente lo que el desarrollador hace es describir de manera declarativa, en un archivo xml, la correspondencia entre los objetos o las clases de dominio y las tablas, y Hibernate se encarga de administrar toda la persistencia de los datos. También se integra con Spring Framework.

3.3.3.4 ANT

ANT *[44, sitio Web]* es una herramienta para automatizar el proceso de construcción, básicamente es una herramienta parecida al make del lenguaje C en donde se describe, en un archivo xml, el equivalente del make file en lo que se refiere a las reglas de construcción. Cada archivo de construcción contiene un proyecto y por lo menos un objetivo (target) que se lleva a cabo por default. Los objetivos pueden depender unos de otros y ANT asegura que se respete la secuencia.

El archivo de construcción comúnmente es llamado build.xml. En este trabajo este archivo describe, de acuerdo a las dependencias de sus objetivos, la secuencia de construcción siguiente:

1. Se asignan valor a las propiedades, por ejemplo, rutas a los archivos fuente.
2. Se crea el directorio donde se almacenan los archivos java compilados.
3. Se copian, al directorio de archivos java compilados, recursos para la ejecución.
4. Se compilan los archivos java.
5. Se ejecutan los archivos java compilados.

3.3.3.5 JPDL.

Para la Arquitectura de la Suite BPM se tomó el componente DiseñadorDeProcesos (Designer) de la Suite Jboss JBPM (ver Apéndice A). Este componente incluye a JPDL.

JPDL es un lenguaje de ejecución de procesos (ver sección 2.3.1), con capacidad de modelado de procesos que se integra completamente con la tecnología Java *[30, sitio Web]*. En esta definición se habla de capacidad de modelado porque JPDL provee su propia *notación* para el modelo gráfico (la definición de notación fue vista en la sección 2.3.1). Esta notación es mapeada a XML. Varios de los elementos de JPDL son mapeados directamente de su notación gráfica a XML. La notación de JPDL está compuesta por nodos de distintos tipos, con distintos usos. En general, a pesar de que un nodo se visualiza como una caja de actividad de un Diagrama de Actividades de UML, un nodo se enfoca en la idea de representar la *espera* de que ocurra una acción *[25, pág. 74]*, por ejemplo: “Acción: tarea terminada”, y así moverse al siguiente nodo. A diferencia de una actividad que indica *hacer* una acción, por lo tanto, un nodo en JPDL representa un estado. De esta manera, con los elementos de JPDL se crean “Diagramas de Estados”.

Los elementos de JPDL son los siguientes *[25, págs. 74-77]*, algunos de estos elementos se muestran en la Figura 12.

a) Nodo-Tarea (Task-Node). *Se utiliza para agrupar una o más tareas, este nodo tiene una representación gráfica, es decir, se mapea directamente del gráfico al documento xml. Es utilizado para representar el estado de espera en el que se encuentra el proceso mientras se concluyen las tareas que agrupa este nodo.*

b) Tarea (Task). *Una tarea es una unidad de trabajo. No tiene representación gráfica, solo en el documento xml, donde se encuentra formando parte de un Nodo-Tarea. Cada tarea se añade, durante la ejecución del proceso, a la lista de tareas del rol al cual se le ha asignado esa tarea.*

c) Estado (State). *Un nodo de estado se utiliza para modelar el comportamiento de esperar a que un sistema externo proporcione una respuesta. Si tiene representación* gráfica, es decir, se mapea directamente del gráfico al documento xml.

d) Forks y joins. *Estos elementos permiten modelar vías de ejecución concurrente. Una vez completadas las partes paralelas se unirán en un nodo join del cual seguirá la ejecución su camino. Si tiene representación gráfica.*

f) Decision. *Se usa un nodo de decision cuando la Instancia del Proceso decide una ruta a seguir en función de los datos con los que dispone al momento el nodo. Si tiene representación.*

g) Nodo. *Un nodo de tipo "nodo" es un tipo especial. Permite al desarrollador definir un nodo personalizado.*

h) Transiciones (Transitions). *Especifican la ruta de transición entre los nodos. Es necesario que las transiciones tengan un nombre para que se puedan diferenciar y sea posible elegir entre ellas durante la ejecución de la instancia de un proceso. Si tiene representación gráfica.*

i) Acciones (Actions). *Permiten a un desarrollador añadir código Java a la definición de un proceso en el ámbito de nodo para modelar el comportamiento del proceso, como podrían ser acontecimientos o acciones terminadas que provocan la entrada o salida de un nodo. Dado que son código, no tienen representación gráfica.*

j) Canales (Swimlanes). *Los canales representan los roles en el proceso, y se utilizan para asignar tareas a personas concretas. Esto puede ser un actor o un grupo de grupo de actores. No tienen representación gráfica.*

k) Asignación (Assignment). *Las asignaciones representan un rol específico para una tarea determinada.*

l) Variables de proceso. *Son variables que son declaradas en la definición de una tarea para almacenar datos durante la ejecución de la Instancia del Proceso,*

el ámbito de estas variables puede ser local, afectar sólo al siguiente nodo, o global afectar a todos los nodos subsecuentes. No tienen representación gráfica.

m) Estado del Proceso (Processstate). *Este tipo de nodo se utiliza cuando es necesario agregar al gráfico subprocesos dentro de un proceso de más alto nivel. Así, un nodo Estado del Proceso hace referencia a un subproceso descrito por separado, permitiendo el manejo de procesos anidados. Si tiene representación gráfica.*

n) Super Estado (Superstate). *Super estados son una forma de modelar un grupo de nodos dentro del mismo modelo (a diferencia del processstate en donde el grupo de nodos se encuentra descrito externamente). Si tiene representación gráfica.*

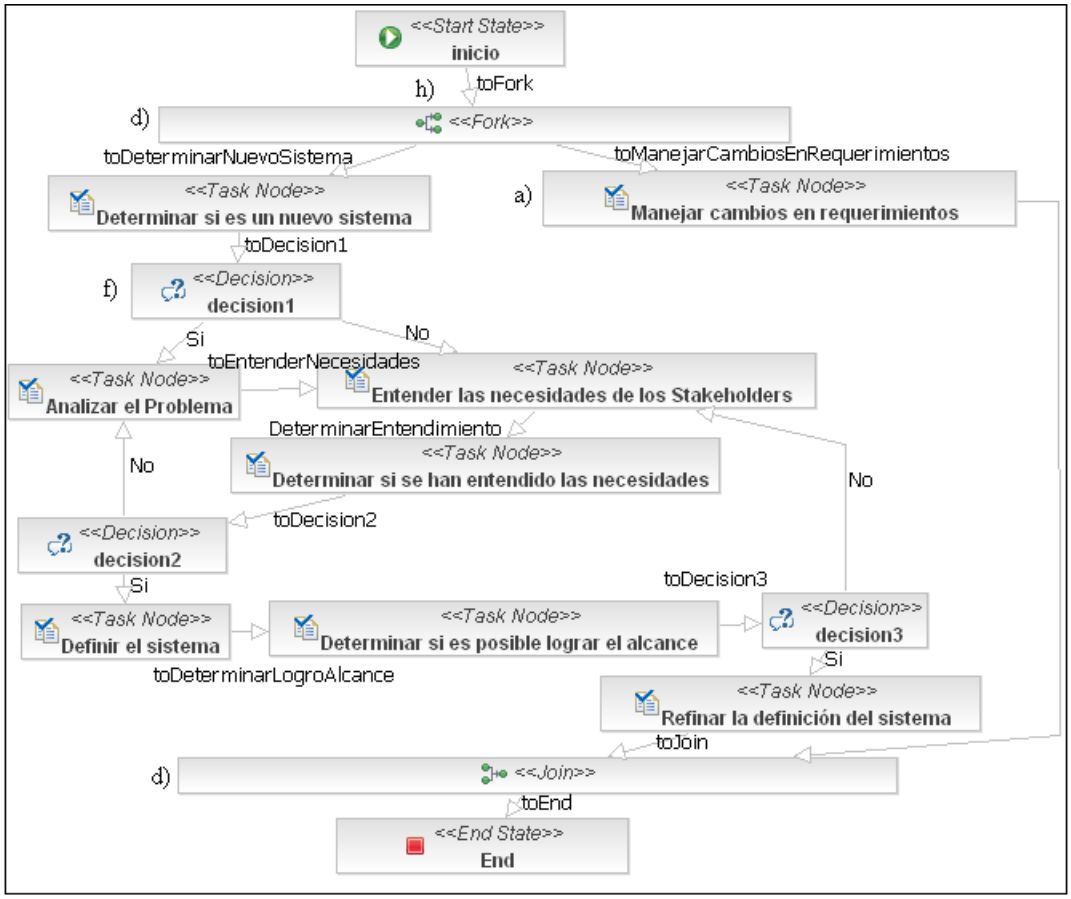


Figura 12: Proceso de levantamiento de requerimientos con notación JPDL

Las principales limitaciones observadas en JPDL son:

- No todos sus elementos tienen mapeo directo de la representación gráfica, como ocurre con las tareas y los roles.
- No existe una manera de representar artefactos.

La Tabla 13 muestra la justificación del uso de: Spring MVC, Hibernate, JPDL, frente a otras opciones.

Tabla 13: Argumentos de selección de tecnologías utilizadas

Tecnología utilizada	Argumentos de la selección.
Spring MVC	Porque provee mecanismos, para interceptar y controlar que permiten factorizar el comportamiento común en el manejo de múltiples peticiones.
Hibernate	Porque la clase Hibernate puede utilizarse en cualquier contexto de ejecución, es decir, no necesita de un contenedor para ser utilizada.
JPDL	Porque desde el inicio se planteo agregar un componente gratuito que utiliza este lenguaje de definición de procesos.

3.3.4 Vistas de la arquitectura.

Un sistema tiene muchos aspectos diferentes: funcionales (estructura estática y las interacciones dinámicas), no funcionales (requisitos de tiempo, modificabilidad, extensibilidad, etc.), y los aspectos de la organización de los desarrolladores (organización del trabajo, la asignación de los módulos de código, etc.). Una descripción del sistema requiere una serie de vistas, donde cada vista representa una estructura que es una proyección de todo el sistema que muestra un aspecto particular. Puesto que la arquitectura del sistema está compuesta por varias estructuras, la arquitectura completa del sistema se representa a través de varias vistas *[36, pág. 21]*. En las siguientes secciones se presentan las vistas de la Arquitectura de la Suite BPM.

3.3.4.1 Vista Lógica.

En una Vista Lógica se presenta la forma en que se proporciona la funcionalidad del sistema. En esta vista se describe la estructura estática (clases y relaciones), se definen propiedades como la persistencia y la

concurrencia, así como también las interfaces y la estructura interna de las clases *[36, pág. 23]*.

En la Figura 13 se muestra la Vista Lógica de la Suite BPM mostrando su estructura estática y la persistencia. También se observa el uso del Patrón de Capas y el Patrón MVC, presentando con las palabras “Modelo”, “Vista” y “Control”, en los componentes: Controlador, Proceso, Tarea, Actor, BaseDeDatosJBPM, Subversión y BaseDeDatos. El Patrón Observador está implementado con los componentes DespachadorDeEventos y ObservadorRegistroTiempo. El Patrón DAO esta implementado con los componentes que forman parte de la Capa de Acceso a la Base de Datos.

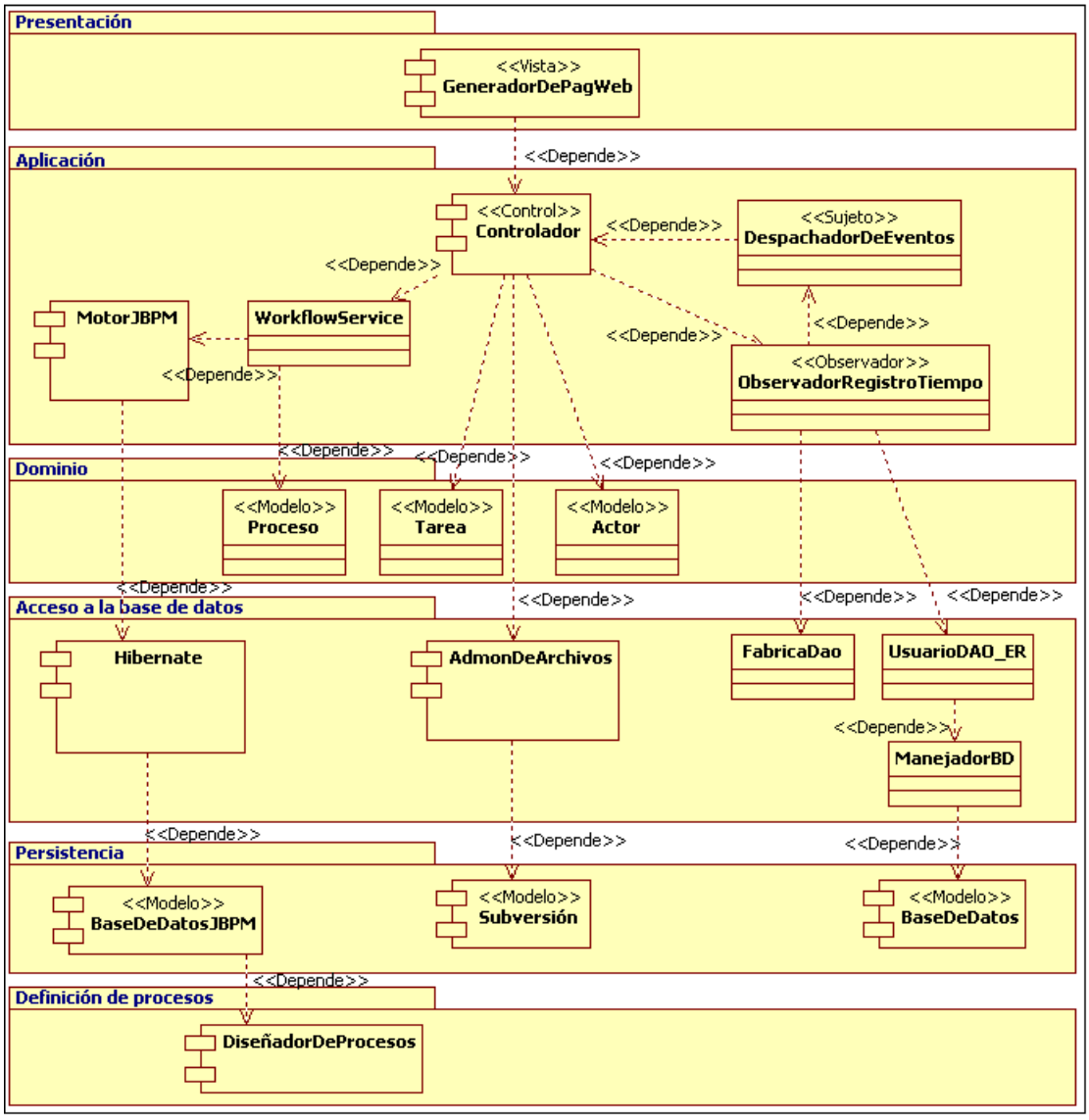


Figura 13: Vista Lógica de la Arquitectura de la Suite BPM construida

También con referencia a la Figura 13, en la Vista Lógica se observa en la Capa de Persistencia, dos bases de datos. En la *BaseDeDatosJBPM* se almacenan los cambios sobre la Instancia del Proceso y se tiene acceso con Hibernate; en la *BaseDeDatos* se almacena la información de las métricas que serán recolectadas y se usa JDBC. Esta *BaseDeDatos* está por separado porque forma parte del componente "Colector y Presentador de Métricas" que sirve para monitorear la Instancia del Proceso (ver Figura 4 de la sección 2.4.2), este componente se hizo al final con escaso recurso de tiempo y se requería una base de datos. Al no contar con la documentación de la *BaseDeDatosJBPM,* no se invirtió tiempo en investigar su implementación para utilizarla, y se integró una base de datos, con la cual ya se contaba.

En la Vista Lógica, también se muestra que la Arquitectura de la Suite BPM tiene componentes de la Suite Jboss JBPM. De acuerdo a las características de esta suite presentadas en la Tabla 6 de la sección 2.4.4, las razones por las que la fue elegida para tomar de ella algunos componentes son las siguientes: es open source, aspecto importante porque este trabajo está orientado a las organizaciones de limitados recursos económicos, y es orientada al Analista Técnico, es decir, permite al programador codificar aspectos del Proceso de Negocio. El detalle de la Arquitectura de Suite Jboss JBPM se presenta en el Apéndice A.

Los componentes tomados de la Suite Jboss JBPM se presentan en la Tabla 14.

Tabla 14: Componentes seleccionados de la Suite Jboss JBPM

Componente	**Descripción**
DiseñadorDeProcesos (Designer)	Es un Modelador Gráfico de Procesos. En el caso de la Suite Jboss JBPM es un plugin del Entorno de Desarrollo Integrado Eclipse. El diseñador de procesos realiza una traducción hacía JPDL de los elementos que tienen representación gráfica (ver sección 3.3.3.5).
BaseDeDatosJBPM	La base de datos almacena tanto definiciones de procesos (puede almacenar múltiples versiones de una misma definición), como

	datos relacionados con instancias de estas definiciones.
MotorJBPM	Es un Motor de Ejecución de Procesos visto en la sección 2.4.2.

La descripción de los demás componentes que integran la Vista Lógica de la Suite BPM se presenta en la Tabla 15.

Tabla 15: Componentes desarrollados para la Suite BPM

Componente	**Descripción**
GeneradorDePagWeb	Este componente se encarga de generar de manera dinámica las páginas Web que el participante solicita. Estas páginas despliegan la información solicitada y capturan la información requerida por la Suite BPM y proporcionada por el participante.
Controlador	Este componente se encarga de responder la petición del participante.
WorkflowService	Este componente encapsula el comportamiento del proceso. Como por ejemplo: crear, modificar o terminar una instancia. Es decir, este componente maneja las funciones que están a disposición del participante. Cada función representa una o más peticiones al MotorJBPM.
AdmonDeArchivos	Este componente se encarga del almacenamiento local de los artefactos del proceso, de las descripciones de las tareas, de las contraseñas y de la verificación de estas contraseñas.
DespachadorDeEventos	Este componente se encarga de recibir los eventos que sobre la Instancia del Proceso suceden (sujeto del Patrón Observador) y notificarlos a los observadores.
ObservadorRegistroTiempo (Recolector de métricas)	Este componente se encarga de mandar a almacenar los datos de los eventos que recibe del DespachadorDeEventos. (Observador del Patrón Observador).
Hibernate	Es un framework que se encarga de hacer el mapeo de un modelo orientado a objetos (Programación en Java) a un modelo relacional (tablas de la base de datos).
FabricaDAO	Se encarga de proveer un nuevo objeto usuarioDAO.
UsuarioDAO_ER	Se encarga de proveer a la Suite BPM los métodos de *crear* y *modificar* la tarea encapsulando los detalles de acceso a la base de datos (Patrón DAO).
ManejadorBD	Conecta a la Suite BPM con la base de datos.
Proceso	Es la clase donde se instancia un objeto del dominio del problema en este caso el Proceso. Es un objeto del dominio del problema

	porque cada vez que se instancia un proceso de su definición adquiere valores específicos, como id por ejemplo, que son asignados como atributos en un objeto de la clase Proceso.
Actor	Es la clase donde se instancia un objeto del dominio del problema en este caso el Actor. Es un objeto del dominio del problema porque cada vez que ingresa un Participante del Proceso, sus datos, como rol por ejemplo, son atributos de un objeto de la clase Actor.
Tarea	Es la clase donde se instancia un objeto del dominio del problema en este caso la Tarea. Es un objeto del dominio del problema porque cada vez que se inicia una tarea, sus datos, como fecha y hora, por ejemplo, son atributos de un objeto de la clase Tarea.
Subversión	Componente implementado para ser el cliente por parte de la Suite BPM y conectarse con el proveedor del sistema de control de versiones, para el almacenamiento de los artefactos.
BaseDeDatos	Componente que representa el almacenamiento local de los artefactos del proceso y de la información de las métricas previamente establecidas que serán recolectadas.

En el Apéndice C se presenta la Vista Lógica a nivel de Modelo de Dominio. Este modelo representa el área principal del sistema, a través de conceptos y sus relaciones; los conceptos son clases y las relaciones son asociaciones *[20, pág. 26]*.

3.3.4.2 Vista de Implantación

La Vista de Implantación muestra el aspecto físico del sistema. En esta vista se presentan elementos tales como equipos y dispositivos (nodos) y las conexiones entre ellos. También se especifica la plataforma de ejecución (sistema operativo o sistema de gestión de base de datos) que se encuentra dentro de los distintos procesadores *[36, pág. 24]*.

En la Figura 14 se muestra la Vista de Implantación de la Suite BPM. Los componentes que la forman se describen en la Tabla 16.

Tabla 16: Descripción de los componentes de la Vista de Implantación

Componente	Descripción
Terminal	Las terminales con los componentes Navegadores Web 1, 2 y 3 representan a los nodos de los usuarios que de manera remota se conectan al nodo-Servidor que contiene la Suite BPM.
Suite BPM	Como su nombre lo indica representa el programa Suite BPM que corre en el nodo-Servidor. Todas las capas de la Vista Lógica se encuentran contenidas dentro de este componente.
Base de datos 1	Representa el almacenamiento físico dentro del nodo-Servidor de la Definición del Proceso y el estado de las distintas instancias que se hayan creado.
Base de datos 2	Representa el almacenamiento físico dentro del nodo-Servidor de la información referente al monitoreo del proceso, es decir, los datos de las métricas previamente establecidas que serán recolectados.
Repositorio remoto	Representa el sistema de control de versiones, en su aspecto de proveedor, que se encuentra en otro nodo. Un sistema de control de versiones tiene dos partes. La parte cuando funge como servidor, donde se encarga de almacenar la información y llevar propiamente el control de versiones y la parte como cliente que se encarga de mandar y recibir información. La parte de cliente se encuentra en la Suite BPM.

En la Figura 14 se observan dos bases de datos. La Base de Datos 1 que corresponde a la *BaseDeDatosJBPM* (ver sección 3.3.4.1), que como ya se ha mencionado, almacena los cambios sobre la Instancia del Proceso; la Base de Datos 2 corresponde a la *BaseDeDatos* (ver sección 3.3.4.1) que almacena la información de las métricas. También como ya se ha mencionado, esta *BaseDeDatos* está por separado porque forma parte del componente “Colector y Presentador de Métricas” que sirve para monitorear la Instancia del Proceso (ver Figura 4 de la sección 2.4.2), este componente requiere una base de datos. Al no contar con la documentación de la *BaseDeDatosJBPM,* no se invirtió

tiempo en investigar su implementación para utilizarla; por lo que se integró una base de datos con la cual ya se contaba.

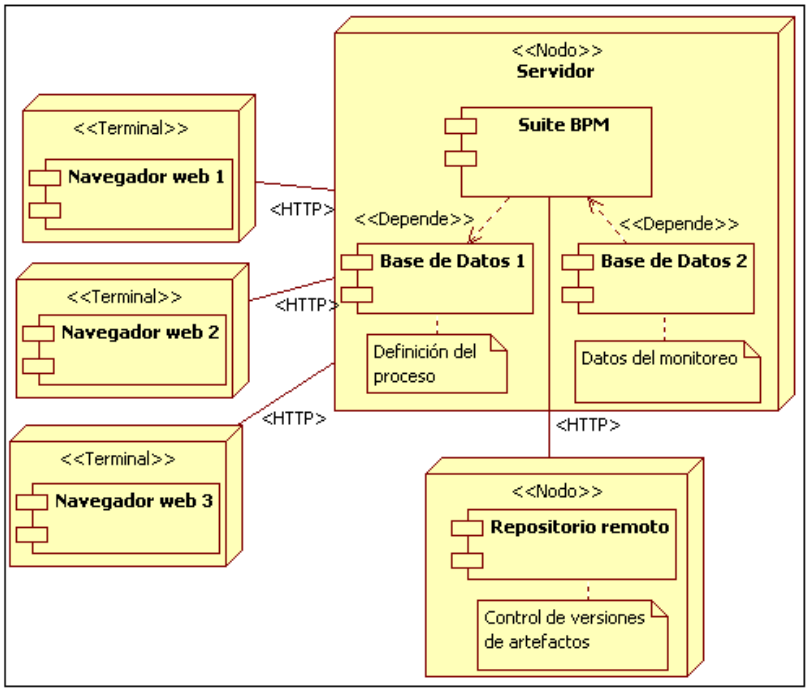


Figura 14: Vista de Implantación de la Suite BPM

3.3.4.3 Vista Dinámica.

La Vista Dinámica representa las interacciones entre los elementos clave de la estructura de la arquitectura que realizan funcionalidades relevantes *[37, pág. 144]*.

En la Figura 15 se presenta la Vista Dinámica del caso de uso de *Iniciar Tarea* utilizando un Diagrama de Secuencia, que se presenta dividido por carecer de espacio suficiente; el punto A de la Figura 15 indica la continuidad en el diagrama.

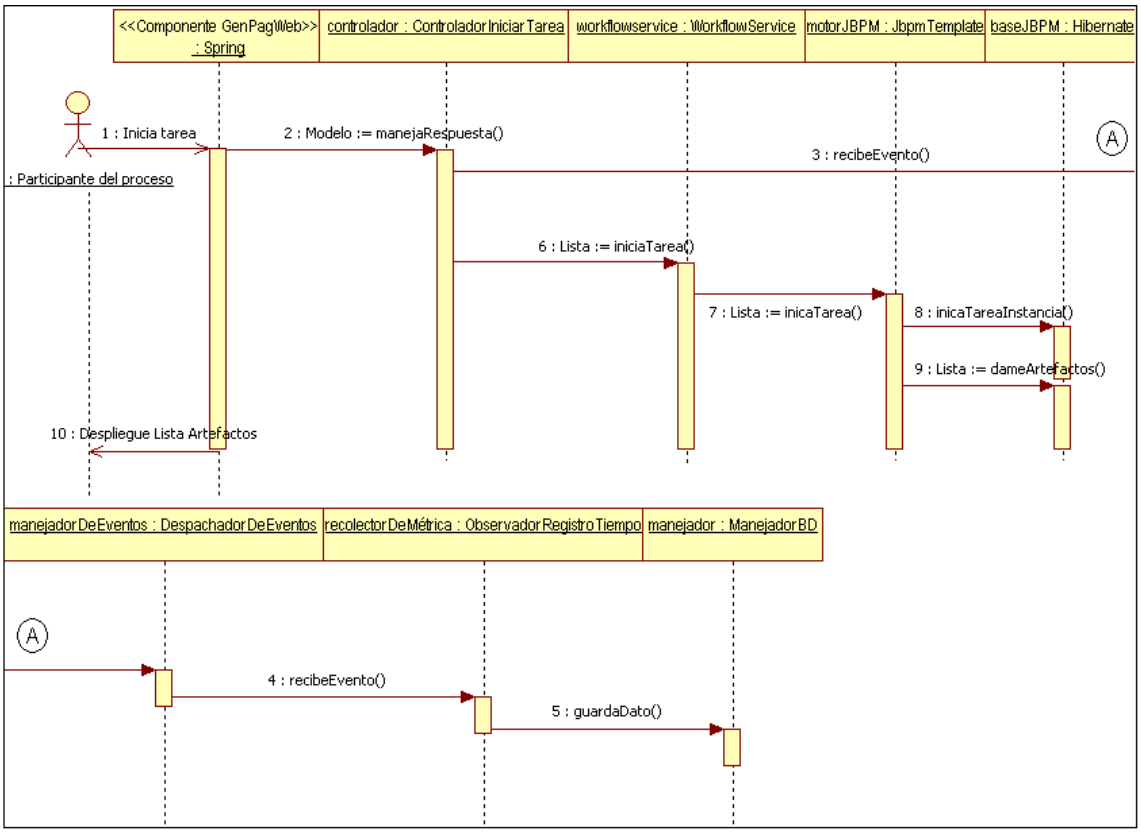


Figura 15: Vista Dinámica del Caso de Uso de *Iniciar Tarea*

En los Diagramas de Secuencia, se presentan los objetos que se crean durante la ejecución del programa y las clases a las que corresponden. La secuencia en la Vista Dinámica de la Figura 15 es la siguiente:

1. El **GeneradorDePagWeb** recibe la petición del Participante del Proceso.
2. El **controlador** recibe la petición y procede a su manejo.
3. El **manejadorDeEventos** recibe el evento de inicio de tarea.
4. El re**colectorDeMetricas** como observador recibe el evento de inicio de tarea y guarda la información de inicio.
5. La **baseDeDatos** almacena los datos correspondientes a la métrica de "Tiempo de desarrollo" que le proporciona el manejador.
6. El **controlador** maneja la petición haciendo solicitudes al workflowservice.
7. El **workflowservice** hace petición al motor para que inicie la tarea y provea la lista de artefactos.
8. El **motorJBPM** solicita el estado de la Instancia del Proceso, lo afecta con el inicio de la tarea y guarda el nuevo estado. También solicita la lista de artefactos de la tarea.
9. La **baseJBPM** provee y almacena el estado de la Instancia del Proceso y provee la lista de artefactos de la tarea.

10. La lista de artefactos regresa hasta el GeneradorDePagWeb que la despliega al Participante del Proceso.

En la siguiente sección se explica cómo se satisfacen las Directrices Arquitectónicas integradas por: Casos de Uso Primario, Atributos de Calidad y Restricciones.

