AUTOMATIZACIÓN DE FLUJOS DE PROCESOS UTILIZANDO REDES DE PETRI

Br. Rafael A. Guerrero G.
Tutor: Prof. Edgar Chacón

COMO REQUISITO PARA OBTENER
EL GRADO DE
INGENIERO DE SISTEMAS
DE LA
UNIVERSIDAD DE LOS ANDES
MÉRIDA, VENEZUELA
OCTUBRE 2006

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERíA

El jurado aprueba el proyecto de grado titulado "**Automatización de Flujos de Procesos utilizando Redes de Petri**" realizado por **Br. Rafael A. Guerrero G.** como requisito parcial para la obtención del grado de **Ingeniero de Sistemas**.

Fecha: Octubre 2006

Tutor: _____
Prof. Edgar Chacón

Jurado: _____
Prof. Judith Barrios

_____
Prof. Dulce Milagros Rivero

Índice general

Índice de cuadros

Índice de figuras

Agradecimientos

A Dios, Por haberme dado: la oportunidad de estar aquí, las fuerzas para de atravesar este camino y hacer que mis manos cada día sean mejores instrumentos pare seguir sirviendole.

A mis padres, Quienes me apoyaron en todo momento y nunca han dejado de hacerlo. Gracias por mostrarme el camino.

A Paola, quien estuvo en la recta final dándome su apoyo incondicional para poder alcanzarla.

A mis compañeros de trabajo, Demián por darme la luz para cambiarme la forma de ver las cosas; Ángel por su sencillez para hacerme entender lo inexplicable; Alejandro por hacer que siempre haga lo mejor y a Mario por enseñarme que la humildad es la mejor bondad que uno puede tener.

A mamá ULA, simplemente por que dentro de ella aprendí a ser lo que ahora soy.

Capítulo 1

Prefacio

1.1. Introducción

En los años 90's surge el término "Workflow" dentro del área de las ciencias computacionales y con él toda una serie de conceptos, herramientas de software, metodologías, trabajos de investigación, estándares y organizaciones sobre "Workflow", entre otros elementos, suficientes para que la tecnología "Workflow" sea de vital importancia en la actualidad para la industria de software y para las organizaciones que hacen uso de tecnología "Workflow".

Con la incorporación de la tecnología como ayuda en las actividades del ser humano, la forma de llevar a cabo estas ha cambiado y evolucionado de acuerdo a las necesidades que se presentan en la vida diaria. Una de las direcciones de esta evolución ha sido tratar de automatizar las actividades lo más completamente posible con ayuda de tecnologías de información. Organizaciones de diferentes sectores incorporan tecnología en sus procesos, con el fin de obtener dicha automatización y mejorar los resultados en sus objetivos de negocio. Como consecuencia a lo anterior, las organizaciones poseen sistemas de información que actualmente funcionan con el fin de organizar, automatizar y ejecutar sus procesos de negocios. La evolución de dichos sistemas de información puede verse por etapas:

- 1975 – 1985 Administradores de bases de datos.

- 1985 – 1995 Administración de interfaces de usuario.
- 1995 – Hoy Administración de "Workflow".

Con los sistemas de bases de datos en la primera etapa se tuvo la existencia de diversos sistemas de información, en los cuales se manejaba y administraba toda la información necesaria para llevar a cabo la producción de las empresas. Se lograron automatizar tareas, que antes se realizaban manualmente. En las siguientes etapas se buscó mejorar el flujo de la información, la obtención de la información de manera rápida y eficiente. Surgió la necesidad de incrementar la eficiencia, optimizar la productividad, acortar tiempos de procesos, tener un control sobre estos, así como también de reducir los costos y mejorar la gestión. Todo lo anterior como consecuencia al incremento de la competitividad que a su vez es resultado de la globalización que experimenta el mundo. Específicamente con la segunda etapa se logró mayor facilidad de utilización de los sistemas para los usuarios finales, logrado gracias a interfaces cada vez más interactivas y menos susceptibles de aceptar errores humanos. La administración basada en "Worklfow" ayuda precisamente en la administración del control de procesos de negocio en las organizaciones.

1.2. Motivación y Objetivos

En el presente la gran mayoría de sistemas empresariales se están orientando a la automatización de sus procesos, debido crecimiento de competitividad en el mercado; con la evolución de la informática y la aparición de nuevas tecnologías, permiten desarrollar nuevas aplicaciones para el mantenimiento y control de sus procesos de gestión y así poder mantenerse, crecer y desarrollarse en el entorno competitivo donde se encuentra.

