INSTITUTO POLITÉCNICO NACIONAL

Secretaría de Investigación y Posgrado

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

IMPLEMENTACIÓN DE UN MOTOR DE WORKFLOW BASADO EN REDES DE PETRI Y PATRONES

T E S I S

QUE PARA OBTENER EL GRADO DE MAESTRO
EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A:

LIC. FELIX GONZÁLEZ DEL ANGEL

D I R E C T O R E S:

DR. AGUSTÍN FRANCISCO GUTIÉRREZ TORNÉS

M. EN C. JOSÉ RAFAEL CEN ZUBIETA

MÉXICO, D.F. MARZO 2007

SIP-14

INSTITUTO POLITECNICO NACIONAL

SECRETARIA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 18:00 horas del día 22 del mes de Marzo de 2007 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis de grado titulada:

"IMPLEMENTACIÓN DE UN MOTOR DE WORKFLOW BASADO EN REDES DE PETRI Y PATRONES"

GONZÁLEZ	DEL ÁNGEL	FELIX
Apellido paterno	materno	nombre(s)

Con registro: B 0 2 0 9 3 0

aspirante al grado de: ***MAESTRO EN CIENCIAS DE LA COMPUTACIÓN***

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Presidente

Dr. José de Jesús Medel Juárez

Secretario

M. en C. Rubén Peredo Valderrama

Primer vocal
(Director de Tesis)

Dr. Agustín Francisco Gutiérrez Tornés

Segundo vocal
(Director de Tesis)

M. en C. José Rafael Cen Zubieta

Tercer vocal

M. en C. César Saúl Guzmán Rentería

Suplente

M. en C. Sandra Dinora Orantes Jiménez

El Presidente del Colegio

Dr. Hugo César Coyote Estrada

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día once del mes de Enero del año 2008, el que suscribe Félix González del Ángel alumno del Programa de Maestría en Ciencias de la Computación con número de registro B020930, adscrito al Centro de Investigación en Computación, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del Dr. Agustín Francisco Gutiérrez Tornés y del M. en C. José Rafael Cen Zubieta y cede los derechos del trabajo intitulado Implementación de un Motor de Workflow Basado en Redes de Petri y Patrones, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección fgonzalemx@hotmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Félix Gnzález del Ángel

Agradecimientos

A mis directores de tesis, M. EN C. JOSÉ RAFAEL CEN ZUBIETA y al DR. AGUSTÍN FRANCISCO GUTIÉRREZ TORNÉS por sus consejos, apoyo en la realización de esta tesis.

A mis profesores y sinodales de tesis DR. JESÚS MEDEL JUÁREZ, M. EN C. SANDRA DINORA ORANTES JIMÉNEZ, M. EN C. RUBÉN PEREDO BALDERRAMA Y AL M. EN C. CESAR GUZMÁN RENTERÍA por sus enseñanzas y comentarios durante la revisión de esta tesis

Al Instituto Politécnico Nacional y al Centro de Investigación en Computación por haberme dado la oportunidad de seguir creciendo profesionalmente.

Dedicatoria

A mis Padres:
María Teresa y Pablo,
por sus consejos y ayuda incondicional.

A mis hermanos:
Yolanda,
Cipriano,
Carmen e
Isabel
por todo su apoyo.

A mis compañeros de clase que siempre me alentaron para seguir adelante.

ÍNDICE

RESUMEN ..... **v**

ABSTRACT ..... **vi**

GLOSARIO ..... **vii**

ÍNDICE ..... **ix**

LISTA DE FIGURAS ..... **xiii**

CAPITULO 1. INTRODUCCIÓN **1**

1.1. Motivación 2
1.2. Planteamiento del problema y justificación 3
1.3. Objetivos 3
1.3.1. Objetivo general *3*
1.3.2. Objetivos específicos *3*
1.4. Planteamiento de hipótesis 4
1.4.1. Hipótesis positiva 4
1.4.2. Hipótesis negativa 4
1.5. Alcances y límites 4
1.6. Organización de la tesis 4

CAPITULO 2. MARCO TEÓRICO **7**

2.1. Estado del Arte 7
2.1.1. Sistemas Manejadores de Worklfow de código abierto *7*
2.1.2. Sistemas Manejadores de Worklfow comerciales *8*
2.1.3. Estándares de Workflow *11*
2.2. ¿Qué es Workflow? 12
2.2.1. Funciones de construcción *13*
2.2.2. Funciones de control del proceso en tiempo de ejecución *14*
2.2.3. Interacción de las actividades en tiempo de ejecución *15*
2.2.4. Interfaces de distribución y del sistema *16*
2.3. Modelo de referencia de la Coalición de Manejadores de Workflow (Workflow Management Coalition (WfMC)) 17
2.3.1. Modelo propuesto por la WfMC *17*
2.4. Perspectivas de Workflow 18
2.5. Introducción a las redes de Petri 18
2.6. Redes de Petri clásicas 19
2.7. Definición formal 20
2.8. Redes de Petri de alto nivel 20
2.9. Mapeo de conceptos de Workflow en redes de Petri 21
2.9.1. El proceso *21*
2.9.2. Ruteo *21*
2.9.3. Enactment (Lanzamiento) *23*
2.10. Patrones de Workflow 23
2.10.1. Patrones de control de flujo básicos *23*
2.11. Lenguaje de modelado con redes de Petri (PNML) 24
2.12. Motor de redes de Petri 25

CAPITULO 3. Modelo a implementar **27**

3.1. Introducción 27

3.2. Meta-modelo básico ... 29

CAPITULO 4. Análisis y Diseño ... 31

4.1. Arquitectura general del sistema propuesto ... 31
4.2. Subsistema de Diseño y Lanzamiento de Procesos ... 31
4.2.1. Definición de la red de Petri WF-Net mediante XML ... 32
4.2.2. Diagrama de Contexto del Subsistema de Diseño y Lanzamiento de Procesos ... 32
4.2.3. Casos de Uso del Submódulo de Diseño y Lanzamiento de Procesos ... 33
4.2.4. Diagrama de Secuencia del Lanzamiento de Procesos ... 33
4.2.5. Lanzamiento de Procesos (Enactment) ... 34
4.2.6. Diagramas de Clase ... 35
4.2.7. Diagrama de Paquetes para el Subsistema de Diseño y Lanzamiento de Procesos ... 36
4.3. Subsistema de Seguimiento de Actvidades (Worklist) ... 36
4.3.1. Diagrama de contexto general del Subsistema de Seguimiento de Actividades ... 37
4.3.2. Diagramas de Clase ... 37
4.3.3. Diagrama de Actividades para la ejecución de una tarea ... 38
4.3.4. Diagramas de Secuencia ... 38
4.4. Diagrama de componentes ... 40
4.5. Diagrama de despliegue ... 41
4.6. Diagrama Entidad Relación ... 41
4.7. Implementación ... 42
4.7.1. Contenedor de Servlets ... 42
4.7.2. Servlets ... 43

CAPITULO 5. Pruebas y Resultados ... 45

5.1. Pruebas al Subsistema de Diseño y Lanzamiento de Procesos ... 45
5.1.1. Descripción del Proceso de Negocio a automatizar ... 45
5.1.2. Diseño del proceso con redes de Petri ... 46
5.2. Pruebas al Subsistema de Seguimiento de Actividades ... 49
5.2.1. Pruebas de Integración del Subsistema de Diseño y Lanzamiento de Procesos y del Subsistema de Seguimiento de Procesos ... 49
5.2.2. Pruebas al Subsistema de Seguimiento de Actividades ... 50
5.2.3. Pruebas de Rendimiento del Motor de Workflow ... 51
5.3. Resultados ... 51
5.4. Comparativo con otros productos ... 52

CAPITULO 6. Conclusiones ... 53

6.1. Conclusiones ... 53
6.2. Aportaciones ... 53
6.3. Trabajos futuros ... 54

Bibliografía ... 55

Referencias bibliográficas ... 55
Páginas electrónicas ... 57

Apéndice A - Listado de código en PNML ... 59

Apéndice B - Código Escrito utilizando PNK ... 60

Apéndice C - WfNet.xml ... 61

Apéndice D - Manual de Usuario ... 62

Manual de usuario del Subsistema de Diseño y Lanzamiento de Procesos ... 62
Ejecución del proceso desde el Subsistema de Lanzamiento de Procesos ... 62
Manual de usuario del Subsistema de Diseño y Lanzamiento de Procesos ... 63
Seguimiento a las actividades del proceso ... 63

Apéndice E - Diccionario de datos..... 65

Apéndice F - Código del Subsistema de Lanzamiento de Procesos (Motor Workflow) 67

Apéndice G - Código del Subsistema de Seguimiento de Actividades 80

LISTA DE FIGURAS

Figura 1. Esquema general de un Workflow 2
Figura 2. BPMS enfocados a tareas ejecutadas por personas 9
Figura 3. Características de un sistema de Workflow 14
Figura 4. Distribución dentro del Workflow Enactment Service 16
Figura 5. Modelo de referencia del Workflow (Componentes e Interfaces) 17
Figura 6. Red de Petri clásica 19
Figura 7. Red de Petri indicando que hay un caso en revisión 20
Figura 8. Ruteo Secuencial 21
Figura 9. Ruteo Paralelo 22
Figura 10. Ruteo Condicional 22
Figura 11. Ruteo Iterativo 23
Figura 12. Proceso Secuencial 25
Figura 13. Modelo conceptual del núcleo de redes de Petri PNK 26
Figura 14. Estructura genérica de un producto Workflow 27
Figura 15. Arquitectura física del sistema propuesto 29
Figura 16. Diagrama general de la base de datos para un Sistema Manejador de Workflow 29
Figura 17. Caso de Uso general del Submódulo de Diseño y Lanzamiento de Procesos . 32
Figura 18. Casos de Uso del Submódulo de Diseño y Lanzamiento de Procesos 33
Figura 19. Diagrama de Secuencia del Submódulo de Lanzamiento de Procesos 34
Figura 20. Diagrama de general de Actividades del Submódulo de Lanzamiento de Procesos 34
Figura 21. Diagrama de clases para el Submódulo de Lanzamiento de Procesos (Clases de control) 35
Figura 22. Diagrama de clases para el Submódulo de Lanzamiento de Procesos (Clases de entidad) 35
Figura 23. Diagrama de clases para el Submódulo de Lanzamiento de Procesos 36
Figura 24. Diagrama de contexto de Submódulo de Seguimiento de Actividades 37
Figura 25. Diagrama de clases del Submódulo de Seguimiento de Actividades 37
Figura 26. Diagrama de actividades de la ejecución de una tarea 38
Figura 27. Diagrama de secuencia del caso de uso Ingresar al Sistema 38
Figura 28. Diagrama de secuencia del caso de uso Actualizar Lista de Trabajo 39
Figura 29. Diagrama de secuencia del caso de uso Seleccionar Elemento de Trabajo 39
Figura 30. Diagrama de secuencia del caso de uso Ejecutar Actividad 40
Figura 31. Diagrama de componentes 40
Figura 32. Diagrama de despliegue 41
Figura 33. Diagrama Entidad-Relación 42
Figura 34. Flujo de información entre el cliente, el Servidor Web y el Contenedor e Servlets 43
Figura 35. Proceso para el pago de reclamaciones para seguros de autos 47
Figura 36. Proceso para el pago de reclamaciones para seguros de autos diseñado mediante el sistema 48
Figura 37. Resultados obtenidos después de haber lanzado el proceso 49
Figura 38. Worklist el usuario. 50
Figura 39. Tiempos de respuesta del sistema 51

Figura 40. Opción para abrir archivo 62
Figura 41. Navegador de archivos mostrando procesos disponibles 62
Figura 42. Opción para lanzar proceso 62
Figura 43. Opción que da inicio al lanzamiento de proceso 63
Figura 44. Página de inicio para el ingreso al sistema de seguimiento de actividades 63
Figura 45. Página del administrador de elementos de trabajo 64
Figura 46. Página que muestra el detalle de la actividad seleccionada 64

RESUMEN

En la última década se han usado varios conceptos y tecnología referente a Workflow en diversos sistemas de información. Aunque no es una tecnología nueva, no es sino hasta ahora que se ha retomado, gracias a la aparición redes de datos más rápidas, y sobretodo, a la importancia que se le ha venido dando a los procesos de negocio como parte fundamental de las aplicaciones.

El propósito general de este trabajo, es la implementación de un motor de Workflow en Java. Java es un lenguaje portable; esto significa que podemos compilar un programa en Java en cualquier sistema y correr el ejecutable en el mismo sistema o en cualquier otro, ya sea Macintosh, Windows, Unix o Linux. También es un lenguaje orientado a objetos. Permite representar objetos de la vida real a través de clases. Además es un lenguaje interpretado; el intérprete de Java es conocido como la máquina virtual de Java, y es la encargada de interpretar y ejecutar los programas.

Se desarrollan dos aplicaciones, una para el diseño, simulación y ejecución de procesos y otra para el seguimiento de las actividades de dichos procesos. Se utilizan redes de Petri y patrones básicos de Workflow para el desarrollo de la aplicación de modelado y Servlets para dar seguimiento a las actividades de un proceso.

ABSTRACT

During the last decade Workflow concepts and technologies have been applied in many information systems. Although this is not a new technology, now it has been retaken, thanks to the existence of faster computer networks and organizations which are aware of the importance of business processes in applications.

The main goal of this thesis is to build a Workflow engine using Java. Java is a portable programming language; that means that we can compile a Java program on any system and run the resulting binary executable file on the same o any other system, on Macintosh, on Windows, on Unix, and on Linux. Also Java is an object oriented programming language. It permits us to represent real objects through classes. Besides, Java is an interpreted language, we need the Java Virtual Machine to interpret and run an application.

Two applications will be developed, one for the design, simulation and execution of a process; and other one to track its activities. To develop the design application, Petri nets will be used and basic Workflow patterns, as well as Servlets to track the activities of a process.

CAPITULO 1. INTRODUCCIÓN

La aparición de nuevas tecnologías y redes informáticas más rápidas, han hecho posible que la distribución del trabajo sea una realidad. Sin embargo para llegar hasta este punto, la forma en la que se han venido desarrollando los sistemas de información también ha evolucionado [1].

Inicialmente se ubicaron los sistemas operativos y las aplicaciones que corrían sobre ellos. El problema era como desligar las aplicaciones de los sistemas operativos. Posteriormente los sistemas manejadores de bases de datos hicieron su aparición, y con ello se logro separar los datos de la aplicación. Mas tarde la atención se centro en las interfaces de usuario, donde se buscaba separar la interfaz de usuario de las aplicaciones. Finalmente aparecieron los Sistemas de Manejadores de Workflow, que buscan separar los procesos de negocio de las aplicaciones.

Los procesos de negocio son la parte medular de cualquier sistema de información. Si son identificados y delimitados adecuadamente permiten acelerar el desarrollo de los sistemas de información de la organización, además facilitan el mantenimiento de los procesos de negocio.

Si la administración de una organización desea hacer cambios en los procesos de negocio implica invertir en tiempo y costo, sin contar las posibles repercusiones que tenga este cambio dentro del sistema. Como resultado de esto, el cambio no se realiza. Los Sistemas de Manejadores de Workflow resuelven este problema. Un Sistema Manejador de Workflow administra los procesos de negocio, y distribuye, entre los recursos humanos y aplicaciones, los datos de cada uno los casos que se presenten.

Los Sistemas de Manejadores de Workflow no son un tema nuevo [9]. La idea de tener herramientas genéricas para soportar procesos de negocio surgió en los años 70 con pioneros como Skip Ellis [2] y Michael Zisman [3], sin embargo no tuvieron el éxito deseado debido principalmente a que en esa época no se contaba con la infraestructura de red adecuada. Fue hasta los años 90 que las redes de datos se volvieron más sólidas. Adicionalmente no se tenían en mente sistemas de información que contemplaran a los procesos de negocio como una parte fundamental del sistema, ni las organizaciones les daban la importancia que merecían.

Es hasta ahora que se ha alcanzado, con los Sistemas de Manejadores de Workflow [1, 10,14, 16], un punto en donde primero se diseña el proceso de negocio de la manera más abstracta, sin considerar la implementación, y entonces diseñamos el sistema de información junto con los recursos de la organización. De hecho es cuando decidimos si las tareas de un proceso deben ser realizadas por un sistema de información o por una persona.
Los Sistemas de Manejadores de Workflow aseguran que la información correcta llegue a la persona correcta en el tiempo correcto, o que sea enviada a la aplicación correcta en el

momento correcto.

Es importante hacer notar que un Sistema Manejador de Workflow no realiza ninguna tarea de un proceso. Precisamente en esto radica su fortaleza. Es un software genérico y se puede utilizar en muchas situaciones. Su debilidad por otro lado, radica en que se hacen necesarias aplicaciones adicionales.

La Figura 1 muestra el contexto de un Sistema Manejador de Workflow.

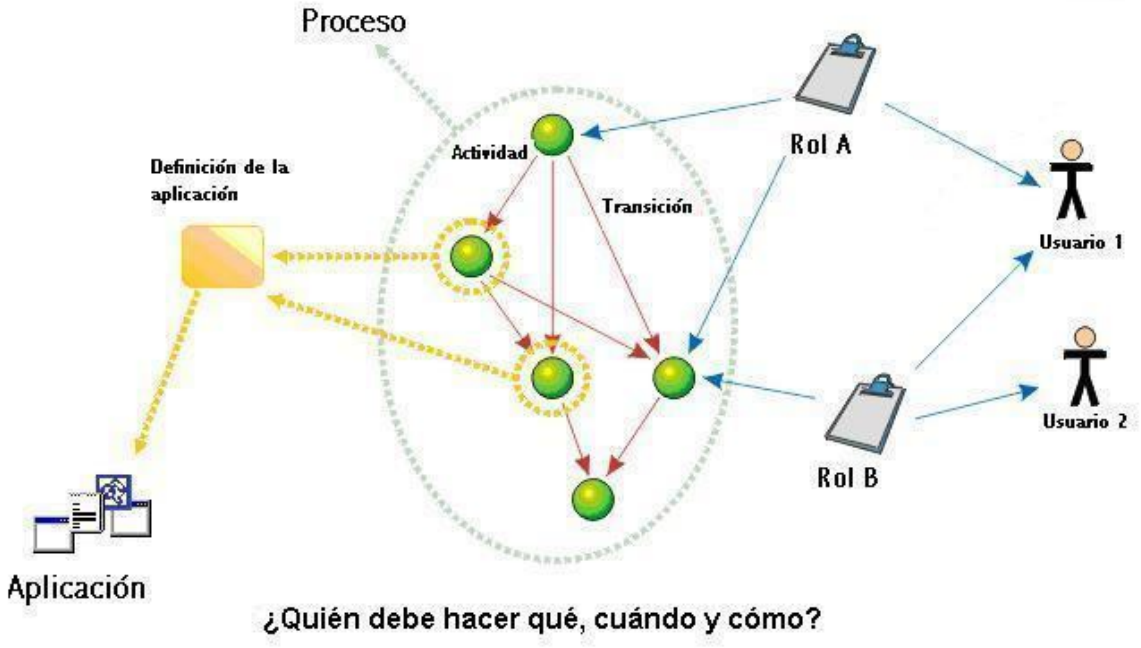


Figura 1. Esquema general de un Workflow

1.1. Motivación

Las sociedades modernas se han vuelto tan complejas que pocas veces se conoce el rol que a cada quien le toca desempeñar en su trabajo. Incluso en las grandes compañías, donde hay un gran nivel de especialización en el trabajo, resulta que muchos empleados no lo realizan como debe ser o no hacen lo que deben de hacer, ocasionando una disminución de la productividad. Es por esto que en muchas compañías se está organizando el trabajo de tal forma que sus empleados sepan para quién están trabajando, con tal de mejorar los niveles de satisfacción de sus clientes.

Para llevar un control del trabajo y asegurar la comunicación entre los empleados, los Sistemas de Manejadores de Workflow han aparecido, dando lugar a un nuevo tipo de sistema de información, el cual construye un puente entre el trabajo de los empleados y las aplicaciones

1.2. Planteamiento del problema y justificación

Al realizar un nuevo sistema de información, pocas veces se abstraen los procesos de negocio de la aplicación, es decir, no se considera la parte que da soporte a las aplicaciones.
Si se toma como premisa que los procesos de negocio son la base fundamental de los sistemas de información, se requiere de una herramienta que permita diseñarlos, documentarlos, probarlos y ejecutarlos.

Si se expresa de forma clara la lógica y los procesos del negocio a través de un Sistema Manejador de Workflow, se puede conocer el comportamiento del sistema bajo diferentes circunstancias.

Estas herramientas están disponibles en el mercado, se trata de los Sistemas Manejadores de Workflow, sin embargo, su alto costo los hace inaccesibles para la mayoría de las organizaciones.
Aunque también las hay de software libre, la mayoría esta basada en bibliotecas que requieren programación adicional para que pueda incluirse funcionalidad Workflow en los sistemas de información.
El componente principal de los Sistemas Manejadores de Workflow es su motor, sin él solo podríamos diseñar y documentar procesos de negocio, en otras palabras, solo sería una base de datos de procesos.

Este trabajo tiene como justificación el requerimiento de una herramienta de bajo costo que permita diseñar, documentar, probar y ejecutar procesos de negocio, además de acelerar el desarrollo de sistemas y reducir el costo de mantenimiento de los mismos.

1.3. Objetivos

1.3.1. Objetivo general

Implementar un motor de Workflow en Java con la funcionalidad para generar nuevas instancias de procesos, disparar sus actividades y asignarle recursos, así como darle seguimiento a los procesos a través de un administrador de actividades. El motor de Workflow se basará en redes de Petri y en patrones de control de Workflow básicos: el patrón de actividades secuenciales, el patrón de actividades paralelas compuesto por un AND-Split y un AND-Join, y el patrón de decisión compuesto por un OR-Split y un OR-Join. Estos patrones son propuestos en [6].

1.3.2. Objetivos específicos

- Extender el kernel de redes de Petri PNK [L-1], herramienta que permite modelar y simular procesos, para que sea capaz de soportar y ejecutar el tipo de redes de Petri de Workflow Wf-Nets [13], e implementar el motor de Workflow.

- Implementar un Worklist (Administrador de actividades) para el seguimiento de los procesos

1.4. Planteamiento de hipótesis

1.4.1. Hipótesis positiva

Los procesos de negocio son la parte fundamental de los sistemas de información y es posible construir una herramienta que permita diseñarlos, simularlos y ejecutarlos.

1.4.2. Hipótesis negativa

Los procesos de negocio son la parte fundamental de los sistemas de información pero no es posible construir una herramienta que permita diseñarlos, simularlos y ejecutarlos.

1.5. Alcances y límites

El presente trabajo busca implementar un motor de Workflow que provea de los mecanismos necesarios para el lanzamiento y seguimiento de procesos. Para ello se desarrollarán dos sistemas: El Sistema de Lanzamiento de Procesos que permitirá la creación de nuevas instancias de procesos, el disparo de sus actividades y la asignación de recursos, a su vez, el Sistema de Seguimiento de Actividades permitirá a los usuarios, a través de un Worklist basado en Web, consultar sus actividades pendientes.

El Sistema de Seguimiento de Actividades se basará en Web para permitir a los usuarios consultar sus pendientes a través de cualquier computadora que tenga acceso a Internet.

El motor de Workflow no contempla la ejecución de software adicional durante la ejecución de las actividades del proceso, ya que solo se podrán definir actividades manuales y no automáticas. Y solo podrá ejecutar procesos que incluyan patrones básicos de control de Workflow debido a que forma parte del objetivo general de esta tesis.

No se contará con una herramienta de administración, propia de los Sistemas Manejadores de Workflow, debido a que esta no es requerida para probar la funcionalidad del motor de Workflow.

1.6. Organización de la tesis

El capítulo 1 presenta un panorama de la evolución de los sistemas de información, comenzando por los sistemas operativos hasta llegar a los Sistemas Manejadores de Workflow. El capítulo 2 contiene el marco teórico, donde inicialmente, se presenta el estado del arte de los Workflows para posteriormente mencionar aspectos importantes de de la tecnología Workflow como son el modelo de referencia propuesto por la Coalición

de Administración de Workflows (WfMC, por sus siglas en inglés), también se mencionan algunos lenguajes y estándares empleados actualmente para la descripción de procesos. El capítulo 3 contiene el modelo propuesto para la implementación del motor de Workflow El capítulo 4 contiene el análisis y diseño del sistema. El capítulo 5 contiene información referente a las pruebas y resultados, y finalmente el capítulo 6 contiene las conclusiones al presente trabajo.

CAPITULO 2. MARCO TEÓRICO

2.1. Estado del Arte

Los Sistemas de Manejadores de Workflow pueden presentarse en varias arquitecturas [7]. Los Sistemas de Manejadores de Workflow embebidos son definidos como sistemas que proveen una integración muy estrecha con otras aplicaciones, por ejemplo con aplicaciones para el manejo de documentos. Un tipo de Sistemas de Manejadores de Workflow embebido es aquel, que está construido sobre sistemas groupware, por ejemplo Lotus Notes[L-2]. Otro tipo son aquellos que están embebidos en sistemas ERP, tales como SAP[L-3], People Soft[L-4].

De acuerdo a [4] un Sistema Manejador de Workflow crea y maneja la ejecución de Workflows a través del uso del software. Hay un gran de número de Sistemas Manejadores de Workflow que van desde prototipos de investigación, pasando por los de código abierto hasta los productos comerciales. Más de una veintena de proyectos de Workflow (Sistemas Manejadores de Workflow Autónomos) están listados en [L-5].

2.1.1. Sistemas Manejadores de Worklfow de código abierto

Dos de ellos, de código abierto, merecen más atención debido a que el primero, el Yet Another Workflow Language [5] (YAWL) se inicio, y continua, como proyecto de investigación sobre patrones de Workflow [6]. En [L-6] se puede ver una lista actualizada de patrones de Workflow. El segundo, el JBoss Java Business Process Managment 2.0 [L-7] (JBoss jBPM) es un sistema flexible y extensible programado en Java, el cual es parte importante de JBoss [L-8], el cual es el servidor de aplicaciones de software libre mas utilizado [17] para aplicaciones J2EE (Java 2 Enterprise Edition, Java 2 Edición Empresarial) [L-9], y por ende su distribución e integración es transparente.

En [L-5] podemos ver un resumen de Sistemas Manejadores de Workflow comerciales y prototipos disponibles. Aunque dado el dinamismo que han tenido los Workflows, esta lista no se encuentra actualizada, sin embargo muestra un cronograma general de los productos disponibles.