3.3.5 Evaluación de la Arquitectura

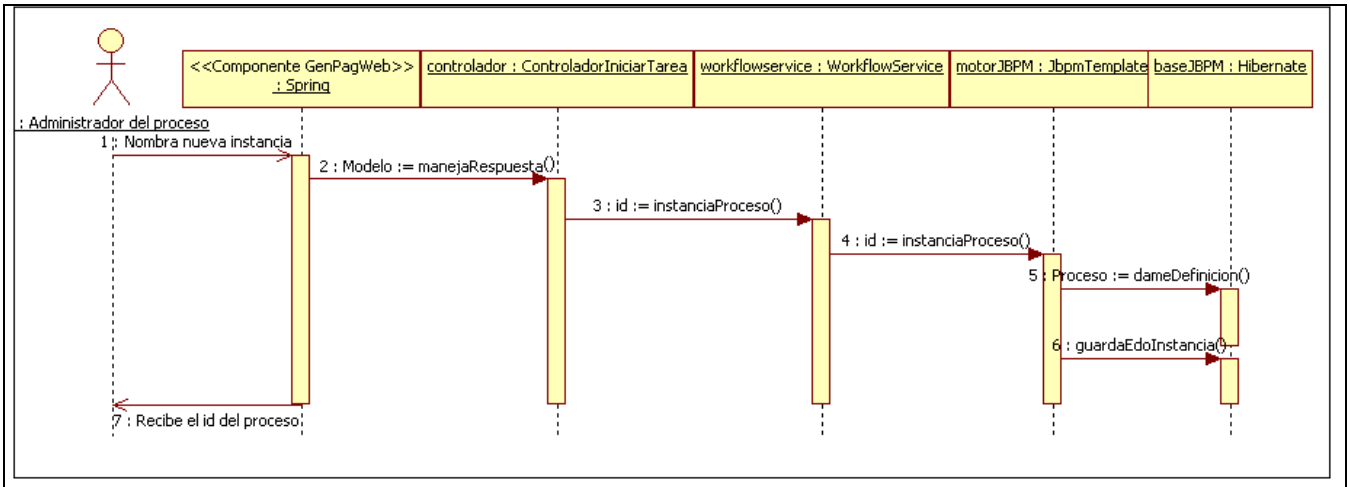
A continuación se presenta una evaluación de la Arquitectura de la Suite BPM mostrando la manera en que se satisfacen las Directrices Arquitectónicas.

3.3.5.1 Cumplimiento de los Casos de Uso Primarios.

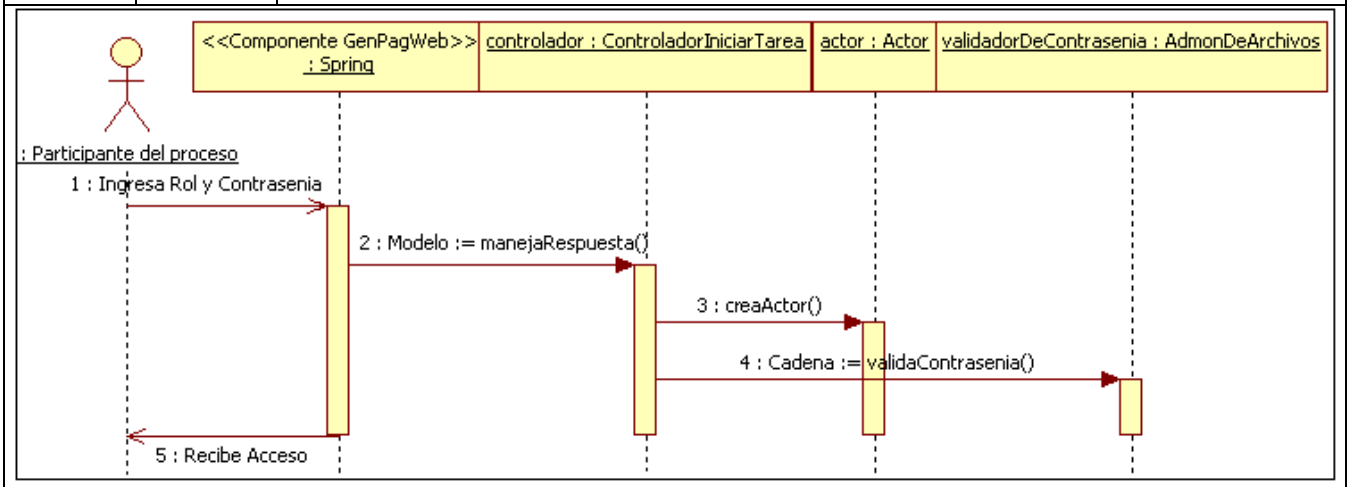
En la Tabla 17 se muestran los Casos de Uso Primarios con los correspondientes componentes de la arquitectura que los satisfacen. A modo de nota, en la Tabla 17 también se muestra el uso del componente BaseDeDatos, para simplificar el diagrama sólo se presenta este componente, sin embargo, el uso de este componente involucra el uso de los componentes FabricaDAO, UsuarioDAO_ER y ManejadorBD.

Tabla 17: Casos de Uso Primarios y los componentes que los satisfacen

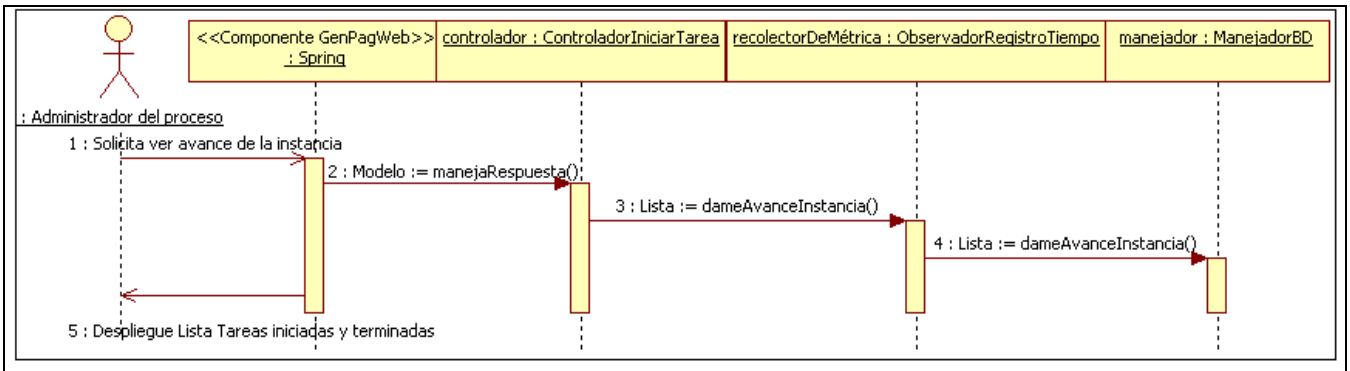
Identificador	Caso de Uso	Componentes y Descripción
E1	Instanciar definición del proceso	**1. El Administrador del Proceso.** Crea una nueva instancia y le da nombre. **2. GeneradorDePagWeb.** Recibe petición del Administrador del Proceso. **3. Controlador.** Maneja petición y hace su solicitud al WorkflowService. **4. WorkflowService.** Interactúa con el MotorJBPM para instanciar la Definición del Proceso. **5. MotorJBPM.** Solicita a la BaseDeDatosJBPM la Definición del Proceso y crea la instancia. **6. BaseDeDatosJBPM.** Provee la Definición del Proceso y almacena el estado inicial de la Instancia del Proceso.



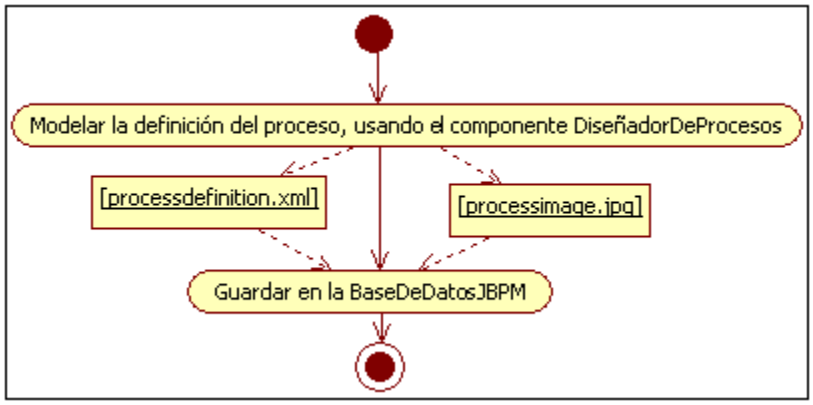
A2	Ingresar al sistema por rol	**1. El Participante del Proceso.** Ingresa a la Suite su rol y contraseña. **2. GeneradorDePagWeb.** Recibe rol y contraseña del participante. **3. Controlador.** Crea el actor y Verifica la contraseña. **4. Actor.** Se crea el objeto actor. **5. ValidadorDeContrasenia.** Valida la contraseña ingresada y devuelve acceso (o negación).



A3	Iniciar tarea	Explicado en la sección 3.3.4.3.
A4	Terminar tarea	Utiliza los mismos componentes que el Caso de Uso iniciar tarea.
B7	Monitorear el avance de la instancia del proceso	**1. El Administrador del Proceso.** Solicita ver avance de la instancia. **2. GeneradorDePagWeb.** Recibe petición del Administrador del Proceso. **3. Controlador.** Maneja petición y hace su solicitud al recolectorDeMétrica (objeto de la clase ObservadorRegistroTiempo). **4. RecolectorDeMétrica.** Solicita a la Base de datos la métrica de “Tiempo de desarrollo”. **5. BaseDeDatos.** Provee la información de la métrica de “Tiempo de desarrollo” que consiste en una lista de tareas iniciadas y terminadas, con sus respectivos registros de tiempos.



E8	Actualizar e instalar nueva versión de la definición del proceso	**1. DiseñadorDeProcesos.** Provee los elementos gráficos de modelado para actualizar la Definición del Proceso. **2. BaseDeDatosJBPM.** Almacenamiento de la Definición del Proceso. Se utiliza un Diagrama de Actividades a continuación, y no un Diagrama de Secuencia porque este Caso de Uso no se lleva a cabo durante la ejecución del programa.



3.3.5.2 Cumplimiento de los Atributos de Calidad

En esta sección se presenta a un alto nivel el cumplimiento de los Atributos de Calidad de la Suite BPM de las categorías: Modificabilidad, Extensibilidad y Seguridad.

3.3.5.2.1 Modificabilidad

El Atributo de Calidad RNF-01 visto en la Tabla 8, de la categoría de Modificabilidad, se refiere a que el sistema debe permitir a un analista del proceso crear o modificar el modelo de la Definición del Proceso, y que el

sistema, sin haber requerido cambios en su codificación, debe conservar la misma funcionalidad, es decir, debe seguir presentando las mismas pantallas con el contenido de acuerdo a los cambios realizados. Este Atributo de Calidad se satisfizo con la utilización de los componentes de la Tabla 18.

Cabe mencionar que los componentes DiseñadorDeProcesos y BaseDeDatosJBPM, intervienen durante la creación o modificación del modelo de la Definición del Proceso creando una nueva versión, una vez realizado esto, se instancia esta nueva versión del proceso, y es cuando intervienen los componentes: GeneradorPagWeb, Controlador, Workflowservice, MotorJBPM y BaseDeDatosJBPM. Ninguno de estos componentes tiene alguna modificación en su código para presentar los cambios que tuvo el modelo.

Tabla 18: Componentes que implementan RNF-01

Componente	Descripción
DiseñadorDeProcesos	La Definición del Proceso se modela con el Modelador Gráfico de Procesos (ver sección 2.4.2), este, se encuentra implementado con el DiseñadorDeProcesos (ver Tabla 14 de la sección 3.3.4.1). Cuando se crea o cambia el modelo de la Definición del Proceso, es decir, se utilizan los elementos gráficos JPDL, el DiseñadorDeProcesos crea o modifica el documento XML que contiene la nueva versión de la Definición del Proceso y la almacena en la BaseDeDatosJBPM.
BaseDeDatosJBPM	Almacena el modelo de la Definición del Proceso.
MotorJBPM	Básicamente instancia el proceso y lee o hace sobre la instancia lo que el Workflowservice le solicita.
Workflowservice	Esta clase se creó para encapsular el comportamiento de los procesos, tales como: *instanciar proceso*, *leer actividades*, *iniciar tarea del proceso.* Dentro del contenido de los métodos de esta clase se invoca al MotorJBPM, que se conecta con la BaseDeDatosJBPM via Hibernate (ver Tabla 15 de la sección 3.3.4.1), también contienen para cada elemento de JPDL utilizado en el modelado del proceso, una codificación adicional para crear el comportamiento que el Participante o Administrador del Proceso espera.
Controlador	Recibe y da respuesta a las peticiones del Participante o Administrador del Proceso. Implementado de acuerdo a Spring MVC (del que se habló

	en la sección 3.3.3.2). El controlador que implementa la interfaz de Spring MVC esta implementado de manera dinámica, para dar respuesta independientemente del contenido.
GeneradorPagWeb	Las páginas Web están codificadas de manera dinámica para dar respuesta independientemente del contenido.

El GeneradorDePagWeb se divide en los componentes: mappings.xml, workflowControllers.xml, tareas.jsp (el nombre del archivo jsp es de acuerdo a la pantalla que se desea mostrar en la Suite BPM). En la Figura 16, se describe la interacción entre estos componentes de acuerdo al siguiente ejemplo.

El analista del proceso modifica el modelo de la Definición del Proceso en lo concerniente a la redacción de una tarea. El Participante del Proceso ingresa a la Suite BPM y selecciona una actividad para conocer las tareas correspondientes a esa actividad y a su rol. Al desplegarse las tareas, la redacción de la tarea ha cambiado de acuerdo con las modificaciones realizadas sin que se hayan realizado cambios en la codificación de la Suite BPM, sólo en el modelo de la Definición del Proceso.

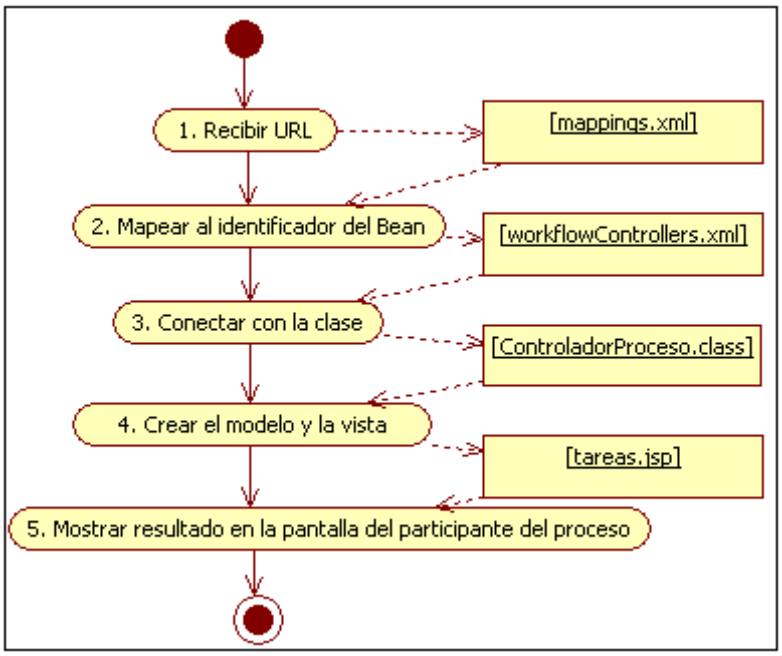


Figura 16: Actividades en la secuencia de interacción de los componentes que conforman el GeneradorPagWeb

En la Figura 16 se incluye un controlador y la descripción de las actividades es la siguiente:

1. El Participante del Proceso selecciona una actividad eligiendo el Localizador Uniforme de Recursos (URL, por sus siglas en inglés) controladorproceso.htm (liga en la página de actividades).
2. Con el archivo mappings.xml, se hace el mapeo de “controladorproceso.htm” a ”controladorProceso” que es el identificador del bean. Un bean es un componente reutilizable.
3. El workflowControllers.xml contiene los beans, en este ejemplo, el bean “controladorProceso” está conectado al controlador ControladorProceso (clase codificada en java) que resuelve la petición.
4. El controlador ControladorProceso crea el modelo que contiene la lista de tareas específicas al rol solicitante (para lo cual interactuó con el Workflowservice) e incluye la Vista, que en este caso es una página JSP (JavaServerPages), llamada tareas.jsp.
5. El Participante del Proceso mira en su pantalla la lista de tareas que le corresponden a su rol.

La Tabla 19 muestra la descripción de los componentes del GeneradorDePagWeb.

Tabla 19: Implementación de los archivos que integran el componente GeneradorPagWeb

Componentes	Descripción
mappings.xml	Este archivo contiene los mapeos entre los URL con extensión .htm y los identificadores de los beans que corresponden a los controladores. A continuación se muestra como ejemplo el mapeo del URL controladorproceso.htm mapeado al identificador del bean que en este ejemplo es controladorProceso: `<propkey="/controladorproceso.htm">controladorProceso</prop>`
workflowControllers.xml	Este archivo contiene los Beans (con sus respectivos identificadores) para crear objetos en este caso de los controladores. A continuación se muestra como ejemplo el bean con identificador “controladorProceso” conectado con la clase “ControladorProceso” que es el nombre del controlador encargado de resolver la petición del usuario cuando solicita la lista de tareas.

	`<bean id="controladorProceso" class="mx.uam.izt.prototipo.jbpm.web.ControladorProceso"></bean>`
ControladorProceso.class	Es el archivo compilado del archivo java que contiene la clase ControladorProceso, que como ya se mencionó es el encargado de resolver la petición del participante, para esto solicita al componente "WorkflowService" la lista de tareas y crea el modelo. **(este archivo no forma parte del componente GeneradorPagWeb).**
tareas.jsp	Este archivo, como todos los archivos con extensión JSP de la Suite BPM, contiene etiquetas de JSTL (JSP Standard Tag Library), que es un conjunto de librerías de etiquetas y estándares que encapsulan la funcionalidad de la página, usada comúnmente para escribir páginas JSP. En este ejemplo, es la página Web "tareas.jsp". En esta página JSP se encuentra las etiquetas JSTL que permiten extraer la lista de tareas del modelo y presentarlas independientemente del contenido. Cada página JSP tiene uno o más URL con extensión .htm que son seleccionadas por los usuarios. Al ser elegidos los URL son mapeados a los controladores correspondientes que atienden las peticiones.

3.3.5.2.2 Extensibilidad

El Atributo de Calidad RNF-02 visto en la Tabla 8, de la categoría de Extensibilidad, se refiere a que el sistema debe permitir agregar componentes, para recolectar métricas que permitan la realización de distintos tipos de análisis, sin modificar más de un componente de la Arquitectura de la Suite BPM. Este Atributo de Calidad se satisfizo con la utilización del Patrón Observador (ver sección 3.3.2.3).

En el Patrón Observador existe un sujeto, en este caso el DespachadorDeEventos, que registra a los observadores en una lista dinámica, estos observadores implementan la interfaz IObservadorEventoTarea, cada observador puede tener una distinta implementación de la interfaz de acuerdo a la métrica que se quiera recolectar. De esta manera, se pueden agregar

observadores (componentes) sólo afectando a un solo componente, el DespachadorDeEventos (sujeto), incluyéndolos en su lista dinámica.

En la Suite BPM, se cuenta con un solo observador, el ObservadorRegistroDeTiempo, que implementa la interfaz IObservadorEventoTarea para almacenar en la Base de Datos la fecha y hora de inicio y terminación de las tareas, eventos que le son notificados por el DespachadorDeEventos (ver Figura 17).

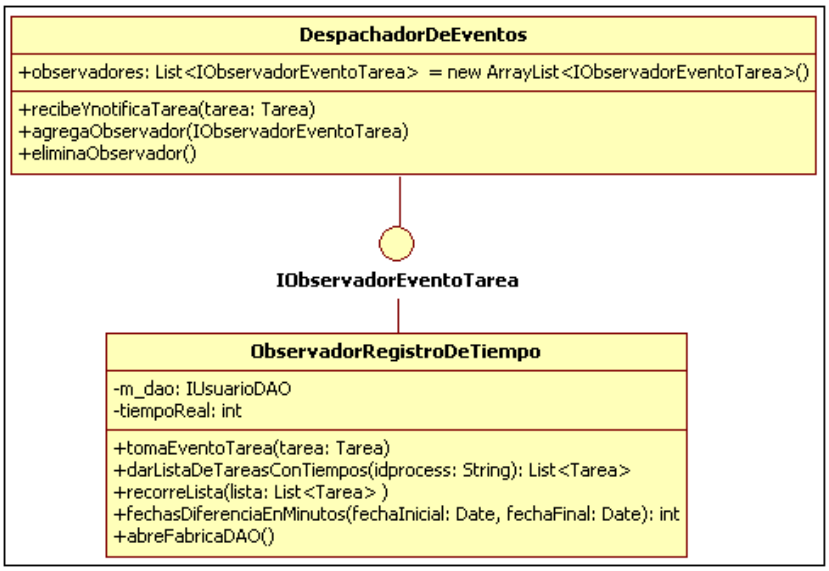


Figura 17: Implementación del Patrón Observador para la Suite BPM

3.3.5.2.3 Seguridad

El Atributo de Calidad RNF-03 visto en la Tabla 8, de la categoría de Seguridad, se refiere a que el sistema debe restringir el acceso a sus servicios. Este Atributo de Calidad se satisfizo con la utilización de los componentes que se describen en la Tabla 20.

Tabla 20: Componentes que implementan RNF-03

Controlador	Recibe y da respuesta a las peticiones del Participante o Administrador del Proceso. Implementado deacuerdo a Spring MVC (del que se habló en la sección 3.3.3.2). El controlador que implementa la interfaz de Spring MVC de formulario en este caso, esta implementado para recibir y validar la contraseña.
GeneradorPagWeb	Recibe el rol y la contraseña.
Actor	Cada vez que ingresa un Participante del Proceso sus datos, como rol y contraseña, son atributos de un objeto de la clase Actor.
AdmonDeArchivos	Valida la contraseña

Para implementar el Atributo de Calidad RNF-03 de la categoría de Seguridad se utiliza el componente GeneradorDePagWeb que se divide en los componentes: mappings.xml, workflowControllers.xml, entradaactor.jsp, actorcargado.jsp (el nombre de los archivos jsp es de acuerdo a la pantalla que se desea mostrar en la Suite BPM). En la Figura 18, se describe la interacción entre estos componentes y los de la Tabla 20 de acuerdo al siguiente ejemplo.

Un Participante del Proceso estando en la página de bienvenida ingresa a la página donde introduce su rol y su contraseña. El participante ve la ventana de éxito o fracaso según la respuesta de la validación de su contraseña.

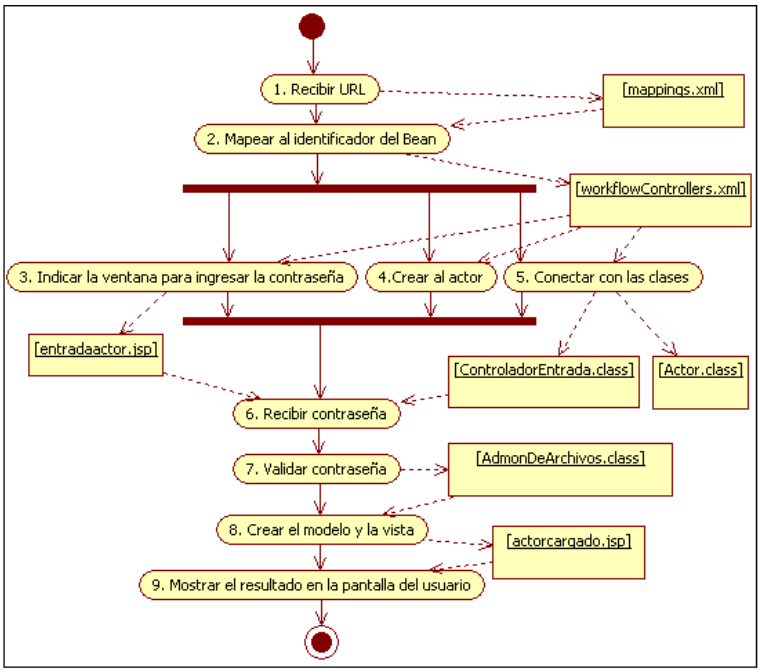


Figura 18: Diagrama de Actividades del acceso para implementar el Atributo de Calidad de la categoría de Seguridad.

La descripción de las actividades de la Figura 18 es la siguiente:

1. El Participante del Proceso ingresa a la página de validación de contraseña a través del Localizador Uniforme de Recursos (URL, por sus siglas en inglés) entradaactor.htm (liga en la página principal).
2. Con el archivo mappings.xml, se hace el mapeo de “entradaactor.htm” a “controladorEntrada” que es el identificador del bean.
3. El workflowControllers.xml contiene los beans, en este ejemplo, el bean “controladorEntrada” tiene un conjunto de propiedades entre ellas el nombre del jsp que se debe presentar al Participante del Proceso para que ingrese su contraseña, en este caso entradaactor.jsp.
4. El workflowControllers.xml crea el objeto actor.
5. El bean “controladorEntrada” está conectado a la clase (controlador) ControladorEntrada (ver sección 3.3.2.2 el Patrón Modelo Vista Controlador) que resuelve la petición.
6. El ControladorEntrada recibe la contraseña ingresada por el Participante del Proceso.
7. El ControladorEntrada interactúa con el AdmonDeArchivos, este último abre el archivo que contiene las contraseñas que corresponden a cada rol, devuelve la respuesta al ControladorEntrada.
8. El ControladorEntrada pone en el modelo la respuesta de la validación y el nombre del jsp (actorcargado.jsp) que mostrará al Participante del Proceso si su contraseña es válida o no.
9. Se le presenta al Participante del Proceso una página de éxito con liga a la lista de instancias del proceso o una página con una liga a la ventana principal en caso de fracaso.

El detalle de la implementación que contienen los componentes que integran la secuencia de acceso al sistema se describen en la Tabla 21.

Tabla 21: Implementación de los componentes que integran la secuencia de acceso restringido.

Archivo	Descripción
mappings.xml	Este archivo contiene los mapeos entre los URL con extensión ".htm" y los identificadores de los beans que corresponden a los controladores. A continuación se muestra como ejemplo el mapeo del URL entradaactor.htm mapeado al identificador del bean que en este ejemplo es controladorEntrada: `<propkey="/entradaactor.htm">controladorEntrada</prop>`
WorkflowControllers .xml	Este archivo contiene los Beans (con sus respectivos identificadores) para crear objetos en este caso de los controladores. A continuación se muestra como ejemplo el bean con identificador "controladorEntrada" conectado con la clase "ControladorEntrada" que es el nombre del controlador encargado de resolver la petición. `<bean id="controladorEntrada" class="mx.uam.izt.prototipo.jbpm.web.ControladorEntrada"> <property name="sessionForm"><value>true</value></property> <property name="commandName"><value>actor</value></property> <property name="commandClass"><value>mx.uam.izt.prototipo.jbpm.Actor</value></property> <property name="formView"><value>entradaactor</value></property> </vean>` La propiedad "commandName" tiene de valor el nombre del objeto que va a recibir los datos del formulario, en este caso, actor, "commandClass" tiene como valor la clase a la que pertenece el objeto, en este caso, Actor, "formView" su valor es el nombre del JSP que tiene el formulario, en este caso, entradaactor, y "sessionForm" su valor es verdadero, lo que significa que se utiliza la misma instancia del bean para toda solicitud, en caso de falso se crearía una nueva instancia para cada solicitud.
entradaactor.jsp	Este archivo con extensión jsp, como ya se mencionó, contiene etiquetas de JSTL (JSP Standard Tag Library), que es un conjunto de librerías de etiquetas y estándares que encapsulan la funcionalidad de

	la página, usada comúnmente para escribir páginas JSP. En este ejemplo es la página Web “entradaactor.jsp”. En esta página JSP se encuentra las etiquetas JSTL que permiten enviar la contraseña al controlador.
ControladorEntrada.class	Es el archivo compilado del archivo java que contiene la clase ControladorEntrada, crea el objeto de la clase Actor, recibe la contraseña e invoca un método de la clase AdmonDeArchivos para validar la contraseña
Actor.class	Es el archivo compilado del archivo java que contiene la clase Actor.
AdmonDeArchivos.class	Es el archivo compilado del archivo java que contiene la clase AdmonDeArchivos que contiene el método que abre el archivo de contraseñas para validar la contraseña.
actorcargado.jsp	En esta página JSP se encuentra las etiquetas JSTL que permiten extraer la respuesta de la validación y en función a ella presentar una ventana de éxito o de rechazo.

3.3.5.3 Satisfacción de restricciones

En la Tabla 22 se muestran las restricciones de la Suite BPM y la explicación de cómo se satisfacen.

Tabla 22: Cumplimiento de las restricciones de la Suite BPM

Restricción	Explicación
El sistema debe estar construido usando componentes open source (para soportar bajo costo).	Se tomaron componentes de una suite open source existente.
El sistema debe permitir el acceso vía Web (para soportar trabajo en equipo).	El servidor donde se ejecuta la Suite BPM también es un Servidor Web.
El soporte del control de versiones debe realizarse a través de la integración con un sistema de control de versiones existente.	La Suite BPM está conectada al sistema de control de versiones Subversión.
El sistema se debe ejecutar en ambientes Linux y Windows.	La Suite BPM está programada en java ejecutable en los sistemas operativos Linux y Windows.