Hoy en día existen gran cantidad de herramientas para el modelado de procesos, pero muy pocas son las herramientas que facilitan el control y monitoreo de estos procesos; el motivo de esta investigación es el desarrollo de un motor que

ayude a controlar y supervisar los procesos de negocios de sistemas administrativos de una forma sencilla y simple, dicha herramienta debe permitir:

a) Describir "Workfolw´s" y que en algunos casos se dan de manera paralela.

b) Monitorear el estado de los procesos desde el punto de vista del personal de programación de la producción. Varias personas intervienen en esta labor, y siguen una serie de normas que deben ser conocidas por los distintos actores.

c) Integrar el sistema de gestión con el sistema de producción.

El objetivo es el de construir una herramienta que facilite la integración entre los dos niveles mencionados anteriormente.

1.3. Objetivo

Diseñar un esquema de interacción entre las unidades de gestión de la producción con las unidades de producción a través del uso de los conceptos de "Workflow", donde existe paralelismo en la producción.

1.3.1. Objetivos Específicos

- Profundizar en el manejo de sistemas de "Workflow".
- Modelar los "Workflow" mediante Redes de Petri jerárquicas.
- Construir un motor que permita establecer las interacciones entre distintos actores que participan en los procesos de negocio, y que han sido definidas en el objetivo anterior mediante el uso de servicios WEB.
- Realizar pruebas sobre un conjunto de "Workflow´s".

1.4. Metodología

Para lograr los objetivos propuestos se han establecido los siguientes pasos:

- Conocer el estado del arte en el modelado de "Workflow".
- Definición del comportamiento de los distintos actores, y como interactúan entre ellos.
- Modelar flujos de documentos de trabajo en las áreas de producción. (El problema del modelado del proceso productivo está fuera del alcance del sistema a ser desarrollado).
- Selección de una plataforma de implantación.
- Construcción del sistema.

Capítulo 2

Antecedentes

2.1. Sistemas Dinámicos de Eventos Discretos (SDED)

En un sentido natural la evolución o dinámica de un sistema está determinada por el paso del tiempo. Pero la dinámica de los SDED no se define naturalmente como los tradicionales sistemas dinámicos. Los estados de un SDED evolucionan en instantes discretos de tiempo a diferencia de los Sistemas Dinámicos de Variables Continuas (SDVC).

Es de notar que en algunos casos los sistemas modelados por ecuaciones diferenciales son a tiempo discreto lo cual no debe confundirse con sistemas de eventos discretos, básicamente los sistemas a tiempos discreto tienen una estructura afín con SDVC a pesar de la semejanza con el nombre de SDED, en los cuales el tiempo no determina la evolución o transición de un estado a otro, sino la ocurrencia de un evento. Los SDED han sido modelados principalmente como autómatas y en términos de lenguajes formales, y han sido construidos por científicos e ingenieros queriendo modelar los sistemas de tipo tecnológico y de automatización de nuestra vida moderna.

El desarrollo de los SDED recibe una gran importancia en el campo de estudio de la teoría de Sistemas de Control. En particular, las definiciones de la dinámica,

las constantes de tiempo, controlabilidad y observabilidad, clásicamente estudiadas en ingeniería, han jugado y continuarán jugando papeles importantes en el desarrollo de los modelos y herramientas de los SDED.

Desde que los SDED se han considerado sistemas artificiales (creados por el hombre) su aplicación está orientada a la descripción de sistemas tecnológicos y no a los de tipo físico-natural. Si consideramos, un sistema industrial flexible el cual actúa recíprocamente bajo el mando de las computadoras y los operadores humanos a través de pantallas de despliegue o mando de materiales real, veremos que ellos son modelados apropiadamente como SDED. Esto tiene implicaciones en las aplicaciones prácticas de los SDED en dos sentidos. El primero, que la utilización de procesos de interfaz y el lenguaje son factores más importante en los SDED que en los SDVC, como ejemplo podemos observar que un documento de recepción de inventario o creación de una orden de trabajo (proceso de automatización industrial), necesita intervención humana con el computador considerado como un SDED y ello requiere un lenguaje mixto de tipo jerárquico. El segundo, que los SDED son sistemas hechos por el hombre que escapan generalmente de las leyes físicas del entorno, las soluciones óptimamente pueden ser encontradas por un conveniente algoritmo artificial y no por leyes naturales invariables de la física; en este sentido, la acción de control o supervisión de los SDED tiene más grado de libertad que en los SDVC, ya que se ejecutan a conveniencia humana. Los ejemplos mas utilizados de SDED son:

- Sistema de Manufactura (máquinas, secuencia de operadores, operadores de inspección, etc.).Los problemas a considerar son las propiedades de frontera de las colas, o si una cola tiene eventualmente un nivel más alto que algún limite que podamos llevar a cabo para una cola de longitud fija, problemas de producción óptima.