2.1.1.1. YAWL

El YAWL surge en respuesta a las limitaciones de los Sistemas Manejadores de Workflow y lenguajes de Workflow para modelar patrones de Workflow existentes descritos en [6]. Una evaluación de varios produtos de Workflow y estándares relevantes (XPDL, BPML, BPEL4WS) mostró que hay diferencias considerables en su habilidad para capturar flujos de control para procesos de Workflow no triviales. También modelos teóricos como las redes de Petri de alto nivel tienen problemas para soportar algunos patrones. Esto originó que se desarrollara un nuevo lenguaje, el cual provee soporte directo para los patrones de workfow identificados en [5]. Para probar el concepto, fue

desarrollado un Sistema Manejador de Workflow basado en YAWL [8] el cual está disponible en [L-10, L-11].

YAWL está basado en redes de Petri, pero las extiende con algunas características para facilitar la implementación de patrones que involucran múltiples instancias, sincronización avanzada y cancelación. YAWL es de hecho, un nuevo lenguaje con su propia semántica y específicamente diseñado para Workflows.

2.1.1.2. jBPM

El objetivo de jBMP es proveer un Sistema Manejador de Workflow en Java que sea flexible y extensible. Está ideado para proveer de un mecanismo muy simple para empezar con una máquina de estados simple, que permite a los desarrolladores en Java agregar jBPM en sus proyectos. Por otro lado debiera escalar a los procesos de Workflow más complejos y patrones de Workflow. En Octubre de 2004 el proyecto jBPM unió sus esfuerzos con JBoss para que fuera una pieza crítica de su Plataforma de Middleware Empresarial.

En la documentación de jBPM el concepto de actividad es remplazado por un estado y una acción. Un estado en un proceso especifica una dependencia con un actor externo. En tiempo de ejecución del proceso, significa que el motor de Workflow tiene que esperar hasta que el actor notifique al Sistema Manejador de Workflow que el estado ha terminado. Una acción es una pieza de código que debe ser ejecutada por el Sistema Manejador de Worflow a través de un evento especificado que ocurre durante la ejecución del proceso. El Sistema Manejador de Workflow inicia la ejecución de la acción en un evento especificado durante la ejecución del proceso.

El Java Process Definition Language (jPDL) permite agregar una declaración del proceso de negocio a un servidor JBoss. El diseñador de Workflow tambien debe anexar la lógica del programa a través de un conjunto de clases en Java. Los autores aseguran que durante el diseño de jBPM se tuvo en cuenta los patrones de Workflow. El modelo de estados de jBPM está basado en grafos con nodos y transiciones. Estos son los elementos principales en la definición de un proceso. Un estado es un ejemplo de un nodo. El grafo de estado, provee la estructura del proceso. Las acciones son piezas de código que pueden ser ejecutadas por eventos en los procesos.

2.1.2. Sistemas Manejadores de Worklfow comerciales

Actualmente existe en el mercado una gran variedad de productos comerciales, la mayoría de ellos brinda soporte para BPEL (Bussines Process Execution Language, Lenguaje de Ejecución de Procesos de Negocio). Adicionalmente, un nuevo paradigma ha venido cobrando importancia en la integración de servicios. La Arquitectura Orientada a Servicios (SOA, por sus siglas en inglés) busca organizar y optimizar los procesos de negocio a través de la implementación de Servicios Web que hagan disponible el proceso en forma interna o externa a la organización.

Los Sistemas Manejadores de Procesos de Negocio (BPMS, por sus siglas en inglés) tienen una funcionalidad similar a los Sistemas Manejadores de Workflow. En ambos es posible diseñar probar y usar procesos de negocio. Sin embargo, en los BPMS el grado de automatización del proceso es mayor, es decir, las tareas que conforman el proceso que se automatiza son realizadas en gran medida por sistemas, y no por humanos como sucede con los Sistemas Manejadores de Workflow. Adicionalmente en los BPMS existe cierto grado de especialización para el tipo de procesos que automatizan. Algunos ejemplos de especialización de BPMS son: los enfocados a manejo de documentos, los enfocados a procesos y los enfocados a tareas ejecutadas por personas.
La Figura 2, tomada de [23], muestra algunos BPMS enfocados a tareas ejecutadas por personas. Estos guardan gran similitud con lo Sistemas Manejadores de Workflow.

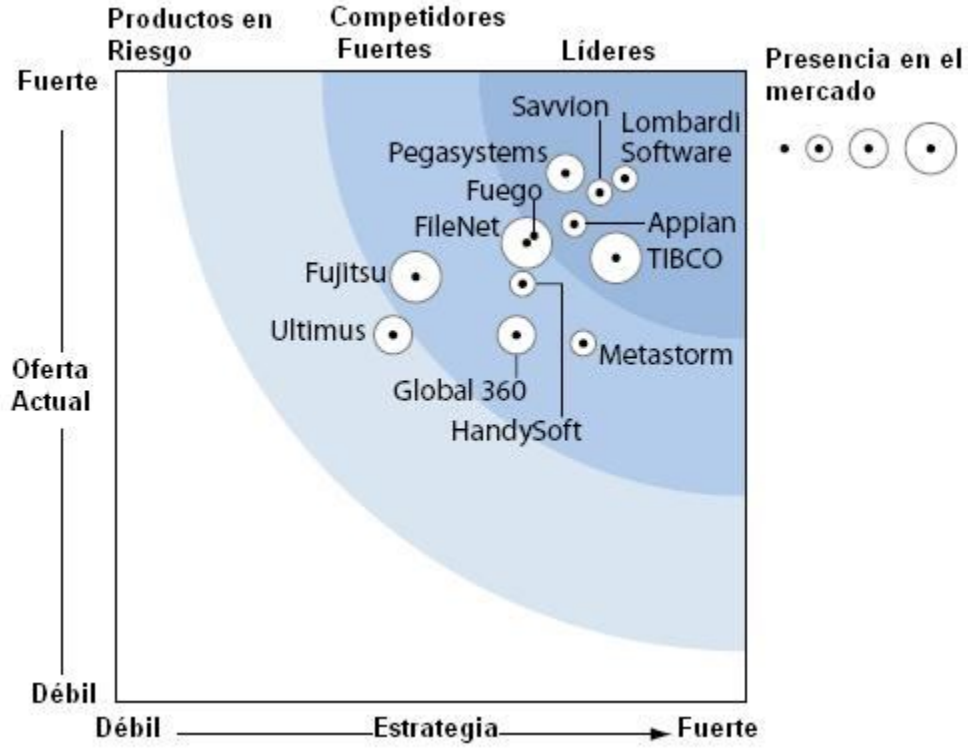


Figura 2. BPMS enfocados a tareas ejecutadas por personas

2.1.2.1. FileNET

Es una herramienta que permite definir, administrar y ejecutar los procesos de negocio a través de una interfaz 100% web. Permite definir procesos dentro de archivos HTML o en Applets. Los procesos se pueden habilitar como Servicios Web para que sean accesibles por otras aplicaciones. También permite simular los procesos antes de que se ejecuten e integra un manejador de documentos.

2.1.2.2. Fujitsu I-Flow

Es una herramienta con una interfaz sencilla para la definición de procesos. Permite modificar los procesos durante el tiempo de ejecución. Las condiciones y expresiones son expresadas como reglas del negocio a través de su propio motor de reglas.
Está totalmente escrito en Java, lo que le permite disponer de un conjunto de bibliotecas (API) para su integración con otras aplicaciones. Soporta SOA.

2.1.2.3. COSA Workflow

Es una herramienta sofisticada que permite definir procesos de negocio complejos. Integra un mecanismo poderoso para definir actividades temporizadas a través de calendarios. Se pueden definir recursos de forma compleja a través de organizaciones, grupos, proyectos y roles. Incluye un manejador de actividades (Worklist). Permite su integración con Lotus Notes. Incluye una API que permite integrarlo con otras aplicaciones.

2.1.2.4. IBM WebSphere

Es un producto Workflow que permite su integración con otras aplicaciones. Los procesos definidos con la herramienta se pueden reutilizar. Permite definir procesos inmunes a cambios organizacionales. Incluye herramientas para el análisis y simulación de procesos. Una desventaja de esta herramienta es que depende de otras aplicaciones IBM, lo que la hace dependiente de otros productos de esta compañía.

2.1.2.5. SAP WebFlow

Permite la automatización de procesos que incluyan aplicaciones desarrolladas en SAP. Incluye herramientas para generar formularios en HTML. Dado que es un Workflow embebido, no permite diseñar Workflows fuera de la herramienta.

2.1.2.6. Staffware

Es una herramienta con una excelente funcionalidad administrativa. Incluye un conjunto de procesos administrativos comunes que pueden utilizarse para generar prototipos de forma rápida. Permite su integración con Lotus Notes y Microsoft Exchange.
Incluye su propio lenguaje de scripts y un conjunto de herramientas orientada a objetos que permite integrarlo con otros lenguajes como Visual Basic, Java y tecnologías Microsoft .Net.

2.1.2.7. Fuego

Una de las fortalezas de Fuego radica en su plataforma de desarrollo orientada a procesos y en su cobertura para el diseño de procesos orientados a personas o sistemas. Además, Fuego provee una forma avanzada de organizar y conectar objetos desarrollados en Java, así como un ambiente de desarrollo amigable. Otra característica que lo diferencia de otros productos es la integración de su ambiente de diseño de procesos con Eclipse[L-20] Quizá una de las limitantes de su modelo de desarrollo basado en Java, hace que se vuelva una herramienta compleja para desarrollar procesos simples.

2.1.2.8. Tibco: In-Concert

Es un producto ideal para procesos de negocio con cambios constantes en tiempo de ejecución. Integra un mecanismo para el manejo de excepciones, en los procesos, a través de eventos. Provee una API orientada a objetos lo que permite integrarlo con otros lenguajes. Además, soporta el manejo de documentos.

2.1.2.9. Appian

Su especialidad es en Inteligencia de Negocios (BI, por sus siglas en inglés), así como en la gran cantidad de usuarios permitidos. Su producto es el más fácil para construir procesos que incluyan trabajo humano, análisis y herramientas de colaboración. También es extremadamente poderoso para la administración de usuarios.

2.1.2.10. Lombardi Software

Su fortaleza radica en el conjunto de herramientas para modelar procesos basados en Sistemas como Servicios. Además de incluir herramientas para el análisis de procesos a través de simulaciones basadas en datos.

2.1.2.11. Pegasystems

Su herencia de reglas de negocio lo hace un producto único, aunque suena como una pequeña diferencia contra otros BPMS, ésta es en realidad una gran ventaja. Los desarrolladores que diseñan soluciones de forma adecuada con está herramienta, se benefician de un producto con una arquitectura poderosa y flexible.

2.1.2.12. Savvion

Permite crear, rápidamente y a través de su ambiente de desarrollo unificado, soluciones poderosas y escalables para todo tipo de procesos. Maneja procesos de manufactura y de tecnología que otros productos no soportan.

2.1.3. Estándares de Workflow

La diversidad de técnicas para modelar procesos de negocio es un problema notorio en los Sistemas Manejadores de Workflow y en los BPMS. Aunque la estandarización se ha discutido por más de 10 años, sigue haciendo falta un formato de intercambio común [20]. La razón del porque el intercambio sigue siendo un problema no puede ser atribuido por lo menos a las diferentes perspectivas que tienen los analistas y diseñadores de procesos de negocio [21].

Recientemente, el modelado de procesos de negocio ha sido sujeto de varios esfuerzos de estandarización. Diferentes organizaciones incluyendo el Grupo para el Manejo de Objetos (OMG, por sus siglas en inglés) [L-14], la Organización para el Avance de Estándares de Información Estructurados (OASIS, por sus siglas en inglés) [L-15], la Iniciativa de Manejo de Procesos de Negocio (BPMI, por sus siglas en inglés) [L-16], el Centro de las Naciones Unidas para la Facilitación del Intercambio Comercial y Comercio Electrónico (UN/CEFACT, por siglas en inglés) [L-17], el Consejo para la

Telaraña Mundial de la Información (W3C, por su acrónimo) [L-18], y la Coalición para el Manejo de Workflows (WfMC, por sus siglas en inglés) [L-19], así como compañías de software y grupos de académicos han propuesto metamodelos y formatos de intercambio para el modelado de procesos de negocio.

Actualmente la Notación Para Procesos de Negocio (BPMN por sus siglas en inglés), una propuesta de BPMI y de la OMG, es el mejor candidato para establecerse como estándar. Ahora que la OMG controla el BMPN, se espera que haya un esfuerzo para combinarlo con UML [24] y sus diagramas de actividades. De hecho en su forma más simple BPMN y los diagramas de actividades de UML pueden ser usados por la gente que diseñe procesos de negocio.

Finalmente la Arquitectura de Sistemas de Información Integrados (ARIS, por sus siglas en inglés) [25] es una herramienta y notación para diagramar sistemas ERP. Es usado por SAP y ha sido adoptado por Oracle. En su forma de ERP, es el único lenguaje que entienden los desarrolladores de software.

2.2. ¿Qué es Workflow?

La tecnología Workflow está vinculada con la automatización de aquellos procedimientos donde las tareas son ejecutadas entre participantes de acuerdo a un conjunto definido de reglas para alcanzar, o contribuir con las metas de la empresa. Un Sistema Manejador de Workflow es aquel que define la automatización de un proceso de negocio, total o parcialmente [1].

Un Workflow puede ser organizado manualmente, aunque en la práctica la mayoría de los Workflow están organizados normalmente dentro del contexto tecnológico para proveerlo del soporte de tecnologías de información y computadoras para la automatización de sus procedimientos.

Los sistemas de Workflow son frecuentemente asociados con la Reingeniería de Procesos. Ésta está compuesta de análisis, modelado, definición y posteriormente la implementación operacional de los procesos de negocio medulares de la organización. Sin embargo no todas las actividades de la Reingeniería de Procesos resultan en una implementación de Workflow. La tecnología de Workflow es frecuentemente una solución apropiada que provee la separación de la lógica de los procesos de la empresa y su soporte operacional de la tecnología de información, permitiendo que los cambios subsecuentes, sean incorporados dentro de la definición de las reglas definidas en los procesos de la empresa.

Un Sistema Manejador de Workflow es aquel que provee la automatización de un proceso de negocio a través del manejo de las actividades subsecuentes y la invocación de un recurso, humano o tecnológico apropiado, asociado a cada actividad del proceso. Su función principal, es la de administrar las tareas de los procesos y controlar su orden de ejecución, definida previamente durante el diseño del proceso.

En general, todos los Sistemas de Manejadores de Workflow deben tener características para proveer de soporte en 3 áreas funcionales:

- Funciones de construcción: constituido por la definición, y posiblemente el modelado, del proceso del Workflow y sus actividades componentes,
- Funciones de control del proceso en tiempo de ejecución: se encarga del manejo de los procesos del Workflow en tiempo de ejecución en un ambiente operacional y del manejo de las actividades de cada proceso, e
- Interacciones en tiempo de ejecución: se encarga de comunicar a los usuarios con las aplicaciones necesarias para completar las actividades.

2.2.1. Funciones de construcción

Las funciones de construcción son aquellas que dan como resultado una definición automatizada de un proceso de negocio. Durante esta fase, un proceso de negocio es trasladado desde el mundo real a una definición formal, procesable mediante una o más, técnicas de modelado. El resultado es algunas veces llamado modelado del proceso, una plantilla del proceso, metadatos del proceso, o una definición del proceso.

Una definición de proceso normalmente comprende un número de actividades con operaciones de computadora y/o humanas y reglas que gobiernan el avance del proceso a través de sus actividades. La definición del proceso puede ser expresada en forma gráfica, textual o una notación de lenguaje formal. Algunos sistemas de Workflow pueden permitir alteraciones dinámicas a la definición del proceso durante el tiempo de ejecución, como se ve en la flecha de ida y vuelta en la Figura 3.

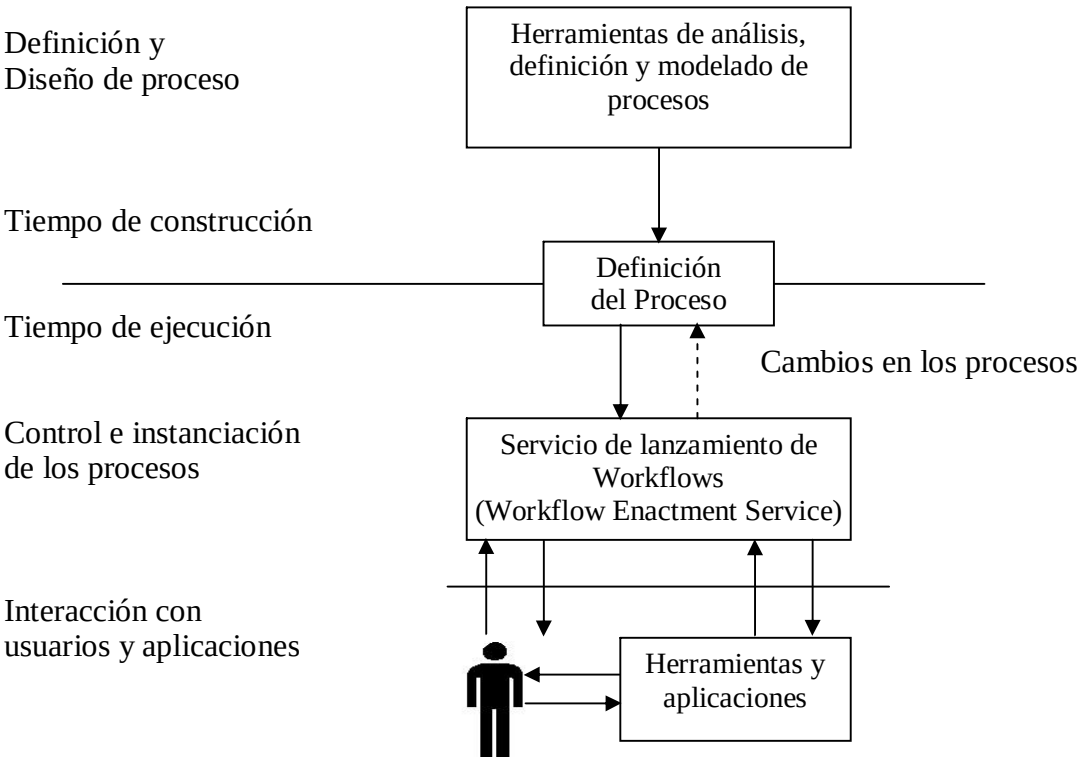


Figura 3. Características de un sistema de Workflow

2.2.2. Funciones de control del proceso en tiempo de ejecución

En tiempo de ejecución la definición de procesos es interpretada por el software el cual es responsable de crear y controlar la operación de las instancias de los procesos, la secuencia de las actividades dentro del proceso e invocar los recursos humanos y tecnológicos (aplicaciones) apropiadas, etc. Éstas funciones de control de procesos en tiempo de ejecución actúan como el vínculo entre el modelado del proceso con el mundo real, reflejado en las interacciones de los usuarios y las aplicaciones. El principal componente es el software de control (o motor de Workflow), responsable de la creación y eliminación de procesos, del control de la secuencia de actividades dentro de un proceso y de la interacción con los recursos humanos y/o tecnológicos.

Como se puede ver, la parte más importante de un Sistema Manejador de Workflow es su motor. En términos generales su función principal es la interpretación del proceso y es el encargado de distribuir las actividades, entre los recursos, a lo largo de todo el proceso. Es el software que provee el control del ambiente de ejecución para una instancia de Workflow.

Un motor de Workflow es responsable en parte (o todo) del control del ambiente de tiempo de ejecución en un Enactment Service (Servicio de Lanzamiento de Procesos).

Típicamente un motor de Workflow provee facilidades para:

- Interpretación de la definición de procesos.
- Control de las instancias de los procesos: creación, activación, suspensión, terminación, etc.
- Navegación entre actividades, las cuales pueden involucrar operaciones secuenciales o paralelas, calendarización de plazos, interpretación de datos relevantes, etc.
- Conexión y desconexión de participantes específicos.
- Identificación de actividades para la atención de algún usuario y una interfaz para soportar la interacción con el usuario.
- Mantenimiento de los datos de control y datos relevantes del Workflow, así como del intercambio de datos relevantes del Workflow hacia o desde aplicaciones o usuarios.
- Una interfaz para invocar aplicaciones externas y el vínculo con algún dato relevante del Workflow.
- Acciones de supervisión para el control, administración y auditoria.

Un motor de Workflow puede controlar la ejecución de un conjunto de procesos, subprocesos o instancias con un alcance definido determinado por el rango de tipos de objetos, y sus atributos, el cual puede ser interpretado dentro de la definición del proceso(s).

Un Servicio de lanzamiento de Workflows (Workflow Enactment Service) consiste de múltiples motores de Workflow. Hay una partición de la ejecución del proceso a través de los motores que lo forman. Este puede ser por tipo de proceso, con un motor particular controlando un tipo particular de proceso en su totalidad; por la distribución de funcionalidad, con un motor particular controlando aquellas partes de un proceso que requieren la intervención del usuario o del uso de recursos dentro de su control de dominio, o algún otro mecanismo de partición.

2.2.3. Interacción de las actividades en tiempo de ejecución

Las actividades dentro de un proceso de Workflow están típicamente ligadas con operaciones humanas, frecuentemente realizadas en conjunción con el uso de una herramienta particular (por ejemplo, el llenado de una forma) o con operaciones de proceso de información que requieren una aplicación particular para operar sobre información definida (por ejemplo, actualizar una base de datos de órdenes de compra con un nuevo registro). La interacción con las funciones de control de procesos es necesaria para transferir el control entre actividades, conocer el estatus de los procesos, invocar aplicaciones y pasar los datos apropiados, etc.

2.2.4. Interfaces de distribución y del sistema

La habilidad para distribuir tareas e información entre participantes es una de las principales características distintivas de la infraestructura en tiempo de ejecución del Workflow. Las funciones de distribución pueden operar en una variedad de niveles (desde grupos de trabajo a organizaciones) dependiendo del alcance de los Workflows. Para ello éste puede usar una variedad de mecanismos de comunicación (correo electrónico, envío de mensajes, tecnología de objetos distribuidos, etc.) La Figura 4 muestra la arquitectura del Workflow, vista desde alto nivel, haciendo énfasis en el aspecto de distribución.

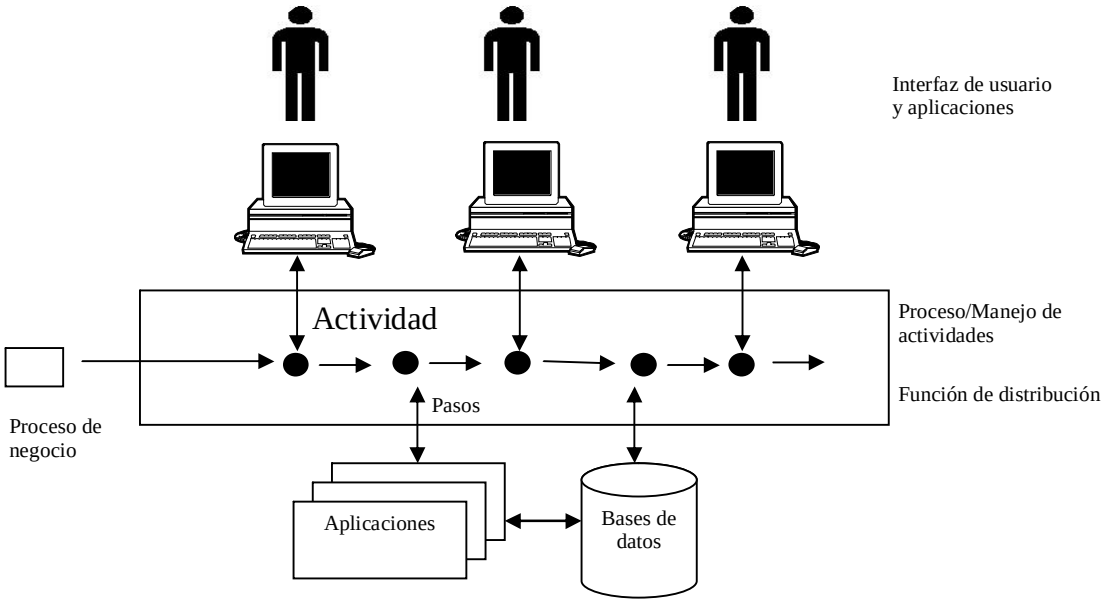


Figura 4. Distribución dentro del Workflow Enactment Service

El Workflow Enactment Service es mostrado como el componente principal de la infraestructura con la interfaz de usuario y aplicaciones, distribuidas a través del dominio del Workflow. Cada una de las interfaces es un punto de integración potencial entre el Worflow Enactment Service y otros componentes de infraestructura o aplicaciones.

2.3. Modelo de referencia de la Coalición de Manejadores de Workflow (Workflow Management Coalition (WfMC))

El modelo de referencia de Workflow fue desarrollado desde estructuras genéricas de aplicaciones de Workflow, identificando las interfaces con estas estructuras, las cuales permiten a los productos comunicarse a distintos niveles. Todos los sistemas de Workflow contienen componentes genéricos que interactúan de forma definida. Para poder tener cierto nivel de interoperabilidad entre los diversos productos de Workflow, es necesario definir un conjunto de interfaces y formatos para el intercambio de datos entre dichos componentes.

2.3.1. Modelo propuesto por la WfMC

El modelo propuesto por la WfMC incluye 5 interfaces. Estas interfaces cubren:

- ✓ Especificaciones para los datos de la definición del proceso y su intercambio,
- ✓ Interfaz para soportar interoperabilidad entre diferentes sistemas de Workflow,
- ✓ Interfaz para soportar la interacción con una variedad de aplicaciones,
- ✓ Interfaz para soportar las funciones de interfaz con el usuario, e
- ✓ Interfaz para proveer de monitoreo del sistema y funciones de medición para facilitar el manejo de ambientes mixtos de Workflow.

La Figura 5 ilustra los componentes principales e interfaces dentro de la arquitectura Workflow propuestos por la WfMC.