3.4 Implementación de la Suite BPM

En esta sección se presenta la implementación de la Suite BPM para el Caso de Uso *Iniciar Tarea*; para satisfacer el mencionado Caso de Uso (ver flujos del caso de uso en el Apéndice B) se requirió adicionar funciones a la Suite BPM, es decir, estas funciones no están incluidas en su funcionamiento tradicional (ver sección 2.4.2). A estas implementaciones adicionales en secciones posteriores se les nombra como adaptaciones.

3.4.1 Primera adaptación: Bajar y subir artefactos

En el caso de uso *Iniciar Tarea*, se hicieron dos adaptaciones: *bajar y subir artefactos* y *control de versiones*. A continuación se presenta la implementación de bajar y subir artefactos. La Figura 19 muestra las actividades que lleva a cabo el sistema cuando se inicia una tarea y los archivos involucrados que contienen el código implementado.

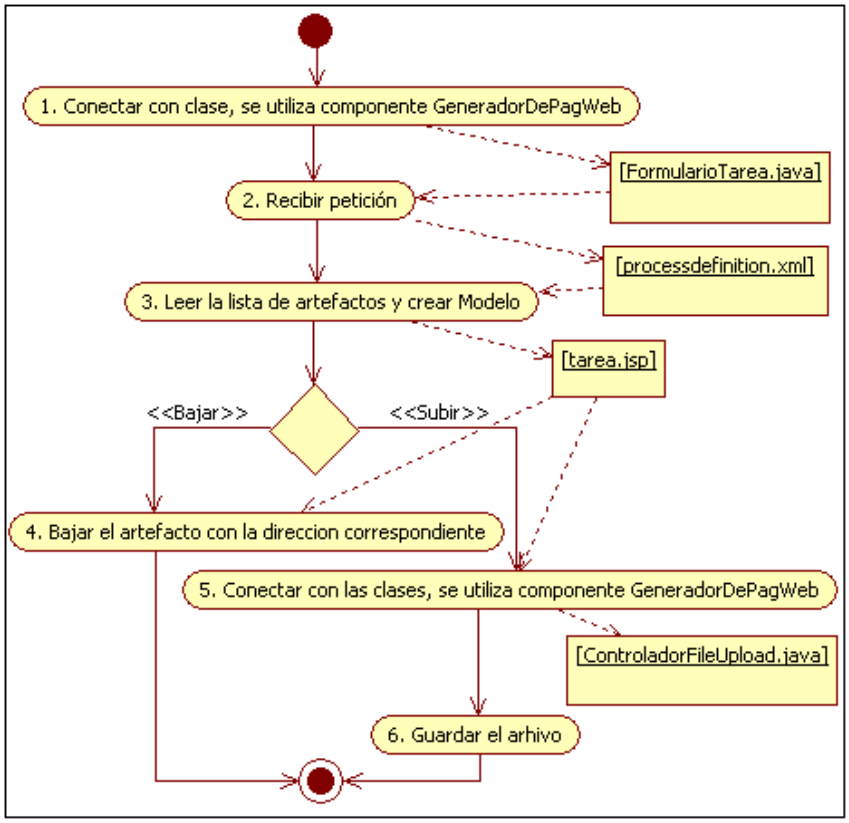


Figura 19: Diagrama de Actividades para explicar la implementación de la adaptación de bajar y subir artefactos

A continuación se comenta el código implementado para cada actividad del Diagrama de Actividades de la Figura 19:

1. El código implementado para el componente GeneradorPagWeb, fue presentado en la sección 3.3.5.2.1.

2. La clase FormularioTarea, que es un controlador, recibe la petición en su método formBackingObject (ver Figura 21) y del parámetro *request* obtiene la información para dar respuesta a la petición.

3. Es necesario mencionar que JPDL permite agregar variables a las tareas (esto se explica con más detalle en la sección 5.2.1.1). Estas variables se les llamó de acuerdo a los nombres de los artefactos que le corresponden a la tarea, éste fue el único mecanismo disponible para tenerlos asociados a cada tarea. En la Figura 20 se muestra un ejemplo de una tarea con sus variables; este ejemplo también muestra como se ve un fragmento del archivo processdefinition.xml que lo crea el componente DiseñadorDeProcesos (ver sección 3.3.4.1); este archivo es almacenado en la BaseDeDatosJBPM.

```
<task-node name="Nombre de la actividad">
    <task name="B.Nombre de la tarea">
        <assignment actor-id="Nombre del rol"></assignment>
        <controller>
            <variable access="read" name="Nombre del artefacto 1"></variable>
            <variable access="read" name="Nombre del artefacto 2"></variable>
        </controller>
    </task>
</task-node>
```

Figura 20: Fragmento del archivo processdefinition.xml, ejemplo de una tarea con sus variables

El código del método formBackingObject, se ocupa de leer los nombres de las variables y los guarda en el objeto llamado *control* que es devuelto (ver Figura 21).

```
/** Método: formBackingObject
Maneja la peticion ha este controlador para desplegar las variables
Entrada: request de tipo HttpServletRequest
Salida: Object Regresa control*/
protected Object formBackingObject(HttpServletRequest request) throws ServletException {
  //Se crea objeto control para almacenar los nombres de los artefactos
  InterTarea control=new InterTarea();
  //Se extraen los parámetros de navegabilidad y se almacenan en el objeto control
  control.setActR(request.getParameter("actR"));
  control.setCmdR(request.getParameter("cmdR"));
  control.setId(request.getParameter("id"));
  control.setRol(request.getParameter("rol"));
  //Se extrae el nombre de la tarea
  nombreTarea= request.getParameter("tarea");
  //La primera letra del nombre de la tarea indica, si ésta tarea
  //tiene artefactos para ser subidos, bajados o ambos.
  String bandera = nombreTarea.substring(0,1);
  //Se le dice al workflowService que obtenga, de la BaseDeDatosJBPM,
  //y entregue una tarea específica de la actividad donde se encuentra el proceso
  TaskInstance task=workflowService.getSubTaskInstance(idProcessInstance,nombreTarea,rol);
  //Se obtiene la colección de variables asociadas a esa tarea.
  Collection var =task.getVariableInstances().values();
  //Se itera y se guarda en el objeto control el nombre de cada variable
  //que es también el nombre de cada artefacto.
  for (Iterator it = var.iterator(); it.hasNext();) {
      Variable variable = new Variable();
      VariableInstance tok=(VariableInstance)it.next();
      variable.setNombre(tok.getName());
      control.addObj(variable);
  }
  control.setVariables(task.getVariableInstances());
  control.setNombre(task.getName());
  control.setBandera(bandera);
  return control;
```

Figura 21: Código del método formBackingObject de la clase FormularioTarea

El archivo tarea.jsp contiene (ver Figura 22) el código para mostrar la lista de artefactos. En este código se hace una iteración sobre los nombres de las variables que se encuentran en el objeto “control”. Con cada nombre se forma la ruta donde se ubica el artefacto. La bandera sirve para determinar si en la tabla de artefactos, al participante se le permitirá subirlos, bajarlos o ambas opciones.

```
<!-- Se crea el encabezado de la tabla-->
    <td align="center">Nombre</td>
    <td align="center">Plantilla</td>
    <td align="center">Subir Nueva version</td>
    </tr><!-- Se Itera para conocer cada variable-->
    <c:forEach items="${control.variables}" var="num" varStatus="i">
    <tr><!-- Se muestra al usuario el nombre de cada artefacto-->
        <td><c:out value="${num.key}"/></td>
        <!-- Se decide si se trata de un artefacto a ser bajado-->
        <c:if test="${control.bandera=='B'}">
        <!-- Se muestra al usuario la palabra Bajar como una liga-->
        <!-- Ésta esta asociada a la dirección donde se encuentra el artefacto-->
            <td><a href="<c:url value="Artefactos/${control.id}/${num.key}"/>">Bajar</a></td>
            <!-- Se muestra en otro color la palabra Subir al usuario-->
            <td><font color='#CCCCCC'>Subir</font></td>
        </c:if>
        <!-- Se decide si se trata de un artefacto a ser subido-->
        <c:if test="${control.bandera=='S'}">
            <!-- Se le da a Spring el nombre del archivo a subir-->
            <spring:bind path="control.lista[${i.index}].valor">
                <c:if test="${status.value==null}">
                <!-- Se muestra en otro color la palabra Bajar al usuario-->
                <td><font color='#CCCCCC'>Bajar</font></td>
                <td>
                <!-- Se muestra al usuario la palabra Subir como una liga-->
                <!-- Ésta esta asociada a la URL upload.htm que se mapea al Bean -->
                <!-- que se conecta al controlador que atenderá la petición -->
                <a href="<c:out value="upload.htm?id=${control.id}&tarea=${num.key}&actR=
                ${control.actR}&cmdR=${control.cmdR}&rol=${control.rol}"/>">Subir</a>
                </td>
            </c:if>
            </spring:bind>
        </c:if>
```

Figura 22: Código del archivo tarea.jsp que implementa el despliegue de los nombres de los artefactos

4. En el código de la Figura 22 se observa el siguiente renglón:<td><a href="<c:urlvalue="Artefactos/${control.id}/${num.key}"/>">Bajar</td>. Este código corresponde a la ruta de la carpeta donde se ubica el artefacto que se desea bajar.

5. El código implementado para el componente GeneradorPagWeb, fue presentado en la sección 3.3.5.2.1.

6. Para implementar la opción de subir los artefactos se utilizó código que proporciona Spring Framework (ver sección 3.3.3.1). En la Figura 23 se muestra el código del ControladorFileUpload para guardar los archivos localmente y en el repositorio remoto de archivos (ver sección 3.3.4.2).

```
/*---------------------------------------*/
/**Método:onSubmit
Este método se encarga de subir a la carpeta local el archivo que el usuario sube
de su sistema también invoca al método de la clase para que lo suba al repositorio
Entrada: HttpServletRequest req, HttpServletResponse res,Object command,
BindException errors
Salida: ModelAndview, el modelo para el jsp*/
@Override
protected ModelAndView onSubmit(HttpServletRequest req,
                                HttpServletResponse res,
                                Object command,
                                BindException errors) throws Exception {
    String mensaje;
    FileUploadControl upload = (FileUploadControl) command;
    MultipartFile file = upload.getFile();
    //Guardar el archivo localmente
    boolean respuesta = adminArchivos.guardaArchivoUpload(file,uploadName,id);
    if(respuesta == true)
        mensaje="true";
    else
        mensaje="false";
    //Se manda al repositorio que le corresponde al archivo
      adminSubversion.comitArchivoUpload(id,uploadName);
```

Figura 23: Código del método onSubmit de la clase ControladorFileUpload

3.4.2 Segunda adaptación: Control de versiones

La segunda adaptación para implementar el Caso de Uso *Iniciar Tarea*, corresponde al *control de versiones* de los artefactos de las tareas; la implementación fue una adaptación del código de Subversión *[45, sitio Web]*. Como ejemplo de este código la Figura 24 muestra el método comitArchivoUpload utilizado para subir el artefacto al repositorio.

En la Figura 24 se observa cómo se crean las condiciones para subir el archivo, estas consisten en proporcionar: la URL donde se ubica el repositorio, el nombre de usuario y la palabra de acceso para cumplir con los requisitos de seguridad, entre otros. Después propiamente se envía el archivo.

```
/*---------------------------------------*/
/**Método:comitArchivoUpload
Para subir el archivo al repositorio remoto
Entrada: id, uploadName
Salida: no devuelve nada**/
public void comitArchivoUpload(String id, String uploadName){
    setupLibrary();
    try {
        ISVNEditor editor = creaCondiciones(id);
        SVNCommitInfo commitInfo = comitUpload(editor, id, uploadName);
    } catch (SVNException e) {
        SVNErrorMessage err = e.getErrorMessage();
        while(err != null) {
            System.err.println(err.getErrorCode().getCode() + " : " + err.getMessage());
            err = err.getChildErrorMessage();
        }
        System.exit(1);
    }
}
```

Figura 24: Código del método comitArchivoUpload de la clase AdministradorDeSubversion

3.4.3 Tercera adaptación: Descripción de las tareas

La tercera adaptación consistió en agregar la descripción de las tareas. La Figura 25 muestra el código para asignarle a una tarea específica la descripción que le corresponde, esto sucede cuando la tarea es iniciada por el participante. Este código pertenece a la clase FormularioTarea.

```
//Se solicita al workflowService que obtenga de la BaseDeDatosJBPM
//una tarea específica.
TaskInstance task=workflowService.getSubTaskInstance(idProcessInstance,nombreTarea,rol);

if(task.getStart()==null){
    //Se inicia la tarea
    task.start();
    //Se obtiene la descripción de la tarea tomada del archivo de descripciones
    String descripcion = adminDescripcionTEstimado.darDescripcion(nombreTarea, rol);
    //Se asigna la descripción a la tarea
    task.setDescription(descripcion);
    //Se solicita al workflowService que guarde la tarea en la BaseDeDatosJBPM
    //ya iniciada y conteniendo la descripción
    workflowService.saveTask(task);
}
```

Figura 25: Líneas de código de la clase FormularioTarea

Como se observa en la Figura 25 cuando la tarea es “marcada” como iniciada y le es asignada su descripción se guarda en la BaseDeDatosJBPM. El encargado de desplegar las tareas en la pantalla del participante es el

ControladorProceso. Éste le solicita al workflowService que extraiga de la BaseDeDatosJBPM las tareas (iniciadas y no iniciadas) que le corresponden al rol del participante y de una actividad y proceso determinados, para colocarlas en el modelo (ver Patrón MVC de la sección 3.3.2.2). Lo anterior se implementó como se muestra en la Figura 26.

```
if(workflowService.getTaskInstances(Long.valueOf(id))!=null){
        myModel.put("tareas", workflowService.getTaskInstances(Long.valueOf(id),act,rol));
}
```

Figura 26: Código de la clase ControladorProceso para colocar las tareas en el modelo

La Figura 27 presenta el código para desplegar la tabla de tareas.

```
<Table bgcolor= "#FFFFFF" border = 1>
    <tr bgcolor= "#ADDFFF" >
        <td align="center">Tareas</td>
        <td align="center">Descripción</td>
        <td align="center">Inicio</td>
        <td align="center">Fin</td>
        <td align="center">Iniciar</td>
        <!--<td align="center">Ver</td>-->
        <td align="center">Terminar</td>
    </tr>
    <c:forEach items="${model.tareas}" var="tarea" >
    <tr>
        <td><c:out value="${tarea.name}"/></td>
        <td><c:if test="${ !empty tarea.start }"><c:out value="${tarea.description}"/></td></c:if>
        <td><c:out value="${tarea.start}"/></td>
        <td><c:out value="${tarea.end}"/></td>
        <td><c:if test="${ empty tarea.end }"><a href="<c:out value="iniciartarea.htm?id=${model.id}&tarea=
        ${tarea.name}&actR=${model.actividad}&rol=${model.rol}&cmdR=verTareas"/>">Accesar</a></c:if></td>
        <td><c:if test="${ !empty tarea.start and empty tarea.end }"><a href="<c:out value="editartarea.htm?id=
        ${model.id}&tarea=${tarea.name}&act=${model.actividad}&rol=${model.rol}&cmd=terminar"/>">Terminar</a></c:if>
    </tr>
    </c:forEach>
</Table>
```

Figura 27: Código de la implementación de la tabla de tareas que se muestra al Participante del Proceso

3.5 Síntesis del capítulo

El desarrollo de la Suite BPM que se construyó partió de la identificación de las Directrices Arquitectónicas que incluyen Casos de Uso Primarios, Atributos de Calidad y restricciones (ver Tabla 23).

Tabla 23: Directrices Arquitectónicas

Casos de Uso Primarios	Atributos de Calidad y categoría.	Restricciones
(E1) Instanciar definición del proceso. (A2) Ingresar al sistema por rol. (A3) Iniciar tarea. (A4) Terminar tarea. (B7) Monitorear el avance de la instancia del proceso. (E8) Actualizar e instalar nueva versión de la definición del proceso.	(RNF-01) Modificabilidad. (RNF-02) Extensibilidad. (RNF-03) Seguridad.	(RES-01) El sistema debe estar construido usando componentes open source (para soportar bajo costo) (RES-02) El sistema debe permitir el acceso vía Web (para soportar trabajo en equipo). (RES-03) El soporte del control de versiones debe realizarse a través de la integración con un sistema de control de versiones existente. (RES-04) El sistema se debe ejecutar en ambientes Linux y Windows.

Las Directrices Arquitectónicas guiaron las decisiones del diseño que fueron las siguientes:

- Uso del Patrón de Capas
- Uso del Patrón Modelo Vista Controlador
- Uso del Patrón Observador
- Uso del Patrón de DAO

La descripción del sistema requirió una serie de vistas, donde cada vista representa la estructura de un aspecto particular del sistema, en este caso las vistas que se construyeron son:

- La Vista Lógica
- La Vista de Implantación

- La Vista Dinámica.

En este capítulo también se presentó la evaluación de la arquitectura con el cumplimiento de las Directrices Arquitectónicas. En la Tabla 24 se muestra el resumen de la explicación de cómo se satisfacen los Atributos de Calidad de la Suite BPM de las categorías: Modificabilidad, Extensibilidad y Seguridad.

Tabla 24: Cumplimiento de los Atributos de Calidad

ID	Categoría	Explicación
RNF-01	Modificabilidad	Este requerimiento se cumple con el componente DiseñadorDeProcesos. Porque los cambios en el modelo de la Definición del Proceso pasan automáticamente a la BaseDeDatosJBPM, también se satisface porque un cambio en la Definición del Proceso no afecta el funcionamiento de la Suite BPM debido a que el componente de GeneradorDePagWeb, produce páginas Web de manera dinámica y el Controlador también esta codificado de manera dinámica.
RNF-02	Extensibilidad	Este requerimiento se satisface con el uso del Patrón Observador, que permite agregar de manera dinámica otros observadores, es decir, otros componentes para recolectar métricas que permitan la realización de distintos tipos de análisis.
RNF-03	Seguridad	Este requerimiento se satisface porque restringe el acceso a la Suite BPM.

El cumplimiento de los Requerimientos Funcionales se presenta más adelante en la sección 5.2.

Las tecnologías ocupadas en la implementación de la Suite BPM, son las siguientes:

- Spring.
- Spring MVC.
- Hibernate.
- ANT.
- JPDL.

Capítulo

4 Modelado con la Suite BPM construida

Como ya se ha mencionado antes, una Suite BPM apoya la administración de los procesos en las etapas de: Modelado, Ejecución y Monitoreo. En la etapa de Modelado, es necesario definir los procesos de manera gráfica. En el caso de la Suite BPM construida se utiliza JPDL. Sin embargo, no todos los elementos de JPDL descritos en la sección 3.3.3.5 fueron utilizados, esto se debió a que para satisfacer los Atributos de Calidad y las restricciones, se introdujo entre otros el Patrón MVC (ver 3.3.2), y fue necesario realizar una programación adicional para acoplar los elementos de JPDL a este patrón, programación que se llevó a cabo sólo para los elementos que permitían modelar los procesos de acuerdo a los Requerimientos Funcionales de la Suite BPM. Las características de los procesos de los Modelos de Mejora de Procesos observadas y que sirvieron de criterio para seleccionar los elementos de JPDL se muestran en la Tabla 25, también en esta tabla se muestran los elementos JPDL utilizados y las restricciones en el modelado que surgieron al adaptar los elementos JPDL al patrón de diseño MVC.

Tabla 25: Elementos de JPDL utilizados en la Suite BPM

Características observadas en los procesos de los Modelos de Mejora de Procesos estudiados en este trabajo	**Elemento JPDL que modela esta característica con programación adicional**	**Restricción para el modelado de un proceso**
Existen procesos que están divididos en fases.	**Estado del proceso (Processstate).** Guarda una referencia hacia un subproceso, que no es un subproceso propiamente dicho, sino una fracción del proceso que puede ser aislada como una fase y sólo puede estar constituido por actividades (no por otros	El proceso para su modelado tiene que ser dividido en fases (al menos una fase).

	subprocesos).	
Existen procesos que tienen fases divididas en actividades.	**Nodo-Tarea (Task-Node).** Modela una actividad o subproceso	Las fases del proceso tienen que dividirse en actividades (al menos una actividad).
Existen procesos que tienen actividades divididas en tareas.	**Task**	Las actividades del proceso tienen que dividirse en tareas (al menos una tarea).
Cada tarea es asignada a un rol.	**Asignación (Assignment)**	Es necesario identificar un responsable para cada tarea.
Cada Instancia del Proceso tiene una o más entradas que son utilizadas por una o más tareas y cada una tiene una o más salidas producidas por una o más tareas de la misma Instancia del Proceso.	**Variable**	No hay restricción al modelar este elemento. Es posible modelar tareas con o sin entradas y salidas.
Existen procesos que tienen actividades paralelas.	**Fork, Join**	No hay restricción al modelar este elemento. Es posible modelar o no actividades paralelas, según la Definición del Proceso que se modele.
Existen procesos que tienen actividades iterativas.	**Decision**	No hay restricción al modelar este elemento. Es posible modelar o no actividades iterativas, según la Definición del Proceso que se modele.

Los puntos de la Tabla 25 señalan que la Suite BPM, sólo modela un nivel de profundidad en la jerarquía de procesos, es decir, no modela procesos que contengan subprocesos (dos niveles de jerarquía o más).

Los puntos de la Tabla 25 también muestran la necesidad de hacer una traducción, del modelo de representación del proceso hacia los elementos JPDL utilizados por la Suite BPM. Esta traducción es lo que en este trabajo se llama Procedimiento de Traducción de un Proceso, el cual se detalla en la siguiente sección.

4.1 Procedimiento de Traducción de un Proceso

El Procedimiento de Traducción de un Proceso se refiere a las actividades que hay que llevar a cabo para modelar los procesos con la Suite BPM construida. Estas actividades se presentan en la Tabla 26 y deben ser realizadas. Sin embargo, dependiendo de la información presente en el modelo fuente, es posible que algunas de las actividades no necesiten ser realizadas.

Tabla 26: Actividades del Procedimiento de Traducción de un Proceso

No. De Actividad	Actividad	Elemento identificado	Elementos JPDL que lo modela
1	Se deben Identificar las fases del proceso a modelar.	Fase	Estado del Proceso (Processstate)
2	Se deben identificar las actividades agrupadas en cada fase.	Actividad	Nodo-Tarea (Task-Node)
3	Se deben identificar las actividades secuenciales y/o paralelas.	Paralelismo	Fork Join

4	▪ Se deben identificar las tareas que integran cada actividad (pueden ser una o más tareas). ▪ Se deben identificar los artefactos o plantillas de artefactos que "requiere" cada tarea. ▪ Se deben identificar los artefactos que se "generarán" en cada tarea. ▪ Se deben identificar los casos en que la tarea consista en bajar una plantilla de artefacto y subir esa misma plantilla con datos.	Tarea	Tarea (Task) Escribir el nombre de la tarea precedido por una de las siguientes letras: B. Si la tarea requiere bajar del sistema un artefacto o plantilla. S. Si la tarea requiere subir al sistema un artefacto o plantilla. A. Si la tarea requiere bajar y subir al sistema un artefacto o plantilla. 0. Si no requiere artefacto. Ejemplo: B.Recibir Agenda de trabajo. Esta tarea requiere que el participante baje un artefacto del sistema.
5	En la actividad 4 se debe identificar el nombre del artefacto o plantilla de artefacto con su correspondiente extensión.	Artefacto	Variable
6	Se debe identificar el rol al que se le asigna cada tarea (sea el rol que lleve a cabo la tarea o sea el responsable de ella).	Rol	Asignación (Assignment)
7	Se deben identificar las actividades que se realizan de manera iterativa.	Decisión	Decision
8	De manera breve se debe redactar una descripción de cada tarea.	Descripción de tarea.	No hay un elemento JPDL que modele este elemento.
9	Por cada tarea se debe identificar su tiempo estimado con el que se llevará a cabo.	Tiempo estimado de duración de la tarea.	No hay un elemento JPDL que modele este elemento.

En la actividad 8 de la Tabla 26 se indica que no hay un elemento JPDL que permita hacer una representación textual de las descripciones de las tareas. Estas se tienen por separado, esto forma parte de las adaptaciones que se hicieron a la Suite BPM para que modelara procesos de los Modelos de Mejora de Procesos. Las descripciones de las tareas se almacenan en un archivo (ver el componente “AdmonDeArchivos” en la Vista Lógica de la sección 3.3.4.1) que durante la Instancia del Proceso, el sistema consulta para presentarlas a los participantes del proceso. También en este archivo se anota el tiempo estimado de duración de cada tarea.

Otra adaptación a la Suite BPM fue agregar la funcionalidad de bajar y subir artefactos, este funcionamiento se explica más adelante en la sección 5.2.5.

Para ejemplificar los elementos de JPDL de la Tabla 26, se presenta como ejemplo el Proceso de Contratación de un empleado. En este ejemplo, se entrevistan a varios candidatos llevando a cabo la actividad de reclutamiento de forma iterativa, posteriormente se presenta y entrena al candidato elegido de forma simultánea. Se aplicó el *Procedimiento de Traducción de un Proceso* y los resultados se presentan en la Tabla 27. Los elementos de JPDL con representación gráfica se presentan en la Figura 28 y los que no tienen representación gráfica como la task, assignment y variable, se presentan en la Figura 29.

Tabla 27: Resultados del ejemplo del procedimiento de traducción.

Elemento a identificar	**Elemento identificado**	**Elemento JPDL**
Fase	Contratación	ProcessState
Actividades	Reclutamiento Presentación Entrenamiento	Task-node
Actividades secuenciales	No hay.	
Actividades paralelas	Presentación y Entrenamiento	- Task-node - Fork - Join

Tareas	En la actividad Reclutamiento: - 0.Contactar al candidato - B.Entrevistar al candidato, requiere bajar un artefacto, “guia_ de_entrevista.doc”.	Task
Artefacto	“guia_ de_entrevista.doc”.	Variable
Rol	- La tarea 0.Contactar al candidato es asignada al “Auxiliar de recursos humanos”. - La tarea B.Entrevistar al candidato es asignada al “Gerente de recursos humanos”.	Asignación (Assignment)
Actividades iterativas	Reclutamiento	- Task-Node - Task-Node (preguntar otra iteración, con la task: otra iteración?) - Decision
Descripción de tarea	Tarea 0.Contactar al candidato: el responsable de la tarea llama por teléfono al candidato. Tarea B.Entrevistar al candidato: el responsable de la tarea sigue los lineamientos de la Guía de Entrevista.	
Tiempo estimado de duración de tarea	Tarea 0.Contactar al candidato: 5 min. Tarea B.Entrevistar al candidato: 20 min.	

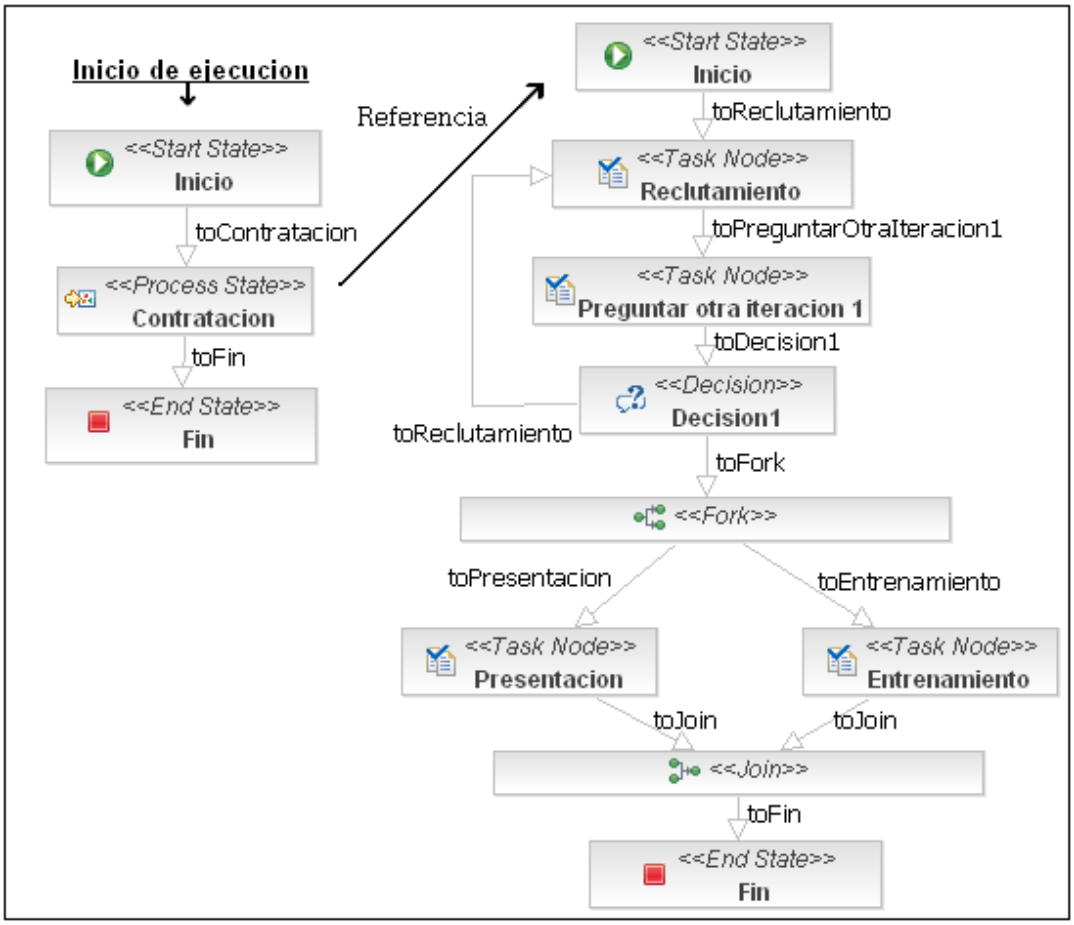


Figura 28: Elementos de JPDL del ejemplo del Proceso de Contratación.

```
<task-node name="Reclutamiento">
  a) <task name="0.Contactar al candidato">
       b) <assignment actor-id="Auxiliar de recursos humanos"></assignment>
          <controller></controller>
     </task>
     <task name="B.Entrevistar al candidato">
          <assignment actor-id="Gerente de recursos humanos"></assignment>
          <controller>
            c) <variable access="read" name="guia_ de_entrevista.doc" ></variable>
          </controller>
     </task>
     <transition to="Otra iteracion" name="toOtraIteracion"></transition>
</task-node>
```

Figura 29: Elementos de JPDL que no tienen representación gráfica, a) task, b) assignment c) variable.