- Sistema de Robótica (operación de lugar, operación de ensamblaje, etc.). Si la sucesión de las operaciones alcanza el objetivo y tiene éxito a distancia,

aún cuando hay ciertas perturbaciones o si una operación de ensamblaje o de inspección se ejecuta con las exigencias establecidas.

- Las computadoras y redes de computadoras (ejemplos, circuitos lógicos, interfaz, los protocolos de comunicación, etc.). Lógicamente la importancia es que reconozca las posibles sucesiones de entradas transformando el problema de la caja negra en una transmisión confiable entre las computadoras, es un reto de dimensión real que motiva al diseño adecuado de las redes.
- Comunicación en redes (el modo de traslado asíncrono (MTA) de redes).
- Sistemas de Tráfico (señales de tráfico, la rampa de medida, los sistemas automatizados para el funcionamiento de las carreras en grupo, etc.).
- Las operaciones en lote en procesos industriales (la secuencia de mezclar las operaciones, o las secuencias de operadores de procesos, etc.).
- Ecuación Diferencial General no Lineal (los sistemas de control espacial, o sistemas de directores como controladores con una planta de eventos discretos).

2.2. Sistemas de Eventos Discretos (SED)

Un Sistema de Eventos Discretos se define como un sistema donde la evolución de los estados depende de la ocurrencia de eventos en un conjunto discreto dado. Un evento se puede pensar como una ocurrencia instantánea y es causada por una transición de valor (discreto) de estado de un sistema. Un evento puede ser identificado como:

1. El activar una tecla o botón.
2. Una ocurrencia espontánea, (por ejemplo encender una máquina).
3. O el resultado de varias condiciones satisfechas en un instante (por ejemplo el nivel de un tanque).

2.2.1. Propiedades de los SED

Se puede resumir en tres puntos característicos los que determinan un SED:

1. Los espacios de estado son discretos.
2. Los mecanismos de transición de estados son dirigidos por eventos.
3. La dinámica está determinada por el conjunto de eventos y el orden en que ocurren.

Ejemplo. Sistema de Almacenamiento. En la Figura 2.1 se muestra un sistema de almacenamiento con la siguente dinámica.

$x(t)$: el número de productos almacenados.

$\sum = a, b$, el alfabeto :

- a : La llegada del producto al almacén.
- b : La salida del producto del almacén.

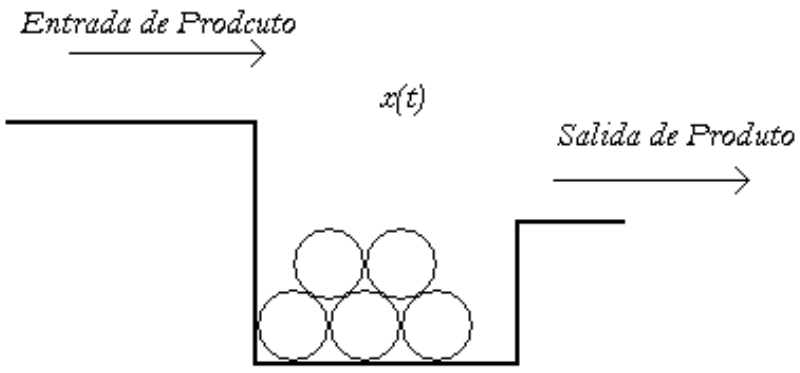

Figura 2.1: Sistema de Almacenamiento

2.2.2. Clasificación de los modelos para SED

Los modelos de los SED pueden ser clasificados como:

1. Modelos Temporizados o No Temporizados: Los Modelos Temporizados permiten especificar los instantes de intervalos de tiempo en que ocurren los eventos. Mientras que los Modelos no Temporizados no especifican los instantes de tiempo en que ocurren los eventos.

2. Modelos Lógicos y Modelos de Rendimiento: Los Modelos Lógicos son aquellos (Temporizados o no Temporizados) que están asociados a abordar el análisis del comportamiento lógico de un SED y de la secuencia de estados por evento generado. Los Modelos de Rendimiento son a su vez asociados a abordar el análisis desempeñado, y permiten responder preguntas específicas.

3. Modelos Algebraicos: Son aquellos que visualizan o capturan y describen las trayectorias de SDED a través de ecuaciones algebraicas, del mismo modo que SDVC son descritos por ecuaciones diferenciales o en diferencias. Los modelos lógicos estudian y clasifica la secuencia de cadenas de eventos que el proceso pueda generar. Los eventos son elementos de un conjunto finito.