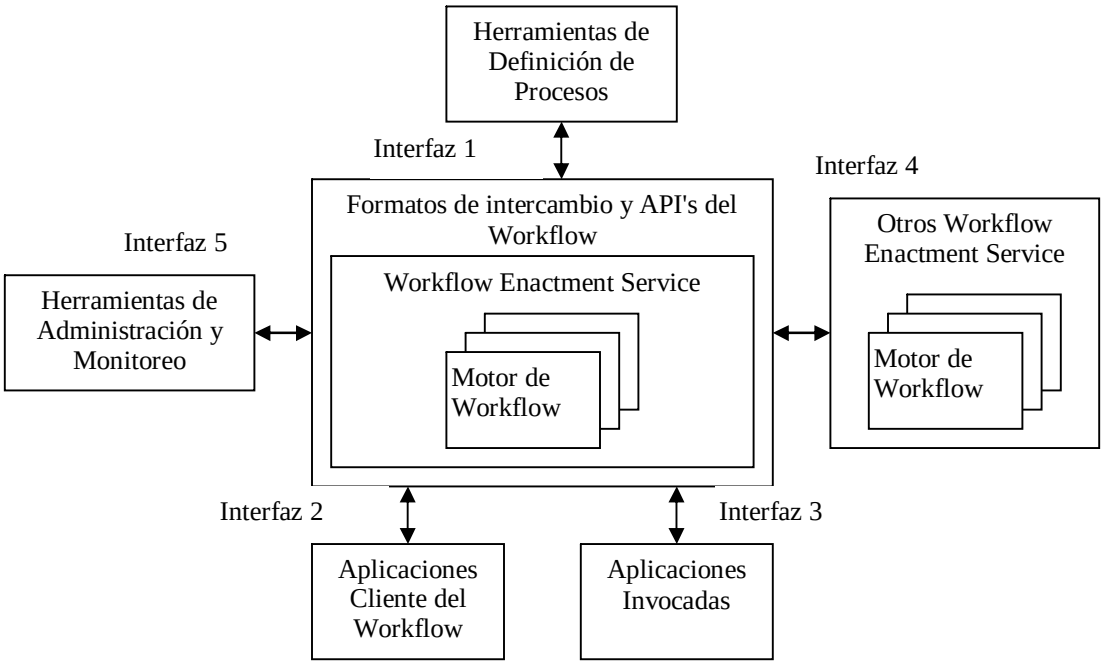


Figura 5. Modelo de referencia del Workflow (Componentes e Interfaces)

2.4. Perspectivas de Workflow

Los procesos de Workflow son manejados por casos, es decir, las tareas son ejecutadas para casos específicos. Por ejemplo, el reclamo a una póliza de seguro, la compra de un libro a través de Internet, la solicitud de una licencia de manejo, son tipos de procesos manejados por casos, los cuales son usualmente soportados por Sistemas de Manejadores de Workflow.
En los procesos manejados por casos, se pueden distinguir tres perspectivas de modelado [10]: la perspectiva de control del flujo del proceso, la perspectiva de recursos del proceso, y la perspectiva del caso. La perspectiva del control del flujo del proceso se refiere al orden en que deben ejecutarse las tareas de un proceso. Las tareas que necesitan ser ejecutadas son determinadas por estructuras de ruteo especificadas en el proceso. Las estructuras de ruteo comunes son: la secuencia, la condición, el paralelistmo y la iteración Las tareas de un proceso son ejecutadas por recursos. Estos pueden ser humanos o técnicos. En la perspectiva de recursos, estos son agrupados en roles (basados en responsabilidades o funciones dentro de la empresa) y su ubicación dentro de la organización (grupos, departamentos). Estas dos perspectivas, la perspectiva de control del flujo del proceso y la de recursos, son genéricas, es decir no están ligadas un caso específico. La tercera perspectiva se refiere a los casos individuales que son ejecutados de acuerdo al proceso definido (primera perspectiva) por los recursos adecuados (segunda perspectiva).

En la implementación de esta tesis se abordaran solo las primeras dos primeras perspectivas (flujo del proceso y recursos), ya que son las mínimas soportadas por los motores de Workflow de software libre y comerciales, y ambas perspectivas permiten, probar la correcta ejecución y distribución de las tareas entre los participantes.

2.5. Introducción a las redes de Petri

El éxito o fracaso de un sistema de Workflow radica en la calidad del modelo del flujo de trabajo. Las redes de Petri son una herramienta útil para esta labor. Con su ayuda podemos representar un proceso, analizarlo, simularlo y ejecutarlo.

El uso de un concepto formal tiene ciertas ventajas. En primer lugar, fuerza a una definición formal. Las posibles ambigüedades, incertidumbres y contradicciones son prevenidas, en contraste con otras técnicas de diagramación informales. En segundo lugar, se pueden establecer ciertos patrones en los procesos. Además, el hecho de que se utilice una técnica formal, permite utilizar técnicas de análisis para conocer la validez del proceso.

Las redes de Petri fueron concebidas en el año de 1962 por Carl Adam Petri [22] como una herramienta de modelado y análisis de procesos. Una de sus fortalezas es el poder describir los procesos de forma gráfica y tienen además una base matemática sólida. A diferencia de otras técnicas, las redes de Petri están completamente formalizadas lo que

permite a través de varias herramientas y técnicas de análisis establecer sus propiedades. A través de los años, el modelo ha sido extendido, por lo que es posible modelar sistemas complejos.

2.6. Redes de Petri clásicas

Una red de Petri consta de *lugares* y *transiciones*. Se utiliza un círculo para indicar un lugar y un rectángulo para mostrar una transición. Los lugares y las transiciones se unen a través de un arco dirigido. Hay dos tipos de arcos: aquellos que van de los lugares a las transiciones y los que van de las transiciones a los lugares. No se permiten los arcos que vayan de lugar a lugar o de transición a transición.

Tomando como base los arcos, podemos determinar los lugares de entrada de una transición. Un lugar l es un lugar de entrada de la transición t, si y sólo si, hay un arco que conecta el lugar l con la transición t.
De igual manera, podemos saber los lugares de salida de una transición. Un lugar l es un lugar de salida para la transición t, si y solo si, hay un arco de salida que va de la transición t al lugar l.

Los lugares pueden contener fichas. Estos se indican con puntos negros. La estructura de una red de Petri es fija; sin embargo, la distribución de sus fichas a través de los lugares puede cambiar. En la Figura 6, la tansición Guardar puede tomar fichas del lugar de entrada Reclamo y ponerlos en Revisión. A esto se le llama disparo.

El estado de una red de Petri, usualmente llamado marcaje, esta indicado por la distribución de fichas entre sus lugares. En la Figura 6 se usaría el vector (3,0,0) para representar su estado, indicando que hay tres fichas en Reclamo y cero fichas en Revisión y Listo.

Una transición puede ser disparada si está *habilitada*. Esto sucede cuando hay al menos un token en cada uno de sus lugares de entrada. En la Figura 6 la transición Guardar está habilitada. Las otras dos no.

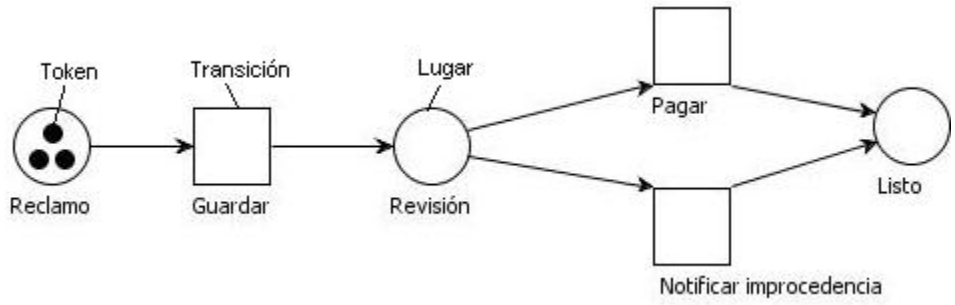


Figura 6. Red de Petri clásica

Una transición puede ser disparada desde el momento en que es habilitada. El disparo de una transición es una operación atómica. Cuando una transición se dispara, se remueve

una ficha de cada uno de los lugares de entrada y se agrega una ficha a cada uno de los lugares de salida de la transición en una sola acción.

La Figura 7 muestra el estado de la red una vez que se disparó la transición Guardar.

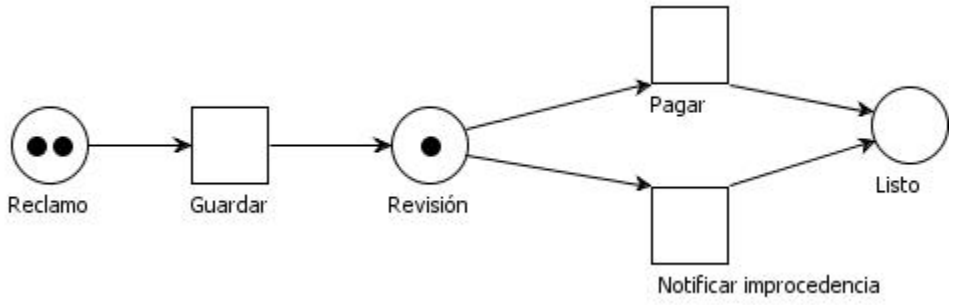


Figura 7. Red de Petri indicando que hay un caso en revisión

Las transiciones son los componentes activos en las redes de Petri. Cuando se dispara una transición, el proceso modelado cambia de un estado a otro. Por lo tanto, una transición frecuentemente representa un evento, operación, transformación o medio de transporte. Los lugares en una red de Petri son pasivos, en el sentido de que no cambian estado.

2.7. Definición formal

Las redes de Petri clásicas son un grafo dirigido que tiene dos tipos de nodos llamados lugares y transiciones.

Definición 1. Una red de Petri es una tripleta (L, T, F)

- L es un conjunto finito de lugares,
- T es un conjunto finito de transiciones ($L \cap T = \varnothing$),
- $F \subseteq (P \times T) \cup (T \times P)$ es un conjunto de arcos

Un lugar l es llamado un lugar de entrada de la transición t si y sólo si, existe una arco dirigido desde l a t. El lugar l es llamado lugar de salida de la transición t si y solo si, existe un arco dirigido desde t a l. Se usa $\cdot t = \{ l \in L : E(t,l) > 0\}$ y $t\cdot = \{ l \in L : S(t,l) > 0\}$ para denotar el conjunto de lugares de entrada y salida de la transición t.

2.8. Redes de Petri de alto nivel

Las redes de Petri clásicas tienen defectos en varias situaciones prácticas. Los modelos representados son muy largos e inaccesibles, o no es posible modelar un caso particular. Es por esto que las redes de Petri clásicas han sido extendidas de varias maneras [11]. De está forma, es posible modelar situaciones complejas en una forma estructurada y accesible. Las extensiones más importantes son: color, tiempo y jerarquía. En está tesis se limitará al uso de las redes de Petri Coloreadas (CPN´s por sus siglas en inglés), es decir, aquellas que han sido extendidas con color [12].

Las fichas son usadas para modelar varias cosas. En un modelo pueden representar el reclamo a una compañía de seguros, en otro la inscripción a un curso. Sin embargo, en las redes de Petri clásicas es imposible distinguir entre dos fichas. Por definición dos fichas en el mismo lugar son indistinguibles. Por ejemplo, en el caso de dos reclamos a una compañía de seguros deseamos saber las características de cada caso. Para poder diferenciar un caso de otro, las redes de Petri fueron extendidas usando "color". Esta extensión asegura que a cada token se le asigne un valor o color.

Cuando se dispara una transición, esta produce fichas basados en color de aquellos que consumió. Por lo tanto, el valor de las fichas producidos depende de las fichas consumidas. A diferencia de las redes de Petri clásicas, el número de fichas producidos es variable.

2.9. Mapeo de conceptos de Workflow en redes de Petri

2.9.1. El proceso

Cuando se usa un proceso en un Sistema Manejador de Workflow, se está indicando de que manera un tipo de caso particular debe manejarse. Los procesos definen que tareas deben ejecutarse. Además, un proceso contiene información acerca de las condiciones. De esta manera, se define el orden en que deben ejecutarse dichas tareas. Podemos observar que un proceso se puede especificar usando una red de Petri. Esta red tendrá un lugar de entrada y un lugar de salida. Las condiciones se modelan como lugares y las tareas como transiciones.

2.9.2. Ruteo

Algunas tareas dentro del proceso pueden ser opcionales. Para algunos casos será necesario llevar a cabo determinadas tareas. El orden en la cual las tareas deben llevarse a cabo dependerá del caso. Si se agregan opciones de ruteo en algunas tareas, podrá determinar que tareas necesitan ser ejecutadas y en qué orden.

Son cuatro las construcciones de ruteo principales:

a) Secuencia. Es aquella en la que una tarea se ejecuta después de otra.

La Figura 8 muestra una red de Petri con actividades secuenciales.

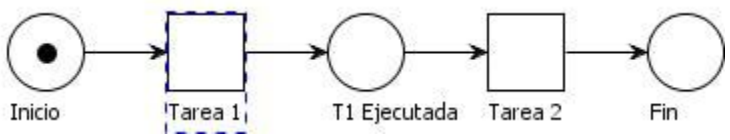


Figura 8. Ruteo Secuencial

b) Ruteo paralelo. Es aquella en que dos o más tareas necesitan ser ejecutadas al mismo tiempo o en cualquier orden. Se utiliza un AND-split para dividir las tareas

y un AND-join para sincronizarlas.

La Figura 9 muestra una red de Petri con ruteo paralelo.

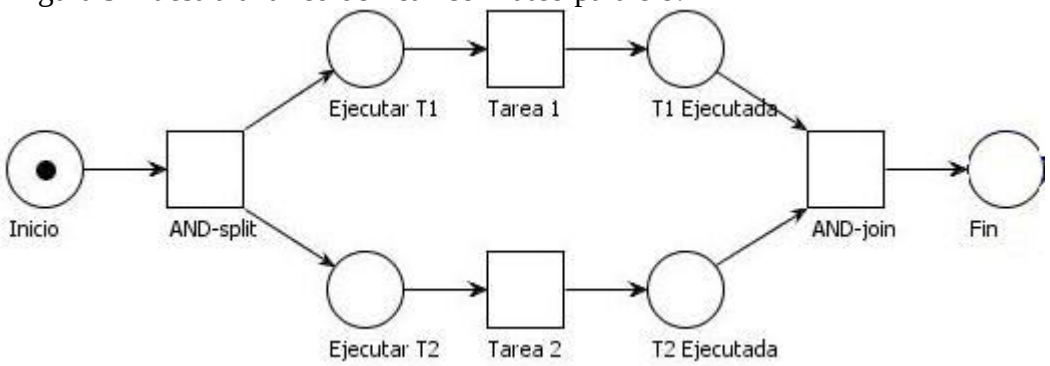


Figura 9. Ruteo Paralelo

c) Ruteo de selección. Es aquella que involucra una opción entre dos o más tareas. Se utiliza un OR-Split para elegir que tarea ejecutar y un OR-Join en espera de que una de las tareas sea ejecutada.

La Figura 10 muestra una rede de Petri con ruteo de selección.

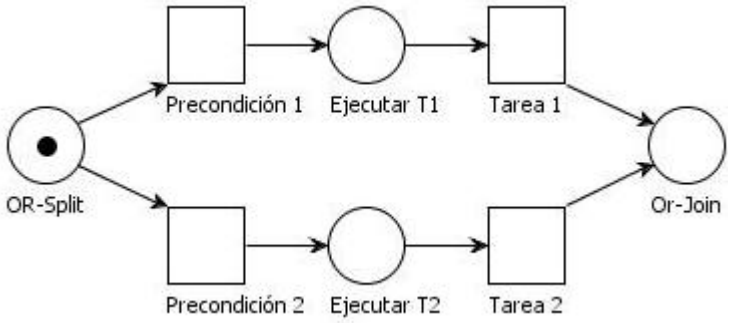


Figura 10. Ruteo Condicional

d) Ruteo iterativo. Esta última forma de ruteo sirve para ejecutar una tarea específica en repetidas ocasiones.

La Figura 11 muestra una red de Petri con ruteo iterativo

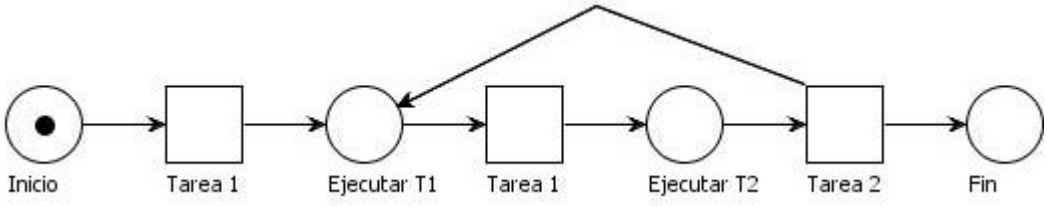


Figura 11. Ruteo Iterativo

2.9.3. Enactment (Lanzamiento)

Como se menciona en [15], un Enactment Service (Servicio de Lanzamiento de Procesos) está compuesto de uno o varios motores de Workflow. Es aquí donde se implementa una de las dos aplicaciones que componen el sistema de Workflow.

2.10. Patrones de Workflow

En el desarrollo de software, los patrones son descritos como una solución bien definida y entendida a un tipo particular de problema. En [18] describe 23 patrones utilizados en la programación orientada a objetos. Los patrones ofrecen mayor flexibilidad y reusabilidad a los programas. Un enfoque similar fue seguido en [19] donde se identificaron veintiún patrones de Workflow existentes en diversos Sistemas Manejadores de Workflow, y su trabajo ha acelerado la estandarización de procesos en estos sistemas.

Desde 1999 se han venido elaborando patrones de Workflow [6]. En [L-6] se presenta información actualizada sobre patrones de Workflow. Los patrones van desde los más simples como el patrón secuencial (Patrón 1) hasta los complejos, por ejemplo, el patrón de sincronización. Los patrones de Workflow pueden ser clasificados en las siguientes categorías:

1. Patrones de control de flujo básicos. Estos patrones están presentes en la mayoría de los lenguajes de Workflow, y sirven para modelar procesos secuenciales, paralelos, o aquellos que incluyan alguna decisión,
2. Patrones de ramificación avanzada y sincronización. Estos patrones superan a los patrones de control de flujo básicos al permitir tipos avanzados de bifurcación y sincronización,
3. Patrones estructurales. Estos patrones permiten terminar un subproceso cuando ya no haya nada que hacer, o permiten definir ciclos de forma arbitraria,
4. Patrones que manejan múltiples instancias. Cuando se le da seguimiento a un caso, algunas veces es necesario que el proceso sea instanciado muchas veces.

2.10.1. Patrones de control de flujo básicos

A continuación se definen las estructuras de control básicas. Éstas se asemejan a las definiciones encontradas en [4].

Los patrones de control básicos, presentes en la mayoría de los sistemas manejadores de Workflow, permiten modelar procesos sencillos. Sin embargo, con estos patrones se pueden modelar procesos complejos que involucren actividades paralelas para agilizar la ejecución del proceso. Asimismo, se pueden modelar procesos que involucren decisiones y elegir, de manera manual por parte del usuario o de manera automática por parte del motor de Workflow, entre varias rutas disponibles dentro del proceso.

1. Patrón 1 (Secuencia). Ejecuta actividades en forma secuencial. Ejemplo: El reclamo a una póliza de seguro es evaluada una vez que sea capturada toda la información del siniestro,

2. Patrón 2 (Bifurcación paralela). Ejecuta actividades en forma paralela.
   Permite que al término de una actividad, ésta genere varias actividades, las cuales pueden ser ejecutadas de forma paralela, es decir de forma simultánea o en cualquier orden. Ejemplo: Al terminar de capturar los datos de un siniestro, se pueden generar varias actividades, por ejemplo, una para verificar que el titular de la póliza este vigente, y otra para verificar que el pago sea procedente,

3. Patron 3 (Sincronización). Permite establecer un punto de convergencia para las actividades que venían ejecutándose de forma paralela. Ejemplo: En el ejemplo anterior, es necesario establecer un punto donde converjan la actividad de verificación de la vigencia de la póliza y la procedencia del pago,

4. Patrón 4 (Decisión). Es un punto en el Workflow donde se lleva a cabo una decisión, basada en la elección del usuario o en los datos de control del Workflow, una de varias ramas es escogida. Ejemplo: El usuario puede elegir entre declarar un siniestro como procedente o improcedente,

5. Patrón 5 (Fusión simple). Es un punto en el Workflow donde varias ramas de decisión se unen sin una sincronización. Este patrón asume que ninguna de las ramas de decisión se ejecuta de forma paralela. Ejemplo: En el ejemplo anterior hay que esperar que concluyan las actividades relacionadas con la decisión tomada por el usuario.

Estos son los patrones que soportará el motor de Workflow que se implementa en está tesis.

2.11. Lenguaje de modelado con redes de Petri (PNML)

El lenguaje de marcas con redes de Petri (PNML, por sus siglas en inglés) es una propuesta de un formato de intercambio para redes de Petri basado en XML. Originalmente fue planeado para servir como formato de los archivos para la versión en Java de PNK [L-1]. Sin embargo, otros grupos tambien están desarrollando otros formatos de intercambio, por lo que PNML es solo uno de varios lenguajes de intercambio para redes de Petri propuestos.

La característica específica de PNML es que es abierto. Es decir, permite diferenciar entre varios tipos de redes de Petri. Las características específicas para cada red de Petri son definidas en un archivo de Definición de Tipo de Red de Petri (PNTD, por sus siglas en inglés).

En el Apéndice A se muestra el código en PNML de la Figura 12. Se trata de un proceso secuencial con tres nodos (dos lugares y una transición). Se quitaron las marcas gráficas, útiles para una visualización adecuada del proceso cuando se diseña a través de una herramienta gráfica.

Figura 12. Proceso Secuencial

2.12. Motor de redes de Petri

El objetivo principal del Núcleo de Redes de Petri (PNK por sus siglas en inglés) [L-1] es la reducción de tiempo de programación, enfocándose en los algoritmos para nuestra aplicación. El nucleo de redes de Petri PNK nos brinda una infraestructura para la administración y modificación de redes de Petri. Además, soporta una implementación orientada a objetos rápida y modular e integra aplicaciones y algoritmos para redes de Petri. PNK también soporta el formato PNML. PNK nos proporciona una interfaz para obtener información básica de la red.

La Figura 13 muestra el modelo conceptual de PNK, el cual está construido en capas. El PNK es responsable de la administración de grafos y redes. Las clases principales de PNK que se deben tomar en cuenta para el desarrollo de una aplicación son Net, Place, Transition y Arc [L-12]. El editor muestra la red de Petri y permite modificarla. Existe un archivo en el que el programador puede definir sus propios tipos de redes de Petri con extensiones específicas. Las clases principales que permiten extender una red son Parking, Mode, Inscription y Extension [L-12], las cuales son representadas como atributos en la clase Specification [L-12]. Esta especificación del tipo de red se pasa a PNK como parámetro.

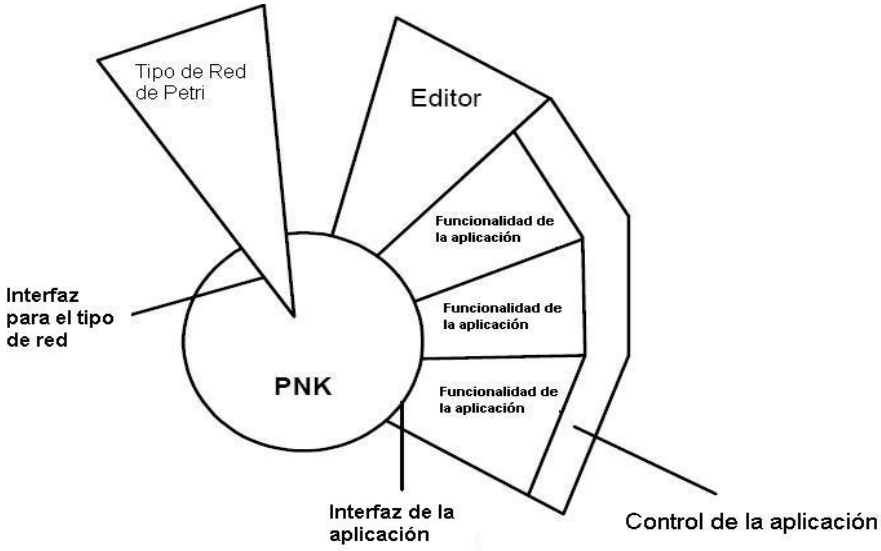


Figura 13. Modelo conceptual del núcleo de redes de Petri PNK

En el Apéndice B se muestra código en Java escrito de acuerdo a [L-12].

CAPITULO 3. Modelo a implementar

3.1. Introducción

A pesar de la gran variedad de productos de Workflow que se encuentra en el mercado, se puede ver que los conceptos y terminologías utilizadas no varían en gran parte. Esto permite que se tienda a realizar un modelo de implementación general. Actualmente se busca identificar los principales componentes de un sistema de Workflow para poder incluirlos dentro de un mismo modelo abstracto. Es necesaria la representación formal de un modelo que permita la realización de sistemas sobre diversos escenarios para tener la posibilidad de que distintos sistemas de Workflow puedan interactuar entre sí. Los principales componentes funcionales de un sistema Workflow están ilustrados en la Figura 14.