Como ya se ha mencionado antes las descripciones de las tareas se modelan escribiendo estas descripciones en un archivo independiente, en la Figura 30 se presenta la imagen de este archivo para las tareas del Proceso de Contratación.

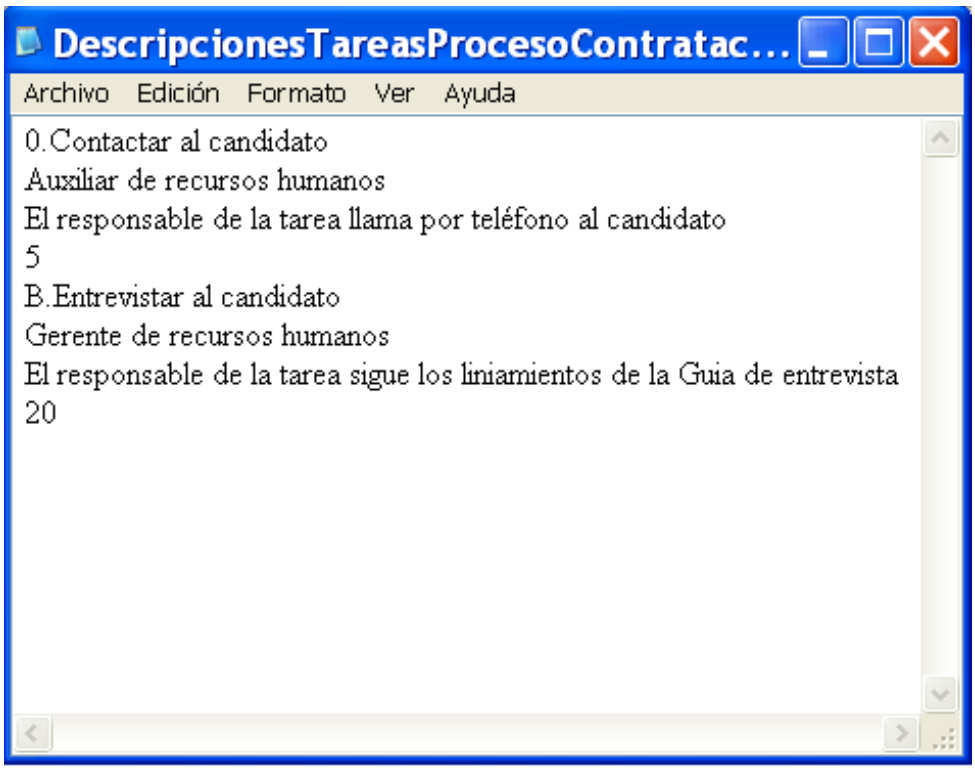


Figura 30: Imagen del archivo que contiene las descripciones de las tareas del Proceso de Contratación de ejemplo.

Al pasar el modelo de un proceso cuya definición no contempla el apoyo de una Suite BPM a un modelo donde la Instancia del Proceso si cuenta con este apoyo, se tienen que hacer algunos ajustes a la definición, como los siguientes:

- Cuando en un paso intervienen más de un rol se divide ese paso en varias task, cada uno con su respectivo rol, de esta forma cada participante es responsable de retroalimentar a la Suite BPM de manera individual, por ejemplo, de indicarle cuando la task es concluida.
- En el caso concreto de la tarea que indica “Enviar un artefacto”, este paso se sustituye en dos o más task, por el task “Subir” un artefacto asignado al rol que tiene la responsabilidad de enviar el artefacto y él o los task “Bajar” asignados a los roles que tienen la responsabilidad de recibir, en este caso, bajar el artefacto del sistema.

- También en el caso de la tarea que indica “Enviar un artefacto” se puede dividir y agrupar esta tarea en dos actividades representadas con dos Task-Node, uno que agrupe la task única de “Subir” el artefacto y el otro Task-Node que agrupe las task de “Bajar” el artefacto asignadas a los roles que tienen la responsabilidad de recibir el artefacto.

Para las pruebas realizadas en este trabajo, se utilizaron procesos modelados con notación EPF. Se creó un mapeo entre los elementos de EPF y los elementos JPDL utilizados; este mapeo se presenta en la siguiente sección.

4.2 Mapeo de los elementos EPF a los elementos de JPDL

Para realizar la evaluación de los objetivos de la investigación, se utilizaron procesos modelados con algunos elementos de la notación EPF. Por esta razón, se determinó un mapeo entre estos elementos de la notación EPF y los elementos JPDL disponibles de la Suite BPM. Este mapeo se presenta en la Tabla 28.

Tabla 28: Mapeo de elementos utilizados entre la notacion EPF y JPDL

Notación EPF	Notación JPDL	Justificación
Fase	Estado del Proceso (ProcessState)	Un “Estado del Proceso” (ProcessState) tiene una referencia de un proceso descrito por separado.
Tarea	Tarea-Nodo (Task-Node)	En notación EPF una tarea es una agrupación de pasos, un Tarea-Nodo (Task-Node), es una agrupación de tareas (Task) y en este trabajo a estas agrupaciones se les llama Actividades. Cuando la Tarea (notación EPF) se refiere a “Enviar artefacto” o “Actualizar y enviar artefacto”, la Tarea ocupa dos Task-Node (notación JPDL), uno para contener la task que indica “Subir” el artefacto y otro que agrupa los task para “Bajar” el artefacto.
Paso	Tarea (Task)	En notación EPF, Paso es la unidad mínima de trabajo en JPDL es una tarea (Task).

Rol	Asignación (Assignment)	En el modelado de un proceso puede existir la posibilidad de responsabilizar a un rol específico por una tarea, en JPDL se modela como la asignación del rol a una tarea.
Artefacto	Variable	En notación EPF se modela el nombre del artefacto y es posible bajarlo de la página Web. En JPDL no es posible bajar el artefacto por lo que el elemento "variable" de JPDL, sólo guarda una referencia al artefacto. Se hizo una programación adicional para que la Suite BPM construida tuviera la funcionalidad de subir y bajar artefactos.

En EPF, hay aspectos de los procesos que se representan de manera textual dentro de las descripciones de los pasos, para modelarlos con la Suite BPM se utilizan elementos de JPDL o se modelan como descripciones de las tareas en un archivo independiente. La Tabla 29 muestra estos aspectos.

Tabla 29: Modelado con la Suite BPM de aspectos textuales de los procesos modelados con EPF

Aspecto de los procesos modelados de manera textual en EPF	**Forma de modelar el aspecto con la Suite BPM.**
Paso opcional	Se realizó presentando las opciones del paso como parte de la descripción del Task (JPDL), dejando al participante elegir la opción, es decir, el sistema no se retroalimenta con la opción que se elige, porque el sistema sólo monitorea el tiempo que le toma hacer el paso (Task en JPDL) al participante, se haya elegido una opción u otra. Las descripciones de los Task se modelan en un archivo independiente.
Pasos iterativos	Se realizó presentando el carácter iterativo del paso como parte de la descripción de la Task (JPDL). El sistema sólo monitorea el tiempo que toma hacer todas las iteraciones del paso (Task en JPDL). Las descripciones de los Task se modelan en un archivo independiente.
Tareas iterativas	Para modelar la iteración de la tarea (Actividad o Task-Node en JPDL) se introduce un Task-Node llamado "Preguntar otra iteración", que incluye una tarea para solicitar al Participante del Proceso una respuesta afirmativa o negativa (si o no). También se introduce un nodo de Decision, que procesa la respuesta y decide si regresar al mismo Task-Node o seguir adelante con el siguiente Task-Node (ver Figura 31).

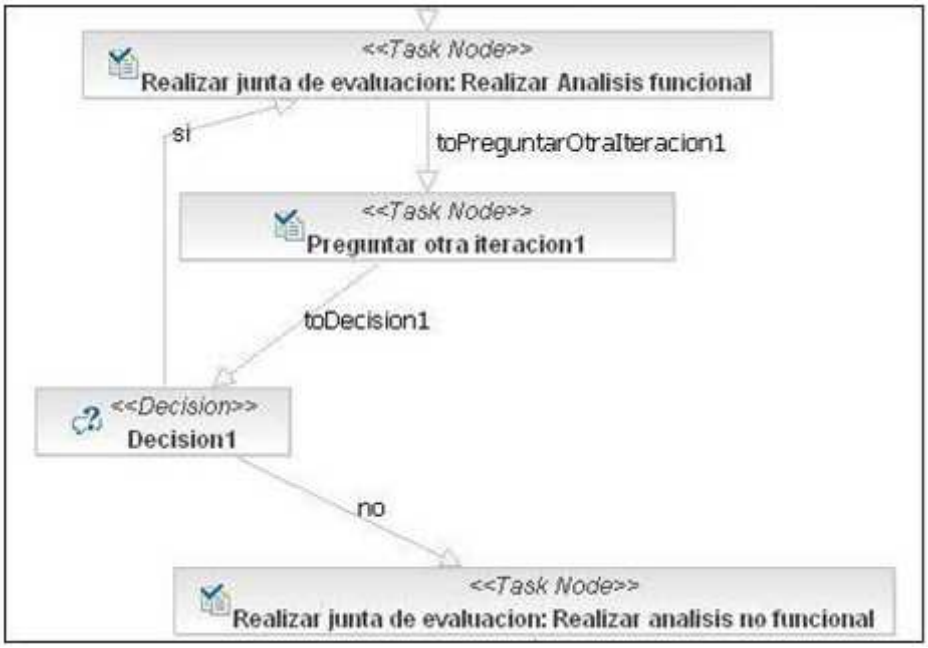


Figura 31: Nodos que modelan las Tareas Iterativas

4.3 Síntesis del capítulo

En este capítulo se presentaron los elementos de JPDL que provee la Suite BPM y que sirven para modelar los procesos, así como la justificación de por qué sólo se ocuparon algunos de los elementos de JPDL y no todos con los que cuenta.

Los elementos que se ocuparon de JPDL son:

- ProcessState
- Task-node
- Fork
- Join
- Task
- Variable
- Asignación (Assignment)
- Decision

Se presentó en forma detallada las actividades que se siguen para traducir del modelo en el que se encuentre modelado un proceso al modelo con la Suite BPM, llamando a esta traducción Procedimiento de Traducción de un Proceso, estas actividades se resumen en la Figura 32.

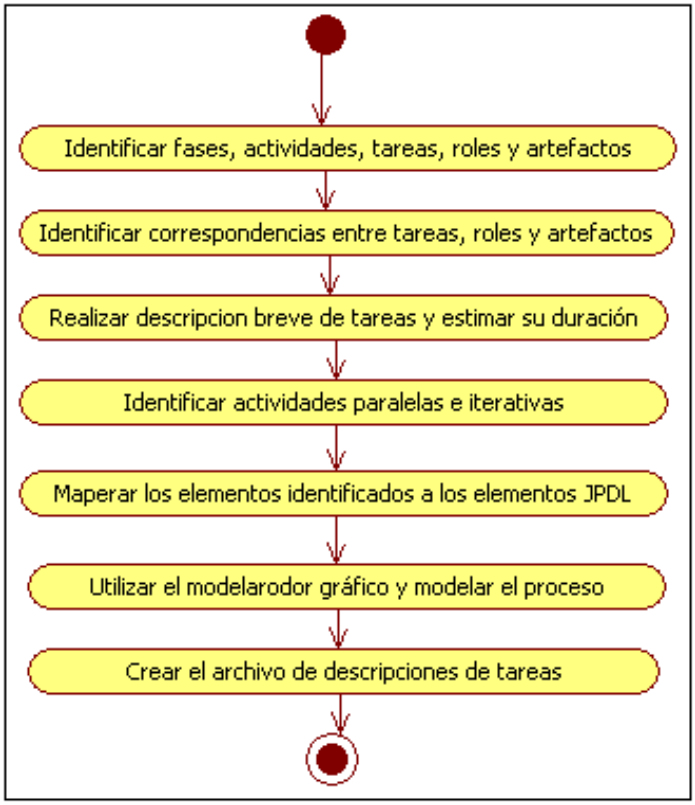


Figura 32: Procedimiento de Traducción de un Proceso, resumido.

Debido a que para evaluar los objetivos de la investigación se modelan procesos que se encuentran modelados con EPF, también se presentó como mapear los elementos de EPF a los elementos de JPDL utilizados.

Capítulo
5 Pruebas (Resultados obtenidos e interpretación)

En este capítulo se presenta la identificación de los procesos seleccionados del modelo Tritón; estos procesos se ocuparon para realizar las pruebas sobre el cumplimiento de los requerimientos. Los resultados de estas pruebas y su interpretación en función de los objetivos del proyecto, se presentan también.

5.1 Selección de los procesos de Tritón

De los 27 procesos del modelo Tritón vistos en la sección 2.2.4, los creadores (CIMAT) han modelado 12 procesos con notación EPF. La Tabla 30 muestra en la columna con encabezado “Procesos” los 12 procesos que se encuentran modelados hasta el momento de realizar este trabajo (el modelo en notación EPF no incluye el número de versión).

Cada uno de los 12 procesos está definido con los mismos elementos de la notación EPF, es decir, todos presentan información respecto a:

- Nombre del proceso
- Información del proceso
- Roles
- Información de los roles
- Artefactos
- Información de artefactos
- Tarea
- Información de tareas
- Paso
- Información de pasos

Al utilizar los mismos elementos para modelar los 12 procesos, no existe una diferencia entre los procesos con respecto a la forma del modelo. Sin embargo,

para llevar a cabo el procedimiento de traducción descrito en el capítulo anterior, se consideró no sólo la forma, es decir, los elementos EPF utilizados, también se consideró el contenido, es decir, la información de las tareas y de los pasos, surgiendo así diferencias entre los procesos que se presentan en la Tabla 30.

Tabla 30: Procesos de Tritón modelados con notación EPF

Procesos	Diferencia						
	No. de Pasos	**No. de Roles que intervienen en los pasos**	**No. De Artefactos producidos en los pasos**	**Tienen Pasos opcionales**	**Tienen Pasos Iterativos**	**Tienen Tareas iterativas**	**Tienen Subprocesos**
Gestión de Procesos	15	2	1	Si	No	No	No
Lanzamiento	29	13	11	No	Si	No	No
Relanzamiento	31	13	14	No	Si	No	No
Postmortem	6	6	4	No	No	No	No
Desarrollo de Software	20	6	5	No	No	No	Si
Requerimientos	27	4	12	Si	No	No	Si
Diseño de Alto Nivel	36	3	12	No	No	No	No
Implementación	21	8	11	No	No	No	Si
Pruebas	77	4	11	Si	No	No	No
Liberación	6	8	3	Si	No	No	No
Aseguramiento de la Calidad del Producto y del Proceso	21	6	4	No	Si	No	No
Administración de la Configuración	21	4	1	No	No	Si	No

El 25% de los procesos de Tritón modelados con notación EPF tiene subprocesos, es decir, dos niveles de jerarquía de procesos y la Suite BPM que se construyó no tiene la capacidad para modelar ese tipo de procesos (ver

capítulo 4). Por lo tanto, la Suite BPM modela el 75% de los procesos de Tritón modelados con notación EPF. Para los procesos con subprocesos se sugiere:

- Modelar el subproceso como una fase del proceso, con la limitante que el subproceso sólo podrá contener actividades, es decir, no es posible modelar fases en el subproceso.

- Modelar el subproceso como un proceso independiente.

Con el análisis de los datos de la Tabla 30 se identificaron cuatro categorías de procesos, estas se muestran en la Tabla 31. Los datos de la columna de Porcentaje de la Tabla 31, se obtienen de calcular el número de procesos de la categoría entre el total de procesos modelados (12). Cabe mencionar que el proceso de "Requerimientos" se encuentra en las categorías: "Tienen pasos opcionales" y en la categoría "Tienen subprocesos", por esta razón, se observa que la suma de los porcentajes no es del 100%.

Tabla 31: Resultados en porcentajes de los datos de los procesos del modelo Tritón.

Categoría	No. De Procesos en esta categoría	Porcentaje
Tienen pasos opcionales	4	33%
Tienen pasos iterativos	3	25%
Tienen tareas iterativas	1	8%
Procesos secuenciales, es decir, que no contienen tareas, pasos iterativos, pasos opcionales o subprocesos.	2	16%
Tienen subprocesos	3	25%

Cada categoría de la Tabla 31 contiene procesos similares no sólo en forma del modelo (elementos EPF utilizados), sino también en el contenido. De cada categoría se seleccionó un proceso que la representara (ver Tabla 32). Los criterios de selección se basaron en elegir los procesos que contuvieran el menor número de pasos y en el menor número de roles, puesto que la Suite BPM cuenta con una Base de Datos embebida de limitada capacidad, exclusiva para prototipos *[25, pág. 149]*.

Tabla 32: Procesos seleccionados del modelo Tritón para ser modelados por la Suite BPM.

	Número de procesos	**Porcentaje que representa del total de procesos modelados**	**Procesos**	**No. de pasos**	**No.de roles**	**Proceso seleccionado**	**Justificación**
Tienen pasos opcionales	4	33%	Gestión de Procesos	15	2	Gestión de Procesos	Proceso con el menor número de pasos y de roles.
			Requerimientos	27	4		
			Pruebas	77	4		
			Liberación	6	8		
Tienen pasos iterativos	3	25%	Lanzamiento	29	13	Aseguramiento de la Calidad del Producto y del Proceso	Proceso con el menor número de pasos y de roles.
			Relanzamiento	31	13		
			Aseguramiento de la Calidad del Producto y del Proceso	21	6		
Tienen tareas iterativas	1	8%	Administración de la Configuración	21	4	Administración de la Configuración	Proceso único.
Procesos secuenciales	2	16%	Postmortem	6	6	Postmortem	Proceso con el menor número de pasos y de roles.
			Diseño de Alto Nivel	36	3		

A continuación se presentan los procesos seleccionados con una breve descripción.

Proceso Postmortem. El propósito del proceso de Postmortem es guiar al equipo en reunir y analizar los datos generados en una iteración o un proyecto con la finalidad de identificar mejoras en el proceso.

Proceso Gestión de Procesos. Es uno de los procesos que tiene pasos opcionales dentro de su única actividad. Este proceso es una guía que contiene los elementos que servirán para la documentación y/o elaboración de los procesos, es decir, su propósito es guiar en la definición y documentación de los procesos.

Aseguramiento de la Calidad del Producto y del Proceso. Es uno de los procesos que tiene pasos iterativos. El principal objetivo de este proceso es encontrar defectos en el producto de trabajo y en el proceso.

Administración de la Configuración. El proceso de Administración de la Configuración, es un proceso que contiene actividades iterativas. El objetivo de este proceso es asegurar la integridad de un producto y hacer su evolución más manejable.

En la siguiente sección se presentan los resultados de las pruebas sobre el cumplimiento de los requerimientos de la Suite BPM.

5.2 Cumplimiento de los requerimientos (Resultados obtenidos)

Los cuatro procesos seleccionados del modelo Tritón, vistos en la sección anterior, se utilizaron para hacer pruebas a la Suite BPM. Las pruebas se realizaron a cada proceso y consistieron en revisar el cumplimiento de los Requerimientos Funcionales enunciados a continuación:

- Actualizar e instalar nueva versión de la definición del proceso.
- Ingresar al sistema por rol.

- Instanciar definición del proceso.
- Iniciar tarea.
- Terminar tarea.
- Monitorear el avance de la instancia del proceso.

A manera de ejemplo, en el Apéndice D, se detallan los *Casos de Prueba Funcional* que se llevaron a cabo para el proceso Postmortem.

En la siguiente sección se presenta el resultado obtenido del cumplimiento del requerimiento “Actualizar e instalar nueva versión de la definición del proceso” para cada proceso; los resultados de los demás Requerimientos Funcionales se presentan en las siguientes secciones con el proceso Postmortem solamente, debido a que no hubo diferencias significativas en los resultados del cumplimiento de estos requerimientos en los demás procesos seleccionados.

El cumplimiento de los Requerimientos No Funcionales se presentó en la sección 3.3.5 con la evaluación de la arquitectura.

5.2.1 Actualizar e instalar nueva versión de la definición del proceso (ID-E8)

Como ya se ha mencionado antes, con el Modelador Gráfico de Procesos se crea el modelo de la definición de un proceso, lo que equivale a crear una versión del modelo. Si se realizan cambios a este modelo se convierte en una versión distinta, es decir, basta con hacer los cambios al modelo con apoyo del Modelador Gráfico de Procesos para tener una versión diferente; para instanciar la nueva versión, se vuelve a iniciar la Suite BPM y la nueva versión queda instalada.

En esta sección se presenta el modelado en JPDL, de los procesos seleccionados del modelo Tritón: Postmortem, Gestión de Procesos, Aseguramiento de la Calidad del Producto y del Proceso y Administración de la Configuración.

5.2.1.1 Traducción y Modelado de Postmortem con la Suite BPM

Como se ha mencionado, el proceso de Postmortem está modelado con la notación EPF, los elementos de esta notación fueron mapeados a los elementos JPDL de acuerdo a la Tabla 28, que se refiere al mapeo de elementos entre la notación EPF a la notación JPDL.

Postmortem cuenta con tres fases:

- ***Preparación.*** Esta fase tiene como propósito guiar al equipo en la preparación de los recursos e infraestructura necesaria para el desarrollo del proceso de Postmortem.

- ***Desarrollo.*** Esta fase tiene como propósito guiar al equipo durante la ejecución del proceso de Postmortem.

- ***Cierre.*** Esta fase tiene como propósito guiar al equipo durante el término de la ejecución del proceso de Postmortem.

Tomando en cuenta el mapeo de la Tabla 28, el modelado en JPDL de las fases del proceso de Postmorten se muestra en la Figura 33.

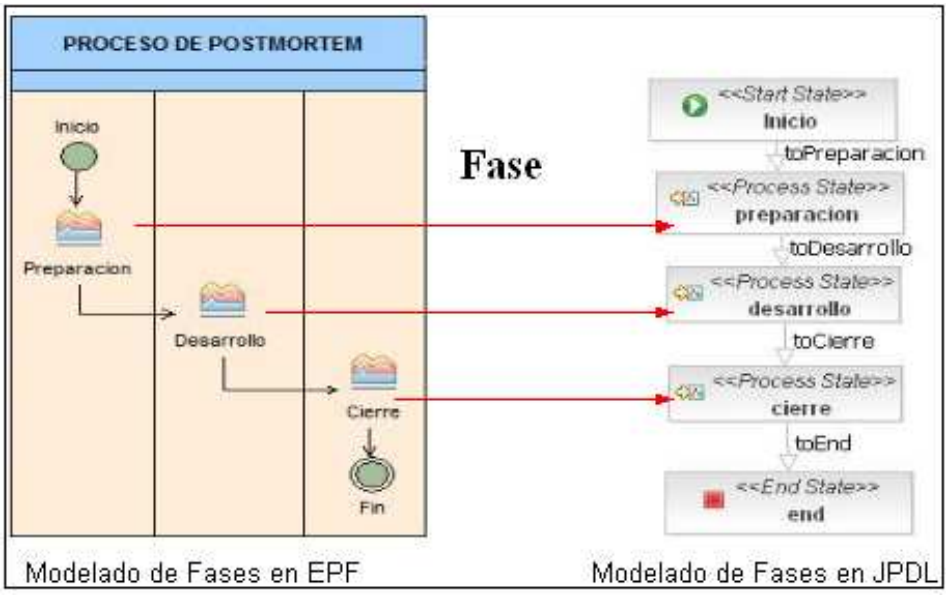


Figura 33: Mapeo de notación EPF a JPDL para fase

Como ya se mencionó, un nodo “Estado del proceso” (ProcessState) (ver Figura 33, lado derecho) tiene una referencia de un proceso descrito por separado, cuando el Motor de Ejecución de Procesos lee este nodo, inicia la lectura del proceso nombrado con el nombre del “ProcessState”. En este caso, el contenido de la Fase Preparación se modela como un proceso llamado Preparación que se muestra en la Figura 34 del lado derecho. El contenido de la Fase Preparación surge del mapeo entre las Tareas (notación EPF), que agrupan pasos, y los Tarea-Nodo (Task-Node), que agrupan tareas. Como se mencionó en la Tabla 28, en este trabajo las Tareas (notación EPF) y las Task-Node (notación JPDL) modelan a las actividades.

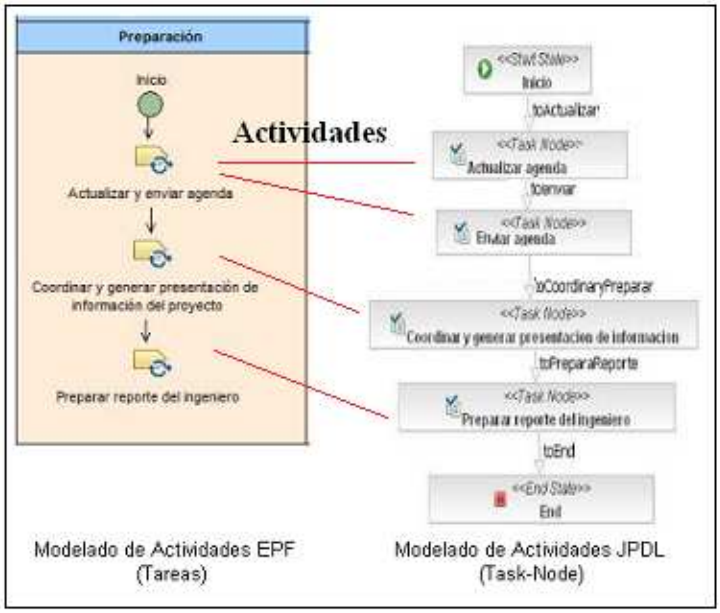


Figura 34: Mapeo de notación EPF a JPDL para actividad

En el modelado del proceso de Postmortem en notación EPF, los Pasos de la Tarea “Actualizar y enviar agenda” de la Figura 34 (lado izquierdo), se muestra en la Figura 35. Los Pasos (Steps), Rol y Artefacto se mapean según la Tabla 28 a los elementos Tarea, Asignación y Variable respectivamente, y que no tienen una representación gráfica, como se explicó en la sección 3.3.3.5. Por lo que en la Figura 36 se presenta el correspondiente mapeo en XML.

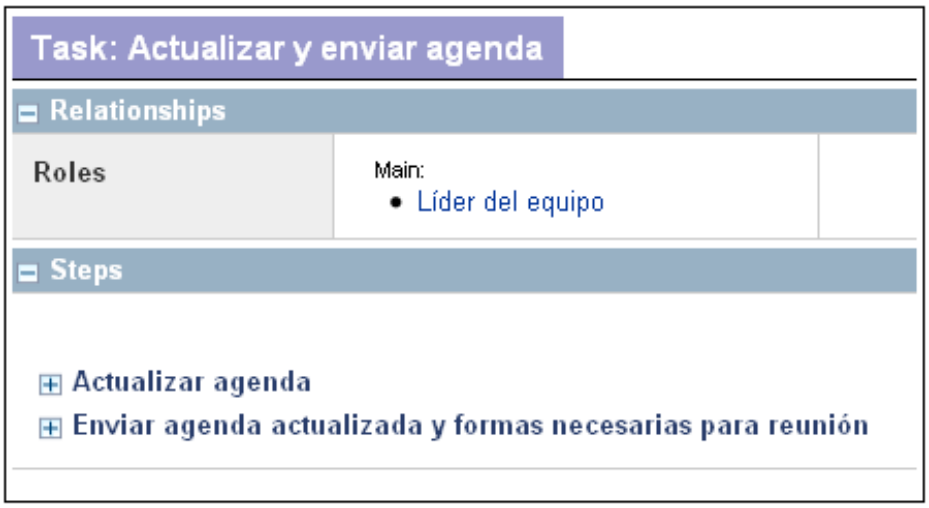


Figura 35: Pasos y rol de la tarea "Actualizar y enviar agenda".

```
<task-node name="Actualizar Agenda">
    <task name="A.Actualizar Agenda de postmortem">  <-- a)

        <assignment actor-id="Lider"></assignment>  <-- b)
        <controller>
c) -->  <variable access="read,write" name="PTLL_AgendaPM_Triton_DAS_v1.0.doc"></variable>
        </controller>
    </task>
    <transition to="Recibir Agenda postmortem" name="toRecibirAgendaPostmortem"></transition>
</task-node>

<task-node name="Recibir Agenda">
    <task name="B.Recibir Agenda postmortem">

        <assignment actor-id="Administrador de soporte"></assignment>
        <controller>
            <variable access="read,write" name="PTLL_AgendaPM_Triton_DAS_v1.0.doc"></variable>
            <variable access="read,write" name="Formas necesarias.doc"></variable>
        </controller>
    </task>
    <task name="B.Recibir Agenda postmortem">

        <assignment actor-id="Administrador de planeacion"></assignment>
        <controller>
            <variable access="read,write" name="PTLL_AgendaPM_Triton_DAS_v1.0.doc"></variable>
```

Figura 36: Tarea (a), asignación (b) y variable (c).

Para modelar la Tarea (notación EPF), "Actualizar y enviar agenda", se llevo a cabo lo que indica la Tabla 28 usando dos Tarea-Nodo (Task-Node). El primer Task-Node llamado "Actualizar Agenda" contiene una sola task: "A.Actualizar agenda de postmortem" (ver Figura 36 inciso a), para modelar el paso "Actualizar agenda" (notación EPF) y otro Task-Node llamado "Recibir Agenda" que contiene los task "B.Recibir Agenda postmortem" para modelar el Paso "Enviar agenda actualizada y formas necesarias para reunión", cada task es asignada a cada rol que el modelo en notación EPF indica tiene que recibir la agenda, en este caso, Administrador de soporte y Administrador de planeación.

El primer Task-Node tiene un task asignado a “el Líder” (ver Figura 36 inciso b) y una variable con valor PTLL_AgendaPM_Triton_DAS_v1.0.doc[2] (ver Figura 36 inciso c), este valor es el nombre del artefacto que tiene que subir el Líder durante la instancia de esta tarea, hasta que termine la tarea y le indique al sistema que terminó, puede pasar al siguiente Task-Node, que agrupa varias Task, con el nombre de “B.Recibir Agenda postmortem” (ver Figura 36).

5.2.1.2 Traducción y Modelado del Proceso de Gestión de Procesos con la Suite BPM

En la Figura 37 inciso a, se muestra el modelado en EPF de la fase del proceso de gestión de procesos llamada “Administración de procesos”, esta fase tiene una única Actividad que es “Definición de procesos” (Figura 37 inciso b), y esta a su vez tiene los pasos (Steps) que se muestran en la Figura 39, entre estos pasos se presenta el paso opcional, que consiste en que el paso presenta dos o más opciones de cómo llevarse a cabo. La fase y la actividad de la Figura 37 se modelaron con JPDL, como se mostró para el proceso de Postmortem y el resultado se muestra en la Figura 38.

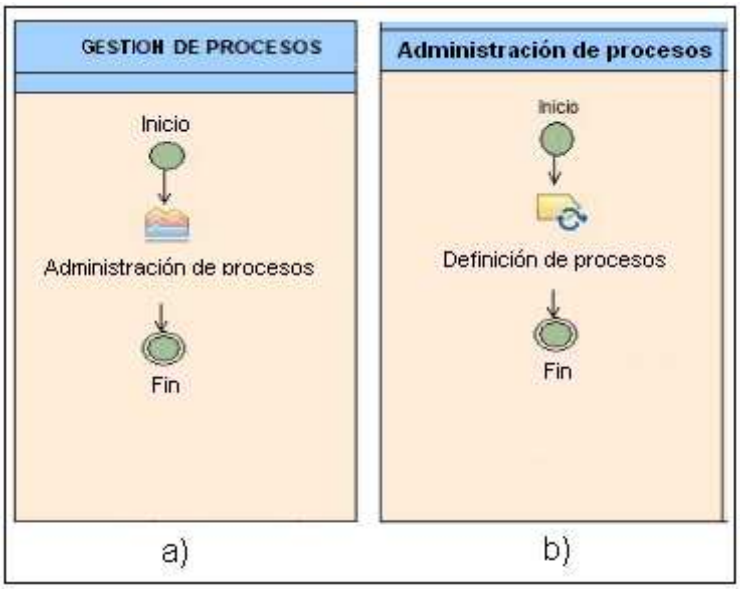


Figura 37: Fase, Administración de procesos (a) Actividad, Definición de procesos (b).

[2] Nota: Esta es la convención de nombrado elegida por los autores de Tritón.

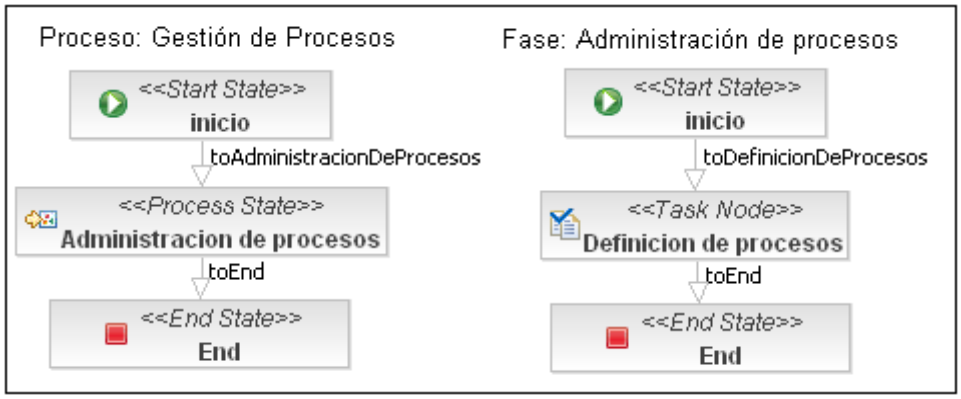


Figura 38: Modelado con JPDL del proceso Gestión de Procesos y su fase.

La Tabla 29 de la sección 4.2, indica que modelar un paso opcional como lo es: "Investigar detalle del proceso y sus pasos", que se muestra en la Figura 39, se realizó presentando las opciones del Paso como parte de la descripción de la task (JPDL) en un archivo independiente, donde además se escribe el nombre de la tarea, el rol y el tiempo estimado de realización, como se muestra en la Figura 40.

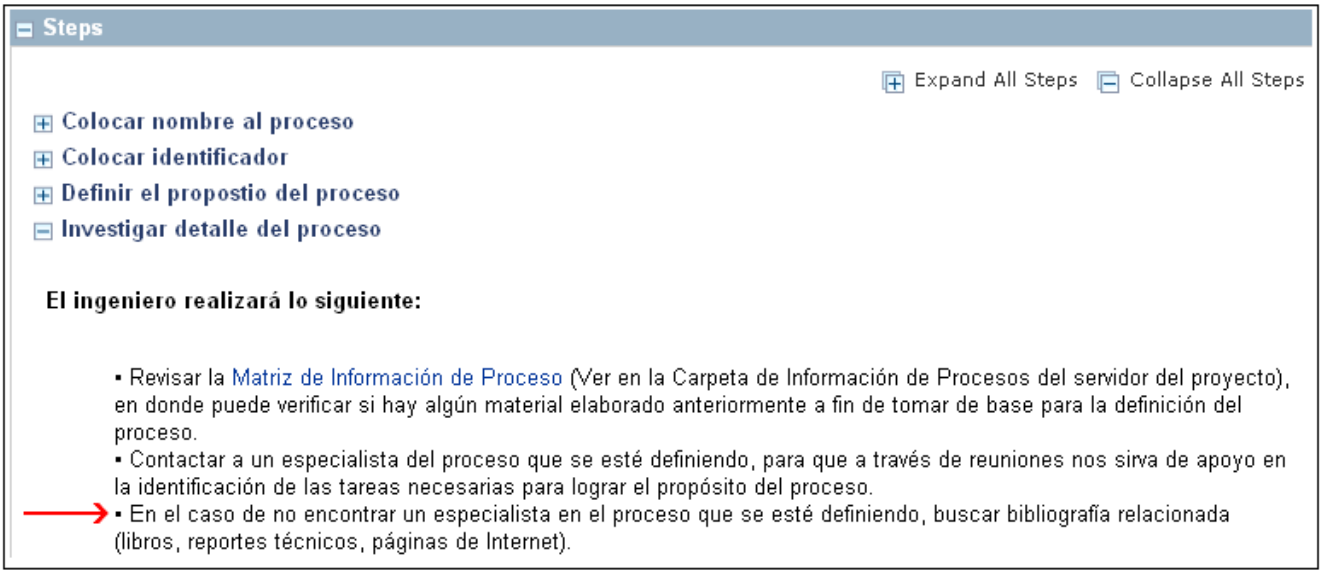


Figura 39: Paso con opciones (notación EPF)

O.Investigar detalle del proceso.
Ingeniero.
- Revisar la Matriz de información de proceso (Ver en la Carpeta de información de procesos del servidor del proyecto), en donde puede verificar si hay algún material elaborado anteriormente a fin de tomar de base para la definición del proceso.
- Contactar a un especialista del proceso que se esté definiendo, para que a través de reuniones nos sirva de apoyo en la identificación de las tareas necesarias para lograr el propósito del proceso.
- En el caso de no encontrar un especialista en el proceso que se esté definiendo, buscar bibliografía relacionada (Libros, reportes técnicos, páginas de Internet).

60

Figura 40: Descripción del task "Investigar detalle del proceso".

5.2.1.3 Traducción y Modelado del Proceso de Aseguramiento de la Calidad del Producto y del Proceso con la Suite BPM

En la Figura 41 inciso a, se muestra el modelado en EPF de la fases del proceso de Aseguramiento de la Calidad del Producto y del Proceso, entre las cuales, la fase llamada "Control calidad del producto", contiene la actividad única que es "Seguimiento al producto de trabajo" (Figura 41 inciso b), y esta a su vez tiene los pasos (Steps) que se muestran en la Figura 43, entre estos pasos se presenta el paso iterativo, que consiste en repetir el paso hasta un criterio de terminación. Las fases y la actividad de la Figura 41, se modelaron con JPDL, como se mostró para el proceso de Postmortem y el resultado se muestra en la Figura 42.

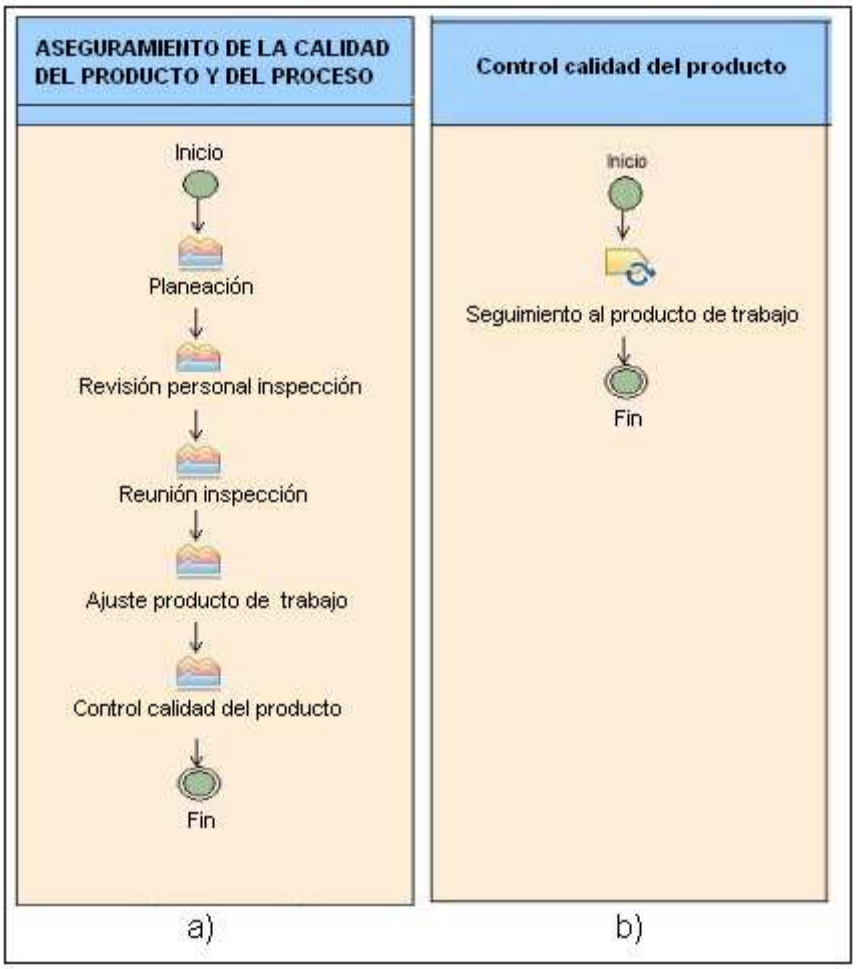


Figura 41: Fase, Control calidad del producto (a) Actividad, Seguimiento al producto de trabajo (b).

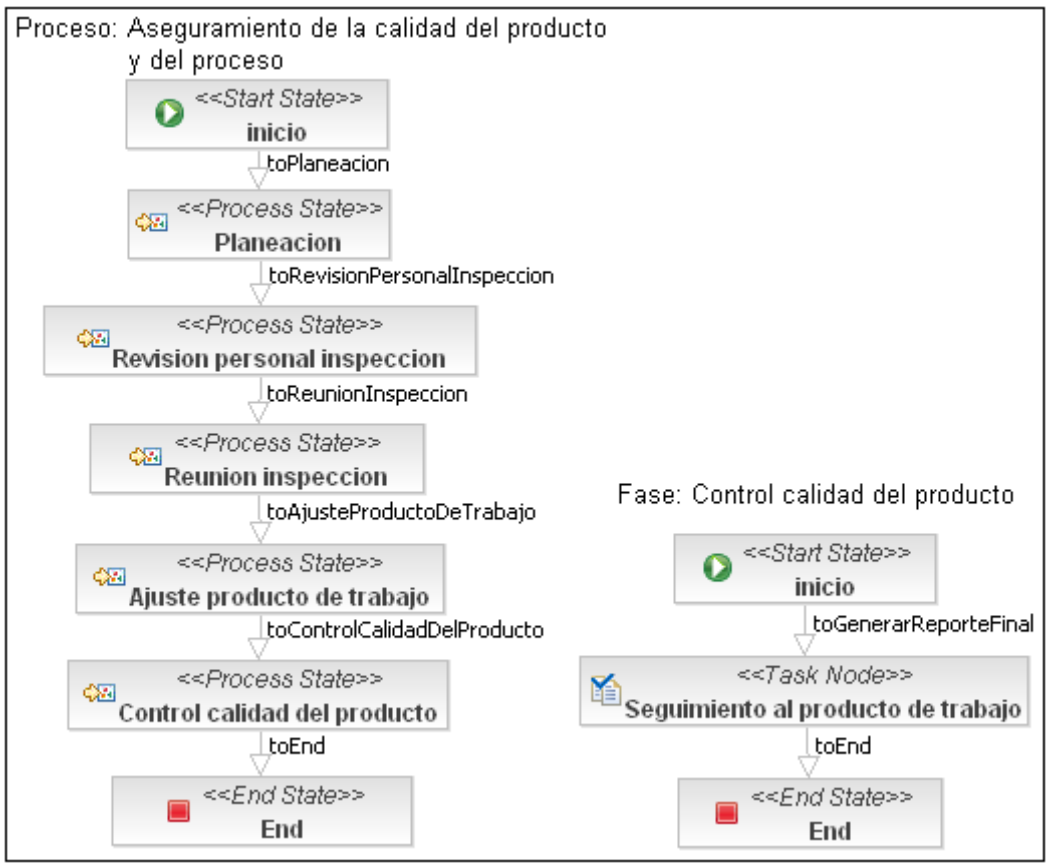


Figura 42: Modelado con JPDL del proceso Aseguramiento de la Calidad del Producto y del Proceso y su fase.

El modelado del paso iterativo, que en este caso es: "El moderador verifica las correcciones del autor" (ver Figura 43) se realizó presentando el carácter iterativo del paso como parte de la descripción del paso (o Task en JPDL), esta descripción como ya se ha mencionado antes, se almacena en un archivo independiente, como lo muestra la Figura 44.

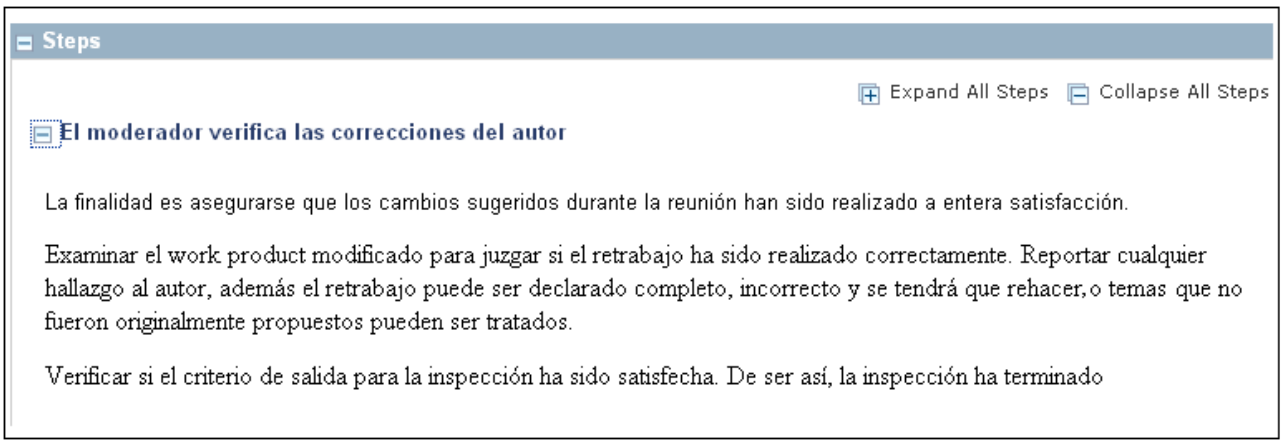


Figura 43: Paso con iteración (notación EPF)

O.El moredador verifica las correcciones del autor
Moderador
La finalidad es asegurarse que los cambios sugeridos durante la reunión han sido realizados a entera satisfacción.
Examinar el work product modificado para juzgar si el retrabajo ha sido realizado correctamente. Reportar cualquier hallazgo al autor, además el retrabajo puede ser declarado completo, incorrecto y se tendrá que rehacer, o temas que no fueron originalmente propuestos pueden ser tratados.
Verificar si el criterio de salida para la inspección ha sido satisfecha. De ser así, la inspección ha terminado.
120

Figura 44: Descripción del task "El moderador verifica las correcciones del autor".

5.2.1.4 Traducción y Modelado del Proceso Administración de la Configuración con la Suite BPM

En la Figura 45 inciso a, se muestra el modelado en EPF de la fases del proceso de Administración de la Configuración, entre las cuales, la fase llamada "Auditoría de la configuración", contiene la actividad única que es "Auditoría de la configuración" (Figura 45 inciso b) y es una actividad iterativa.

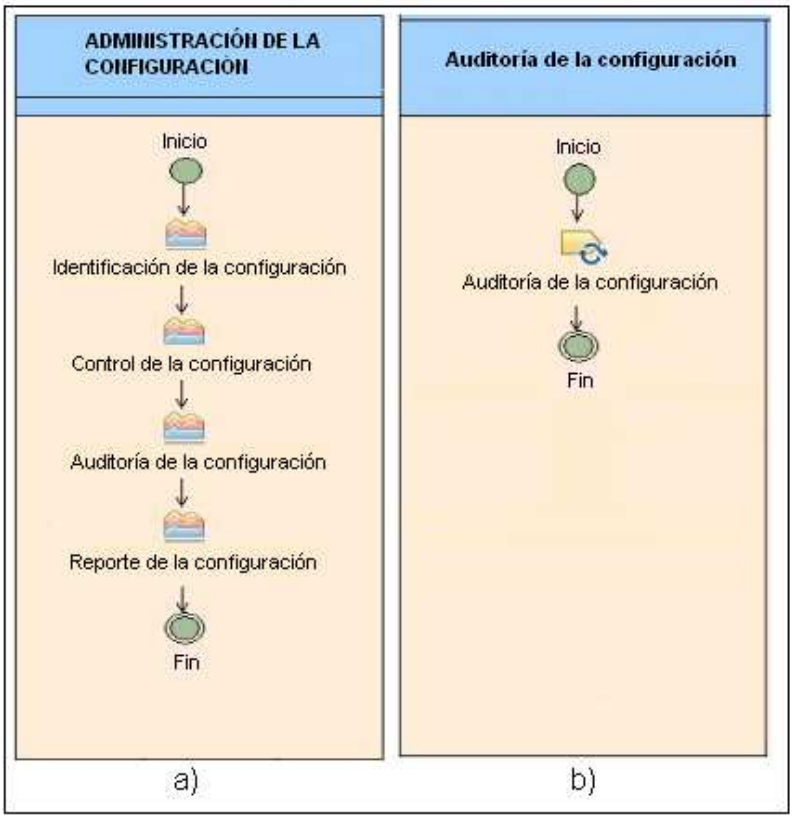


Figura 45: Fase, Auditoría de la configuración (a) Actividad, Auditoría de la configuración (b)

Las fases de la Figura 45 se modelaron con JPDL, como se mostró para el proceso de Postmortem y el resultado se muestra en la Figura 46.

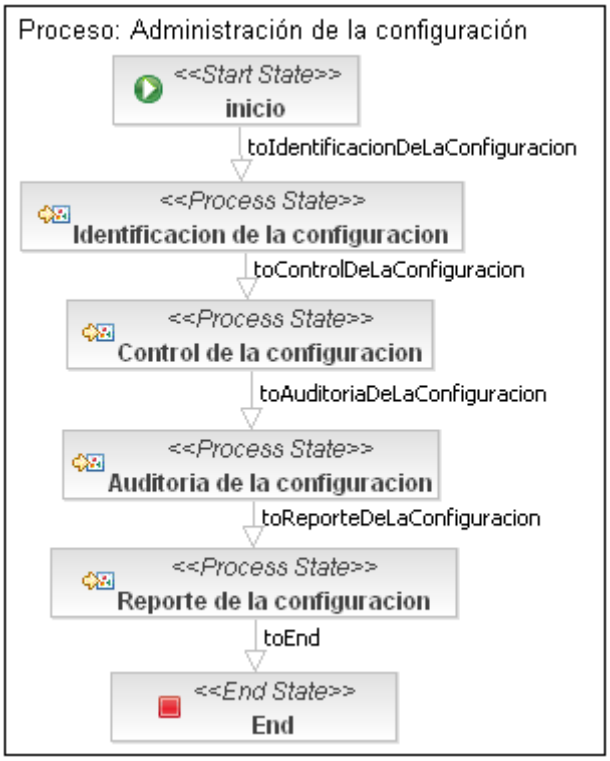


Figura 46: Modelado con JPDL del proceso Administración de la Configuración y su fase.

El modelado de la Tarea Iterativa (notación EPF), llamada en este trabajo Actividad, que en este caso es: "Auditoría de la configuración" (ver Figura 47) se realizó ocupando la Tabla 29 que indica ocupar un Task-Node llamado "Preguntar otra iteración", que incluye una tarea para solicitar al Participante del Proceso una respuesta afirmativa o negativa y el nodo Decision presentado en la sección 3.3.3.5. (ver Figura 48).

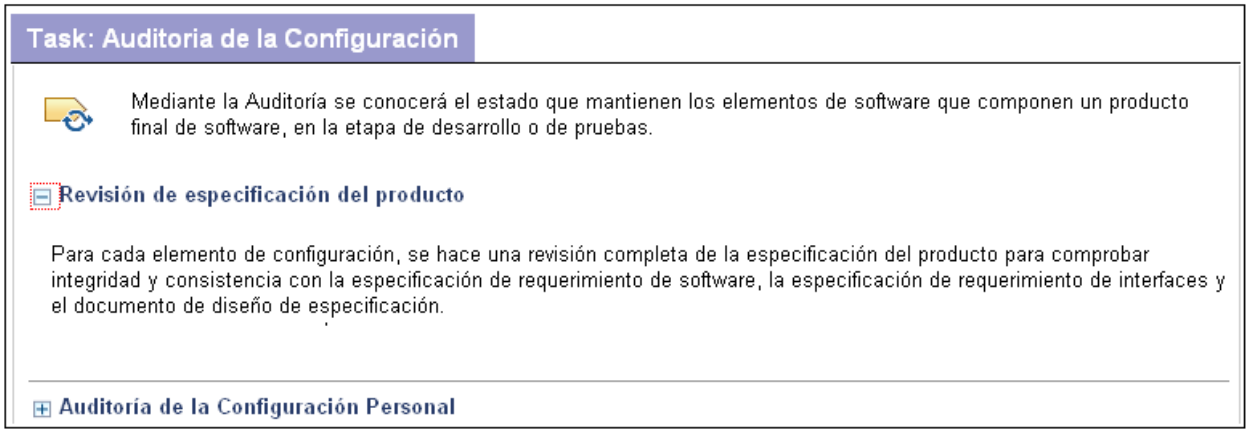


Figura 47: Tarea Iterativa (notación EPF)

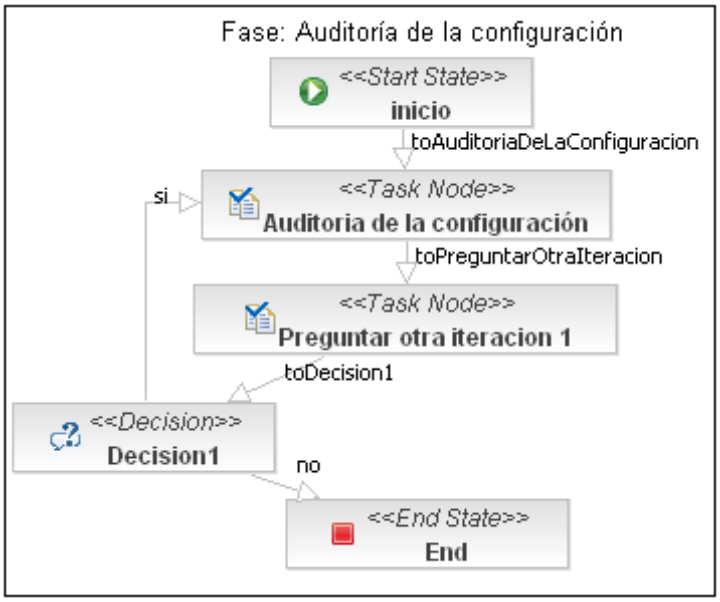


Figura 48: Modelado de la iteración de una Actividad.

5.2.1.5 Análisis cualitativo del modelado en notación JPDL a partir de la notación EPF.

El análisis cualitativo de esta sección es en base a los resultados del modelado de los procesos seleccionados; y consistió en determinar cuáles de los elementos de la notación EPF fueron posibles modelar con la Suite BPM y cuáles no. La Tabla 33 muestra los resultados obtenidos.

Tabla 33: Elementos de la notación EPF que se modelaron con la Suite BPM

Elemento notación EPF	Modelado
Nombre del proceso	Si
Información del proceso	No
Nombre del rol	Si
Información del rol	No
Nombre del artefacto	Si
Información de artefactos	No
Nombre de la tarea	Si
Información de la tarea	No

Nombre del Paso	Si
Información del paso	Si

Cómo se observa en la Tabla 33, de los 10 elementos de la notación EPF, 6 fueron modelados con la Suite BPM. Los 4 elementos restantes no se modelaron porque la creación de la Suite BPM fue acotada a realizarle adaptaciones que permiten el modelado de la descripción del paso (ver sección 3.4.3); no la información del proceso, del rol, de los artefactos y de las tareas (notación EPF). La adaptación para modelar la descripción del paso se llevó a cabo para dar cumplimiento al objetivo 4 de este trabajo (ver sección 1.4).

Los renglones de “Tarea” y “Paso” se examinaron por separado con mayor detalle y los resultados se presentan en la Tabla 34. La mayor cantidad de respuestas afirmativas en esta Tabla 34, corrobora la respuesta “Si” anotada en la Tabla 33 en los renglones de Tarea y Paso. Los aspectos presentados en la Tabla 34 (columna: la Suite BPM permite modelar) se explicaron anteriormente en esta sección 5.2.1, excepto el aspecto de “Ejecución en paralelo” que se presentó en la Tabla 27 de la sección 4.1 del capítulo anterior.

Tabla 34: Aspectos modelados de los elementos de EPF de Tarea y Paso

La Suite BPM permite modelar.	**Tarea (notación EPF)**	**Paso (notación EPF)**
Ejecución	Si	Si
Ejecución en paralelo	Si	Si
Elegir de manera opcional	No	Si
Iterar actividades	Si	Si

En las siguientes secciones se presentan los resultados obtenidos del cumplimiento de los Requerimientos Funcionales restantes.

5.2.2 Ingresar al sistema por rol (ID-A2)

Como se observa en la Figura 49 se ingresa al sistema (Suite BPM) por rol, esto da cumplimiento al Atributo de Calidad RNF-03 de la categoría de Seguridad visto en la Tabla 8, de la sección 3.2.1.2, página 42. El cumplimiento de este atributo consiste en que cada participante ingresa por rol y contraseña, y sólo tiene acceso a las tareas que le corresponden.