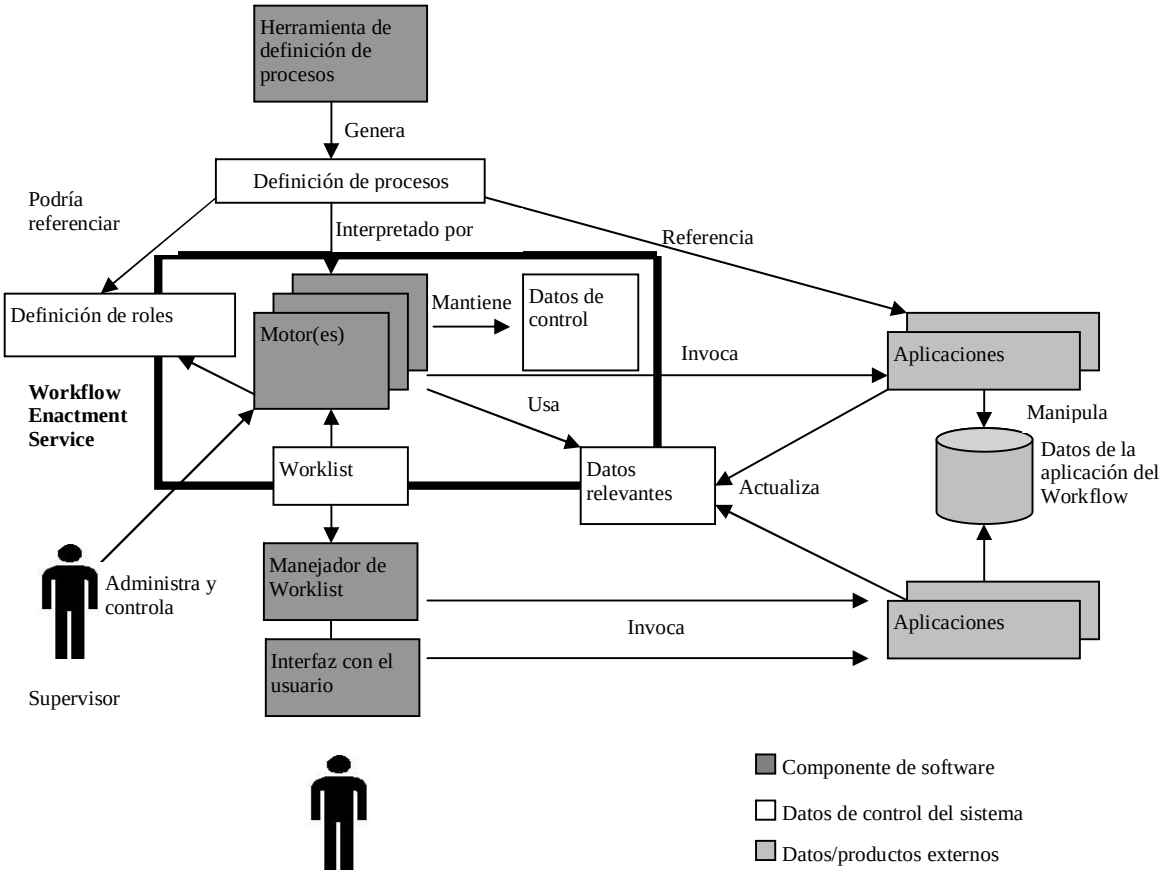


Figura 14. Estructura genérica de un producto Workflow

A continuación se enumeran los componentes funcionales que se implementarán en ésta tesis para la ejecución de procesos a través del motor del Workflow:

✓ **Herramienta de Definición de Procesos.**

Forma parte de los componentes de software del Workflow. Será utilizada para crear una descripción de los procesos en una forma procesable por el motor de Workflow. Esta herramienta estará basada en un lenguaje formal de definición de procesos, en este caso redes de Petri.

Esta herramienta podrá existir por si sola o estar integrada con el motor de Workflow y manejador de actividades.

✓ **Definición de Procesos**

Luego del componente anterior encontramos la Definición de Procesos, que forma parte de los datos del Workflow. Contendrá toda la información necesaria acerca de los procesos e incluirá información de comienzo de actividades, condiciones, y reglas de navegación que serán almacenadas en formato XML.

En la definición de procesos se mencionará que rol participa en la ejecución de cada una de las actividades

✓ **Workflow Enactment Service**

El servicio de lanzamiento de Workflow estará compuesto de un solo motor. El principal objetivo de está tesis, es la implementación de este componente. Los otros componentes se desarrollan para probar la ejecución de los procesos, y la integración entre la herramienta de definición de procesos y el worklist.

✓ **Datos relevantes y datos de la aplicación del Workflow**

Cuando los procesos incluyen decisiones, u otra operación de control dentro del motor de Workflow, estos se basan en datos generados o actualizados por los programas de aplicación del Workflow, tales datos son accesados por el motor del Workflow y llamados datos relevantes del Workflow (también conocidos como "datos de caso"); este es el único tipo de datos de la aplicación accesibles por el motor de Workflow. Los datos de la aplicación del Workflow son manipulados directamente (y solo) por las aplicaciones invocadas, aunque el motor del Workflow puede ser responsable de transferir tales datos entre las aplicaciones (si es necesario), ya que diferentes aplicaciones son invocadas en diferentes actividades dentro del proceso de Workflow.

Para la implementación de está tesis, solo se hará uso de datos relevantes y serán almacenados en la base de datos y accesibles a través del motor de Workflow.

✓ **Worklist (lista de trabajo)**

La Worklist forma parte de los datos del Workflow. Ya que la interacción con los usuarios es necesaria en algunos casos, el motor de Workflow utiliza una worklist manejada por un manejador de worklist para controlar tal interacción. El motor deposita en la worklist elementos ha ser ejecutados para cada usuario. La worklist puede ser visible o invisible para los usuarios depende del caso, muchas veces se deja que el usuario seleccione elementos y los procese en forma individual.

En la implementación que se hará en está tesis, todas las actividades serán ejecutadas por personas por lo que se desarrollará una herramienta que permita la selección y ejecución de las tareas por parte de los usuarios que pertenezcan al rol asociado a cada una de las

actividades.

- ✓ **Manejador de Worklist**

Es un componente de software el cual maneja la interacción entre los participantes del Workflow y el Workflow Enactment Service, vía la worklist.
Se implementará como consultas a la base de datos que permitan actualizar el worklist de los usuarios.

La Figura 15 muestra a la base de datos como intermediario entre los dos subsistemas propuestos que integren los componentes funcionales a desarrollar.

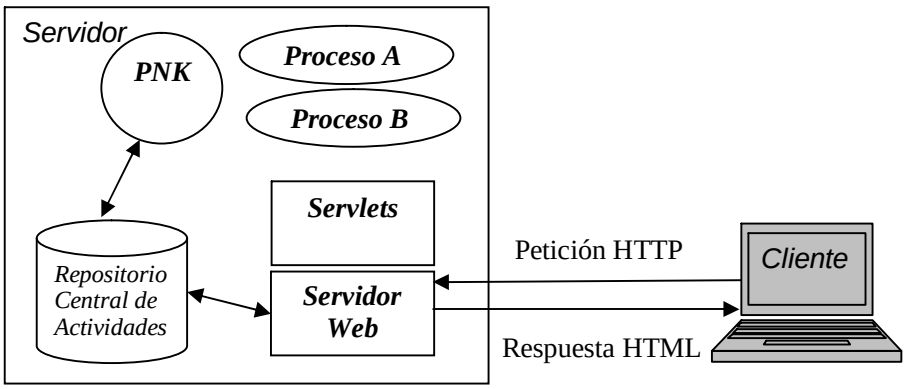


Figura 15. Arquitectura física del sistema propuesto

3.2. Meta-modelo básico

En general los sistemas manejadores de Workflow utilizan un esquema similar al mostrado en la Figura 16 para almacenar los datos del Workflow. Este es el modelo general de la base de datos que se utilizará para la implementación de está tesis. El rol que tendrá dentro del sistema será muy importante, ya que fungirá como intermediario entre los dos subsistemas propuestos.

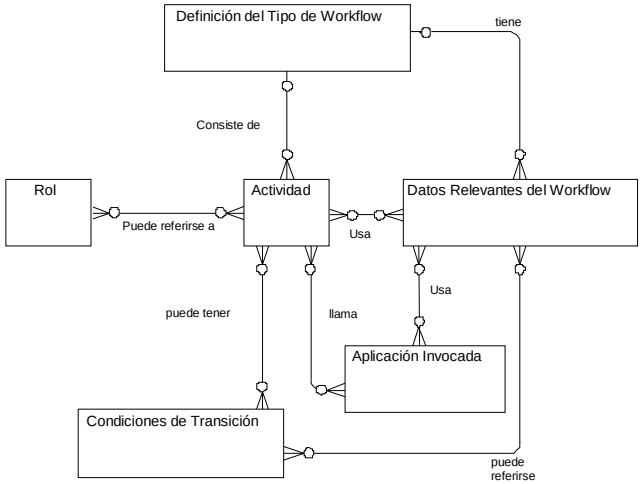


Figura 16. Diagrama general de la base de datos para un Sistema Manejador de Workflow

A continuación se enuncian, en forma general, las propiedades tentativas de cada una de las tablas que se mencionan en la figura anterior. El modelo entidad-relación del sistema se detalla en el capítulo de Análisis y Diseño.

Definición del tipo Workflow
- Nombre del proceso Workflow
- Número de versión
- Condiciones de comienzo y terminación del proceso
- Seguridad, auditoría y otros datos de control

Actividad
- Nombre de la actividad
- Tipo de actividad (subflujo, flujo atómico, etc)
- Condiciones Pre- y posp- actividad
- Otros tiempos obligatorios

Condiciones de transición
- Condiciones de flujo o ejecución

Datos relevantes del Workflow
- Ruta y nombre del dato
- Tipo de dato

Rol
- Nombre y entidad organizacional

Aplicaciones invocadas
- Tipo genérico o tipo
- Parámetros de ejecución
- Localización y ruta de acceso

CAPITULO 4. Análisis y Diseño

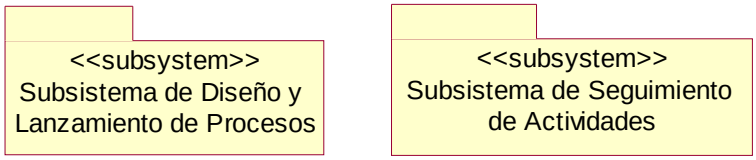
Como se ha visto, el núcleo de redes de petri PNK brinda la infraestructura necesaria para la construcción del motor de Workflow. Sin embargo, cada nuevo tipo de red que se desea que soporte, necesita de una implementación específica de la interfaz FiringRule. Además se necesita de una aplicación que ayude a los usuarios a dar seguimiento a las actividades, de los procesos lanzados, que se vayan generando. La consulta y ejecución de las actividades deberá ser posible desde cualquier computadora conectada a Internet.

4.1. Arquitectura general del sistema propuesto

Para la correcta integración entre la herramienta de definición de procesos, el motor de Workflow y el worklist se propone utilizar la base de datos como medio de interacción entre ellos.

Tomando como base esta premisa, se desarrollarán dos subsistemas y su integración estará mediante la base de datos.

El Submódulo de Lanzamiento de Procesos que incluye al motor de Workflow se encargará de generar, en la base de datos, la actividad inicial de cada proceso. Las actividades subsecuentes de cada proceso lanzado las irá generando el motor de Workflow de acuerdo al diseño del proceso, y conforme el usuario vaya ejecutando dichas actividades. El submódulo de Seguimiento de Actividades permitirá, a través del worklist, consultar, seleccionar y ejecutar dichas actividades.



4.2. Subsistema de Diseño y Lanzamiento de Procesos

Para este sistema se tomará como base el núcleo de redes de Petri PNK [L-1]. Como se menciona en el capítulo anterior, éste nos brinda la infraestructura necesaria para el diseño, simulación y ejecución de procesos. Sin embargo, no soporta las WF-Nets ni su ejecución, por lo se desarrollarán estas dos necesidades.

El núcleo de redes de Petri PNK permite extender, mediante definiciones hechas en XML, los tipos de redes de Petri que soporta. Aquí es donde se define un nuevo tipo de red de Petri, en este caso las WF-Nets para el modelado de procesos de negocio.

El núcleo de redes de Petri PNK utiliza el lenguaje PNML [L-13] para guardar las

definiciones de los procesos.
Los procesos almacenados bajo el tipo WF-Nets son ejecutados mediante un módulo de lanzamiento (Enactment). El objetivo principal de este módulo es crear y guardar los datos de una nueva instancia del proceso lanzado. Además de colocar la ficha inicial en la red de Petri para que la primer actividad o actividades del proceso (transición o transiciones de la red de Petri estén habilitadas), y el usuario que tenga el rol adecuado pueda ver y ejecutar las actividades del proceso. El siguiente apartado se dedica al módulo que da seguimiento a estas actividades

En resumen, sólo se implementaron dos mejoras al núcleo de redes de Petri PNK:

- Se agregó un nuevo tipo de red de Petri (WF-Net) mediante un archivo XML. Este punto no implica la realización de un nuevo módulo, simplemente la definición de un nuevo tipo de red de Petri.
- Se agregó la opción de Lanzamiento (Enactment) para que la WF-Net sea ejecutada. El motor de Workflow interpretará el proceso lanzado y definido.

4.2.1. Definición de la red de Petri WF-Net mediante XML

En el Apéndice C se muestra la especificación del archivo WFNet.xml que deberá ser guardado en el directorio netTypeSpecifications del núcleo de redes de Petri PNK. El archivo está en formato XML y sigue los lineamientos indicados en la documentación de dicho núcleo.

4.2.2. Diagrama de Contexto del Subsistema de Diseño y Lanzamiento de Procesos

La Figura 17 presenta en el Diagrama de Casos de Uso del contexto general del Subsistema de Diseño y Lanzamiento de Procesos.

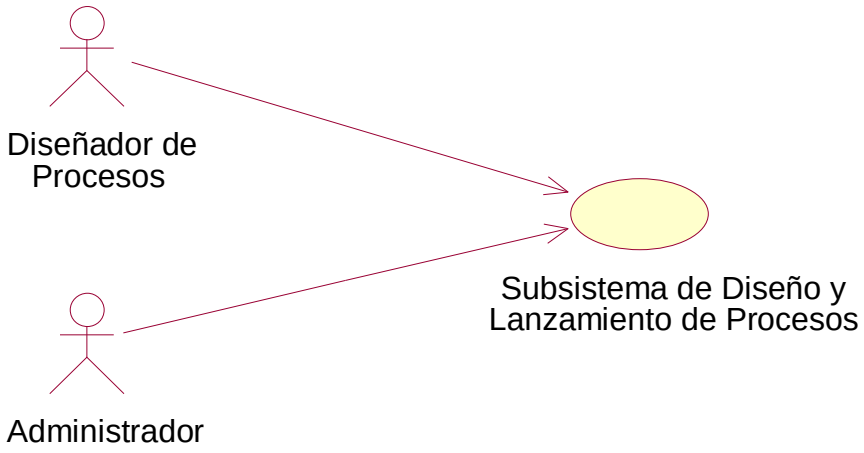


Figura 17. Caso de Uso general del Submódulo de Diseño y Lanzamiento de Procesos

4.2.3. Casos de Uso del Submódulo de Diseño y Lanzamiento de Procesos

Aunque en la Figura 18 se pueden ver los casos de uso del Diseño de Procesos, solo se pone énfasis en el caso de uso Lanzar Proceso, y que es accesible sólo por el Administrador. Este caso de uso sirve para lanzar un proceso y forma parte del motor de Workflow. La funcionalidad del Diseño de Procesos nos la proporciona el Kernel de Redes de Petri PNK. En [15] se encuentra la documentación donde se muestra dicha funcionalidad.

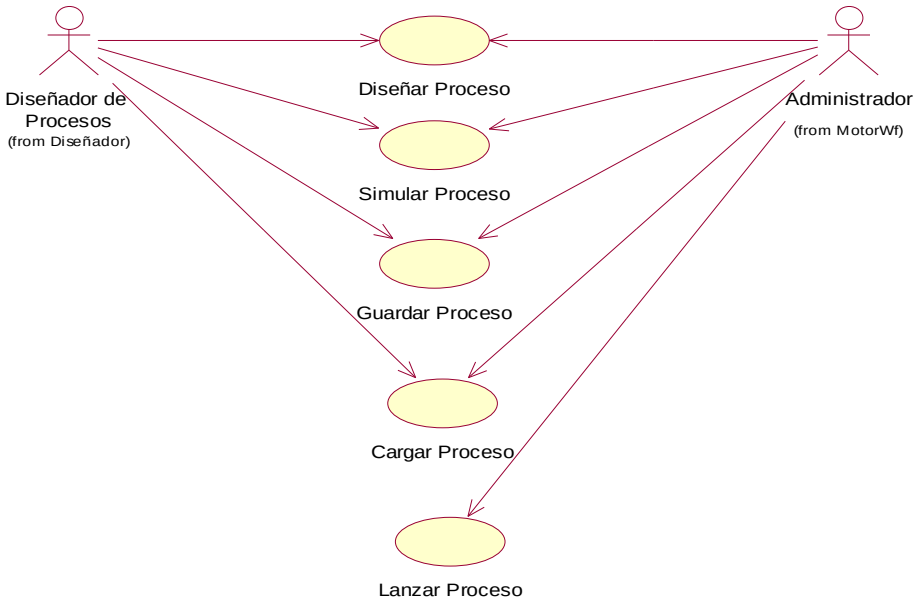


Figura 18. Casos de Uso del Submódulo de Diseño y Lanzamiento de Procesos

A continuación se describen los casos mostrados en la Figura 17:

Diseñar Proceso. El actor Diseñador de Procesos o el Administrador pueden en cualquier momento crear un nuevo proceso o modificar uno existente.
Simular Proceso. Antes de guardar o una vez guardado el proceso, es posible simular el proceso diseñado para saber si se ejecutará de la forma en que lo hemos planeado.
Guardar Proceso. Permite guardar el proceso diseñado, o que estamos diseñando para su posterior modificación o ejecución
Cargar Proceso. Permite cargar un proceso para su modificación, simulación o lanzamiento.
Lanzar Proceso. Permite Lanzar (Ejecutar) un proceso.

4.2.4. Diagrama de Secuencia del Lanzamiento de Procesos

La Figura 19 muestra la secuencia de acciones entre los actores del sistema cuando se decide lanzar un proceso.

1. El Administrador del sistema selecciona, desde el Administrador de Archivos, el archivo PNML que contiene el proceso que desea ejecutar
2. El archivo seleccionado es cargado y mostrado en el diseñador de procesos
3. El Administrador del sistema decide lanzar el proceso
4. El motor de Workflow revisa si se cumplen las condiciones necesarias para lanzar el proceso
5. El motor de Workflow crea en la base de datos una nueva instancia del proceso Lanzado

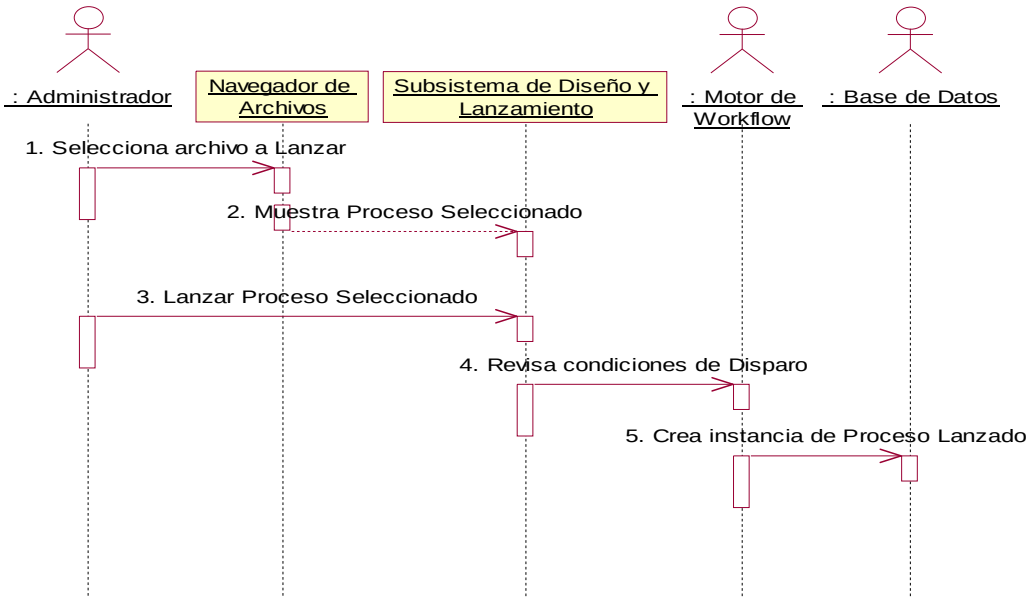


Figura 19. Diagrama de Secuencia del Submódulo de Lanzamiento de Procesos

4.2.5. Lanzamiento de Procesos (Enactment)

La Figura 20 presenta el diagrama de actividades para el lanzamiento de un nuevo proceso.

Figura 20. Diagrama de general de Actividades del Submódulo de Lanzamiento de Procesos

4.2.6. Diagramas de Clase

Las clases MetaApplication y Enactment son proporcionadas por el núcleo de redes de Petri PNK. El motor de Workflow es una implementación de la interfaz propuesta por el mismo núcleo.

La Figura 21 muestra el diagrama de clases para el motor de Workflow.

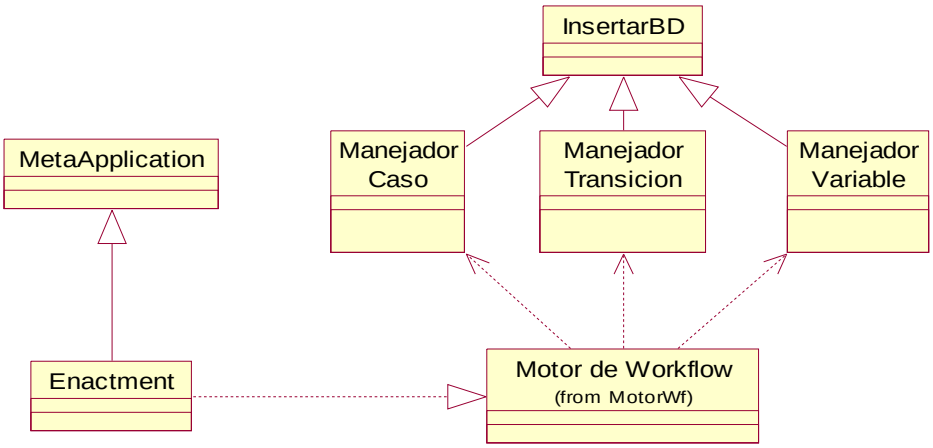


Figura 21. Diagrama de clases para el Submódulo de Lanzamiento de Procesos (Clases de control)

La Figura 22 muestra el Diagrama Entidad-Relación para ambos subsistemas.

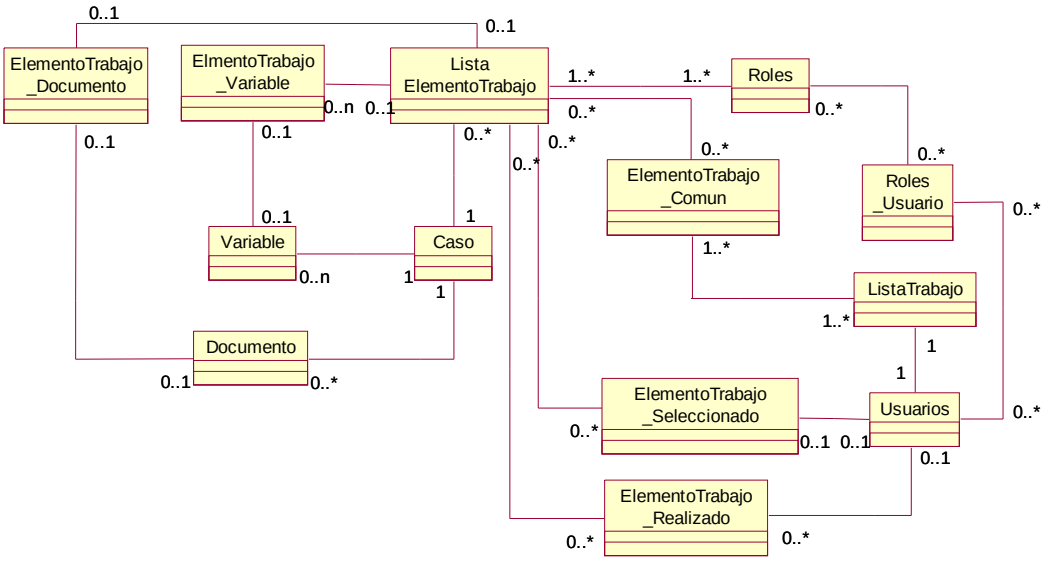


Figura 22. Diagrama de clases para el Submódulo de Lanzamiento de Procesos (Clases de entidad)

4.2.7. Diagrama de Paquetes para el Subsistema de Diseño y Lanzamiento de Procesos

La organización de paquetes para el Subsistema de Diseño y Lanzamiento de Procesos es la siguiente:
PNK: contiene la aplicación del núcleo de Redes de Petri PNK.
MotorWF: contiene las clases que se encargarán de ejecutar los procesos.
Manejador de Datos: contiene las clases que permiten guardar, en la base de datos, nuevas instancias de procesos, actividades y variables de las instancias de los procesos.
La Figura 23 muestra el diagrama de paquetes para el Subsistema de Diseño y Lanzamiento de Procesos.

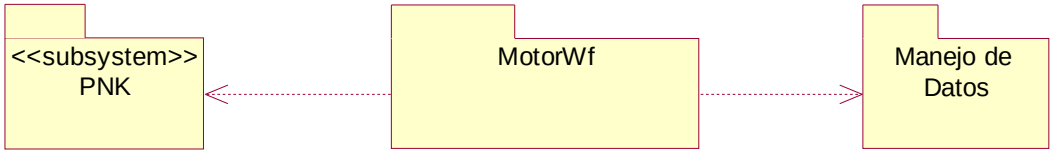


Figura 23. Diagrama de clases para el Submódulo de Lanzamiento de Procesos

4.3. Subsistema de Seguimiento de Actvidades (Worklist)

Una vez lanzada la tarea inicial o tareas iniciales del proceso mediante la opción de Enactment del Submódulo de Diseño y Lanzamiento de Procesos, el usuario o usuarios con los roles adecuados debe darle seguimiento a las actividades que conforman el proceso que fue lanzado.

Inicialmente este subsistema ofrece a los usuarios una pantalla de acceso al sistema que valida contra una base de datos los permisos de acceso del usuario al sistema. Una vez dentro del sistema, y dependiendo del rol o roles asignados a cada usuario, éste ve los elementos de trabajo (WorkItems) disponibles para su ejecución. El usuario selecciona el elemento de trabajo que desea realizar en ese momento, o si desea postergar su realización. La actividad quedará reservada (pasan de ser elementos de trabajo a actividades), y sólo puede ser ejecutada por él. Los elementos de trabajo seleccionados son verificados para confirmar que estén disponibles, es decir, que ningún otro usuario esté trabajando en ellas. El usuario podrá ejecutar sus actividades en el momento que crea conveniente. Cuando el usuario ejecute la actividad seleccionada, deberá indicar explícitamente que la actividad ha finalizado. Una vez que la actividad ha finalizado, las actividades subsecuentes se disparan, tomando como base el propio diseño del proceso, y las condiciones en él establecidas.

En resumen, para el subsistema de Seguimiento de Actividades realizaremos lo siguiente:

- Una manejador de elementos de trabajo (workitems)
- Los mecanismos necesarios para comunicarse y mantener la base de datos, que es compartida por el motor de Workflow y el worklist

4.3.1. Diagrama de contexto general del Subsistema de Seguimiento de Actividades

La Figura 24 muestra los casos de uso para el Subsistema de Seguimiento de Actividades. Es necesario un caso de uso para verificar la disponibilidad de las tareas, ya que el motor de Workflow no asigna tareas de forma automática. Es necesario que el usuario seleccione, entre las tareas disponibles, aquellas que desea realizar.

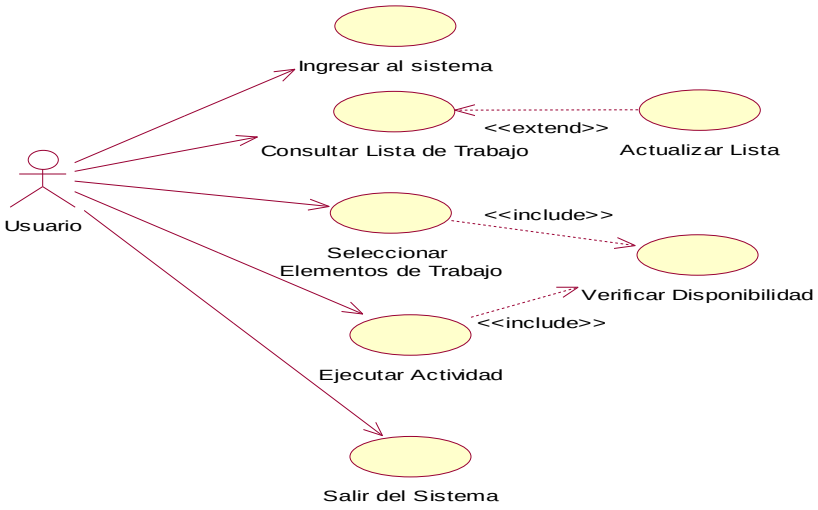


Figura 24. Diagrama de contexto de Submódulo de Seguimiento de Actividades

4.3.2. Diagramas de Clase

La clase HttpServlet extiende a GenericServlet agregando métodos para manejar peticiones http. Las clases que heredan de HttpServlet implementaran el método doGet() para el manejo de las peticiones.

Las clases HTML_UsuarioLogin, HTML_ManejadorListaTrabajo y HTML_MuestraElementoTrabajoSeleccionado implementarán el método desplegarHTML() y son clases auxiliares para la presentación de la interfaz de usuario.

La Figura 25 muestra las clases necesarias para el Submódulo de Seguimiento de Actividades.

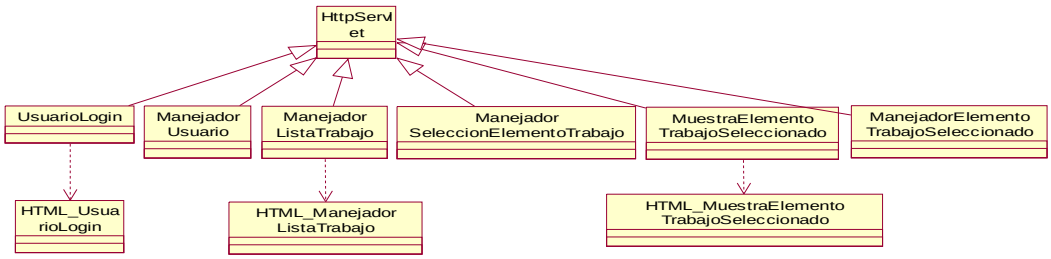


Figura 25. Diagrama de clases del Submódulo de Seguimiento de Actividades

4.3.3. Diagrama de Actividades para la ejecución de una tarea

La Figura 26 muestra las actividades realizadas por el usuario y el sistema al seleccionar una actividad

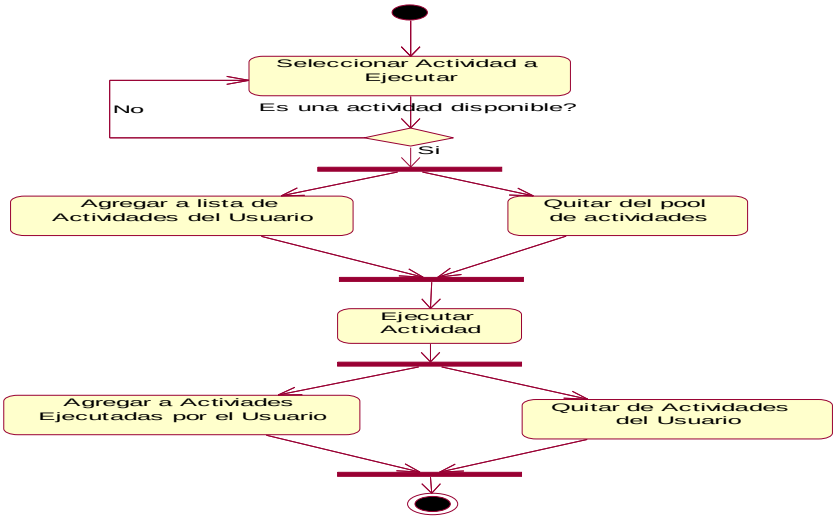


Figura 26. Diagrama de actividades de la ejecución de una tarea

4.3.4. Diagramas de Secuencia

La Figura 27 muestra la interacción entre los distintos actores del worklist inicial

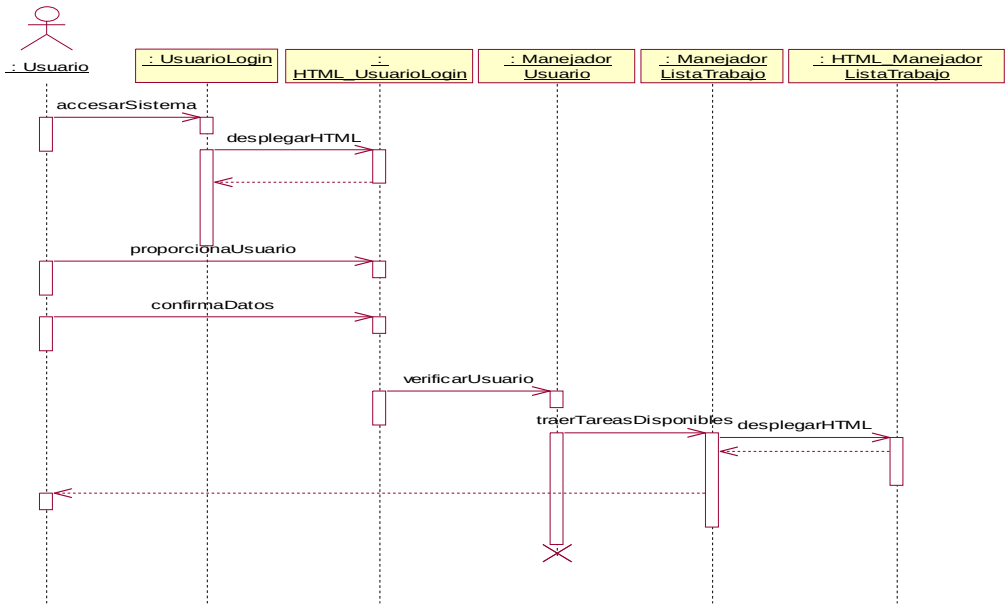


Figura 27. Diagrama de secuencia del caso de uso Ingresar al Sistema

La Figura 28 muestra la interacción entre el usuario y el worklist al actualizar el worklist

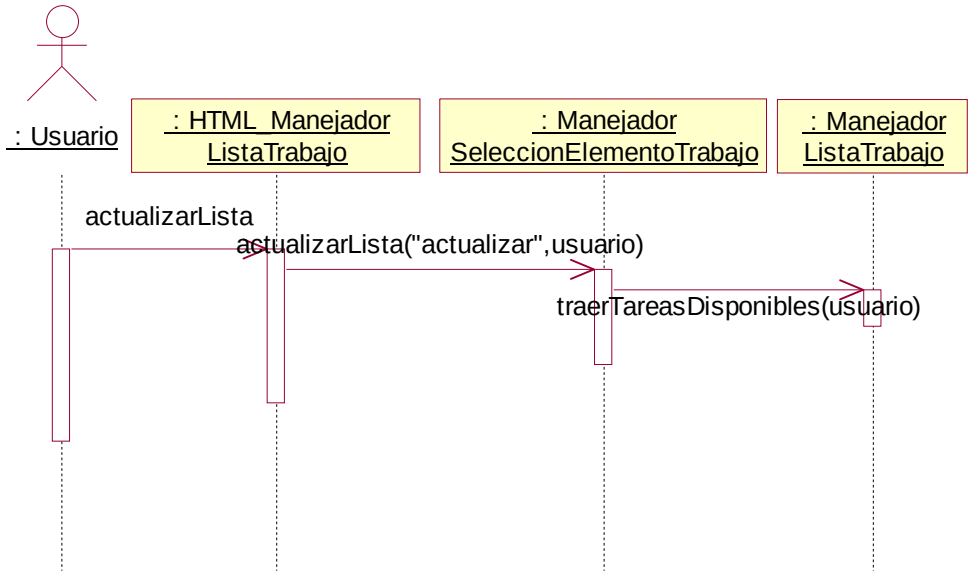


Figura 28. Diagrama de secuencia del caso de uso Actualizar Lista de Trabajo

La Figura 29 muestra la interacción entre el usuario y el worklist al seleccionar una tarea.

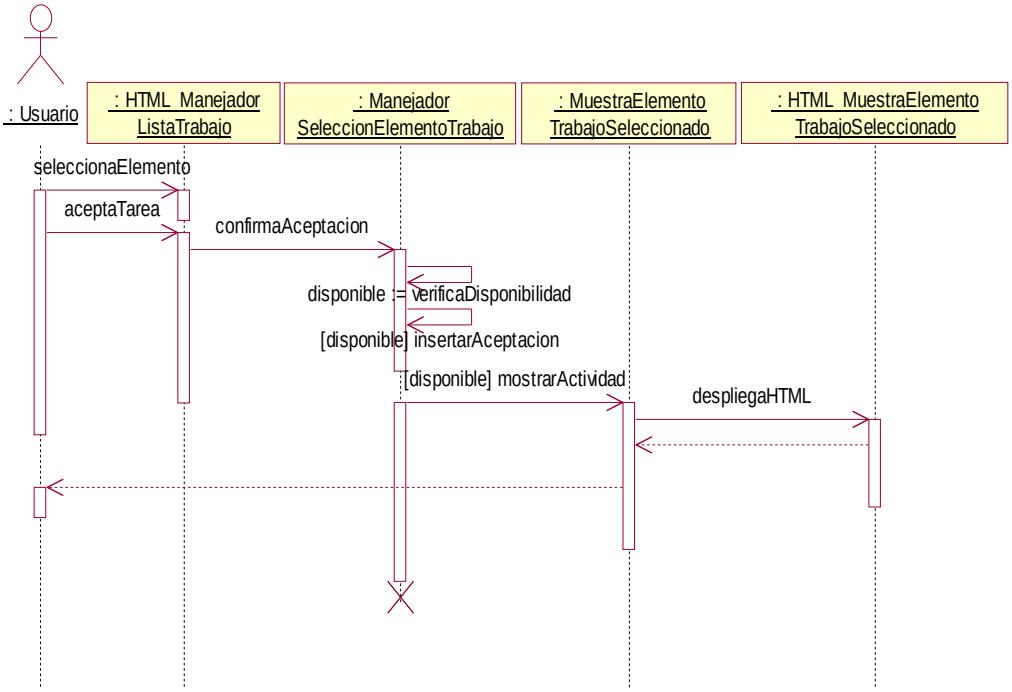


Figura 29. Diagrama de secuencia del caso de uso Seleccionar Elemento de Trabajo

La Figura 30 muestra la interacción entre el usuario, el worklist y el motor de Workflow al ejecutar una actividad.

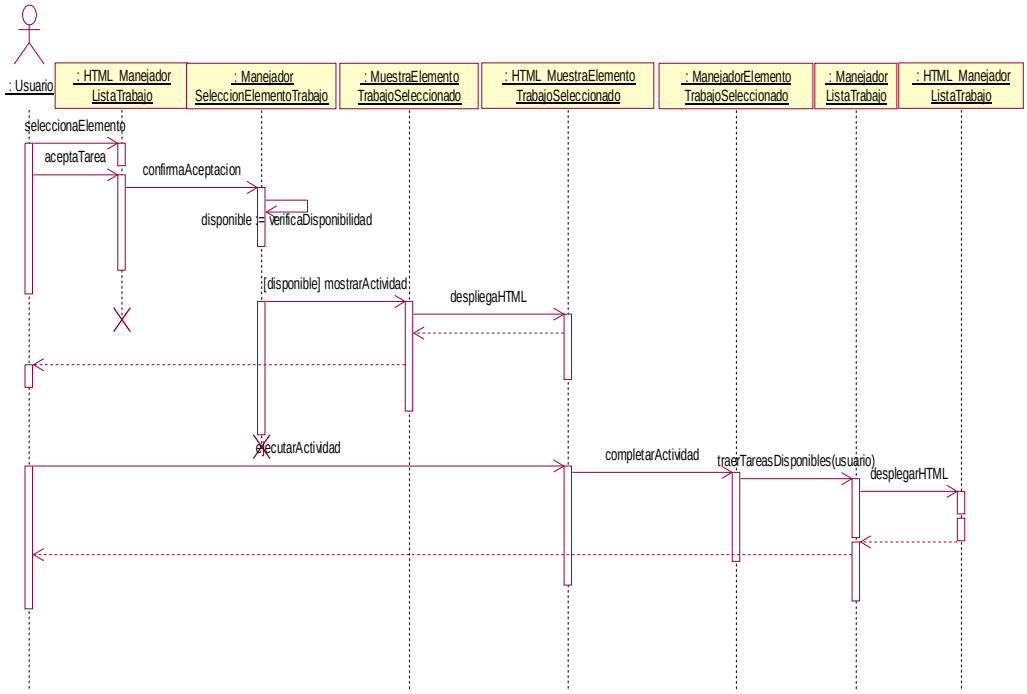


Figura 30. Diagrama de secuencia del caso de uso Ejecutar Actividad

4.4. Diagrama de componentes

La Figura 31 muestra el diagrama de componentes del sistema. Como se observa, el Subsistema de Seguimiento de Actividades depende del Subsistema de Diseño y Lanzamiento de Procesos. Esto se debe que las actividades de un proceso podrán realizarse hasta que el administrador inicie el proceso.

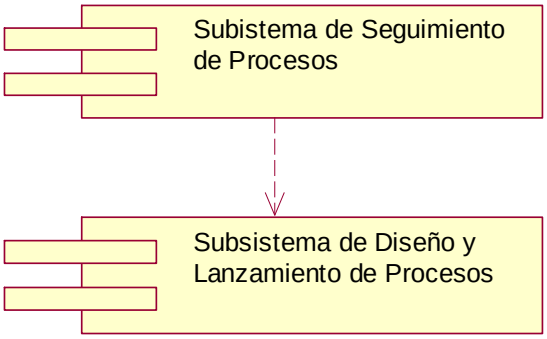


Figura 31. Diagrama de componentes

4.5. Diagrama de despliegue

En la Figura 32 se muestra la instalación que se hará de los componentes. En el cliente no se instalará ningún componente. Solo es necesario que cuente con algún navegador para Internet.
En el Servidor será necesario instalar los dos subsistemas desarrollados, esto debido a que se busca la utilización optima de los recursos.

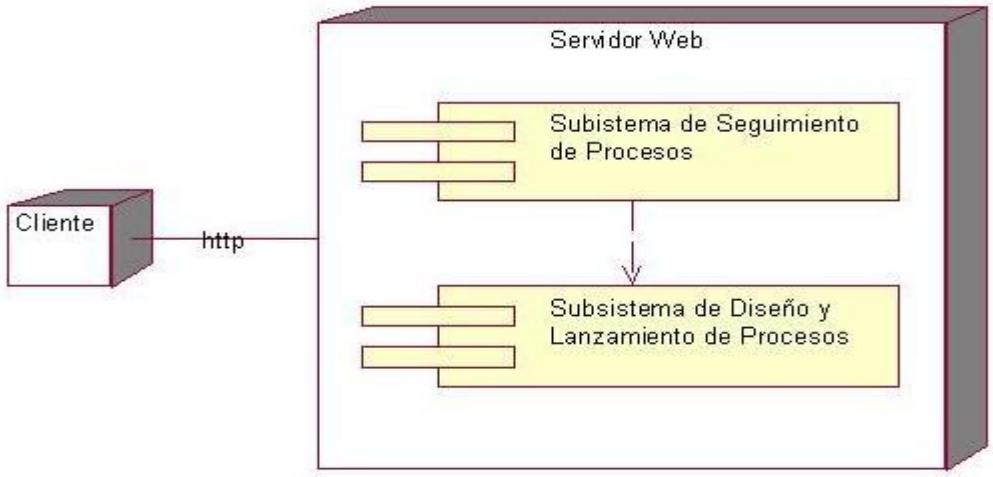


Figura 32. Diagrama de despliegue

4.6. Diagrama Entidad Relación

En el diagrama Entidad Relación, muestra el conjunto de elementos que intervienen en el sistema. En el se pueden observar como elementos principales a los usuarios, roles, casos y listas de trabajo para cada usuario. Como elementos secundarios y de control se encuentran la lista de trabajo común, la de tareas seleccionadas y la de tareas realizadas.

La lista de trabajo inicial, ElementoTrabajo_Lista, es el repositorio que comunica al submódulo de Diseño y Lanzamiento de Procesos con el de Seguimiento de Actividades. El motor de Workflow realiza, en esta tabla, una búsqueda de todas las actividades que pueden realizarse e inserta uno o varios registros, dependiendo del número de usuarios que sean responsables de realizar esta tarea, en la tabla ElementoTrabajo_Comun. Adicionalmente, las tablas de elementoTrabajo_Seleccionado y elementoTrabajo_Realizado, permiten conocer, respectivamente, que actividades se encuentran en proceso de realizarse o ya se realizaron.

La Figura 33 muestra el diagrama Entidad Relación de todo el sistema. La base de datos es el medio de comunicación entre el motor de Workflow y el Worklist del usuario.

En el Apéndice E se muestra el diccionario de datos de ésta base de datos.

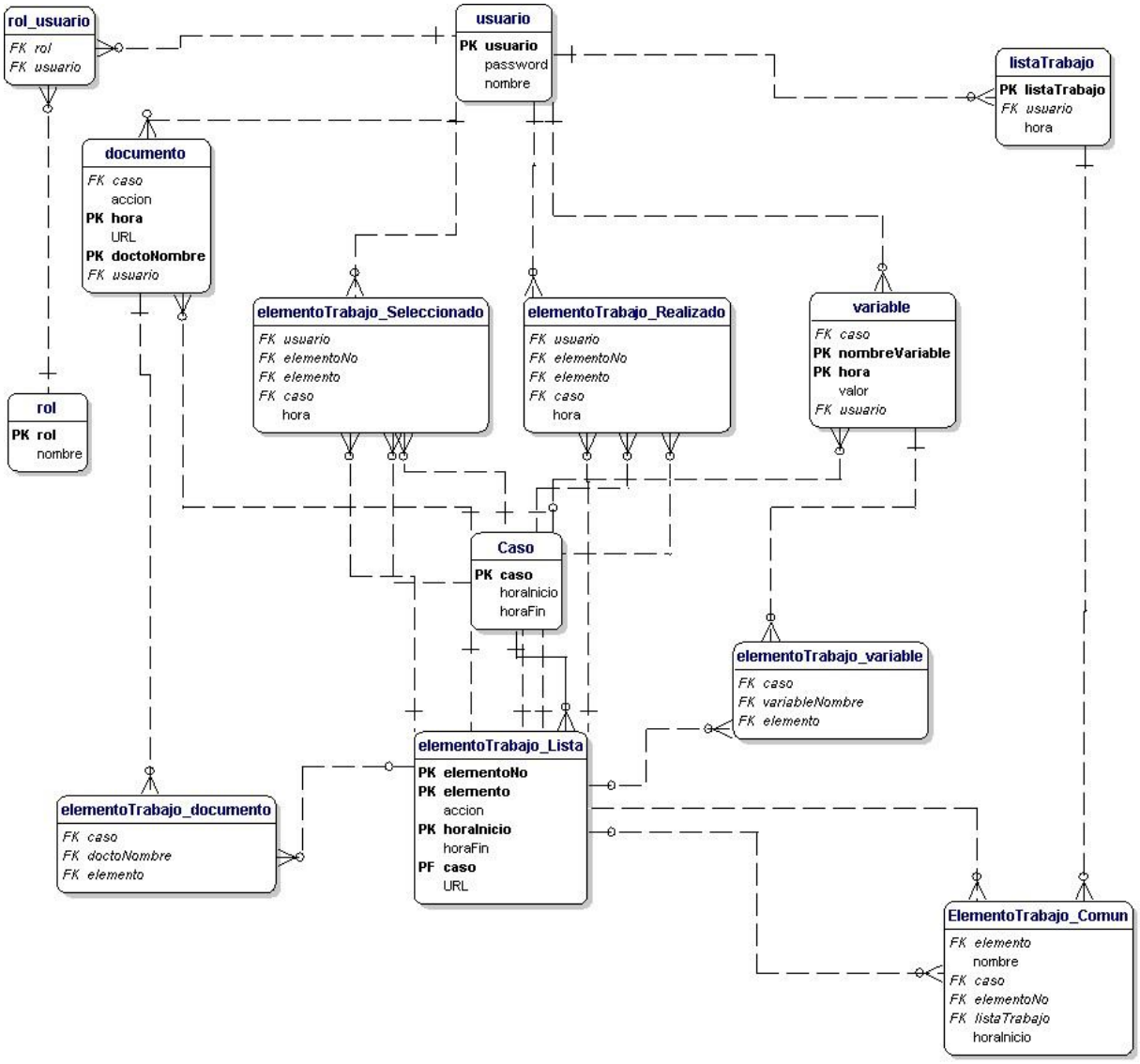


Figura 33. Diagrama Entidad-Relación

4.7. Implementación

Para la implementación de los dos submódulos se utilizan Servlets, los cuales corren sobre el servidor Web Apache Tomcat 5.0; la IDE de desarrollo que se utiliza es Borland JBuilder 2005 y el manejador de base de datos es MySQL 5.0.22. En el cliente no es necesario software adicional, solo se requiere que tenga instalado algún navegador de Internet.

4.7.1. Contenedor de Servlets

Tomcat es un contenedor de JSPs y Servlets, e integra un Servidor Web por lo que no se

necesita instalar uno adicional. Para la implementación del motor de Workflow se eligió este contenedor dada su facilidad para instalarse en forma local, además de no requerir muchos recursos de hardware.
En un ambiente típico, el servidor Web recibe todas las peticiones. El servidor decide, usualmente basado en el nombre del archivo o directorio de la petición, si envía una página estática o pasa el control al contenedor de Servlets. Este crea e invoca al Servlet según sea necesario, y maneja cualquier condición de error. La Figura 34 muestra el flujo de información entre el cliente (navegador), el servidor Web y el contenedor de Servlets.
También es posible que un contenedor de Servlets funcione en modo autónomo, como Servidor Web y administrador de Servlets. Tomcat puede ser usado junto con un Servidor Web o en modo autónomo. Este es el modo que se utiliza para el Submódulo de Seguimiento de Actividades.

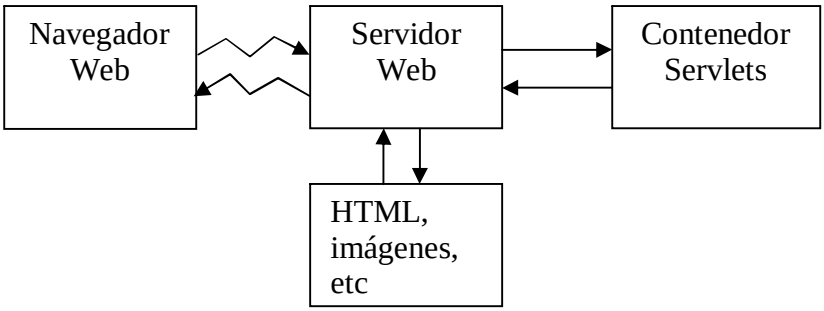


Figura 34. Flujo de información entre el cliente, el Servidor Web y el Contenedor e Servlets

4.7.2. Servlets

Los Servlets son pequeñas unidades de código en Java que se ejecutan rápidamente en el servidor en respuesta a una petición del navegador.

Un Servlet es un módulo cargado dinámicamente que sirve peticiones desde un navegador. El Servlet corre completamente dentro de la máquina virtual de Java. Debido a que el Servlet está corriendo en el servidor, no depende del navegador. Los Servlets nos permitirán mostrar páginas dinámicas con los datos solicitados por el usuario.

Se eligieron los Servlets para la implementación del motor de Workfkow, dada su facilidad para integrarlos en un servidor web de software libre como es Tomcat.

CAPITULO 5. Pruebas y Resultados

Los métodos existentes para el desarrollo de sistemas de información le dan mucha importancia a la definición de estructuras y a la forma en que la aplicación es presentada a los usuarios (interfaz de usuario). Los cambios organizacionales y el diseño o rediseño del proceso reciben poca atención. El uso de sistemas Workflow generalmente va de la mano con una reorganización radical del proceso de negocio.

La forma en que se lleva a cabo el desarrollo de un nuevo proceso debe contemplar a los usuarios en el diseño del proceso. El proceso de desarrollo debe ser de preferencia evolutivo. Esto significa que la funcionalidad del sistema es mejorada a través de acuerdos y revisiones de prototipos, hasta que tenga la funcionalidad deseada.

El proceso de pruebas está compuesto de tres etapas:

- Análisis y diseño del proceso de negocio a rediseñar,
- Implementación a través de redes de Petri del proceso diseñado, y
- Ejecución y pruebas al seguimiento de las actividades del proceso diseñado.

5.1. Pruebas al Subsistema de Diseño y Lanzamiento de Procesos

Las dos primeras etapas y la primera tarea de la tercera etapa, del proceso de pruebas, forman parte de esta prueba.

Durante la ejecución de estas tareas se utilizó equipo con las siguientes características:

Sistema	
CPU	
Tipo de Sistema	Escritorio
Proveedor	AMD
Frecuencia	2.0 GHz x 2
Memoria	2 GB
Disco Duro	160 GB
Sistema Operativo	Windows XP Professional SP2
Servidor Web	Tomcat 5.0
Manejador de base de datos	MySQL 5.0.22
Java	SDK 1.4.2.12

5.1.1. Descripción del Proceso de Negocio a automatizar

Aunque esta etapa no forma parte real de las pruebas del sistema, se incluye ya que el diseño del proceso que se genere servirá como entrada para probar el Submódulo de Lanzamiento de Procesos.