Figura 49: Página Web correspondiente a ingresar al sistema por rol.

5.2.3 Instanciar definición del proceso (ID-E1)

Se dio cumplimiento al requerimiento que indica que sólo a un rol que se encuentre en la categoría de Administrador del Proceso puede crear instancias del proceso, en este ejemplo es el Líder, sólo a él se le presenta la liga de "Nuevo" en la página Web de instancias del proceso, como lo ilustra la Figura 50.

Figura 50: Página Web correspondiente a la creación de una instancia.

En la Figura 50, también se muestra el encabezado de la *Tabla de procesos* que se va llenando con los datos de las instancias de los procesos que se van creando.

5.2.4 Iniciar tarea (ID-A3)

A cada rol se le presenta una *Tabla de tareas*, que contiene las tareas que le corresponden al rol. Al iniciar una tarea, el sistema llena las columnas de "Descripción" de la tarea e "Inicio", donde registra la fecha y hora en la que se inicia la tarea (ver Figura 51).

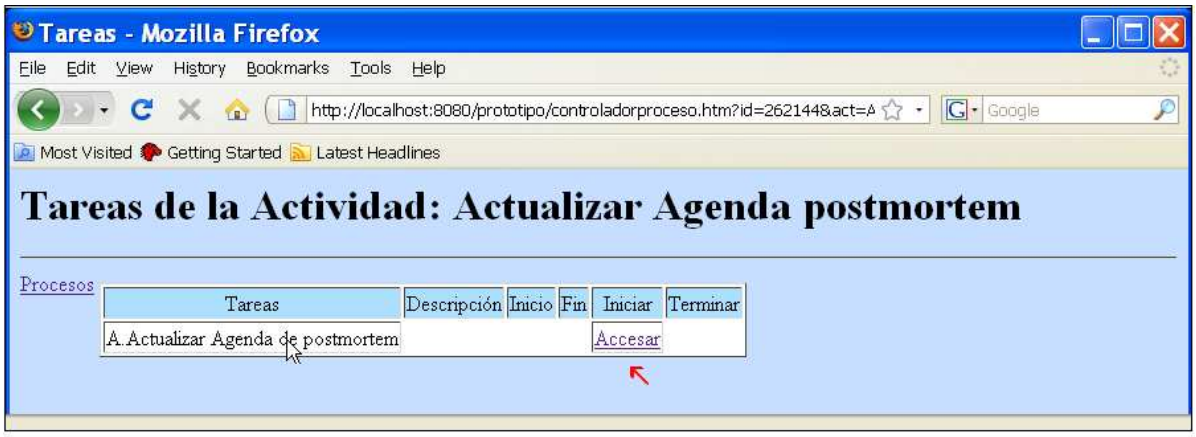


Figura 51: Página Web con la *Tabla de tareas* y la opción para Iniciar/Accesar.

Como se mencionó en la sección 3.4.3, el agregar la descripción a las tareas es una adaptación que se hizo a la Suite BPM, en la siguiente sección se detalla esta adaptación desde el punto de vista funcional.

5.2.4.1 Descripción de tareas.

El DiseñadorDeProcesos, que es un componente de la Suite Jboss JBPM que se utilizó en la Suite BPM (ver sección 3.3.4.1), no permite modelar las descripciones de las tareas, sólo el nombre de la tarea o instrucciones cortas. En la Figura 52 se muestra un ejemplo de una tarea en la Suite Jboss JBPM. La tarea, “Solicitar petición”, tiene dos variables: fecha de inicio y duración, los recuadros son los campos donde el Participante del Proceso ingresa la información. Inclusive, estos recuadros pueden servir para que el Participante del Proceso agregue breves descripciones de las tareas.

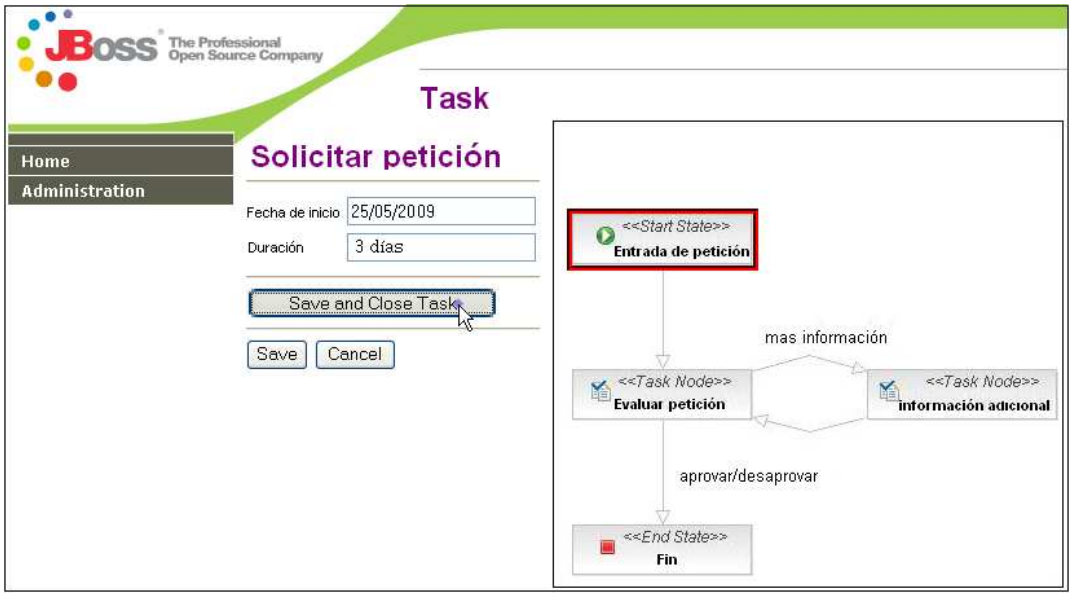


Figura 52: Ejemplo de una tarea en la Suite Jboss JBPM.

La adaptación de la Suite BPM consistió en que no fuera el Participante del Proceso el que agregará la descripción de la tarea, sino la Suite BPM la que presentará la descripción de cada tarea, como se muestra en la columna de “Descripción” de la *Tabla de tareas* de la Figura 54 de la sección 5.2.6. El

componente del DiseñadorDeProcesos no se modificó, las modificaciones fueron en el componente AdmonDeArchivos (ver Tabla 15 de la sección 3.3.4.1 página 61), encargado de leer las descripciones de un archivo, y en la capa de presentación, donde se crearon las páginas Web que de forma dinámica presenta las descripciones de las tareas, ver sección 3.4.3 de la implementación de la Suite BPM.

5.2.5 Bajar y subir artefactos (ID-A5, ID-A6) y control de versiones

Al iniciar una tarea, también se ingresa a la página Web que contiene la *Tabla de artefactos* (ver Figura 53), de la cual es posible bajar y/o subir los artefactos que corresponden a cada tarea, es decir, que de manera dinámica dependiendo de la tarea, la *Tabla de artefactos* se llena con los nombres de los artefactos que intervienen en esa tarea. Si se trata de un artefacto que es una entrada y salida de la tarea, como lo es en este ejemplo donde la tarea tiene como entrada una plantilla y como salida la plantilla completada, aparecen ambas ligas en la *Tabla de artefactos*, para subir y bajar. En otros casos, es posible que sólo aparezca la liga de subir, o sólo la de bajar según sea el artefacto una salida o una entrada.

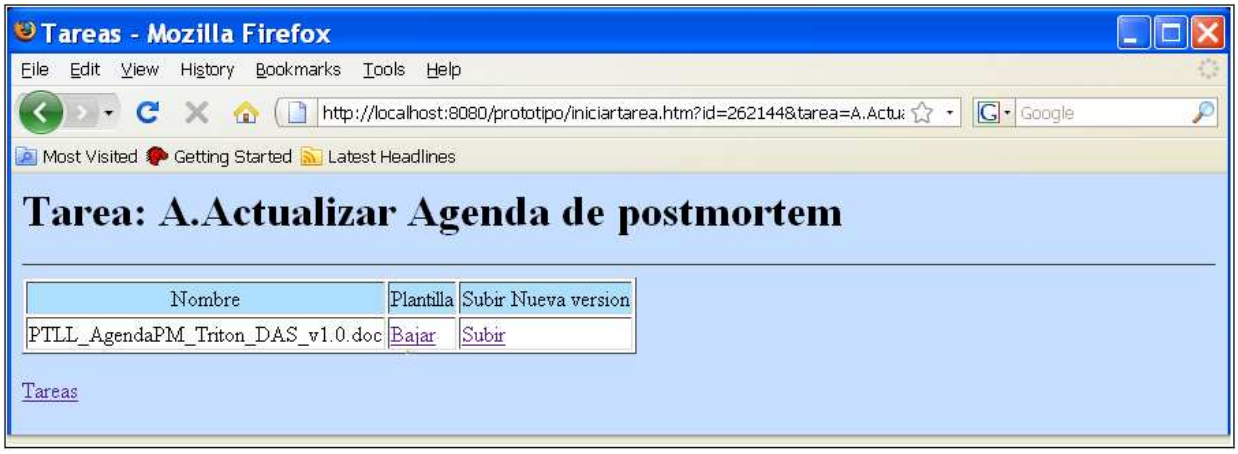


Figura 53: Página Web con la *Tabla de artefactos* y las opciones de Bajar y Subir.

Cabe mencionar que mientras no se termine la tarea es posible subir distintas versiones del mismo artefacto. Las versiones serán administradas por un sistema de control de versiones.

Cabe señalar que subir y bajar artefactos es la primera adaptación que se hizo a la Suite BPM y controlar sus versiones es la segunda. Son adaptaciones porque las Suites BPM estudiadas no cuentan con estas funcionalidades. Estas adaptaciones se realizaron por las razones explicadas en la sección 3.1.

5.2.6 Terminar tarea (ID-A4)

Al terminar la tarea, se registra la fecha y hora en la que termina la tarea, con esta acción, la tarea ya no se desplegará en la *Tabla de tareas*, ésta se llenará con las tareas subsecuentes que le correspondan al rol (ver Figura 54). La fecha y hora de terminación de la tarea al igual que cuando inicia son los datos de las métricas que son conocidos por el DespachadorDeEventos (sujeto) y lo comunica al ObservadorRegistroTiempo (observador).

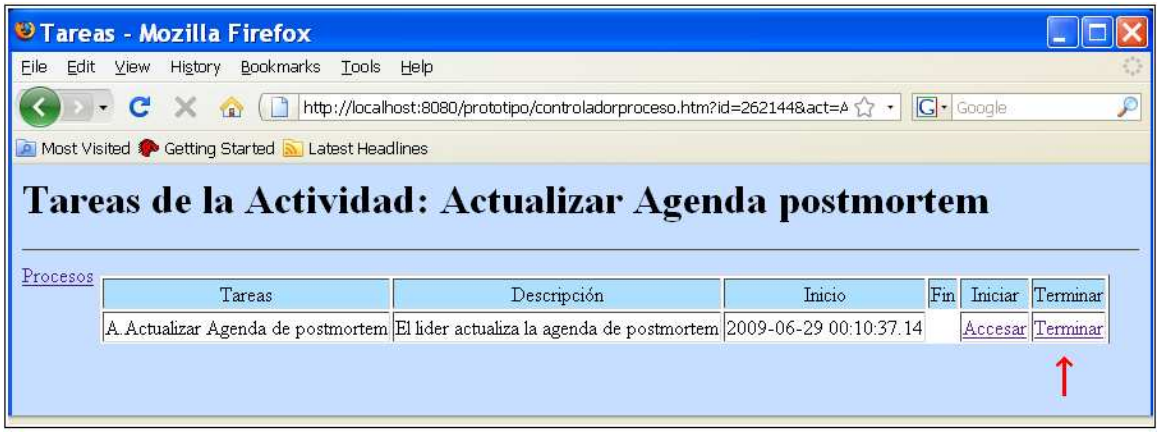


Figura 54: Página Web con la *Tabla de tareas* y la opción para terminar.

5.2.7 Monitorear el avance de la instancia del proceso (ID-B7)

En la Figura 55 se muestra la página Web que contiene la *Tabla de monitoreo*, esta tabla presenta cada tarea iniciada y concluida, con sus respectivas fechas

de inicio y terminación, el rol responsable y la duración de estas, con sus diferencias en tiempo con respecto al tiempo estimado. Esta *Tabla de monitoreo* puede ser vista por todos los roles autorizados a entrar al sistema, para conocer en qué fase y tarea va la Instancia del Proceso y toda la información que la tabla proporciona.

Figura 55: Página Web de monitoreo de avance.

La información que se muestra en la *Tabla de monitoreo*, se almacena en una base de datos independiente a la que contiene la Definición del Proceso.

El resumen de los resultados del cumplimiento de los Requerimientos Funcionales se muestra en la Tabla 35; la forma en que se comprobó este cumplimiento fue a través de los C*asos de Prueba Funcionales* (ver Apéndice D).

Tabla 35: Resumen del cumplimiento de los Requerimientos Funcionales.

ID	Requerimiento Funcional	Cumplido	Resultado (sección)
ID-E8	Actualizar e instalar nueva versión de la definición del proceso	Si	5.2.1
ID-A2	Ingresar al sistema por rol	Si	5.2.2
ID-E1	Instanciar definición del proceso	Si	5.2.3
ID-A3	Iniciar tarea	Si	5.2.4
ID-A5	Bajar artefactos	Si	5.2.5
ID-A6	Subir artefactos	Si	5.2.5
ID-A4	Terminar tarea	Si	5.2.6
ID-B7	Monitorear el avance de la instancia del proceso	Si	5.2.7

5.3 Evaluación de los objetivos de la investigación a través del funcionamiento de la Suite BPM

En esta sección se presenta la evaluación de los objetivos de la investigación; el resultado de esta evaluación busca validar la hipótesis de este trabajo:

> "Una Suite BPM puede ser adaptada para soportar el ciclo de vida de los procesos de un Modelo de Mejora de Procesos en las etapas de Modelado, Ejecución y Monitoreo."

La Tabla 36 muestra los objetivos y sus respectivos mecanismos de evaluación. En esta tabla se observa, que la evaluación del objetivo 3 es de carácter arquitectural, por esta razón, se considera que el resultado de la evaluación se presentó en la sección 3.3.5. La evaluación de los objetivos 1, 2 y 4 se enfocan al funcionamiento de la Suite BPM.

Tabla 36: Objetivos de la investigación con su respectiva evaluación

No. Objetivo	Descripción	Evaluación
1	Soportar dentro de la Suite BPM el modelado de un número variable de procesos de un Modelo de Mejora de Procesos.	La evaluación del alcance de este objetivo se realiza mediante el modelado de procesos seleccionados de un Modelo de Mejora de Procesos. Los procesos se seleccionaron porque incorporan elementos que se encuentran en todos los demás procesos estudiados.
2	Soportar la etapa de Ejecución de procesos operativos, en particular en lo que se refiere al intercambio de artefactos y el control de versiones de los mismos.	La evaluación del alcance de este objetivo se realiza mediante un estudio que permite verificar que la Suite BPM satisface los requerimientos de los procesos operativos de los Modelos de Mejora de Procesos (sección 2.2.6), en particular el intercambio y el control de versiones de artefactos.
3	Soportar mecanismos flexibles de colecta de métricas que permitan el monitoreo durante la etapa de ejecución del proceso.	La evaluación del alcance de este objetivo se realiza mediante una revisión del Diseño arquitectural que demuestre que se han considerado mecanismos que permitan una adición flexible de componentes para la recolecta de métricas. Esto se presentó en la sección 3.3.5.2.2.
4	Proveer información a los participantes del proceso sobre la etapa en la cual se encuentra la Instancia del Proceso durante la etapa de Ejecución del proceso.	La evaluación del alcance de este objetivo se realiza mediante un estudio que permite verificar que la Suite BPM incluye la funcionalidad de proveer a los participantes del proceso la información sobre la fase y tarea en la cual se encuentra la Instancia del Proceso.

Para llevar a cabo la evaluación de los objetivos 1, 2 y 4, mencionados en la Tabla 36, se consideró el modelo Tritón, que como ya se dijo en la sección 2.2.4, en comparación con los Modelos de Mejora de Procesos CMMI y Moprosoft, Tritón provee más detalle sobre la manera de realizar los procesos, siendo esta la razón por la que fue seleccionado este modelo.

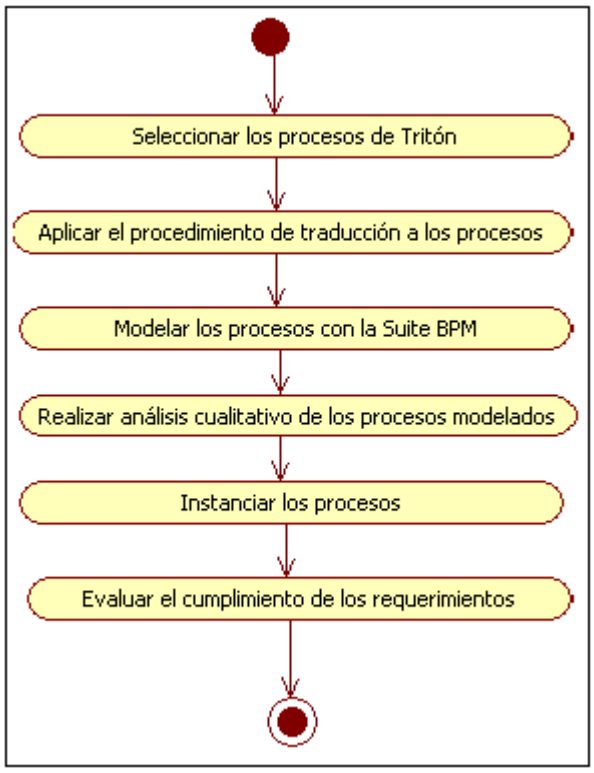


Figura 56: Metodología de la evaluación de los objetivos 1, 2 y 4 de la investigación

Como parte de la metodología que se siguió (ver Figura 56), antes de realizar la evaluación de los objetivos de la investigación, se seleccionaron los procesos de Tritón. A estos procesos se les aplicó el Procedimiento de Traducción de un Proceso y se modelaron con la Suite BPM.

Como parte de la evaluación del objetivo 1, se hizo un análisis cualitativo para determinar que tanto fue posible modelar los procesos con respecto a su modelo en notación EPF (ver sección 5.2.1.5). También siguiendo la metodología se instanciaron los procesos seleccionados, se llevaron a cabo las pruebas del cumplimiento de Requerimientos Funcionales (ver Apéndice D) con la finalidad de realizar las evaluaciones descritas en la Tabla 36 de los objetivos 2 y 4. En la Tabla 35, de la sección anterior, donde se muestra el cumplimiento de los Requerimientos Funcionales, se observa que se cumple con estos objetivos.

5.4 Síntesis del capítulo

Se presentaron los criterios para seleccionar los procesos del modelo Tritón, los seleccionados fueron:

- Postmortem
- Gestión de procesos
- Aseguramiento de la Calidad del Producto y del Proceso.
- Administración de la Configuración

Se presentó el resultado del modelado de los procesos seleccionados, y el análisis cualitativo de este resultado indica, que de los 10 elementos identificados en la notación EPF utilizados para modelar 12 de los procesos de Tritón, 6 elementos se modelan con la Suite BPM. El modelado con la Suite BPM de los procesos seleccionados de Tritón modelados con EPF y su análisis cualitativo forma parte de la evaluación del objetivo 1 de la investigación.

También se presentó la evaluación de los objetivos 2 y 4, a través de mostrar el cumplimiento de los requerimientos de la Suite BPM mostrando, en particular, las adaptaciones que se le hicieron a la Suite BPM para completar la satisfacción de estos requerimientos. Estas adaptaciones son:

Primera adaptación. Dar a la Suite BPM la capacidad que durante la etapa de Ejecución del proceso y si la tarea lo requiere, subir y bajar artefactos a la Suite BPM.

Segunda adaptación. Dar a la Suite BPM la capacidad que durante la etapa de Ejecución del proceso, los artefactos que suban a la Suite BPM, sean administrados por un sistema de control de versiones.

Tercera adaptación. Dar a la Suite BPM la capacidad que durante la etapa de Ejecución del proceso, las tareas que se presentan al Participante del Proceso, de acuerdo a su rol, incluyan las descripciones de las mismas.

En este capítulo se presentó la metodología para realizar la evaluación de los objetivos de la investigación vistos en la introducción.

Capítulo

6 Discusión crítica y Recomendaciones para el Trabajo Futuro

El contenido de esta investigación es el primer acercamiento de utilizar el enfoque BPM para coadyuvar en la implementación y ejecución de un Modelo de Mejora de Procesos. Como se ha mencionado, este enfoque apoya su realización en Suites BPM.

Este proyecto tuvo la complejidad de reconocer las características que diferencian a los procesos de los Modelos de Mejora de Procesos con respecto de los procesos que comúnmente se apoyan en Suites BPM. También fue complejo construir la Suite BPM que tuviera la funcionalidad correspondiente a esas características. Algunos requerimientos de los procesos de los Modelos de Mejora de Procesos que no están contemplados en el funcionamiento de las suites examinadas (ver sección 2.4.4) son los siguientes:

- La función de subir y bajar artefactos.
- Tener un control de versiones de los artefactos.
- La función de proporcionar a los participantes del proceso las descripciones extensas de las tareas que tienen que realizar.

6.1 Desarrollo de la Suite BPM.

Con respecto al desarrollo de la Suite BPM fue importante considerar los siguientes elementos de discusión.

Arquitectura. Como se ha mencionado, para agregar las funciones que satisficieran los requerimientos mencionados en la sección anterior se desarrolló una Suite BPM; se implemento una arquitectura que soportara los cinco elementos que identifican a las Suites BPM (ver sección 2.4.2) y las funciones faltantes identificadas. Para el componente de monitoreo se

implementó la métrica de duración de tareas; la arquitectura posee el Atributo de Calidad de Extensibilidad, es decir, permite que se puedan agregar otras métricas, como por ejemplo, la métrica de duración total de las instancias de los procesos, que apoyaría en la estimación del tiempo total que tomarían otras instancias de un mismo proceso; aumentando la posibilidad de acordar tiempos de entrega mas acertados. Otros Atributos de Calidad son el de Seguridad y de Modificabilidad (ver sección 3.3.5.2).

En la Figura 57 se muestran en color blanco los cinco componentes que corresponden a las arquitecturas de las Suites BPM (ver sección 2.4.2), en el interior de estos se presentan los componentes desarrollados y los tomados de la Suite JBPM: DiseñadorDeProcesos, BaseDeDatosJBPM y MotorJBPM. Los componentes: Encapsulador del Comportamiento del Proceso, Control de Versiones y Base de Datos 2, se implementaron para cumplir con los requerimientos mencionados en la sección anterior, estos componentes se comentan a continuación.

Encapsulador del Comportamiento del Proceso. Integrado por los componentes: WorkflowService y Controlador, explicados en la sección 3.3.4.1, se incorporaron para, como su nombre lo dice, encapsular el comportamiento de los procesos, o también, el comportamiento de las tareas, como por ejemplo: iniciar tarea, donde la petición de un participante requiere varias peticiones al Motor de Ejecución de Procesos, incluyendo las necesarias para presentar la descripción de las tareas.

Control de versiones. Está integrado por los componentes: AdmonDeArchivos y Subversión, explicados en la sección 3.3.4.1. Este componente fue agregado para que la suite permitiera subir y bajar artefactos con su respectivo control de versiones.

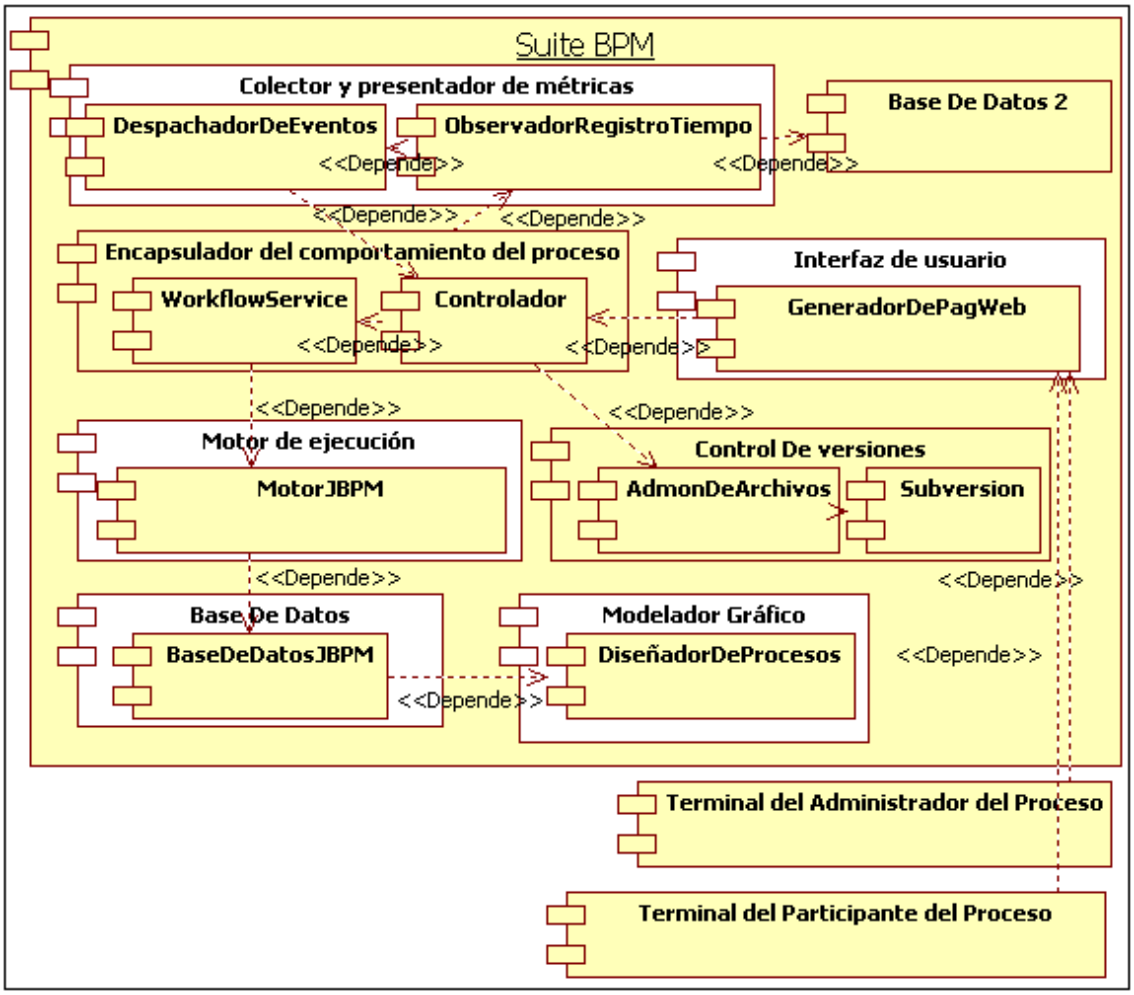


Figura 57: Componentes de la Suite BPM construida

Desarrollo. Como se ha mencionado antes, para desarrollar la Suite BPM se tomaron componentes de la Suite JBPM, acontinuación se presentan algunos comentarios con respecto a haberlos utilizado.

- *DiseñadorDeProcesos.* Permitio modelar los procesos, pero estos modelos fueron incompletos porque no permite agregar descripciones de: tareas, actividades, procesos y roles.
- *BaseDeDatosJBPM.* No cuenta con documentación sobre su implementación.
- *MotorJBPM.* No permite modificaciones de la Definición del Proceso durante su instancia, como por ejemplo: eliminación de tareas o actividades, reasignación de roles.

Metodología. La metodología para construir la Suite BPM presentada en la sección 3.1 se considera acertada por los resultados obtenidos. Sin embargo, el resultado de la actividad*de Identificación de requerimientos*no fue satisfactorio en su totalidad; porque faltaron requerimientos respecto a la usabilidad de la Suite BPM. Esto se debio a que era la primera vez que se construía una suite que apoyara a los procesos de desarrollo de software y también, por falta de experiencia, no se atendio este aspecto; por lo que no se investigó con personas dedicadas a la implementación de Modelos de Mejora de Procesos, como deberían ser las páginas Web de la suite para que les resultaran útiles.

También en la metodología se tiene la actividad de *comprobación del cumplimiento de los requerimientos funcionales;* la cual contemplaba inicialmente la participación de desarrolladores en las instancias de varios procesos. Esto no se realizó porque la Suite BPM construida no satisfizo los requerimientos de usabilidad solicitados por la persona encargada de autorizar las pruebas con sus desarrolladores, además de considerar que si el modelo de la Definición del Proceso no estaba completo, no tenía caso la ejecución. Por esta razón, las pruebas consistieron en revisar el cumplimiento de los requerimientos que si se habían planteado en el inicio.

Tecnologías. Las tecnologias utilizadas: ANT e Hibernate, fueron utilizadas por tener conocimiento de ellas. Spring MVC Framework fue seleccionado porque provee mecanismos para interceptar y controlar, que permite factorizar el comportamiento común en el manejo de múltiples peticiones, en comparación con Struts *[46, sitio Web]*.