La empresa de seguros El Halcón, S.A. de C.V. procesa reclamaciones de seguros de autos de sus clientes, que resultan de accidentes automovilísticos. La empresa procesa las reclamaciones siguiendo el siguiente procedimiento:

1. Guardar el recibo de la reclamación,
2. Establecer el tipo de reclamación,
3. Revisar la póliza de seguro del cliente para confirmar que esta cubre lo reclamado,
4. Revisar la prima del seguro para confirmar que el pago esté al corriente,
5. Se rechaza la solicitud de pago si el paso 4 o 5 tienen resultados negativos,
6. Se genera una carta de rechazo para el cliente explicando los motivos,
7. Se estima del monto que se debe pagar, basado en los detalles de la reclamación,
8. Se solicita la verificación por parte de un asesor para que investigue las circunstancias de del daño y establecer su valor,
9. Considerar las medidas de emergencia para limitar daños mayores o calmar el dolor provocado por posibles lesiones,
10. Tomar las medidas de precaución si la tarea 9,
11. Establecer o revisar el monto a ser pagado y ofrecerlo al cliente,
12. Guardar la respuesta del cliente al ofrecimiento del monto a pagar: aceptación o rechazo,
13. Llegar a un acuerdo de rechazo y revisar nuevamente el monto (tarea 11) o tomar las medidas legales (tarea 14),
14. Tomar medidas legales,
15. Pagar de la suma reclamada, y
16. Cerrar reclamación: colocar fecha de fin de caso.

En el proceso de negocio para el reclamo de una suma asegurada originada por accidente automovilístico se detectaron 16 actividades, aunque no todas ellas se deben llevar a cabo de forma secuencial.

El tiempo que se obtuvo en diseñar este proceso utilizando lápiz y papel fue de treinta minutos. Este tiempo es subjetivo, es decir, depende del tipo de proceso que se esté diseñando y del entendimiento que se tenga de él, por parte de la persona que lo realice.

5.1.2. Diseño del proceso con redes de Petri

Esta prueba consiste en dibujar los nodos (lugares y transiciones) con sus respectivos vértices. En general la herramienta utilizó un tiempo promedio para dibujar cada uno de estos elementos de un milisegundo. Cabe aclarar que este tiempo es solo ilustrativo, ya que el tiempo que le tome a la persona que realice el diseño del proceso, dependerá del número de veces que dibuje o borre cada elemento, así como su familiaridad con la herramienta. Además se deben de considerar las condiciones de disparo que se necesiten especificar para cada transición, así como del número de veces que se vaya simulando el proceso para garantizar que su ejecución sea la esperada.
En el caso del proceso que se viene diseñando, tomo quince minutos terminarlo.

Incluyendo tres simulaciones parciales y una de todo el proceso.

La Figura 35 muestra el mismo diseño del proceso diseñado antes de pasarlo a la herramienta.

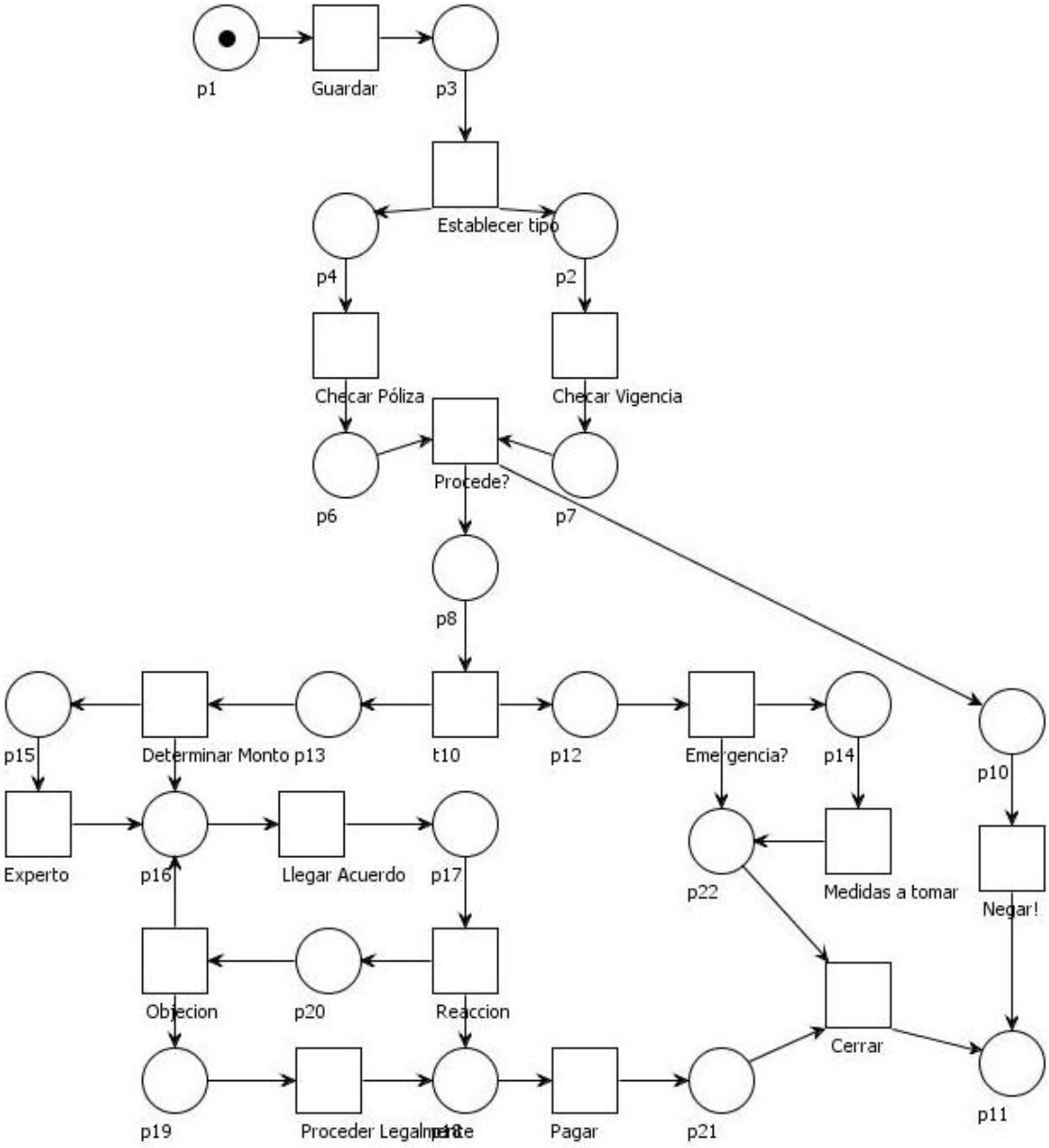


Figura 35. Proceso para el pago de reclamaciones para seguros de autos

El proceso diseñado por medio de la herramienta es el siguiente (Figura 36):

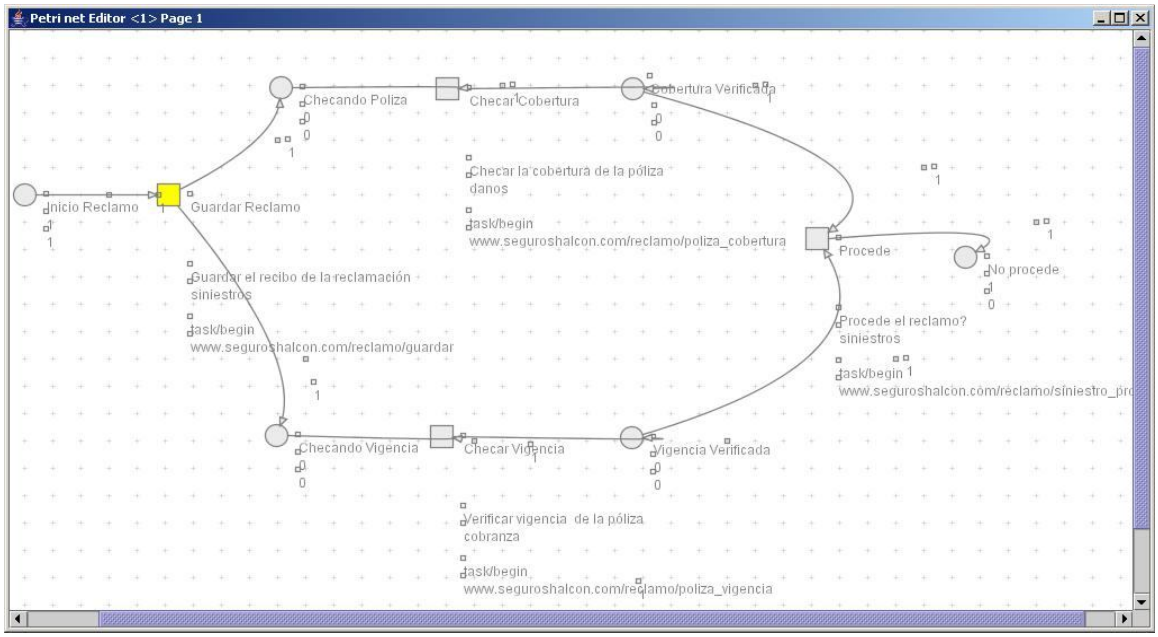


Figura 36. Proceso para el pago de reclamaciones para seguros de autos diseñado mediante el sistema

Para cada transición fue necesario especificar, adicionalmente al nombre de la actividad, la descripción textual de la tarea ejecutar, el rol encargado de ejecutar la tarea, el tipo de tarea, ya sea inicial, intermedia o final, así como una dirección web a la que hay que acceder para completar la tarea.

El archivo se guardó con el nombre *Reclamo 1*. La extensión que por default se coloca al archivo es pnml.

La actividad *Establecer Tipo* es del tipo AND-split, permitiendo que las actividades *Checar Póliza* y *Checar Vigencia* se ejecuten en paralelo. La actividad *Procede?* es del tipo AND-join y OR-Split, con lo que se logra que se sincronicen las actividades que venían ejecutándose en paralelo. Además en esta actividad se determina si la reclamación procede. En caso de que no proceda la reclamación, se ejecuta la actividad *Negar!* Y el caso se da por terminado.
Se creó una actividad vacía (*t10*) del tipo AND-split que permite darle seguimiento al caso en forma paralela. Por un lado se verifican las medidas a caso de seguir en caso de emergencia. Como no todos los accidentes incluyen medidas de emergencia, la actividad *Emergencia?* es del tipo OR-Split e involucra una decisión. Y por otro lado, se determina el monto a pagar. En algunos casos es necesario llamar a un valuador o experto por lo que la actividad *Determinar Monto* es del tipo OR-split. Una vez determinado el monto a pagar, se le hace el ofrecimiento de pago al cliente. Si este acepta la cantidad ofrecida se procede al pago. La actividad *Reacción* guarda la aceptación o no del ofrecimiento, por lo que es una actividad del tipo OR-split que en este caso permite modelar una decisión y un ciclo que se mantendrá hasta que se llegue a un acuerdo con el cliente, o se decida proceder legalmente. La actividad *Objeción* es una actividad del tipo OR-split que

permite volver a consultar con el cliente o proceder con el pago, sin antes turnar el caso al departamento jurídico. Antes de dar el caso por terminado, es necesario haber hecho el pago al cliente y haber guardado las medidas de emergencia adoptadas, es por esto que la actividad *Cerrar* es del tipo AND-join y sirve para sincronizar las actividades que venían ejecutándose de forma paralela.

Como se puede ver, en el diseño del proceso se emplearon los patrones básicos, actividades secuenciales, AND-split/join (actividades paralelas) y OR-split/join (actividades que involucran una decisión), además de que se pueden modelar actividades repetitivas a través de OR-split/join. Con esto se comprobó que es posible modelar procesos de negocio complejos utilizando patrones de control de Workflow básicos.

5.2. Pruebas al Subsistema de Seguimiento de Actividades

Para esta prueba se utilizó una computadora personal con las siguientes características:

Sistema	
CPU	
Tipo de Sistema	Portátil
Proveedor	Intel
Frecuencia	1.0 GHz
Memoria	1 GB
Disco Duro	20 GB
Sistema Operativo	Windows XP Professional SP2
Navegador	Microsoft Internet Explorer 6.0

5.2.1. Pruebas de Integración del Subsistema de Diseño y Lanzamiento de Procesos y del Subsistema de Seguimiento de Procesos

Antes de iniciar las pruebas al Subsistema de Lanzamiento de Procesos se comprobó que hubiera una integración entre ambos Subsistemas. Para esto, se verificó que en la base de datos existieran las tareas iniciales disparadas por el motor de Workflow. La Figura 37 muestra los resultados obtenidos al hacer una consulta a la base de datos.

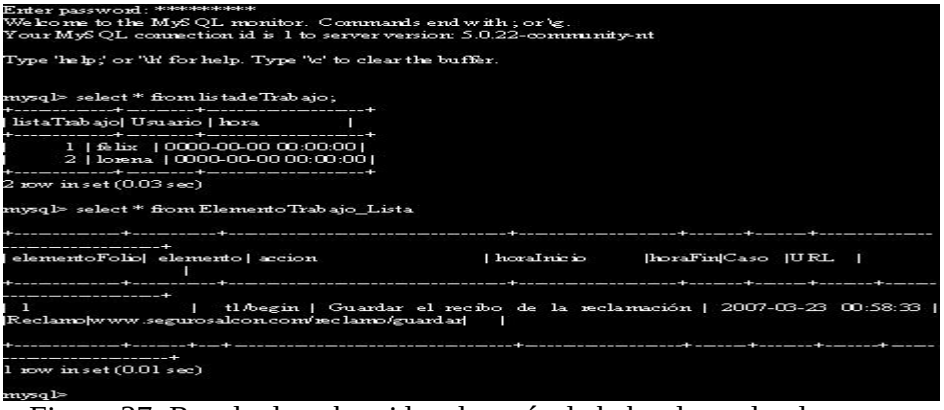


Figura 37. Resultados obtenidos después de haber lanzado el proceso

Los resultados muestran los registros iniciales que fueron insertados durante las pruebas al Subsistema de Lanzamiento de Procesos. Estos datos muestran que el disparó de la actividad inicial del proceso fue satisfactoria. Esta actividad es la que el usuario ve en su Worklist una vez que ingrese al sistema.

5.2.2. Pruebas al Subsistema de Seguimiento de Actividades

En esta prueba se comprobó que el usuario ingresara correctamente al sistema después de teclear su usuario y password. Una vez dentro del sistema se verificó que, de acuerdo al rol del usuario, el workklist solo mostrara las actividades que le corresponden. La Figura 38 muestra el worklist del usuario.

Figura 38. Worklist el usuario.

En esta prueba se observa que el usuario tiene una actividad pendiente, y que es la actividad inicial del proceso que se diseño durante la prueba de Diseño del proceso.

Además se comprobó que una vez terminada está tarea, el motor de Workflow generará las tareas subsecuentes, así sucesivamente, hasta terminar el proceso.
Con estas pruebas se comprobó que el motor de Workflow generó, de acuerdo a lo especificado durante el diseño del proceso, las actividades correctas.

5.2.3. Pruebas de Rendimiento del Motor de Workflow

Como parte de las pruebas del sistema se realizó una serie de mediciones para comprobar el desempeño del sistema con una carga que va de los 10 los 250 usuarios. La Figura 39 muestra gráficamente el tiempo de respuesta del sistema partiendo de 10 hasta 250 conexiones simultáneas. Como se puede observar a mayor número de usuarios, el tiempo de respuesta se incrementa. Sin embargo, el tiempo de respuesta de dos segundos para 250 usuarios consultando sus actividades de forma simultanea, es bastante aceptable, si consideramos que no se utiliza un Servidor Web dedicado.

Clientes	Tiempo de Comunicación en Segundos
10	0.985376
20	1.112799
30	1.38398
40	1.147195
50	1.905239
60	1.677043
70	1.755732
80	1.953528
90	1.955299
100	2.012184
200	2.284536
250	2.128495

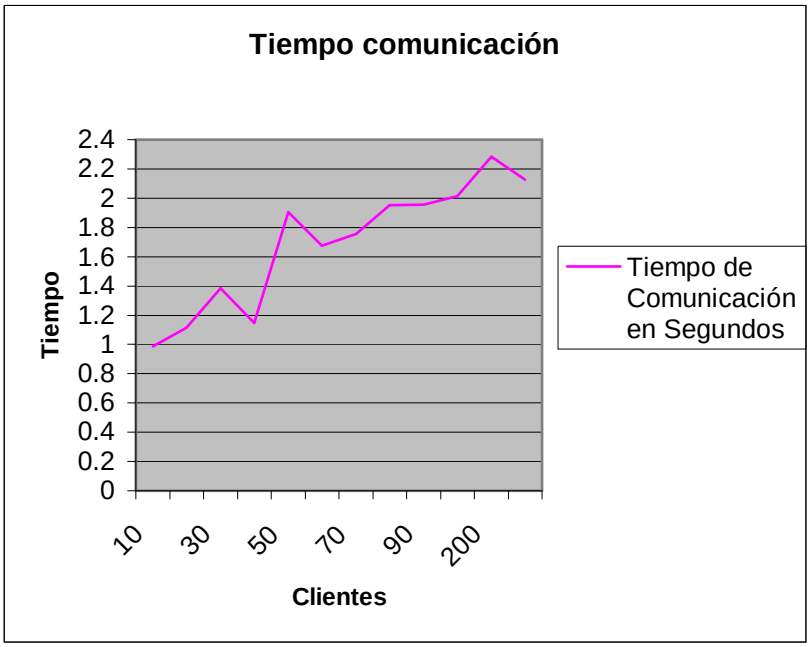


Figura 39. Tiempos de respuesta del sistema

5.3. Resultados

Se lograron hacer pruebas individuales y de integración a ambos subsistemas. En estas pruebas se comprobó que funcionaran correctamente de forma individual y colectivamente, adicionalmente su integración fue satisfactoria. Fue posible diseñar, simular y lanzar un proceso. Por su parte, el motor de Workflow interpretó y generó correctamente las actividades de acuerdo a lo especificado en el diseño del proceso, y se completaron sus actividades satisfactoriamente a través del Administrador de tareas.
Finalmente, los tiempos de respuesta del sistema fueron satisfactorios dado que no se utiliza un Servidor Web dedicado.

5.4. Comparativo con otros productos

La siguiente tabla muestra un comparativo de las características soportadas en la implementación de ésta tesis y dos productos de software libre. Para este comparativo no fue posible obtener una versión de prueba de algún producto comercial.

	Sistema		
Característica	Tesis	WOPED 0.8.0	YAWL
Modelado	Redes de Petri (Wf-Nets)	Redes de Petri Clásicas	Redes de Petri (Wf-Nets) con adecuaciones de redes de Petri Coloreadas
Lenguaje	PNML 1.0	PNML 1.3.2	YAWL
Motor	Implementado en esta tesis	No incluye. Hace uso del motor de YAWL	Motor propio
Patrones Soportados	De control básicos	De control básicos	Todos los patrones de Workflow disponibles
Tiempos de ejecución	Los mostrados en la Figura 40	No disponibles	No disponibles

CAPITULO 6. Conclusiones

6.1. Conclusiones

Al inicio de ésta tesis se plantearon varios objetivos así como dos hipótesis, una positiva y otra negativa.
A lo largo del desarrollo de esta tesis se buscó cubrir dos objetivos específicos, uno de ellos era extender el núcleo de PNK que permitiera diseñar procesos de negocio, así como la creación de un administrador de tareas. El objetivo general fue la creación de un motor de Workflow, que permitiera separar los procesos de negocio de los sistemas, y que estos fueran ejecutados por el motor.

Los objetivos específicos de ésta tesis fueron cubiertos mediante la creación de dos Subsistemas, el primero el de Diseño y Lanzamiento de Procesos, y el segundo, el de Seguimiento de Actividades. Estos, permitieron establecer una base para la automatización de procesos.

Por un lado, el Subsistema de Diseño y Lanzamiento de Procesos fue el punto de partida para el diseño y lanzamiento de los procesos, y por otro, el Subsistema de Seguimiento de Actividades dio continuidad a la ejecución de los procesos por parte del usuario; mientras la base de datos fungió como intermediario entre ambos subsistemas. Esto permite a los usuarios la ejecución de su trabajo desde cualquier computadora conectada a Internet que disponga de un navegador web.

El objetivo general se logró concretar mediante la implementación, a través del Submódulo de Lanzamiento, del motor de Workflow. Este permitió que los procesos previamente diseñados fueran ejecutados por los usuarios mediante su worklist.

Es así como el contar con tecnologías sólidas, como lo son redes de datos, manejadores de bases de datos rápidas y confiables, y tecnología Web permitió en gran medida establecer una arquitectura sencilla pero a la vez robusta para la integración de ambos subsistemas, logrando así una implementación de un motor de Workflow con soporte para los patrones de control de Workflow básicos.

Así comprobamos que nuestra hipótesis fue verdadera mediante la implementación de un motor de Workflow que permite diseñar, documentar, simular y ejecutar procesos de negocio.

6.2. Aportaciones

Este trabajo de tesis permite enfocar el esfuerzo de análisis y diseño hacia los procesos de negocio, parte fundamental de los sistemas de información. Además de conocer su comportamiento antes de lanzarlos a producción, y poder documentarlos.

Al enfocar el esfuerzo hacia los procesos de negocio se logra un mejor diseño de los sistemas, que a su vez, son más fáciles de mantener además de lograr una mejor documentación.

Una de las aportaciones principales es el disponer de una herramienta de bajo costo y con tecnología de software libe que apoya a los diseñadores de procesos y organizaciones en el diseño o rediseño de sistemas de información. El contar con organizaciones que tengan en cuenta a los procesos de negocio como parte fundamental de su operación diaria permitirá que sus sistemas de información sean más confiables.

6.3. Trabajos futuros

Dentro de los trabajos futuros a este trabajo, se incluyen los siguientes:

- Integrar formularios dentro la ejecución de actividades para darle al usuario una herramienta más fácil de usar,
- Permitir la integración de herramientas que generen documentos, basados en plantillas, de forma automática, ahorrando tiempo al usuario,
- Crear una herramienta de monitoreo para saber el desempeño de los empleados,
- Realizar una herramienta para el administrador que permita definir roles y usuarios,
- Permitir la integración del motor de Workflow con otras aplicaciones sin hacer uso del subsistema de lanzamiento inicial
- Permitir al usuario una personalización del administrador de elementos de trabajo de acuerdo a sus preferencias y forma de trabajo, y
- Notificaciones al usuario de tareas inconclusas cuando se acerca su vencimiento

Bibliografía

Referencias bibliográficas

[1] W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, Cambridge, 2002.

[2] C.A. Ellis. Information Control Nets: A Mathematical Model of Office Information Flow. In Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems, pages 225-240, Boulder, Colorado, 1979. ACM Press

[3] M. D. Zisman. Representation, Specification and Automation of Office Procedures. PhD thesis, University of Pennsylvania, Warton School of Business, 1977

[4] WFMC. Workflow Management Coalition Terminology & Glossary, Document Number WFMC-TC-1011, Document Status - Issue 3.0, February. Technical report, Workflow Management Coalition, Brussels, 1999.

[5] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. Accepted for publication in Information Systems.

[6] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. *Workflow Patterns*. Distributed and Parallel Databases, volume 14(3), pages 5-51, July 2003.

[7] Zur Muehlen, M.; Allen, R.: Workflow Classification. Embedded & Autonomous Workflow Management Systems. Workflow Management Coalition White Paper, March 10th 2000.

[8] W.M.P van der Aalst, L. Aldred. M. Dumas, and A.H.M. ter Hofstede. Design and implementation of YAWL system. In the 16th International Conference on Advanced Information Systems Engineering (CAiSE04), Riga, Latvia, June 2004, Springer Verlag.

[9] M Zur Muehlen. Workflow-based Process Controlling: Foundation, Design and Application of Workflow-driven Process Information Systems. Logos, Berlin, 2004.

[10] S. Jablonski and C. Bussler. Workflow Management: Modeling Concepts, Architecture, and Implementation. International Thomson Computer Press, London, UK, 1996

[11] K. Jensen and G. Rozenberg, editors. High-level Petri Nets: Theory and Application, Spfinger-Verlag, Berlin, 1991

[12] W. Reisig and G. Rozenberg, editors. Lectures on Petri Nets I: Basic Models volume 1491 of Lecture Notes in Computer Sciences, Springer-Verlag, Berlin, 1998

[13] Wil M.P. van der Aalst Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management. Department of Technology Management Eindhoven University of Technology

[14] L. Fischer, editor. Workflow Handbook 2005, Workflow Management Coalition. Future Strategies, Lighthouse Point, Florida, 2005.

[15] Workflow Management Coalition. The Workflow Reference Model

[16] F. Leymann and D. Roller. Production Workflow: Concepts and Techniques. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.

[17] Máximo Pezzini Yefim V. Natis. Trends in Platform Middleware: Disruption Is in Sight. ID Number G00152076. Gartner 24 September 2007

[18] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Copyright © 1995 Addison-Wesley Publishing Company. *Design Patterns*. 0-201-63361-2.

[19] W.M.P. van der Aalst, A.H.M ter Hofstede, B. Kiepuszewski, and A.P. Barros. Copyright © 2002 Queensland University of Technology. *Workflow Patterns*.

[20] Delphi Group: BPM 2003 – Market Milestone Report. White Paper. 2003.

[21] zur Muehlen, M. and Rosemann, M.: Multi-Paradigm Process Management. In: Proc. of the Fifth Workshop on Business Process Modeling, Development, and Support – CaiSE Workshops. 2004

[22] C.A. Petri. Comunicación y Automatización. Tesis doctoral, Instituto de Instrumentación Matemática, Bonn, Alemania, 1962.

[23] Connie Moore, The Forrester Wave Human-Centric Business Process Management Suites, Q1 2006. Forrester

[24] OMG, ed: Unified Modeling Language. Version 2.0. Object Management Group, 2004

[25] Scheer, A.-W. ARIS - *Business Process Frameworks* (3rd Ed). Springer, 1999.

Páginas electrónicas

[L-1] Página web del Kernel de Redes de Petri PNK
http://www2.informatik.hu-berlin.de/top/pnk/

[L-2] Página web de Lotus Notes
http://www.ibm.com/software/lotus

[L-3] Página web SAP
http://www.sap.com/mexico/index.epx

[L-4] Página web de People Soft
http://www.peoplesoft.com

[L-5] Página web de motores de Workflow de código abierto escritos en Java
http://www.manageability.org/blog/stuff/Workflow_in_java/view

[L-6] Página web de patrones de Workflow
http://www.Workflowpatterns.com

[L-7] Página web del sistema manejador de Workflow jBPM
http://www.jbpm.org

[L-8] Página web del servidor de aplicaciones JBoss
http://www.jboss.org

[L-9] Página web de Java 2 Enterprise Edition
http://java.sun.com/downdloads/j2ee.html

[L-10] Proyecto YAWL
http://sourceforge.net/projects/yawl

[L-11] Página web oficial del Sistema Manejador de Workflow YAWL
http://yawlfoundation.org/

[L-12] Página web del manual de usuario del kernel de redes de Petri PNK
http://www2.informatik.hu-berlin.de/top/pnk/doku/pnk-guide/quickStart.html

[L-13] Página web del lenguaje de marcas para redes de Petri
http://www2.informatik.hu-berlin.de/top/pnml/about.html

[L-14] Página web de OMG
http://www.omg.org

[L-15] Página web de OASIS
http://www.oasis-open.org

[L-16] Página web de BPMI
http://www.bpmi.org

[L-17] Página web de UN/CEFACT
http://www.uncefact.org

[L-18] Página web de W3C
http://www.w3c.org

[L-19] Página web de WfMC
http://www.wfmc.org

Apéndice A - Listado de código en PNML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<pnml>
  <net id="n1" type="WFNet">
    <place id="p1">
      <marking>
        <value>0</value>
      </marking>
      <name>
        <value>Inicio</value>
      </name>
      <initialMarking>
        <value>1</value>
      </initialMarking>
    </place>
    <transition id="t1">
      <name>
        <value>Tarea 1</value>
      </name>
    </transition>
    <place id="p2">
      <marking>
        <value>0</value>
      </marking>
      <name>
        <value>T1 Ejecutada</value>
      </name>
      <initialMarking>
        <value>0</value>
      </initialMarking>
    </place>
    <arc id="a1" source="p1" target="t1">
      <inscription>
        <value>1</value>
      </inscription>
    </arc>
    <arc id="a2" source="t1" target="p2">
      <inscription>
        <value>1</value>
      </inscription>
    </arc>
    <firingRule>
      <value> </value>
    </firingRule>
    <name>
      <value>Proceso Secuencial</value>
    </name>
  </net>
</pnml>
```

Apéndice B - Código Escrito utilizando PNK

```
1 package de.huberlin.informatik.pnk.app;
2 import de.huberlin.informatik.pnk.app.base.MetaApplication;
3 import de.huberlin.informatik.pnk.appControl.ApplicationControl;
4 import
de.huberlin.informatik.pnk.netElementExtensions.llNet.SimpleRule;
5
6 public class Enactment extends MetaApplication {
7
8 // este es el nombre de la aplicación
9 public static String staticAppName = "Enactment";
10
11 // constructor de la clase
12 public Enactment(ApplicationControl ac) {
13 super(ac);
14 }
15
16 public void run() {
17 SimpleRule rule =
18 (SimpleRule) net.getExtension("firingRule");
19 rule.simulateWithUserInteraction(this);
20 }
21 } // Enactment (Lanzamiento)
```

```
1 public class SimpleRule extends FiringRule {
2 ...
3 public void simulateWithUserInteraction(MetaApplication app) {
4
5 checkContextAndParseExtensions();
6 ApplicationControl ac = app.getApplicationControl();
7 Net net = (Net) getGraph();
8
9 Vector concessioned = getAllConcessioned( ); //Todas las transiciones
habilitadas
10
11 if(concessioned == null || concessioned.isEmpty()) return;
12
13 Transition transition = (Transition)
14 (new SelectObjectAction(ac, net, app, concessioned)).invokeAction();
15
16 while(transition != null) {
17 fire(transition);
18 concessioned = getAllConcessioned();
19 if(concessioned == null || concessioned.isEmpty()) return;
20 transition = (Transition)
21 (new SelectObjectAction(ac, net, app,
22 concessioned)).invokeAction();
23 }
24 }
25 } // clase SimpleRule
```

Apéndice C - WfNet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE netTypeSpecification SYSTEM "netTypeSpecification.dtd">
<netTypeSpecification name="WFNet">
   <extendable class="de.huberlin.informatik.pnk.kernel.Net">
      <extension name="firingRule"
      class="de.huberlin.informatik.pnk.netElementExtensions.WFnetFiringRule"/>
   </extendable>
   <extendable class="de.huberlin.informatik.pnk.kernel.Place">
      <extension name="marking"
      class="de.huberlin.informatik.pnk.netElementExtensions.NaturalNumber"/>
      <extension name="initialMarking"
      class="de.huberlin.informatik.pnk.netElementExtensions.NaturalNumber"/>
   </extendable>
      <extendable class="de.huberlin.informatik.pnk.kernel.Transition">
            <extension name="tipoTransicion"
      class="de.huberlin.informatik.pnk.netElementExtensions.base.StringExtension"/>
            <extension name="accion"
      class="de.huberlin.informatik.pnk.netElementExtensions.base.StringExtension"/>
            <extension name="condicion"
      class="de.huberlin.informatik.pnk.netElementExtensions.base.StringExtension"/>
           <extension name="direccion"
      class="de.huberlin.informatik.pnk.netElementExtensions.base.StringExtension"/>
           <extension name="rol"
      class="de.huberlin.informatik.pnk.netElementExtensions.base.StringExtension"/>
           <extension name="variables"
      class="de.huberlin.informatik.pnk.netElementExtensions.base.StringExtension"/>
           <extension name="leer_doc"
      class="de.huberlin.informatik.pnk.netElementExtensions.base.StringExtension"/>
           <extension name="escribir_doc"
      class="de.huberlin.informatik.pnk.netElementExtensions.base.StringExtension"/>
           <extension name="crear_doc"
      class="de.huberlin.informatik.pnk.netElementExtensions.base.StringExtension"/>
      <extension name="fechaHora"
      class="de.huberlin.informatik.pnk.netElementExtensions.base.StringExtension"/>
      </extendable>
   <extendable class="de.huberlin.informatik.pnk.kernel.Arc">
      <extension name="inscription"
      class="de.huberlin.informatik.pnk.netElementExtensions.NaturalNumber1"/>
   </extendable>
</netTypeSpecification>
```

Apéndice D - Manual de Usuario

Manual de usuario del Subsistema de Diseño y Lanzamiento de Procesos

Ejecución del proceso desde el Subsistema de Lanzamiento de Procesos

Se seleccionamos el archivo que contiene el proceso diseñado, y que deseamos ejecutar. La Figura 40 muestra la opción del menú.

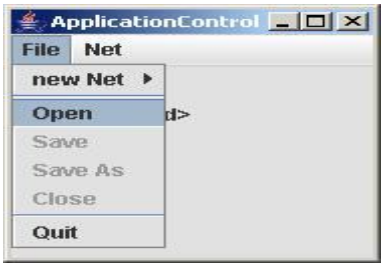


Figura 40. Opción para abrir archivo

La Figura 41 muestra archivos con extensión PNML que es posible cargar.

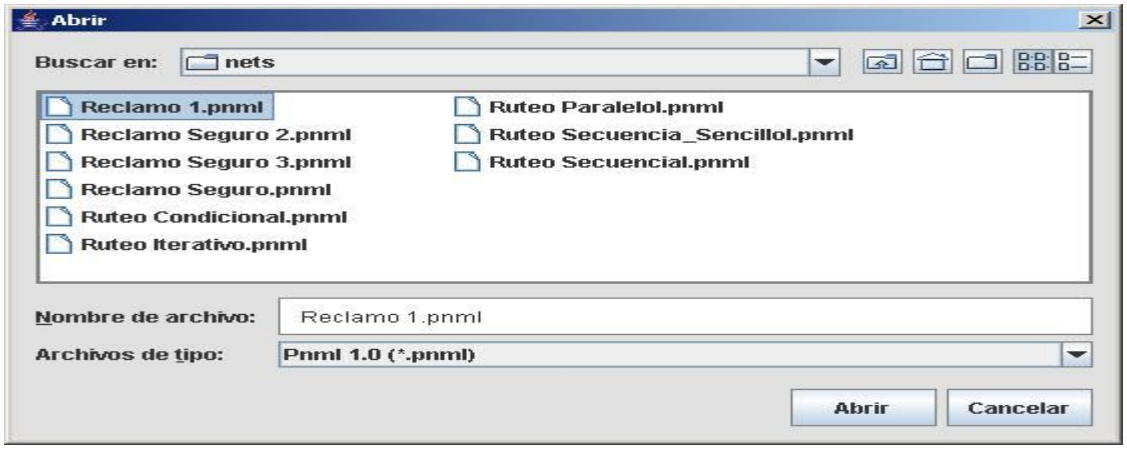


Figura 41. Navegador de archivos mostrando procesos disponibles

La Figura 42 muestra la opción que permite activar la opción de lanzamiento

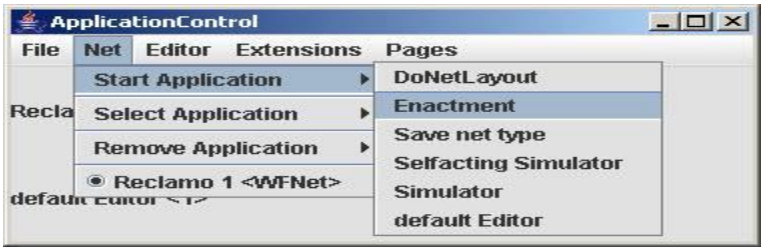


Figura 42. Opción para lanzar proceso

La Figura 43 muestra la opción para iniciar el lanzamiento del proceso previamente cargado.

Figura 43. Opción que da inicio al lanzamiento de proceso

Manual de usuario del Subsistema de Diseño y Lanzamiento de Procesos

Seguimiento a las actividades del proceso

Una vez lanzado el proceso, se puede dar seguimiento a las actividades restantes a través del administrador de elementos de trabajo. Primero es necesario ingresar al sistema desde cualquier navegador. La Figura 44 muestra la pantalla que permite ingresar al sistema.

Figura 44. Página de inicio para el ingreso al sistema de seguimiento de actividades

Una vez dentro del sistema, se muestran las actividades disponibles. La Figura 45 muestra la pantalla que se le presenta al usuario con las actividades disponibles de acuerdo a su rol.

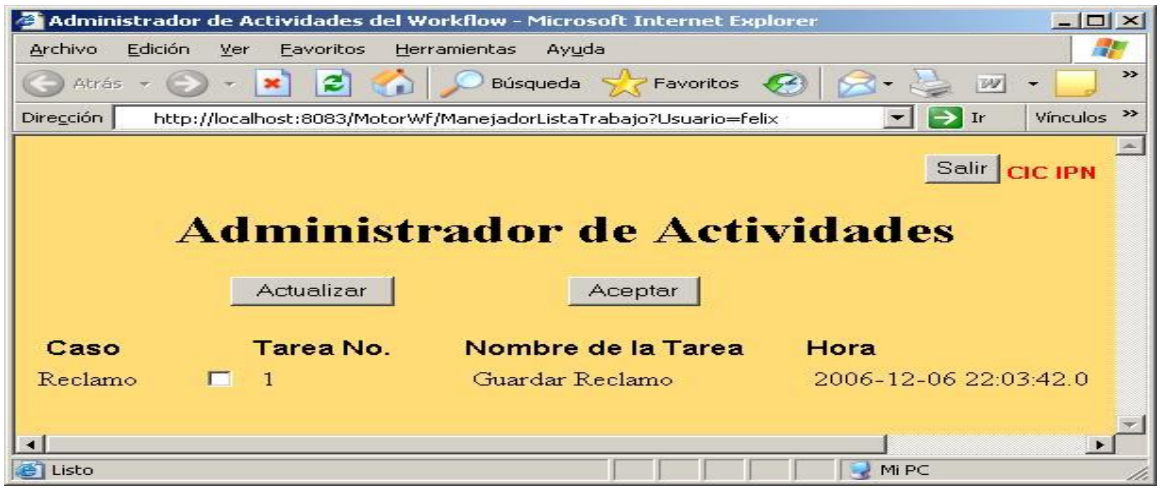


Figura 45. Página del administrador de elementos de trabajo

En está página se selecciona el elemento de trabajo que se desea realizar. Dando paso a la siguiente página que muestra el detalle de la actividad. La Figura 46 muestra la página mostrada al usuario.

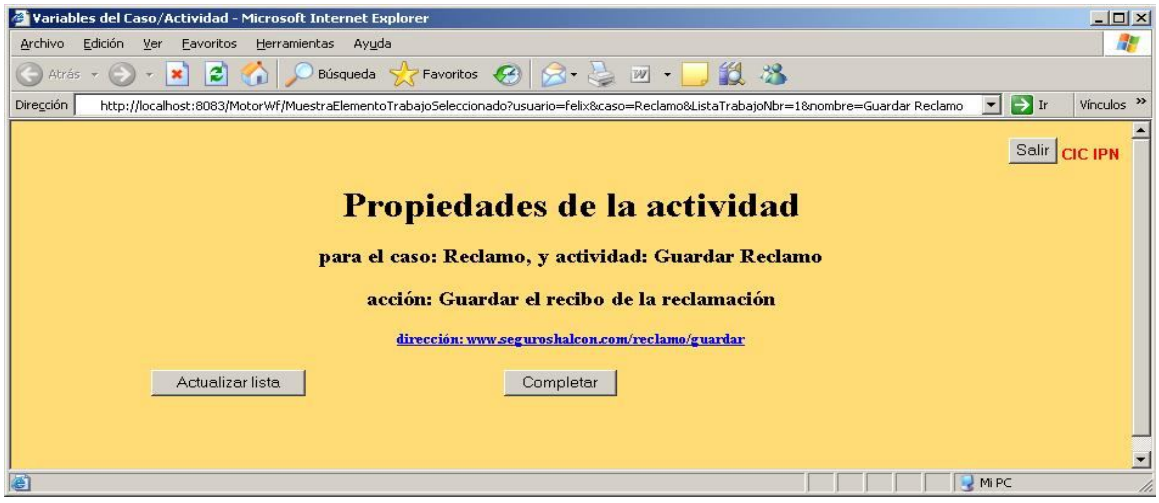


Figura 46. Página que muestra el detalle de la actividad seleccionada

En esta página el usuario puede ver y completar explícitamente la culminación de la actividad, dando paso a la generación de las actividades restantes del proceso. Para las actividades restantes, las páginas mostradas al usuario son similares a las Figuras 45 y 46 por lo que no se muestran.

Apéndice E - Diccionario de datos

Rol			
Campo	Descripcion	Tipo de Dato	Tipo llave
rol	Clave del rol	Autonumérico	PK
nombre	Nombre del Rol	Varchar(255)	

Usuario			
Campo	Descripción	Tipo de Dato	Tipo llave
usuario	Identificador del usuario	Varchar(255)	PK
password	Clave del Usuario	Varchar(20)	
nombre	Nombre completo del Usuario	Varchar(255)	

rol_usuario			
Campo	Descripción	Tipo de Dato	Tipo llave
rol	Clave del rol	Integer	PK,FK
usuario	Clave del Usuario	Varchar(255)	PK,FK

listaTrabajo			
Campo	Descripción	Tipo de Dato	Tipo llave
listaTrabajo	Identificador de la lista de trabajo de cada usuario	Autonumérico	PK
usuario	Clave del Usuario	Varchar(255)	FK
hora	Hora de creación de la lista de trabajo	Datetime	

Caso			
Campo	Descripción	Tipo de Dato	Tipo llave
caso	Nombre del caso	Varchar(255)	PK
horaInicio	Hora de inicio	Datetime	
horaFin	Hora de fin	Datetime	

elementoTrabajo_Lista			
Campo	Descripción	Tipo de Dato	Tipo llave
elementoNo	Consecutivo de la tarea a realizar	Autonumérico	PK
Elemento	Identificador único de la tarea a realizar	Varchar(255)	PK
accion	Descripción de la tarea a realizar	Varchar(255)	
horaInicio	Hora en la que el motor de Workflow inserto la tarea	Datetime	PK
horaFin	Hora en la que se concluyo con la tarea	Datetime	
caso	Caso al que corresponde esta tarea	Varchar(255)	FK

URL	Dirección web a la que hay que acceder para completar la tarea	Varchar(255)	

elementoTrabajo_Comun			
Campo	Descripción	Tipo de Dato	Tipo llave
Elemento	Identificador único de la tarea a realizar	Varchar(255)	PK
nombre	Descripción de la tarea a realizar	Varchar(255)	
caso	Caso al que corresponde esta tarea	Varchar(255)	FK
elementoNo	Identificador de la tarea a realizar	Autonumérico	FK
listaTrabajo	Identificador de la lista de trabajo de cada usuario	Entero Largo	FK
horaInicio	Hora en la que el motor de Workflow inserta la tarea	Datetime	

elementoTrabajo_Seleccionado			
Campo	Descripción	Tipo de Dato	Tipo llave
usuario	Clave del usuario que seleccionó la tarea	Varchar(255)	FK
elementoNo	Identificador de la tarea a realizar	Entero Largo	FK
elemento	Identificador único de la tarea a realizar	Varchar(255)	FK
caso	Caso al que corresponde esta tarea	Varchar(255)	FK
hora	Hora en la que el usuario selecciona la tarea	Datetime	

elementoTrabajo_Realizado			
Campo	Descripción	Tipo de Dato	Tipo llave
usuario	Clave del usuario que realizó la tarea	Varchar(255)	FK
elementoNo	Identificador de la tarea a realizada	Entero Largo	FK
elemento	Identificador único de la tarea realizada	Varchar(255)	FK
caso	Caso al que corresponde esta tarea	Varchar(255)	FK
hora	Hora en la que el usuario realizó la tarea	Datetime	

Apéndice F - Código del Subsistema de Lanzamiento de Procesos (Motor Workflow)

Clase principal para abrir la base de datos:

```
package de.huberlin.informatik.pnk.netElementExtensions.llNet.Wf;

import java.sql.*;
import java.util.Calendar;

public class InsertarBD {
  protected ResultSet resultSet = null;
  protected Connection con = null;

  protected String armaHORAFECHA() {
    String fechaHora = new String();
    Calendar rightNow = Calendar.getInstance();

    dateTime = (new Integer(ahora.get(Calendar.YEAR))).toString() + "-";
    fechaHora = fechaHora + (new Integer(ahora.get(Calendar.MONTH)+1)).toString() +
"-";
    fechaHora = fechaHora + (new
Integer(ahora.get(Calendar.DAY_OF_MONTH))).toString() + " ";
    fechaHora = fechaHora + (new
Integer(ahora.get(Calendar.HOUR_OF_DAY))).toString() + ":";
    fechaHora = fechaHora + (new Integer(ahora.get(Calendar.MINUTE))).toString() +
":";
    fechaHora = fechaHora + (new Integer(ahora.get(Calendar.SECOND))).toString();

    return fechaHora;
  }

  protected void getConexionBD() {
    try {
      Class.forName("com.mysql.jdbc.Driver").newInstance();
      con =
DriverManager.getConnection("jdbc:mysql://localhost/Wf?user=root&password=picasso
74");
    }
    catch (Exception e) {
      System.out.println("InsertaBD METODO getConexionBD: ");
      System.out.println(e);
    }
  }
```

```
  protected void destruyeConexionBD() {
    try {
      if (con != null) con.close();
    }
    catch (SQLException ignored) {
      System.out.println("InsertaBD METODO destruyeConexionBD: ");
    };
  }
}
```

Clase provista por el PNK para lanzar un proceso:

```
package de.huberlin.informatik.pnk.app;

import java.util.*;

import de.huberlin.informatik.pnk.app.base.*;
import de.huberlin.informatik.pnk.appControl.*;
import de.huberlin.informatik.pnk.appControl.base.*;
import de.huberlin.informatik.pnk.kernel.*;
import de.huberlin.informatik.pnk.kernel.base.*;
import de.huberlin.informatik.pnk.netElementExtensions.base.*;
/*
  Petri Net Kernel,
  Copyright 1996-1999 Petri Net Kernel Team,

  Petri Net Technology Group,
  Department of Computer Science,
  Humboldt-Universitaet zu Berlin, Germany
  All Rights Reserved.

  Do NOT modify the following lines!
  They contain CVS management information.
  $Log: Simulator.java,v $
  Revision 1.13  2001/12/12 12:51:22  gruenewa
  Neues tool zum auffalten von dawn netzen.

  Revision 1.11  2001/10/11 16:56:20  oschmann
  Neue Release

  Revision 1.10  2001/06/12 07:02:19  oschmann
  Neueste Variante...

  Revision 1.9  2001/06/04 15:44:59  efischer
  *** empty log message ***
```

```
  Revision 1.8  2001/05/11 17:20:36  oschmann
  Alpha Version... hoffentlich komplett und unzerstvrt.

  Revision 1.6  2001/03/06 09:12:14  mweber
  Patch: simpleRule was moved to llNet

  Revision 1.5  2001/02/27 21:28:22  oschmann
  Version mit viel neu Effekt: Menu bleibt sichtbar, Apps sind
  nummeriert, und es gibt Radiobuttons;-)

  Revision 1.2  2001/02/14 15:14:52  hohberg
  Uses new method simulateWithUserInteraction from SimpleRule

*/

/**
 * SelfActingSimulator.java
 *
 *
 * Created: Wed Jan 24 08:51:50 2001
 *
 * @author Alexander Gruenewald
 * @version
 */
/**
 * This class implements the standard PNK simulator application.
 */
public class Enactment
  extends MetaApplication {

  public boolean startAsThread = true;
  public static String staticAppName = "Enactment";

  public Enactment(ApplicationControl ac) {
    super(ac);
  }
  /**
   * Start the simulator, that means:
   * 1. Get the {@link de.huberlin.informatik.pnk.netElementExtensions.base.FiringRule
firingRule} for the net and
   * 2. call its method simulateWithoutUserInteraction
   */
  public void run() {

    FiringRule rule = (FiringRule) net.getExtension("firingRule");
```

```
    rule.simulateWithoutUserInteraction(this);
  }

} // SelfActingSimulator
```

Clase para Insertar o actualizar un caso en la base de datos:

```
package de.huberlin.informatik.pnk.netElementExtensions.llNet.Wf;

import java.sql.*;

public class ManejadorCaso extends InsertarBD {
    private PreparedStatement pstmtInsertarCasoWf = null;
    private PreparedStatement pstmtActualizarCasoWf = null;

  public ManejadorCaso() {

  }

  public void insertaFinCasoWf(String caso) {
    getConexionBD();
    try {
      pstmtActualizarCasoWf = con.prepareStatement("UPDATE Caso SET horaFin = ?
WHERE caso = ?");

      pstmtActualizarCasoWf.clearParameters();
      pstmtActualizarCasoWf.setString(1, armarHORAFECHA());
      pstmtActualizarCasoWf.setString(2, caso);
      pstmtActualizarCasoWf.executeUpdate();
    }
    catch (SQLException e) {
      System.out.println("ManejadorCaso METODO insertaFinCasoWf: ");
      System.out.println(e);
    }

    destruyeConexionBD();
  }

  public void insertaInicioCasoWf(String caso) {
    getConexionBD();
    try {
      pstmtInsertarCasoWf = con.prepareStatement("INSERT INTO Caso VALUES
(?,?,?)");

      pstmtInsertarCasoWf.clearParameters();
      pstmtInsertarCasoWf.setString(1, caso);
      pstmtInsertarCasoWf.setString(2, armarHORAFECHA());
```

```
      pstmtInsertarCasoWf.setString(3, armarHORAFECHA());
      pstmtInsertarCasoWf.executeUpdate();
    }
    catch (SQLException e) {
      System.out.println("ManejadorCaso METODO insertaInicioCasoWf: ");
      System.out.println(e);
    }

    destruyeConexionBD();
  }
}
```