Bases de Datos. En la sección 2.4.4 se mencionó que la Suite JBPM no contaba con un componente para la recolección de métricas. Inicialmente se pensó en agregar la plataforma SeeWhy *[25, pág. 160]* para cubrir esta función, pero no se encontró una edición gratuita, por lo que de forma un poco tardía se decidio construir el componente “Colector y Presentador de Métricas”. Para lo cual se requería de una base de datos que almacenara los resultados de las métricas. Se contaba con la BaseDeDatosJBPM, pero al carecer de documentación de su implementación, se decidio no invertir tiempo en

investigar como estaba implementada y se agregó otra base de datos que se tenía. Esta segunda base de datos es embebida, es decir, sólo se accede a sus tablas por la aplicación que la crea, en este caso, la Suite BPM; los archivos que genera son con extension .dat.

También se mencionó en la sección 5.1 que la BaseDeDatosJBPM es de capacidad limitada, exclusiva para prototipos, como así lo indica la literatura revisada *[25, pág. 149]*, en ésta referencia también se proporcionan los pasos para cambiar la BaseDeDatosJBPM al sistema de gestión de bases de datos MySQL.

Otras herramientas. Dentro del alcance de este proyecto no estaba contemplado explorar que herramientas podrían trabajar con la Suite BPM. Pero lo que se puede decir al respecto, es que las herramientas que se ocupen podrian cubrir los requerimientos que se han identificado en este trabajo y que no satisfacen las suites investigadas, o sustituir algunos de los componentes de la Suite BPM que se han desarrollado en este proyecto.

Requerimientos para instalar la Suite BPM. Como se ha mencionado antes, la Suite BPM que se construyó es para prototipos, es decir, como para procesos como los modelados en este trabajo. La Suite BPM se instala en una computadora que soporte los pre-requerimientos presentados en la Tabla 37.

Tabla 37: Pre-requerimientos para la instalación de la Suite BPM

Pre-requerimientos	**Referencia para los requerimientos de sistema y espacio en disco.**
JDK 5.0	*[47, sitio Web]*
JBPM JPDL Suite 3.2.x	*[48, sitio Web]*
Eclipse	*[49, sitio Web]*
Apache ANT 1.7	*[50, sitio Web]*

Una vez instalados los pre-requerimientos se importa el proyecto (Suite BPM) en el Entorno de Desarrollo Integrado “Eclipse”. La computadora donde se ha instalado la suite puede estar conectada, a través de una red, con otras computadoras, a fin de que tengan acceso los participantes del proceso a las páginas Web de la suite.

6.2 Modelado de los procesos

En cuanto al modelado de las definiciones de los procesos fue importante considerar los siguientes elementos de discusión para el desarrollo de la Suite BPM.

Elementos de JPDL. JPDL cuenta con varios elementos para representar la definición de los procesos (ver sección 3.3.3.5), en este proyecto se decidió utilizar sólo algunos de estos elementos, porque había que agregarles una programación adicional para cumplir con las funciones que se requerían en la suite; la selección se basó en ocupar los que permitieran modelar los aspectos en común de los Modelos de Mejora de Procesos estudiados (ver sección 2.2.6). Se considera que esta selección fue la apropiada, porque fue posible modelar las definiciones de los procesos seleccionados de Tritón; los aspectos que no se modelaron con estos elementos, como las descripciones de las tareas por ejemplo, fueron limitaciones de JPDL (ver sección 3.3.3.5). Cabe mencionar que desde el inicio no se contempló la posibilidad de modelar subprocesos porque el objetivo de este proyecto no era modelar definiciones de procesos con todas las características posibles, era saber si las Suites BPM, con sus cinco elementos (ver sección 2.4.2), se podían adaptar para modelar los procesos de los Modelos de Mejora de Procesos. Se recomienda como trabajo futuro determinar la conveniencia de usar todos los elementos de JPDL.

EPF y JPDL. Como se ha mencionado, los procesos de Tritón están modelados con algunos elementos de la notación EPF y se realizó un mapeo hacia los elementos seleccionados de JPDL, es decir, en este trabajo no se determinó un mapeo entre todos los elementos de la notación EPF y todos los

elementos de la notación JPDL. Se recomienda como trabajo futuro determinar la conveniencia de utilizar todos los elementos de ambas notaciones y la conveniencia de trabajar con ambas notaciones, EPF y JPDL, o en su caso, si existe una notación que ofrezca mayores ventajas.

Descripciones. Al comparar los Modelos de Mejora de Procesos se vio la necesidad de que la Suite BPM que se desarrollaría tenía que presentar al Participante del Proceso las descripciones de las tareas. Se procedió a realizar la adaptación que satisficiera esta necesidad, pero por falta de experiencia en realizar investigación, no se busco la existencia de una herramienta que pudiera ser útil y adaptarse a la Suite BPM. Ahora se sabe que existe la necesidad de crear una capa en la Suite BPM, que contenga una interfaz gráfica (página Web por ejemplo). Que le permita al *Administrador del Proceso* agregar: las descripciones de las tareas, y las plantillas correspondientes a estas; descripciones de actividades, procesos, roles, productos, guías y demás información que permita, que durante la ejecución del proceso, se vaya simultáneamente capacitando al Participante del Proceso. Siendo esta misma capa la que se encargue de almacenar esta información en una base de datos. Se recomienda como trabajo futuro agregar a la Suite BPM ésta capa o una herramienta que cumpla con las funciones mencionadas, pero que además, también cumpla con el Atributo de Calidad RNF-01 visto en la Tabla 8, de la categoría de Modificabilidad (ver sección 3.3.5.2.1). Por el momento, las descripciones de las tareas se agregan en un archivo independiente, pero esto disminuye la usabilidad de la Suite BPM.

Procesos modelados. Las descripciones de los procesos seleccionados de Tritón, están redactadas sin considerar el apoyo de una Suite BPM, por lo que se hicieron algunas adecuaciones, por ejemplo: la palabra “Enviar” (un artefacto) cambia a subir y bajar del sistema un artefacto. Lo que implica que en notación EPF se enuncia como un solo paso, pero al modelarlo se convierte en cuando menos dos tareas, una para el rol encargado de hacer y subir el artefacto y otra para el rol que tiene la obligación de recibirlo. Estas adecuaciones se hicieron en función de lo que JPDL permite hacer para apoyar la ejecución de los procesos. Como trabajo futuro se recomienda investigar si el

estándar BPMN (ver sección 2.3.1), que tiene una mayor cantidad de elementos de modelado que JPDL, puede apoyar de mejor manera a los Modelos de Mejora de Procesos, cuando se ejecutan después de ser implementados.

Control de versiones. Se decidió hacer un control de versiones sobre los artefactos. Sin embargo, no se incluyó en el alcance de este proyecto tener un control de las versiones de las definiciones de los procesos; considerando que estos continuamente son modificados al ser mejorados o adaptados para desarrollar productos de software específicos, se recomienda como trabajo futuro agregar a la Suite BPM ésta función.

Instancia del Proceso. Con las Suites BPM estudiadas (ver sección 2.4.4) no es posible modificar al proceso durante su ejecución. Sólo permiten modificar la Definición del Proceso antes de crear la instancia correspondiente; cualquier modificación a la definición impacta sólo a la siguiente instancia que se cree. Esto sucede por la codificación del Motor de Ejecución de Procesos, que es uno de los componentes representativos en una suite. Tomando en cuenta que el objetivo del proyecto es saber si las Suites BPM, con sus cinco elementos (ver sección 2.4.2), se podían adaptar para modelar los procesos de los Modelos de Mejora de Procesos, no se considero en el alcance hacer la adaptación que permitiera modificar las instancias de los procesos. No obstante, si se identificó la necesidad de crear una capa adicional a la Suite BPM para eliminar tareas o reasignar roles durante la ejecución, con la finalidad de corregir defectos. Se recomienda como trabajo futuro agregar a la Suite BPM la capa o herramienta que cumpla con las funciones mencionadas, pero que además, también cumpla con el Atributo de Calidad RNF-01 visto en la Tabla 8, de la categoría de Modificabilidad (ver sección 3.3.5.2.1).

Modelos de Mejora de Procesos. En este proyecto se investigó como ayudar a las organizaciones pequeñas para la implementación de un Modelo de Mejora de Procesos. En la actualidad, las organizaciones están necesitando herramientas multimodelos, es decir, que apoyen a las organizaciones que utilizan varios Modelos de Mejora de Procesos, en donde se requiere un

procedimiento de traducción para mapeos entre: elementos de los modelos de procesos, elementos de modelos de procesos y una notación de modelado de procesos, y entre elementos de la notación de modelado de procesos y la Suite BPM.

6.3 Desarrollo profesional

Entender y asimilar la teoría y el espíritu de trabajar con procesos no es tarea fácil, mas aún, si se trata de metaprocesos, como lo es en el caso de *Gestion de Procesos* de MoProSoft, que tiene el propósito de establecer los procesos de una organización. Este proyecto me sirvio para tener fundamentos teóricos formales del conocimiento previo que tenía sobre procesos y del nuevo que adquirí. Al redactar la tesis organice la información en una estructura cronológica de resultados, esto me sirvio para tener en mi mente de forma ordenada los conocimientos adquiridos y aplicados. Se destaca lo siguiente:

- Conocimiento de los Modelos de Mejora de Procesos: CMMI, MoProSoft y Tritón.
- Entendimiento del funcionamiento de las Suites BPM.
- Conocimiento de la Suite Jboss JBPM.
- Metodología de desarrollo de un software de calidad.
- Creación de una Arquitectura de Software, con su respectiva evaluación.
- Conocimiento de las tecnologias: Spring MVC Framework, Hibernate, ANT, Subversión.
- Modelado de la definición de procesos de software.

La Suite BPM que construí tiene limitantes, que se plantean como trabajo futuro en las secciones anteriores pero ofrece en mi opinión, entre otras, las siguientes ventajas:

- Obliga la adherencia al proceso por parte de sus participantes.
- Si en la Definición del Proceso se introducen tareas de inspección, los errores son detectados y corregidos antes de que continúe el proceso.

- Evita el estrés que se provoca cuando se tiene que interactuar con otro humano para solicitar reiteradamente información.
- Es posible identificar que rol consumio mayor cantidad de tiempo que el estimado.

Capítulo
7 Conclusiones

En las siguientes secciones para cada objetivo de la investigación se presenta: el planteamiento, la evaluación, el resultado, la justificación, las limitaciones y la conclusión particular. Al final de este capítulo se presenta la conclusión general.

7.1 Conclusión del objetivo uno.

En esta sección se presenta el objetivo 1, su evaluación, resultado, justificación del resultado y limitaciones en el alcance del objetivo.

Objetivo 1. *"Soportar dentro de la Suite BPM el modelado de un número variable de procesos de un Modelo de Mejora de Procesos".*

Evaluación. *"La evaluación del alcance de este objetivo se realiza mediante el modelado de procesos seleccionados de un Modelo de Mejora de Procesos. Los procesos son seleccionados porque incorporan elementos que se encuentran en todos los demás procesos estudiados".*

Resultado. Parcialmente se alcanzo el objetivo 1.

Justificación. La Suite BPM permite modelar 6 de 10 elementos de la notación EPF (ver sección 5.2.1.5). Esto se debió principalmente a que no se modelaron las descripciones de Procesos o Tareas (notación EPF). La interpretación de este resultado es que las Suites BPM estudiadas, se han especializado en la ejecución del proceso, más no así en la capacitación simultanea al Participante del Proceso, es decir, le indican al participante que hacer, pero sin detalladas explicaciones. En este trabajo, con la Suite BPM que se construyó, se demostró que es posible adaptar las Suites BPM para que presenten las descripciones contenidas en los Modelos de Mejora de Procesos.

Limitaciones. La Suite BPM que se construyó no modela subprocesos.

Conclusión particular. Se concluye que la Suite BPM soporta parcialmente el modelado de un número variable de procesos de un Modelo de Mejora de Procesos.

De acuerdo al estado del arte, los Modelos de Mejora de Procesos tienen como propósito capacitar y guiar a los participantes de los procesos. Esto hace que cuenten con descripciones amplias sobre los elementos con los que se definen los procesos, además de otros aspectos como el intercambio y control de versiones de artefactos. Las Suites BPM estudiadas cuentan con los elementos para modelar la ejecución de los procesos de negocios, pero no modelan las descripciones mencionadas y los otros aspectos de los Modelos de Mejora de Procesos. Por esta razón existe la necesidad de hacer adaptaciones a la Suites BPM estudiadas si se desea modelar este tipo de modelos.

7.2 Conclusión del objetivo dos

En esta sección se presenta el objetivo 2, su evaluación, resultado, justificación del resultado y limitaciones en el alcance del objetivo.

Objetivo 2. *"Soportar la etapa de Ejecución de procesos operativos, en particular en lo que se refiere al intercambio de artefactos y el control de versiones de los mismos".*

Evaluación. "*La evaluación del alcance de este objetivo se realiza mediante un estudio que permite verificar que la Suite BPM satisface los requerimientos de los procesos operativos de los Modelos de Mejora de Procesos (sección 2.2.6), en particular el intercambio y el control de versiones de artefactos".*

Resultado. Se alcanzó el objetivo 2.

Justificación. La Suite BPM soporta:

- El acceso restringido a participar en un proceso es una ventaja de seguridad que ofrece ocupar una Suite BPM.
- Crear varias instancias de un mismo proceso de forma tal que con cada instancia, por ejemplo, se puede estar desarrollando un producto de software diferente.
- Que en la etapa de Ejecución de un proceso, al mandar las tareas al monitor de los participantes del proceso, que se adhieren al orden de las tareas. También los participantes del proceso pueden intercambiar artefactos sin tener que enviarlos por correo o de mano en mano. Es posible que el Administrador del Proceso recupere versiones anteriores de artefactos.
- "Cerrar" una tarea sin marcha atrás de forma tal que el Participante del Proceso sólo tiene la opción de proseguir con la siguiente tarea.
- La actualización e instalación de una versión de un modelo de una definición de un proceso, sólo se hace al inicio, o terminación de la Instancia del Proceso, pero no durante la instancia.

Limitaciones. Cuando se instanciaron los procesos seleccionados se observaron las siguientes limitaciones que no se implementaron porque no se identificaron como requerimientos de la Suite BPM en el inicio:

- Cuando un proceso ocupa las salidas (artefactos) de otros procesos, no se implementó que la Suite BPM automatice pasar los artefactos completados por un proceso al proceso que los requiere como entradas.
- La Suite BPM carece de un mecanismo que avise a los participantes del proceso que ya pueden realizar su tarea, porque ya se llegó a la actividad que incluye su tarea.

Conclusión particular. Se concluye que la Suite BPM soporta la etapa de ejecución de los procesos operativos, en particular en lo que se refiere al intercambio de artefactos y el control de versiones de los mismos.

7.3 Conclusión del objetivo tres

En esta sección se presenta el objetivo 3, su evaluación, resultado, justificación del resultado.

Objetivo 3. *"Soportar mecanismos flexibles de colecta de métricas que permitan el monitoreo durante la etapa de Ejecución del proceso"*.

Evaluación. *"La evaluación del alcance de este objetivo se realiza mediante una revisión del Diseño Arquitectural que demuestre que se han considerado mecanismos que permitan una adición flexible de componentes para la recolecta de métricas"*.

Resultado. Se alcanzó el objetivo 3.

Justificación. El diseño que se realizó permite agregar un nuevo componente de colecta de métricas y el impacto de su introducción sólo afecta a un componente. Un nuevo componente de colecta de métricas permite ampliar el monitoreo de la Instancia del Proceso. Los mecanismos fueron presentados en la sección 3.3.5.2.2.

Conclusión particular. Se concluye que la Suite BPM soporta mecanismos flexibles de colecta de métricas que permitan el monitoreo durante la etapa de Ejecución del proceso.

7.4 Conclusión del objetivo cuatro

Objetivo 4. *"Proveer información a los participantes del proceso sobre la etapa en la cual se encuentra la Instancia del Proceso durante la etapa de Ejecución del proceso".*

Evaluación. *"La evaluación del alcance de este objetivo se realiza mediante un estudio que permite verificar que la Suite BPM incluye la funcionalidad de*

proveer a los participantes del proceso la información sobre la fase y tarea en la cual se encuentra la Instancia del Proceso".

Resultado. Se alcanzó el objetivo 4.

Justificación. Una de las ventajas que ofrece ocupar la Suite BPM, que proporciona tanto a los participantes de los procesos como al Administrador del Proceso información sobre la fase y tarea en donde se encuentra la Instancia del Proceso a través de la *Tabla de monitoreo* de la Instancia del Proceso.

Limitaciones. La Suite BPM que se construyó incluye diagramas donde el Participante del Proceso puede observar las actividades del proceso completo, pero no existe un señalamiento que le indique en el diagrama la actividad exacta en la que se encuentra el proceso.

Conclusión particular. Se concluye que la Suite BPM soporta proveer información a los participantes del proceso sobre la etapa en la cual se encuentra la Instancia del Proceso durante la etapa de Ejecución del proceso.

7.5 Conclusión general

Se concluye que es posible construir una Suite BPM de bajo costo que permita soportar el ciclo de vida de los procesos de un Modelo de Mejora de Procesos de acuerdo a las etapas del enfoque BPM de Modelado, Ejecución y Monitoreo.

8 Apéndice A

Jboss JBPM.

Los "*Lenguajes de Ejecución de Procesos"* son especificaciones que contienen la semántica que es entendida por los *motores de ejecución de procesos* *[51, pág. 239]*. Jboss JBPM es una plataforma para *Lenguajes de Ejecución de Procesos*. Estos procesos pueden ser procesos de negocios (BPM), o inclusive, flujos de trabajo para servicios de orquestación *[30, pág. Web]*. Cuando se habla de orquestación se refiere a procesos de negocios ejecutables que tienen interacción con servicios Web internos y externos *[52, pág. 252]*. JBPM tiene su propio *Lenguaje de Ejecución de Procesos* llamado Lenguaje de Definición de Procesos Java (JPDL, por sus siglas en inglés).

Elementos de JBPM

Una suite construida usando JBPM tiene una arquitectura compuesta por los elementos presentados en la Figura 58. Estos elementos son *[25, pág. 54]*:

Diseñador de Procesos (Designer). Es un Modelador Gráfico de Procesos. En el caso de JBPM es un plugin del Entorno de Desarrollo Integrado Eclipse. El Diseñador de Procesos realiza una traducción hacía JPDL de los elementos que tienen representación gráfica.

Base de Datos. La base de datos almacena tanto definiciones de procesos (puede almacenar múltiples versiones de una misma definición), como datos relacionados con instancias de estas definiciones.

Motor de Ejecución de Procesos JBoss JBPM. Es un Motor de Ejecución de Procesos visto en la sección 2.4.2

Servidor de Aplicación Jboss. El motor de procesos Jboss JBPM es ejecutado en el servidor de aplicaciones Jboss. El servidor también

contiene un Servidor Web a través del cual se genera la página Web que es la interfaz con el usuario.

Componente de Presentación. Genera páginas que permiten al usuario tener conocimiento sobre las tareas que debe realizar. También, este elemento recibe datos del usuario que al ser enviados al Motor de Ejecución de Procesos permiten, a este último, el avanzar en la ejecución de la Instancia del Proceso.

Terminal del Usuario (usuario Administrador y usuario Participante del Proceso). Vía Internet el Componente de Presentación, envía las tareas y formas a los usuarios para que se desplieguen en las pantallas de sus terminales.

Colector de Métricas. Es el componente encargado de recolectar los datos de las métricas durante la ejecución del proceso.

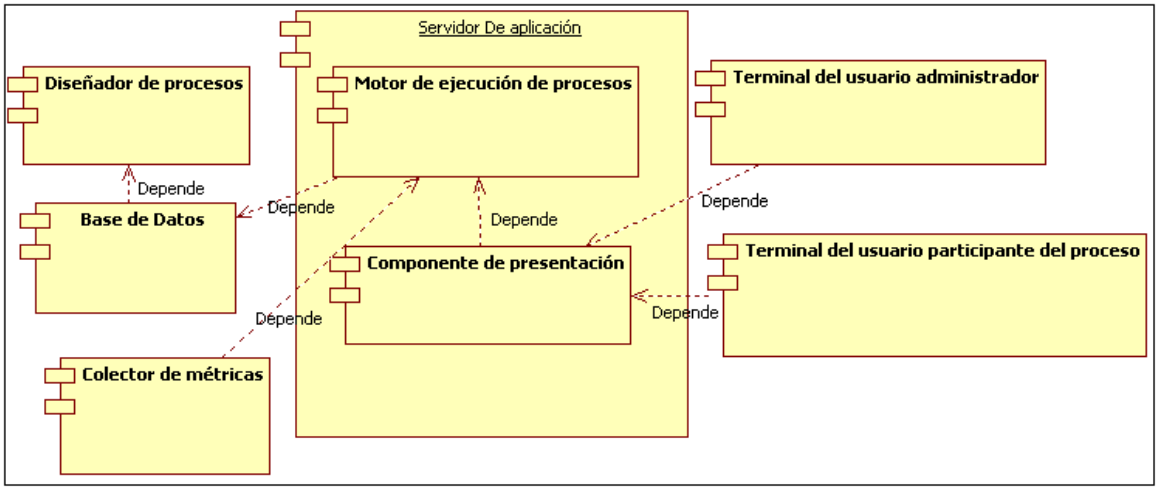


Figura 58: Arquitectura de la Suite JBPM

La base de datos por default es embebida en JBPM, y es excelente para el desarrollo de prototipos y de una aplicación, pero no para el manejo de procesos de nivel empresarial. Sin embargo, JBPM soporta bases de datos empresariales, incluyendo el Oracle, DB2 y Microsoft SQL Server *[25, pág. 149]*.

En el presente trabajo el fin es estudiar el enfoque BPM, por esta razón, se eligió trabajar con la base de datos de JBPM.

Algunas Suites BPM ofrecen interfaces de usuario gráficas para realizar definiciones de procesos con características particulares. JBPM todavía no cuenta con un entorno de desarrollo "arrastrar y soltar" que el desarrollador de procesos pudiera usar para hacer las páginas Web. Si se requiere hacer cambios a las páginas Web que tiene por default, existen dos opciones: editar las interfases de usuarios estándares que tiene en XHTML, o construir páginas Web propias *[25, pág. 146]*. La secuencia de pasos de cómo trabaja JBPM se presenta en la Figura 59 *[25, pág. 54]*.

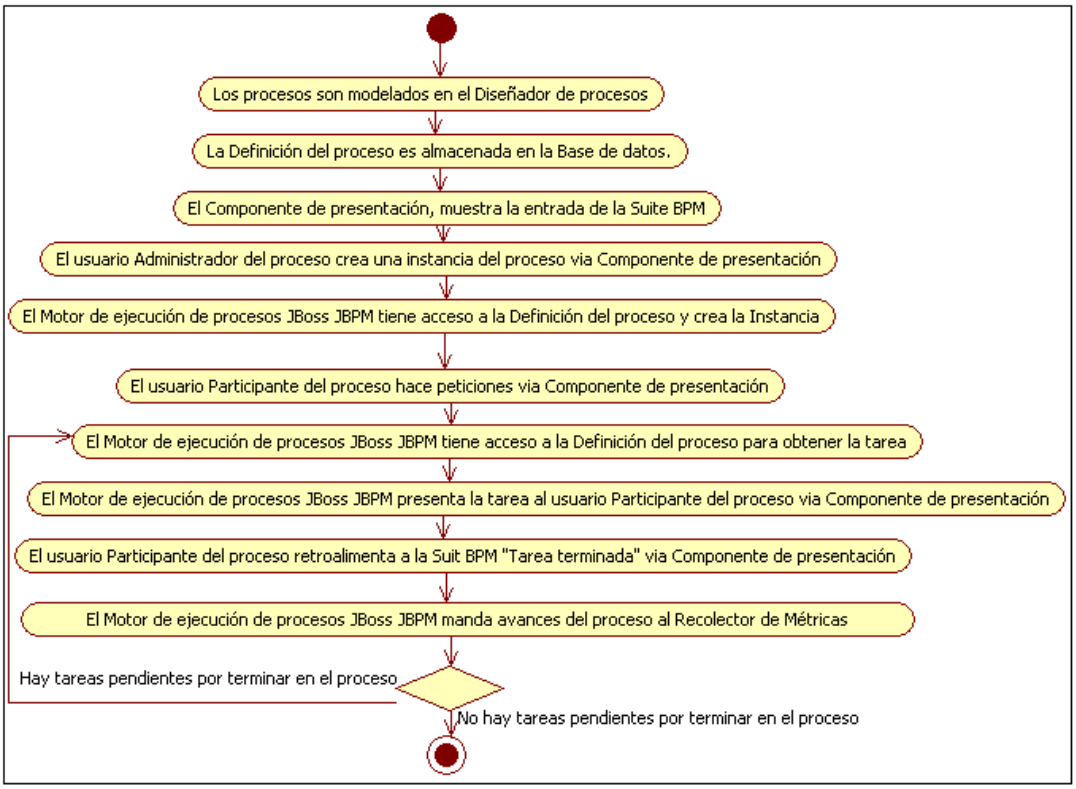


Figura 59: Diagrama de Actividades de la interacción de los componentes JBPM

9 Apéndice B

En este apéndice se presenta el templete que corresponde al detalle de los casos de uso para la construcción de la Suite BPM.

Nombre	E1. Instanciar Definición del Proceso
Descripción	Este caso de uso permite soportar crear nuevas instancias de una Definición del Proceso.
Actores	El Administrador del Proceso
Precondiciones	El sistema se encuentra mostrando la pantalla de instancias del proceso.
Post-condiciones	El sistema se encuentra mostrando la pantalla de instancias del proceso y se ha creado una nueva instancia.
Descripción del flujo principal	1. El Administrador del Proceso selecciona la opción instanciar. 2. El sistema despliega la pantalla de ingreso de nombre del proceso. 3. El Administrador del Proceso escribe el nombre en el campo de nombre del proceso. 4. El sistema crea la Instancia del Proceso.
Flujos alternativos	
Requerimientos No Funcionales	

Nombre	A2. Ingresar al sistema por rol
Descripción	Este caso de uso permite soportar el ingreso restringido al sistema.
Actores	- Administrador del Proceso - Participante del Proceso
Precondiciones	El sistema se encuentra mostrando la pantalla de rol.
Post-condiciones	El sistema se encuentra mostrando la pantalla de acceso exitoso o página principal.
Descripción del flujo principal	1. El actor introduce nombre de rol y contraseña. 2. El sistema verifica la contraseña de acuerdo al rol. 3. El sistema presenta la pantalla de acceso exitoso, dándole acceso a las tareas que le corresponden al rol.
Flujos alternativos	1. El actor introduce nombre de rol y contraseña. 2. El sistema verifica la contraseña de acuerdo al rol. 3. El sistema presenta la pantalla de página principal.
Requerimientos No Funcionales	RNF-03 de la categoría de Seguridad.

Nombre	A3. Iniciar tarea
Descripción	Este caso de uso permite que se registre en el sistema la fecha y hora de inicio de una tarea.
Actores	- Administrador del Proceso - Participante del Proceso
Precondiciones	El sistema se encuentra mostrando la pantalla de tareas
Post-condiciones	El sistema se encuentra mostrando la pantalla de tarea.
Descripción del flujo principal	1. El actor inicia la tarea 2. El sistema le presenta la pantalla de tarea, desplegando la tabla con los artefactos correspondientes a la tarea.
Flujos alternativo No. 1	1. El actor inicia la tarea 2. El sistema le presenta la pantalla de tarea, con la *Tabla de artefactos* vacía.
Flujos alternativo No. 2	1. El actor inicia la tarea 2. El sistema le presenta la pantalla de tarea, desplegando la tabla con los artefactos correspondientes a la tarea. 3. El actor baja un artefacto del sistema.
Flujos alternativo No. 3	1. El actor inicia la tarea 2. El sistema le presenta la pantalla de tarea, desplegando la tabla con los artefactos correspondientes a la tarea. 3. El actor baja un artefacto del sistema y minutos después sube el mismo artefacto con información adicional.
Requerimientos No Funcionales	

Nombre	A4. Terminar tarea
Descripción	Este caso de uso permite que se registre en el sistema la fecha y hora de terminación de una tarea.
Actores	- Administrador del Proceso - Participante del Proceso
Precondiciones	El sistema se encuentra mostrando la pantalla de tareas
Post-condiciones	El sistema se encuentra mostrando la pantalla de tareas
Descripción del flujo principal	1. El actor termina la tarea 2. El sistema le presenta la pantalla de tarea.
Flujos alternativos	
Requerimientos No Funcionales	

Nombre	A5. Bajar artefactos
Descripción	Este caso de uso permite soportar la descarga de archivos

	(artefactos) del sistema.
Actores	- Administrador del Proceso - Participante del Proceso
Precondiciones	El sistema se encuentra mostrando la pantalla de tarea
Post-condiciones	El sistema se encuentra mostrando la pantalla de tarea
Descripción del flujo principal	1. El actor elige bajar artefacto 2. El sistema obtiene del sistema de control de versiones la última versión del artefacto. 3. El actor elige la ubicación para guardar el artefacto. 4. El sistema baja el artefacto.
Flujos alternativos	
Requerimientos No Funcionales	

Nombre	A6. Subir artefactos
Descripción	Este caso de uso permite cargar archivos (artefactos) al sistema.
Actores	- Administrador del Proceso - Participante del Proceso
Precondiciones	El sistema se encuentra mostrando la pantalla de tarea
Post-condiciones	El sistema se encuentra mostrando la pantalla de tarea
Descripción del flujo principal	1. El actor elige subir el artefacto 2. El sistema despliega el recuadro para ubicar el artefacto. 3. El actor ubica el artefacto y sube el artefacto. 4. El sistema sube al sistema de control de versiones el artefacto.
Flujos alternativos	
Requerimientos No Funcionales	

Nombre	B7. Monitorear el avance de la Instancia del Proceso.
Descripción	Este caso de uso permite desplegar un listado de las tareas que han iniciado.
Actores	- Administrador del Proceso - Participante del Proceso
Precondiciones	El sistema se encuentra mostrando la pantalla de instancias del proceso.
Post-condiciones	El sistema se encuentra mostrando la pantalla de monitoreo.
Descripción del flujo principal	1. El actor en la pantalla de instancias del proceso selecciona ver avance. 2. El sistema muestra la pantalla de monitoreo, con la *Tabla de monitoreo*, con cada tarea iniciada y

	concluida, con sus respectivas fechas de inicio y terminación, el rol responsable y la duración de estas, con sus diferencias en tiempo con respecto al tiempo estimado
Flujos alternativos	
Requerimientos No Funcionales	RNF-02 de la categoría de Extensibilidad.