Clase que maneja las transiciones de los procesos:

```
package de.huberlin.informatik.pnk.netElementExtensions.llNet.Wf;

import java.io.*;
import java.sql.*;
import de.huberlin.informatik.pnk.kernel.*;
import java.util.*;

public class ManejaTransicion  extends InsertarBD {

  public ManejaTransicion  () {

  }

  private Vector extraeRoles(String rolString) {
  Vector rolVector = new Vector();

  if (!(roleString.indexOf("AND") > 0) && !(rolString.indexOf("OR") > 0)) {
      rolString.trim();
      rolVector.add(roleString);
    }
    return roleVector;
  }

  private Vector obtieneActores(Vector rolVector) {
    Vector actorVector = new Vector();
    ResultSet resultSet = null;
    int counter = 0;
    PreparedStatement pstmtSelectActores = null;

    try {
      pstmtSelectActores = con.prepareStatement("SELECT usuario FROM rol_usuario
AS RU, rol AS R WHERE R.nombre = ? AND RU.rol = R.rol");
```

```
      while (counter < roleVector.size()) {
        if (((((String)roleVector.get(counter)).indexOf("AND")) < 0) &&
((((String)roleVector.get(counter)).indexOf("OR")) < 0)) {
          pstmtSelectActores.clearParameters();
          pstmtSelectActores.setString(1, (String)roleVector.get(counter));
          resultSet = pstmtSelectActores.executeQuery();
          while (resultSet.next()) {
            actorVector.add(resultSet.getString("usuario"));
          }
        }
        counter++;
      }
    }
    catch (SQLException e) {
      System.out.print("ManejaTransicion  METODO obtieneActores: ");
      System.out.println(e);
    }
    return actorVector;
  }

private void insertaElementoTrabajoLista(int foliosRequeridos, Transition transicion,
String caso) {
  String horaFecha = null;
  PreparedStatement pstmtInsertaFolio = null;
  horaFecha = new String(armarHORAFECHA());

    try {
        pstmtInsertaFolio = con.prepareStatement("INSERT INTO elementoTrabajo_Lista
VALUES (?,?,?,?,?,?,?)");
        pstmtInsertWLIList.clearParameters();
        pstmtInsertWLIList.setInt(1,0);
        pstmtInsertWLIList.setString(2, transicion.getId().toString());
        pstmtInsertWLIList.setString(3, transition.getExtension("action").toString());
        pstmtInsertWLIList.setString(4, horaFecha);
        pstmtInsertWLIList.setString(5, "0000-00-00 00:00:00");
        pstmtInsertWLIList.setString(6, caso);
        pstmtInsertWLIList.setString(7, transition.getExtension("address").toString());

        for (int i =0; i < foliosRequeridos; i++) pstmtInsertaFolio.executeUpdate();
    }
    catch (SQLException e) {
      System.out.print("ManejaTransicion  METODO insertaElementoTrabajoLista: ");
      System.out.println(e);
    }
    try {
      wait(1000);
```

```
    }
    catch(Exception e){}
  }

private Vector obtieneElemento(Transition transitcon, String caso) {
  ResultSet resultSet = null;
  int Elementos = 0;
  Vector elementoVector = new Vector();
  PreparedStatement pstmtObtieneElemento = null;

    try {
      pstmtObtieneElemento = con.prepareStatement("SELECT elementoNo FROM
elementoTrabajo_Lista WHERE elemento = ? AND caso = ? AND horaFin = ?");
      pstmtObtieneElemento.clearParameters();
      pstmtObtieneElemento.setString(1, transicion.getId().toString().trim());
      pstmtObtieneElemento.setString(2, caso);
      pstmtObtieneElemento.setString(3, "0000-00-00 00:00:00");
      resultSet = pstmtObtieneElemento.executeQuery();

      while (resultSet.next()) {
        elementoVector.add(Integer.toString(resultSet.getInt("elementoNo")));
      }
    }
    catch (SQLException e) {
      System.out.print("ManejaTransicion  METODO obtieneElemento: ");
      System.out.println(e);
    }
    return elementoVector;
  }

private void insertaElementoTrabajoComun(Vector actorVector, Transition transicion,
Vector Elemento) {
  int totalActores = 0;
  ResultSet resultSet = null;
  ResultSet resultadoTrabajoComun = null;
  PreparedStatement pstmtSelectListaTrabajo = null;
  PreparedStatement pstmtInsertarElementoComun = null;
  PreparedStatement pstmtChecarElemento = null;

  pstmtSelectListaTrabajo = con.prepareStatement("SELECT listaTrabajo FROM
listaTrabajo WHERE usuario = ?");
      while (totalActores < actorVector.size()) {
        pstmtSelectListaTrabajo.clearParameters();
        pstmtSelectListaTrabajo.setString(1, actorVector.get(totalActores).toString());
        resultSet = pstmtSelectListaTrabajo.executeQuery();
```

```
      if (resultSet.next()) {
        pstmtChecarElemento = con.prepareStatement("SELECT * FROM
ElementoTrabajo_Comun WHERE elemento = ? AND listaTrabajo = ?");
        for (int i = 0; i < Elemento.size(); i++) {
          pstmtChecarElemento.clearParameters();
          pstmtChecarElemento.setInt(1,
Integer.valueOf((String)Elemento.get(i)).intValue());
          pstmtChecarElemento.setString(2, resultSet.getString("listaTrabajo"));
          pstmtChecarElemento.executeQuery();
          resultadoTrabajoComun = pstmtChecarElemento.executeQuery();

          if (!resultadoTrabajoComun.next()) {
            pstmtInsertarElementoComun = con.prepareStatement("INSERT INTO
ElementoTrabajo_Comun VALUES (?,?,?,?,?,?)");
            pstmtInsertarElementoComun.clearParameters();
            pstmtInsertarElementoComun.setString(1, transicion.getId().toString());
            //pstmtInsertarElementoComun.setString(2, transicion.getName().substring(1,
transicion.getName().length() - 7));
            pstmtInsertarElementoComun.setString(2, transicion.getName());
            pstmtInsertarElementoComun.setString(3,
((Net)transicion.getGraph()).getName());
            pstmtInsertarElementoComun.setInt(4,
Integer.valueOf((String)Elemento.get(i)).intValue());
            pstmtInsertarElementoComun.setString(5, resultSet.getString("listaTrabajo"));
            pstmtInsertarElementoComun.setString(6, armarHORAFECHA());
            pstmtInsertarElementoComun.execute();
          }
        }
       }
       totalActores++;
     }
    }
    catch (SQLException e) {
      System.out.println("ManejaTransicion  METODO insertaElementoTrabajoComun:
");
      System.out.println(e);
    }
  }

public void insertaTransicion(Transition transicion, int elementosRequeridos) {
    Vector actorVector = new Vector();
    Vector rolVector = new Vector();
    String fechaHora = new String();
    Vector elementoVector = new Vector();

    getConexionBD();
```

```
    rolVector = extraeRoles(transition.getExtension("rol").toString());
    actorVector = obtieneActores(rolVector);
    insertaElementoTrabajoLista(elementosRequeridos, transicion,
((Net)transicion.getGraph()).getName());
        elementoVector = obtieneElemento (transicion,
((Net)transicion.getGraph()).getName());
    insertaElementoTrabajoComun (actorVector,transicion, elementoVector);
    destruyeConexionBD();
  }
private String creaCadenaElementoComun() {
  String string = new String();

  string = "SELECT ET.elementoNo FROM elementoTrabajo_Lista as ET,
elemetoTrabajo_Seleccionado AS ES WHERE ";
  string = string + "ET.elemento= ? AND ET.caso = ? AND ET.horaFin = ? AND ";
  string = string + "ET.elemento = ES.elemento AND ET.caso = ES.Caso AND
ET.elementoNo = ET.elementoNo";
  return string;
  }

public int existeEnElementoComun(Transition transicion) {
  ResultSet resultSet = null;
  int numeroDeElementos = 0;
  int valorBorrar = 0;
  int totalMarcaje = 1000000;
  PreparedStatement pstmtSelectElementoFin = null;
  PreparedStatement pstmtSelectElementoAceptado= null;

  getConexionBD();

  try {

    pstmtSelectElementoFin = con.prepareStatement("SELECT elementofolio FROM
ElementoTrabajo_Lista WHERE elemento = ? AND caso = ? AND horaFin = ?");
    pstmtSelectWLIEnd.clearParameters();
    pstmtSelectWLIEnd.setString(1, transicion.getId().toString());
    pstmtSelectWLIEnd.setString(2, ((Net)transicion.getGraph()).getName());
    pstmtSelectWLIEnd.setString(3, "0000-00-00 00:00:00");
    resultSet = pstmtSelectElementoFin.executeQuery();
    while (resultSet.next()) {
      numeroDeElementos ++;
    }

    pstmtSelectElementoAceptado = con.prepareStatement(creaCadenaElemComun());
    pstmtSelectElementoAceptado.clearParameters();
    pstmtSelectElementoAceptado.setString(1, transicion.getId().toString());
```

```
    pstmtSelectElementoAceptado.setString(2, ((Net)transicion.getGraph()).getName());
    pstmtSelectElementoAceptado.setString(3, "0000-00-00 00:00:00");
    resultSet = pstmtSelectElementoAceptado.executeQuery();
    while (resultSet.next()) {
      numeroDeElementos --;
    }

    for(int i = 0; i < transicion.getIncomingEdges().size(); i++) {
         if
((Integer.parseInt(((Place)(((Edge)(transicion.getIncomingEdges().get(i))).getSource())).g
etExtension("marking").toString())) < markingAmount) {
        markingAmount =
(Integer.parseInt(((Place)(((Edge)(transicion.getIncomingEdges().get(i))).getSource())).g
etExtension("marking").toString()));
       }
      //markingAmountString =
(((Place)(((Edge)(transicion.getIncomingEdges().firstElement())).getSource())).getExtens
ion("marking").toString());
    }
    destruyeConexionBD();
       if (numeroDeElementos == totalMarcaje) {
      totalMarcaje = 1000000;
      numeroDeElementos = 0;
      return 0;
    }
  }
  catch (SQLException e) {
    System.out.print("ManejaTransicion  METODO existeEnElementoComun: ");
    System.out.println(e);
  }

  valorBorrar = numeroDeElementos - totalMarcaje;
  totalMarcaje = 1000000;
  numeroDeElementos = 0;
  return valorBorrar;
  }
```

Clase del Motor de Workflow:

```
package de.huberlin.informatik.pnk.netElementExtensions.llNet;

/*
  Petri Net Kernel,
  Copyright 1996-1999 Petri Net Kernel Team,
  Petri Net Technology Group,
*/
```

```
import java.util.*;
import java.io.*;
import javax.swing.*;
import de.huberlin.informatik.pnk.app.*;
import de.huberlin.informatik.pnk.app.base.*;
import de.huberlin.informatik.pnk.app.WorklistGUI;

import de.huberlin.informatik.pnk.exceptions.*;
import de.huberlin.informatik.pnk.kernel.*;
import de.huberlin.informatik.pnk.netElementExtensions.base.*;

import de.huberlin.informatik.pnk.appControl.ApplicationControl;
import de.huberlin.informatik.pnk.netElementExtensions.llNet.*;

public class MotorWf extends Extension implements FiringRule
{

/**
   * Constructor specifying the extended extendable (a net).
  */
  public WFnetFiringRule(Extendable ext)
  {
       super(ext);
  }

public void simulateWithoutUserInteraction(MetaApplication app)
{

  ManejadorTransicion manejadorTransicion = new ManejadorTransicion();
  String tipoTransicion = new String();
  ChecarCondicionXpath condicion = new ChecarCondicionXpath ();
  int elementosRequeridos = 0;
  int elementosBorrar = 0;

  checkContextAndParseExtensions();

  ApplicationControl ac = app.getApplicationControl();

  Random integerGenerator =
    new java.util.Random();

  Net net = (Net) getGraph();

  ManejadorCaso manejadorCaso = new ManejadorCaso();
  manejadorCaso.insertaInicioCasoWf(net.getName());
```

```
 Vector conc = getAllConcessioned( );

 if(conc == null || conc.isEmpty()) {
  manejadorCaso.insertaFinCasoWf(net.getName());
  return;
 }
 Transicion tr = (Transicion)
  conc.get(integerGenerator.nextInt(conc.size()));

 while(tr != null && app.letrun) {

  synchronized(this) {
   (new EmphasizeObjectsAction(ac, net, app, tr)).invokeAction();
   tipoTransicion = tr.getExtension("tipoTransicion").toString();

   if (tipoTransicion.equalsIgnoreCase("task/begin")) {

    //Check if WLIs need to be deleted
    elementosBorrar = handleTransicion.existsInDBPool(tr);

    if (elementosBorrar > 0) {
     manejadorTrasicion.borrarTransicion(tr, elementosBorrar);
     try { wait(500); } catch(Exception e){}
    }

    elementosRequeridos = manejadorTransicion.existeEnBD(tr);
   if (elementosRequeridos > 0) {
    manejadorTransicion.insertaTransicion(tr, elementosRequeridos);
   }

   if ((elementosRequeridos == 0) && manejadorTransicion.aceptada(tr) &&
manejadorTransicion.completada(tr)) {
    fire(tr);
     try { wait(1000); } catch(Exception e){}
    conc = getAllConcessioned();
    if (conc.size() !=0) manejadorTransicion.borrarElementosNoHabilitados(conc);
    }
    elementosBorrar = 0;
    elementosRequeridos = 0;
   }

   if (tipoTransicion.length() == 0) fire(tr);

   try {
    wait(1);
```

```
      }
      catch(Exception e){}
      (new ResetEmphasizeAction(ac, net, app)).invokeAction();

    }

    conc = getAllConcessioned();
    if(conc == null|| conc.isEmpty()) {
      manejadorCaso.insertaFinCasoWf(net.getName());
      return;
    }
/*
    if (conc != null|| !conc.isEmpty()) {
      manejadorTransicion.borrarElementosNoHabilitados(conc);
      //try { wait(1000); } catch(Exception e){}
    }
*/
    tr = (Transicion)
    conc.get(integerGenerator.nextInt(conc.size()));
  }
}
```

Apéndice G - Código del Subsistema de Seguimiento de Actividades

```
package WfServlets;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class UsuarioLogin extends HttpServlet implements SingleThreadModel {
  public UsuarioLogin () {
    try {
      jbInit();
    }
    catch (Exception ex) {
      ex.printStackTrace();
    }
  }

  static final private String CONTENT_TYPE = "text/html";
  private HTML_UsuarioLogin despliegaHTML = new HTML_UsuarioLogin() ;

  public void init() throws ServletException {
  }
  //Process the HTTP Get request
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    despliegaHTML. despliegaHTML (out);
  }
  public void destroy() {
  }

  private void jbInit() throws Exception {
  }
}


package WfServlets;

import java.io.*;

public class HTML_ UsuarioLogin {
```

```
  public HTML_ UsuarioLogin() {

  }

  public void despliegaHTML(PrintWriter out) {


    out.println("<HTML>");
    out.println("<HEAD><TITLE>Acceso a Workflow</TITLE></HEAD>");
    out.println("<BODY BGCOLOR=#FFDC75>");
    out.println("<FORM action=loginhandling method=POST>");
    out.println("<H5><CENTER> </CENTER></H5>");
    out.println("<H5><CENTER><FONT COLOR=#ff0000
FACE=Arial></FONT> </CENTER></H5>");
    out.println("<H1><CENTER><FONT COLOR=#ff0000
FACE=Arial></FONT> </CENTER></H1>");
    out.println("<H1><CENTER><FONT COLOR=#ff0000 FACE=Arial>Sistema de
Workflow</FONT></CENTER></H1>");
    out.println("<H6><CENTER><FONT COLOR=#ff0000
FACE=Arial></FONT> </CENTER></H6>");
    out.println("<H1><CENTER><FONT COLOR=#ff0000 FACE=Arial></FONT>");
    out.println("<TABLE WIDTH=360 BORDER=0 CELLSPACING=2
CELLPADDING=0 HEIGHT=107>");
    out.println("<TR><TD WIDTH=38%>");
    out.println("<P ALIGN=RIGHT> <B><FONT
FACE=Arial>Usuario:</FONT></B></TD>");
    out.println("<TD WIDTH=62%> <INPUT NAME=usuario value=\"\"
TYPE=text SIZE=25></TD></TR>");
    out.println("<TR><TD WIDTH=38%>");
    out.println("<P ALIGN=RIGHT><B><FONT
FACE=Arial>Password:</FONT></B></TD>");
    out.println("<TD WIDTH=62%> <INPUT NAME=password value=\"\"
TYPE=password SIZE=25></TD></TR>");
    out.println("</TABLE><FONT COLOR=#ff0000
FACE=Arial></FONT></CENTER></H1>");
    out.println("<H1><CENTER><FONT COLOR=#ff0000 FACE=Arial>");
    out.println("<INPUT NAME=name TYPE=submit VALUE=Aceptar>");
    out.println("<INPUT NAME=name TYPE=reset VALUE=Cancelar>");
    out.println("</FORM></BODY></HTML>");

  }

}

package WfServlets;
```

```
//import com.mysql.jdbc.*;
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;


public class ManejadorUsuario extends HttpServlet implements SingleThreadModel  {
  protected Connection con = null;
  protected ResultSet rs = null;
  protected Statement stmt = null;
  protected PreparedStatement pstmt = null;
  static final private String CONTENT_TYPE = "text/html";
  //Initialize global variables

  public void init (ServletConfig config) throws ServletException {
    super.init(config);
    try{
      Class.forName("com.mysql.jdbc.Driver").newInstance();
      con = DriverManager.getConnection("jdbc:mysql://localhost/Wf ", "root",
"picasso74");
    }
    catch (Exception e) {
      System.out.println(e);
    }
  }

  public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();

    try {
      pstmt = con.prepareStatement("SELECT nombre FROM usuario WHERE usuario=?
AND password=?");
    }
    catch (SQLException e) {
      System.out.println(e);
    }

    String usuario = request.getParameter("usuario");
    String passWord = request.getParameter("password");

    try {
      pstmt.clearParameters();
      pstmt.setString(1, usuario);
```

```
      pstmt.setString(2, password);
      rs = pstmt.executeQuery();

      if (rs.next()) {

response.sendRedirect(("http://localhost:8083/Wf/ManejadorLista_Trabajo?usuario=" +
usuario));
      }

      else {
        printFalseLogin(out);
      }
    }
    catch (Exception e) {
      System.out.print("ManejadorUsuario METODO doPost:");
      System.out.println(e);
    }

  }

  public void destroy() {
    try {
      if (con != null) con.close();
    }
    catch (SQLException ignored) {};
  }
}
```

Clase complemento para mostrar los elementos de trabajo pendientes:

```
package WfServlets;

import java.io.*;

public class HTML_ManejadorListaTrabajo {

  public void despliegaHTML(PrintWriter out) {

       out.println("<HTML>");
    out.println("<HEAD><TITLE>Administrador de Actividades del
Workflow</TITLE></HEAD>");
    out.println("<BODY BGCOLOR=#FFDC75>");

    out.println("<FORM action=ManejadorSeleccionElementoTrabajo method=GET>");
    out.println("<H5 ALIGN=RIGHT><INPUT NAME=submit TYPE=submit
VALUE=\"Salir\">");
```

```
   out.println("<FONT COLOR=#ff0000 FACE=Arial>CIC IPN</FONT></H5>");
   out.println("<H1><CENTER>Administrador de Actividades</CENTER></H1>");
   out.println("<H1><CENTER><TABLE WIDTH=359 BORDER=0
CELLSPACING=2 CELLPADDING=0>");
   out.println("<TR><TD WIDTH=24%><INPUT NAME=submit TYPE=submit
VALUE=\"Actualizar\"></TD>");
   out.println("<TD WIDTH=25%> <INPUT NAME=submit TYPE=submit
VALUE=\"Aceptar\"></TD>");
   out.println("</TR></TABLE><!--SELECTION--><!--/SELECTION--
></CENTER></H1>");
   out.println("<P><CENTER><TABLE WIDTH=700 BORDER=0 CELLSPACING=2
CELLPADDING=0HEIGHT=131>");
   out.println("<TR><TD WIDTH=10% HEIGHT=23> <B><FONT
FACE=Arial>Caso</FONT></B></TD>");
   out.println("<TD WIDTH=5% HEIGHT=23> </TD>");
   out.println("<TD WIDTH=15% HEIGHT=23><B><FONT FACE=Arial>Tarea
No.</FONT></B></TD>");
   out.println("<TD WIDTH=25% HEIGHT=23><B><FONT FACE=Arial>Nombre de
la Tarea</FONT></B></TD>");
   out.println("<TD WIDTH=40% HEIGHT=23><B><FONT
FACE=Arial>Hora</FONT></B></TD></TR>");
  }

  public void finHTML(PrintWriter out, String usuario, String action, String usuario) {

     out.println("<INPUT TYPE=hidden VALUE=" + usuario +" NAME=usuario>");
     //out.println("<INPUT TYPE=hidden VALUE=" + caseName +"
NAME=caseName>");
     out.println("<INPUT TYPE=hidden VALUE=\"" + accion +"\" NAME=accion>");
     //out.println("<INPUT TYPE=hidden VALUE=" + caso +" NAME=caso>");
     out.println("</TABLE></CENTER></FORM>");
     out.println("</BODY></HTML>");
  }
}
```

Clase que muestra los elementos de trabajo pendientes:

```
package WfServlets;

import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ManejadorListaTrabajo extends HttpServlet implements SingleThreadModel
{
```

```
  protected Connection con = null;
  protected ResultSet  pstmtElementoComun = null;
  static final private String CONTENT_TYPE = "text/html";
  private HTML_|ManejadorListaTrabajo despliegaHTML = new
HTML_ManejadorListaTrabajo();

  public void init (ServletConfig config) throws ServletException {
    super.init(config);
    try{
      Class.forName("com.mysql.jdbc.Driver").newInstance();
      con =
DriverManager.getConnection("jdbc:mysql://localhost/Wf?user=root&password=picasso
74");
    }
    catch (Exception e) {
      System.out.println(e);
    }
  }

  private void printNoElementosHTML(PrintWriter out) {
     http://localhost:8083/Wf/usuarioLogin\"");
    out.println("<H5><CENTER> </CENTER></H5>");
    out.println("<H2>Noy hay Actividades para ti.<BR>");
  }

  private String crearSelectElemComun() {
    String selectStatement = new String();

    selectStatement = "SELECT EC.caso, EC.Elemento, EC.nombre, EC.horaInicio ";
    selectStatement = selectStatement + " FROM ElementoTrabajo_Comun AS EC,";
    selectStatement = selectStatement + " listaTrabajo AS LT";
    selectStatement = selectStatement + " WHERE LT.usuario = ?";
    selectStatement = selectStatement + " AND LT.listaTrabajo = EC.listaTrabajo";

    return selectStatement;
  }

private ResultSet GetElementoTrabajo(String usuario) {
    ResultSet resultSet = null;
    try {
      pstmtGetElementoTrabajo.clearParameters();
      pstmtGetElementoTrabajo.setString(1, actorID);
      pstmtGetElementoTrabajo.executeQuery();
      resultSet = pstmtGetElementoTrabajo.getResultSet();
    }
    catch (Exception e) {
```

```
      System.out.print("ManejadorListaTrabajo METHOD GetElementoTrabajo: ");
      System.out.println(e);
    }
    return resultSet;
  }

public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    int Elemento = 0;
    String horaInicio = new String();
    String caso= new String();
    String nombre = new String();
    boolean resultados = true;

    String actorID = request.getParameter("usuario");
    try {
      pstmtElementoComun= con.prepareStatement(crearSelectElemComun());
    }
    catch (SQLException e) {
      System.out.println(e);
    }
    rsElementoTrabajo = getElementoTrabajo (usuario);

    try {
      if (!rsElementoTrabajo.next()) {
        despliegaHTML.despliegaHTML(out);
        printNoElementosHTML (out);
        resultados = false;
      }
      if (resultados) {
         despliegaHTML.despliegaHTML(out);
         do {
            caso = rsCaseIDwliID.getString(1);
            elemento = rsCaseIDwliID.getInt(2);
            nombre = rsCaseIDwliID.getString(3);
            horaIinicio = rsCaseIDwliID.getString(4);

            out.println("<TR><TD WIDTH=10% HEIGHT=23>"+ caseID +"</TD> ");
            out.println("<TD WIDTH=5% HEIGHT=23> ");
            out.println("<INPUT TYPE=checkbox VALUE=\"" + elemento +"\" NAME=\"
elemento\"></TD> ");
            out.println("<TD WIDTH=15% HEIGHT=23> "+ elemento +"</TD>");
            out.println("<TD WIDTH=25% HEIGHT=23> "+ nombre +"</TD>");
            out.println("<TD WIDTH=40% HEIGHT=23> "+ horaInicio
```

```
+"</TD></TR>");
          //}
        } while(rsElementoTrabajo.next());
      }
    }
    catch (Exception e) {
      System.out.print("ManejadorListaTrabajo METODO doGet: ");
      System.out.println(e);
    }
    despliegaHTML.finHTML(out, actorID, action, caseID);
  }

  //Clean up resources
  public void destroy() {
    try {
      if (con != null) con.close();
    }
    catch (SQLException ignored) {};
  }
}
```

GLOSARIO

Actividad de un Proceso: Es un paso lógico o descripción de un trabajo que contribuye a realizar el proceso. Una actividad puede ser manual o automática.
Ad Hoc Routing: Es la habilidad de definir el ruteo del Workflow en tiempo de ejecución. En este caso la construcción de los caminos del Workflow se basa en la información ingresada por el usuario en tiempo real.
Análisis: La parte del proceso de desarrollo de software cuyo propósito principal es realizar un modelo del dominio del problema. El análisis hace énfasis en qué hacer, el diseño hace énfasis en cómo hacerlo.
AND-Join: Dos o más actividades que convergen a un punto de control común dentro del proceso.
AND-Split: Es el punto dentro del flujo del proceso donde se abre el flujo en varios caminos a seguir simultáneamente.
API: Application Programming Interface. Un conjunto de llamados a funciones de software los cuales les permiten a otras aplicaciones acceder a las funciones de un programa.
Arquitectura: La estructura organizacional de un sistema. Una arquitectura puede ser descompuesta recursivamente en: partes que interactúan entre si por medio de interfaces, relaciones que conectan las partes, y restricciones para ensamblar las partes.
Base de Datos: Datos que han sido organizados y estructurados de una manera disciplinada para que el acceso a la información de interés sea tan rápido como sea posible.
Business Process Reengineering (BPR): El acto de analizar un proceso de empresa existente, juzgando su conveniencia para el ambiente existente en el negocio, y cambiar este proceso radicalmente. BPR y Workflow son mutuamente exclusivos; ellos pueden ser llevados a cabo independientemente uno del otro.
Contenedor de Servlets: Es un agregado para un servidor web que ejecuta un servlet y responde enviando el contenido en html.
Diagrama: La representación gráfica de una colección de elementos del modelo, frecuentemente plasmada como un grafo conectado de arcos (relaciones) y vértices (otros elementos del modelo).
Diseño: La parte del proceso de desarrollo de software cuyo propósito principal es decidir cómo se construirá el sistema. Durante el diseño se toman decisiones estratégicas y tácticas para alcanzar los requerimientos funcionales y la calidad esperada.
ERP: (Enterpise Resource Planning, Planeación de Recursos Empresariales). Sistemas de información que permiten integrar y disponer de información generada por el sistema en forma inmediata.
Groupware: Aplicación que permite a los usuarios compartir datos a través de la red.
html: (Hypertext markup language). Lenguaje de marcas de hipertexto. Es un lenguaje de marcas que permite escribir páginas web con contenido estático.
http: (Hypertext Transfer Protocolo). Protocolo de transferencia de hipertexto. Protocolo que permite enviar y recibir peticiones de páginas web escritas en html.
Implementación: La definición de cómo está construido o compuesto algo. Por ejemplo: una clase es una implementación de un tipo, un método es una implementación de una operación.

Interacción: Una especificación de comportamiento cuyo fin es lograr un propósito específico. Abarca un conjunto de intercambios de mensajes entre un conjunto de objetos dentro de un contexto particular.
Metamodelo: Un modelo que define el lenguaje para expresar un modelo.
Modelo: Una abstracción semánticamente consistente de un sistema. Ver sistema.
Navegador: Es un programa que envía peticiones http a un servidor web, y despliega los datos que el servidor regresa.
OR-Join: Punto en común donde se espera a que una actividad, de varias disponibles, termine.
OR-Split: Es el punto dentro del flujo del proceso donde se sigue solo un camino de varios disponibles.
Procesos: Una serie de tareas realizadas por dos o más individuos o recursos en serie o en paralelo para alcanzar una meta común. También llamados Procesos de Negocio, Workflow, Procesos de Workflow o Workflow del Negocio.
Rol: El nombre que identifica al comportamiento especifico de una entidad, cuando ésta participa de contexto particular. Para una empresa: la responsabilidad asignada a un individuo en una organización o un grupo.
Servlet. Es un programa escito en Java que permite generar contenido dinámico.
Sistema: Una colección de unidades conectadas entre sí, que están organizadas para llevar a cabo un propósito específico. Un sistema puede describirse mediante uno o más modelos, posiblemente desde puntos de vista diferentes.
Servidor Web: Es un programa en espera de peticiones a través del puerto http, y que responde enviando contenido en html al solicitante.
Tecnología: Sistematización de los conocimientos y prácticas aplicables a cualquier actividad.
Tomcat: Es un motor de JSP y Servlets de código abierto, escrito en Java. Incluye un servidor web básico que hace fácil su uso para desarrollo.
WAPI*:* Workflow Application Programming Interface.
Workflow: La automatización de un proceso de negocio, total o parcialmente.