Nombre	E8. Actualizar e instalar nueva versión de la definición del proceso
Descripción	Este caso de uso permite soportar agregar, quitar o cambiar el modelo gráfico de la Definición del Proceso en el sistema, y reiniciar el sistema para que reconozca la nueva versión de la Definición del Proceso.
Actores	Analista del proceso.
Precondiciones	El sistema muestra la pantalla para modelar el proceso con notación JPDL.
Post-condiciones	La Definición del Proceso ha sido almacenada en la base de datos.
Descripción del flujo principal	1. El analista del proceso diseña el proceso 2. El sistema almacena la Definición del Proceso en la base de datos.
Flujos alternativos	
Requerimientos No Funcionales	RNF-01 de la categoría de Modificabilidad.

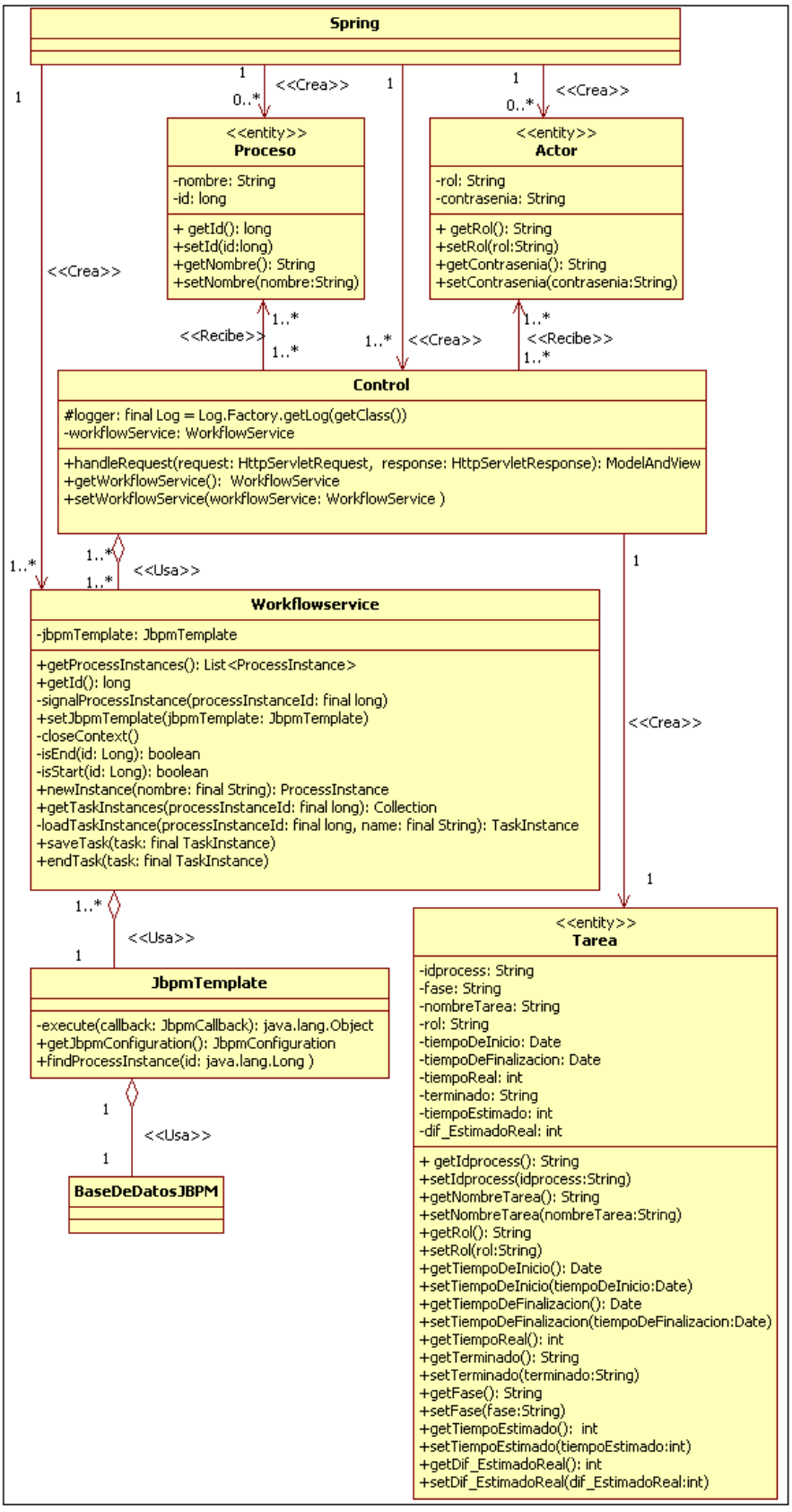


Figura 60: Modelo de Dominio de la Suite BPM construida

11 Apéndice D

A manera de ejemplo, en este apéndice, se presentan los siete *Casos de Prueba Funcional* que se llevaron a cabo para el proceso Postmortem.

Caso de uso: Ingresar al Sistema por Rol (ID-A2)	
Caso de prueba funcional: 01	
Propósito:	Probar si se cumple el flujo principal del caso de uso Ingresar al Sistema por Rol (ver apéndice B).
Prerrequisitos:	Estar situado en la página principal de la Suite BPM.
Datos de prueba:	Rol: Lider Contraseña: 1111
Pasos:	1. Dar click en “continuar” en la página principal. 2. Seleccionar el rol de “Líder”. 3. Ingresar la contraseña que le corresponde al rol de líder.
Resultados esperados:	Como la contraseña es correcta debe mostrarse la página con título “Sistema de automatización de procesos” que contiene el mensaje *“¡Bienvenido! Esta autorizado a ingresar en Procesos”*. La palabra procesos de este mensaje es una liga a la *Tabla de Instancias de procesos*.
Pregunta:	Se aprobó el caso de prueba funcional?
Respuesta:	Aprobado.
Descripción del defecto:	

Caso de uso: Ingresar al Sistema por Rol (ID-A2)	
Caso de prueba funcional: 02	
Propósito:	Probar si se cumple el flujo alternativo del caso de uso Ingresar al Sistema por Rol (ver apéndice B).
Prerrequisitos:	Estar situado en la página principal de la Suite BPM.
Datos de prueba:	Rol: Lider Contraseña: 1112
Pasos:	1. Dar click en “continuar” en la página principal. 2. Seleccionar el rol de “Líder”. 3. Ingresar la contraseña incorrecta.

Resultados esperados:	Como la contraseña es incorrecta debe mostrarse la página principal.
Pregunta:	Se aprobó el caso de prueba funcional?
Respuesta:	Aprobado.
Descripción del defecto:	

Caso de uso: Instanciar Definición del Proceso (ID-E1)	
Caso de prueba funcional: 03	
Propósito:	Probar si se cumple el flujo principal del caso de uso Instanciar Definición del Proceso (ver apéndice B).
Prerrequisitos:	El Líder (rol autorizado para crear instancias de la Definición del Proceso) ingresa a la Suite BPM con una contraseña válida, y se encuentra situado en la página “Procesos”.
Datos de prueba:	Rol: Líder Contraseña: 1111 Nombre del proceso: p1
Pasos:	1. Da click en “Nuevo” 2. En la página “Nueva instancia de proceso”, se introduce en el recuadro el nombre del proceso. 3. Se da click en “Execute”
Resultados esperados:	1. Aparece el mensaje “¡Proceso creado con éxito! Ir a Procesos” 2. Al ir a la página de “Procesos”, en la *Tabla de instancias de procesos* se han registrado: - Fecha y hora de inicio de la nueva instancia. - Nombre del proceso, en este caso, Postmortem. - Fase, en este caso, Inicio. - Identificador de la instancia. - Nombre de la instancia, en este caso, P1. - Aparece en la columna de “Ver actividades” la liga “Ver”. - Aparece en la columna de “Ver avance” la liga “Ver”.
Pregunta:	Se aprobó el caso de prueba funcional?
Respuesta:	Aprobado
Descripción del defecto:	

Caso de uso: Iniciar Tarea (ID-A3)	
Caso de prueba funcional: 03	
Propósito:	Probar si se cumple el flujo principal del caso de uso Iniciar Tarea (ver apéndice B).
Prerrequisitos:	El Líder ha ingresado satisfactoriamente a la Suite BPM y a creado una Instancia del Proceso y se encuentra en la página: “Tareas de la actividad: Actualizar agenda postmortem” que contiene la “Tabla de tareas”, y ésta incluye: - En la columna “Tareas”: A.Actualizar Agenda de postmortem. - En la columna “Iniciar”: La liga “Accesar”. Las columnas: “Descripción”, “Inicio”, “Fin” y “Terminar” están vacías.
Datos de prueba:	A.Actualizar Agenda de postmortem.
Pasos:	1. Dar click en “Accesar” de la tarea “A.Actualizar la agenda de postmortem”. 2. En la página “Tarea: A.Actualizar Agenda de postmortem” dar click en la liga “Tareas” para regresar a la página: “Tareas de la actividad: Actualizar agenda postmortem”
Resultados esperados:	Se despliega la página “Tarea: A.Actualizar Agenda de postmortem”, que contiene la “Tabla de artefactos”, y ésta incluye: - En la columna “Nombre”: El nombre de la plantilla, “PTLL_AgendaPM_Triton_DAS_v1.0.doc”. - En la columna “Plantilla”: La liga “Bajar”. - En la columna “Subir Nueva Versión”: La liga “Subir”. Al regresar a la página: “Tareas de la actividad: Actualizar agenda postmortem” contiene la “Tabla de tareas” que incluye: - En la columna “Descripción”: El líder actualiza la agenda de postmortem. - En la columna “Inicio”: La fecha y hora de inicio de la tarea. - En la columna “Terminar”: La liga “Terminar”. - La columna “Fin” está vacía.
Pregunta:	Se aprobó el caso de prueba funcional?
Respuesta:	Aprobado.
Descripción del defecto:	

Caso de uso: Bajar Artefactos (ID-A5)	
Caso de prueba funcional: 04	
Propósito:	Probar si se cumple el flujo principal del caso de uso Bajar Artefactos (ver apéndice B).
Prerrequisitos:	El Líder ha ingresado satisfactoriamente a la Suite BPM, ha creado una Instancia del Proceso, ha iniciado una tarea y se encuentra en la página: "Tarea: A.Actualizar Agenda de postmortem".
Datos de prueba:	La plantilla PTLL_AgendaPM_Triton_DAS_v1.0.doc
Pasos:	1. Da click en "Bajar" de la columna de "plantilla". 2. En la ventana emergente llamada "opening PTLL_AgendaPM_Triton_DAS_v1.0.doc" se selecciona la opción "Save File" y se pulsa en el botón "ok"
Resultados esperados:	Al minimizar la ventana, el archivo PTLL_AgendaPM_Triton_DAS_v1.0.doc se encuentra en el escritorio
Pregunta:	Se aprobó el caso de prueba funcional?
Respuesta:	Aprobado.
Descripción del defecto:	

Caso de uso: Subir Artefactos (ID-A6)	
Caso de prueba funcional: 05	
Propósito:	Probar si se cumple el flujo principal del caso de uso Subir Artefactos (ver apéndice B).
Prerrequisitos:	El Líder ha ingresado satisfactoriamente a la Suite BPM, ha creado una Instancia del Proceso, ha iniciado una tarea y se encuentra en la página: "Tarea: A.Actualizar Agenda de postmortem".
Datos de prueba:	El artefacto "PTLL_AgendaPM_Triton_DAS_v1.0.doc".
Pasos:	1. Dar click en la liga "Subir" en la columna "Subir Nueva Versión". 2. Ventana que se despliega, pulsamos el botón "browser" 3. En la ventana emergente ubicamos el artefacto "PTLL_AgendaPM_Triton_DAS_v1.0.doc" y pulsamos el botón "abrir". 4. Pulsamos el botón "SubmitQuery".
Resultados esperados:	Aparece el mensaje "¡Archivo cargado exitosamente!".

Pregunta:	Se aprobó el caso de prueba funcional?
Respuesta:	Aprobado.
Descripción del defecto:	

Caso de uso: Terminar Tarea (ID-A4)	
Caso de prueba funcional: 06	
Propósito:	Probar si se cumple el flujo principal del caso de uso Terminar Tarea (ver apéndice B).
Prerrequisitos:	El Líder ha ingresado satisfactoriamente a la Suite BPM y a creado una Instancia del Proceso, una tarea fue iniciada y se encuentra en la página: "Tareas de la actividad: Actualizar agenda postmortem"
Datos de prueba:	A.Actualizar Agenda de postmortem (tarea iniciada).
Pasos:	Dar click en "Terminar" de la tarea "A.Actualizar la agenda de postmortem".
Resultados esperados:	En la "Tabla de tareas": En la columna "Fin" están registradas la fecha y hora de terminación. En la columna "Iniciar" no se muestra la liga "Accesar". En la columna "Terminar" no se muestra la liga "Terminar".
Pregunta:	Se aprobó el caso de prueba funcional?
Respuesta:	Aprobado.
Descripción del defecto:	

Caso de uso: Monitorear el Avance de la Instancia del Proceso (ID-B7)	
Caso de prueba funcional: 07	
Propósito:	Probar si se cumple el flujo principal del caso de uso Monitorear el Avance de la Instancia del Proceso (ver apéndice B).
Prerrequisitos:	El Líder ha ingresado satisfactoriamente a la Suite BPM, ha creado una Instancia del Proceso, ha iniciado una tarea y se encuentra en la página: Procesos.
Datos de prueba:	Ninguno.
Pasos:	1. Dar click en "Ver" en la columna de "Ver avance"
Resultados	El despliegue de la página: Monitoreo de avance, que contiene la "Tabla de

esperados:	monitoreo” con las siguientes columnas y la siguiente información. - En la columna “Idproceso”: El identificador de la Instancia del Proceso al que corresponde la tarea iniciada. - En la columna “Fase”: Preparación. - En la columna “Nombre de tarea”: A.Actualizar Agenda de postmortem. - En la columna “Rol”: Líder. - En la columna “Tiempo de inicio”: La fecha y hora de inicio de la tarea: - La columna “Tiempo de finalización” está vacía. - En la columna “Tiempo total”: El tiempo que le tomo realizar la tarea. - En la columna “Tiempo estimado”: 1 minuto. - En la columna “Tiempo dif.EstimadoTotal”: Resultado de la resta entre tiempo total y el tiempo estimado.
Pregunta:	Se aprobó el caso de prueba funcional?
Respuesta:	Aprobado.
Descripción del defecto:	

12 GLOSARIO:

Adoptar. Recibir, haciéndolos propios, métodos, doctrinas, ideologías, modas, etc., que han sido creados por otras personas o comunidades.

Administración de Procesos de Negocio (BPM, por sus siglas en inglés). Es el enfoque que plantea que el ciclo de vida de los procesos debe consistir en una serie de etapas, estas son: Diseño, Modelado, Ejecución, Monitoreo y Optimización.

Administrador del Proceso. Es la categoría donde se agrupan el o los roles autorizados para crear Instancias de la Definición del Proceso y para modificar definiciones del proceso.

Analista de Proceso. Se encarga del diseño, modelado y optimización del proceso.

Analista Técnico. Es un desarrollador de software.

Aplicable. El grado en el que un objeto puede adaptarse a las circunstancias, condiciones o resultados esperados.

Calidad. Es el conjunto de características específicas permisibles en el producto.

Ciclo de Vida de un Proceso. Son las etapas por las que pasa un proceso desde que nace hasta que es desechado.

Defecto. Característica errónea del producto.

Definición del Proceso. Descripción del proceso que detalla lo que se hace en el proceso, quien lo hace, los materiales que se necesitan y que es lo que se produce.

Ejecución de un Proceso. Sinónimo de Instancia del Proceso.

Implementación de un Proceso. Cubrir los requerimientos para que el proceso pueda ser ejecutado.

Instancia del Proceso. LLevar a cabo las actividades y las tareas del proceso, involucrando todos los elementos de la Definición del Proceso

Mejorar un Proceso. Consiste en estudiar, documentar (definir y modelar), el proceso, medir sus resultados y encontrar soluciones más eficientes y eficaces.

Modelos de Mejora de Procesos para el Desarrollo del Software. Son modelos que integran un conjunto de procesos que guían a las organizaciones a implementar y adaptar a sus organizaciones, los procesos que reúnen las mejores prácticas en el desarrollo de software.

Monitoreo. Tomar valores de las características específicas del producto y/o proceso.

Notación. Una gráfica es una descripción que se representa por medio de figuras o signos. En las técnicas de modelado de procesos, al conjunto de estas figuras o signos y sus nombres, se llama “notación”. Cuando se modelan los procesos, la notación representa los elementos de la Definición del Proceso. También existen signos para expresar el orden secuencial de las tareas.

Participante del Proceso. Es la categoría donde se agrupan los distintos roles que intervienen en el proceso.

Proceso. Un proceso es un conjunto de actividades que se llevan a cabo para lograr un propósito.

Proceso de Desarrollo de Software. Son un conjunto de actividades, métodos, prácticas y modificaciones que las personas realizan para desarrollar y mantener productos de software.

Proceso de Negocio. Es un conjunto de actividades o tareas ordenadas para producir un determinado producto o servicio de utilidad para un cliente interno o externo de la organización.

Seguimiento de un Modelo de Mejora de Procesos. Poner en práctica lo que el modelo dice.

Software de Calidad. Productos de software libres de defectos que cumplen con sus requerimientos

Suite BPM. Es la herramienta que apoya el enfoque BPM en las etapas de: Modelado, Ejecución y Monitoreo.

13 Referencias

[1] Taz Daughtrey, *Fundamental Concepts for the Software Quality Engineer*, Editor American Society for Qualit, 2001. Pág. 4.

[2] Emanuel R. Baker, Matthew J. Fisher and WolfhartGoethert, *Basic Principles and Concepts for Achieving Quality*, Nota Técnica CMU/SEI-2007-TN-002, Editor Lisa Marino, December 2007. Pág. 11, Capítulos 1 y 2.

[3] Rosalind L. Ibrahim, Iraj Hirmanpour, *The Subject Matter of Process Improvement: A Topic and Reference Source for Software Engineering Educators and Trainers*, Technical Report CMU/SEI-95-TR-003 ESC-TR-95-003, May 1995. Págs. 15, 18.

[4] Thomas H. Davenport, *Process innovation: reengineering work through information technology*, Editor Harvard Business Press, 1993. Pág. 5.

[5] José Antonio Pérez Fernández de Velasco, *Gestión por procesos*, Editor ESIC, 2007. Págs. 83, 87.

[6] James R. Persse, PhD, *Process Improvement Essentials*, Editor O'Reilly, September 2006. Capítulos 1, 2 (sección 2.5.) y 6.

[7] Philippe Kruchten, *The rational unified process: an introduction*, Editor Addison-Wesley, 2004.

[8] John S. Oakland, *Statistical Process Control*, Editorial Butterworth Heinemann, Quinta edición 2003. Págs. 14, 17, 107, 153. Capítulos 1 y 2.

[9] W. Wayt Gibbs, *Software's Chorinic Crisis*, Artículo de la revista Scientific American, September 1994. Pág. 1.

[10] William A. Florac, Anita D. Carleton, *Measuring the software process: statiscal process control for software process improvement SEI series in software engineering* , Editor Addison-Wesley, 1999. Págs. 2, 3.

[11] Francisco J. Pino, Félix García, Mario Piattini*, Revisión sistemática de mejora de procesos software en micro, pequeñas y medianas empresas*,

Revista Española de Innovación, Calidad e Ingeniería del Software, Vol 2, No. 1, 2006. Pág. 7.

[12] José Antonio Heredia Álvaro, *Sistema de indicadores para la mejora y el control integrado de la calidad de los procesos*, Editor Universidad Jaume I, 2001. Pág. 44.

[13] Eduardo Andreu Alabarta, Rafael Martínez-Vilanova Martínez, *Cómo gestionar una PYME mediante el cuadro de mando*, Editor ESIC, 2007. Pág. 170.

[14] Ian Sommerville, *Ingenería de software*, Ediciones Pearson Addison Wesley, 2005. Págs. 109, 111, 608, 609.

[15] Real Academia Española, *Diccionario de la lengua española,* Volumen 1, Espasa calpe, Edición 22, 2001.

[16] Software Engineerig Institute, *Capability Maturity Model Integration Version 1.1*, Manual técnico, March 2002. Págs. 1, 11-13.

[17] Directora del proyecto Dra. Hanna Oktaba, *Modelos de Procesos para la Industria del Software (MoProSoft) Por niveles de capacidad de procesos*, Versión 1.3, Agosto 2005. Págs. 3, 10, 12.

[18] Centro de investigación en matemáticas (CIMAT), *Proyecto Tritón,* Version 1.2, Documento técnico, Mayo 2008.

[19] Object Management Group, *Business Process Modeling Notation Specification*, dtc/06-02-01, February 2006. Págs. 1, 15.

[20] Sinan Si Alhir, *Learning UML*, Editor O'Reilly Media, Inc, 2003. Págs. 4, 26.

[21] Lambert M. Surhone, Miriam T. Timpledon, Susan F. Marseken, *XPDL*, Editor VDM Verlag Dr. Mueller AG & Co. Kg, 2010.

[22] Matjaz B. Juric, Benny Mathew, Poornachandra Sarang, *Business Process Execution Language for Web Services*, Editor Packt Publishing Ltd, 2006.

[23] Watts S. Humphrey, *The Personal Software Process℠ (PSP℠)*, Technical report CMU/SEI-2000-TR-022 ESC-TR-2000-022, Noviembre 2000. Págs. ix, 9.

[24] http://www.eclipse.org/epf/, Página Web de Eclipse Process Framework.

[25] Matt Cumberlidge, *Business Process Management with JBoss jBPM*, Editor Packt, July 2007. Págs.6, 12, 13, 16, 54, 74, 75, 76, 77, 146, 149, 160. Capítulos 1, 2, 3, 4, 5 y 6.

[26] Arthur ter Hofstede, *Business Process Management Workshops: revised selected papers*, Editorial Springer, 2008. Pág. 273.

[27] Jean-noel Gillot, *The Complete Guide to Business Process Management: Business Process Transformation Or a Way of Aligning the Strategic Objectives of the Company and the Information System Through the Processes*, Editor Lulu.com, 2008. Pág. 92.

[28] http://www.intalio.com, Página Web de Intalio

[29] http://swik.net/bpm/BPM+bookmarks+from+del.icio.us/BONITA:+Open+Source+Workflow+%2F+BPM+Solution+XPDL+Open+Source+Workflow/ckwlh, Página Web de Bonita.

[30] http://www.jboss.org/jbossjbpm/, Página Web de JBoss JBPM.

[31] Len Bass, Paul Clements, Rick Kazman, *Software Architecture in Practice*, Editorial Addison Wesley, April 2003. Pág. 3, sección 4.3. Capítulo 1.

[32] RikardLand, *A BriefSurvey of Software Architecture*, Artículo del Departamento de Ingeniería de informática de la Universidad de Mâlardalen, Vâsteras, Suecia, Febrero 2002. Pág. 7.

[33] Frank Buschmann, Kevlin Henney and Douglas C. Schmidt, *Pattern-Oriented Architecture: A Pattern Language for Distributed Computing*, Editorial John Wiley and Sons, Volumen 4, 2007.

[34] Alfredo Weitzenfeld, *Ingeniería de software orientada a objetos con UML, Java e Internet*, Thomson, 2005. Pág. 235.

[35] Harver M. Deitel, Paul J. Deitel, Traducción Alfonso Vidal Romero, *Cómo programar en Java*, Editorial Pearson/Educación, 2004. Pág. 910.

[36] Hans-Erik Eriksson; et al, *UML 2 toolkit,* Editor Wiley Publishing, 2004. Págs. 21, 23, 24.

[37] Benjamin A. Lieberman, *The art of software modeling*, Editorial CRC Press, 2006. Pág. 144

[38] http://static.springsource.org/spring/docs/2.5.x/reference/mvc.html#mvc-introduction, Página Web de Spring MVC Framework.

[39] Rob Harrop, Jan Machacek, *Pro Spring*, Apress, 2005. Pág. 49.

[40] Andres Gomez De Silva Garza, Ignacio De Jesús Ania Briseño, *Introducción a la computación*, Cengage Learning Editores, 2008. Pág. 34.

[41] David Robinson, *Aspect-oriented programming with the e verification language: a pragmatic guide for testbench developers*, Editor Morgan Kaufmann, 2007.

[42] http://www.springsource.org, Página Web de Spring Framework.

[43] http://www.hibernate.org, Página Web de Hibernate.

[44] http://www.ant.apache.org, Página Web de ANT.

[45] http://svnkit.com, Página Web de SVNKit.

[46] http://struts.apache.org/, Página Web de Struts.

[47]http://www.oracle.com/technetwork/java/javase/install-windows-139179.html, Página Web de Oracle, documentos de instalación.

[48] http://docs.jboss.org/tools/3.1.0.GA/en/jboss_jbpm_ref_guide/html_single/, Página Web de documentos JBPM.

[49] http://help.eclipse.org/help33/index.jsp, Página Web de la documentación de Eclipse.

[50] http://ant.apache.org/manual/install.html, Página Web del manual de instalación de ANT.

[51] Michael Lang, Wita Wojtkowski, *Information Systems Development: Challenges in Practice, Theory and Education*, Editorial Springer, 2008. Pág. 239.

[52] In Lee, *E-business models, services, and communications Gale Refernce*, Editorial Idea Group Inc (IGI), 2008. Pág. 252.

UNIVERSIDAD AUTÓNOMA METROPOLITANA
UNIDAD IZTAPALAPA División de Ciencias Básicas e Ingeniería
Casa abierta al tiempo

"Desarrollo de una Suite BPM para el modelado, ejecución y monitoreo de los procesos de un Modelo de Mejora de Procesos de Desarrollo de Software"

Tesis que presenta:
C.P. y Lic. Silvia Nagheli Márquez Solís

Para obtener el grado de:
MAESTRA EN CIENCIAS
DEL
POSGRADO EN CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

Asesores:

Dr. Humberto Cervantes Maceda
Dr. Carlos Montes de Oca Vázquez

Jurado calificador:

Presidente: M. en C. Alfonso Martínez Martínez
Secretario: Dr. Humberto Cervantes Maceda
Vocal: M. en C. Mery Helen Pesantes Espinoza

MEXICO, D.F. JULIO 2011

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

ACTA DE EXAMEN DE GRADO

No. 00006
Matrícula: 207380924

DESARROLLO DE UNA SUITE BPM PARA EL MODELADO, EJECUCION Y MONITOREO DE LOS PROCESOS DE UN MODELO DE MEJORA DE PROCESOS DE DESARROLLO DE SOFTWARE.

En México, D.F., se presentaron a las 11:00 horas del día 25 del mes de julio del año 2011 en la Unidad Iztapalapa de la Universidad Autónoma Metropolitana, los suscritos miembros del jurado:

M. EN C. ALFONSO MARTINEZ MARTINEZ
M. EN C. MERY HELEN PESANTES ESPINOZA
DR. HUMBERTO GUSTAVO CERVANTES MACEDA

Bajo la Presidencia del primero y con carácter de Secretario el último, se reunieron para proceder al Examen de Grado cuya denominación aparece al margen, para la obtención del grado de:

MAESTRA EN CIENCIAS (CIENCIAS Y TECNOLOGIAS DE LA INFORMACION)

DE: SILVIA NAGHELI MARQUEZ SOLIS

y de acuerdo con el artículo 78 fracción III del Reglamento de Estudios Superiores de la Universidad Autónoma Metropolitana, los miembros del jurado resolvieron:

APROBAR

Acto continuo, el presidente del jurado comunicó a la interesada el resultado de la evaluación y, en caso aprobatorio, le fue tomada la protesta.

SILVIA NAGHELI MARQUEZ SOLIS
ALUMNA

REVISÓ

LIC. JULIO CESAR DE LARA ISASSI
DIRECTOR DE SISTEMAS ESCOLARES

DIRECTOR DE LA DIVISIÓN DE CBI

DR. JOSE ANTONIO DE LOS REYES HEREDIA

PRESIDENTE

M. EN C. ALFONSO MARTINEZ MARTINEZ

VOCAL

M. EN C. MERY HELEN PESANTES ESPINOZA

SECRETARIO

DR. HUMBERTO GUSTAVO CERVANTES MACEDA