Una clasificación de los sistemas dinámicos se muestra en la figura Figura 2.2

2.3. Autómatas

En electrónica un autómata es un sistema secuencial, aunque en ocasiones la palabra es utilizada también para referirse a un robot. Puede definirse como un equipo electrónico programable en lenguaje no informático y diseñado para controlar, en tiempo real y en ambiente industrial, procesos secuenciales. Sin embargo, la rápida evolución de los autómatas hace que esta definición no esté cerrada. Los Autómatas se pueden clasificar en:

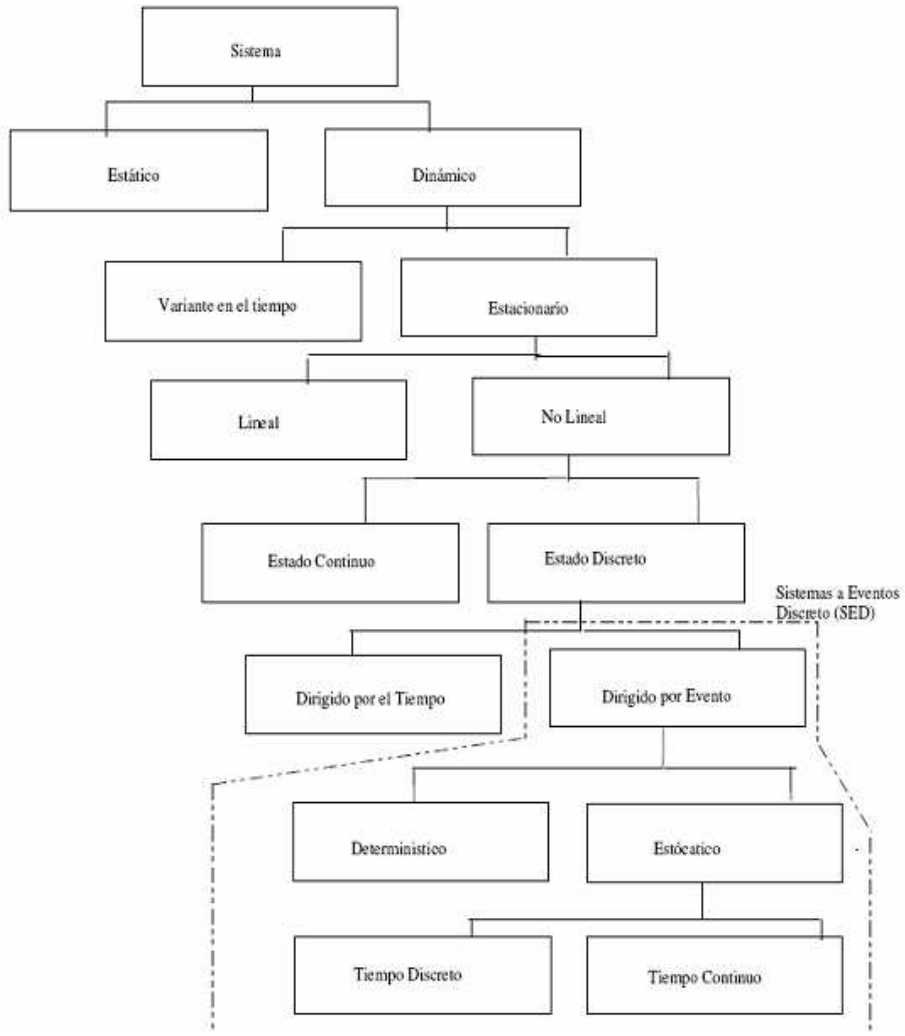

Figura 2.2: Clasificación de los Sistemas

2.3.1. Autómatas de Estado Finitos

Un autómata finito o máquina de estado finito es un modelo matemático de un sistema que recibe una cadena constituida por símbolos de un alfabeto y determina si esa cadena pertenece al lenguaje que el autómata reconoce.

2.3.2. Autómatas Finitos Deterministas

Los autómatas finitos son máquinas abstractas que procesan palabras, las cuales son aceptadas o rechazadas; ver Figura 2.3.

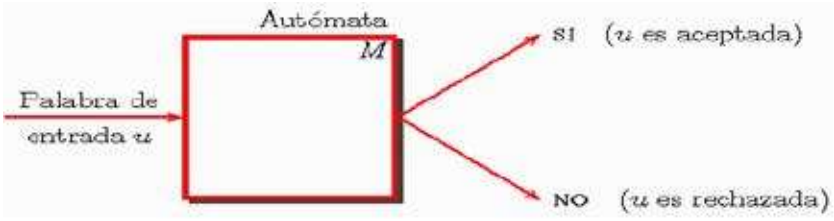

Figura 2.3: Autómata Finito Determinista

2.4. Redes de Petri

Las redes de Petri representan una alternativa para modelar sistemas, sus características hacen que, para algunos problemas las redes de Petri funcionen de una manera natural. Las PN (Petri Net) fueron inventadas por el alemán Carl Adam Petri en 1962. En su tesis doctoral "Kommunikation mit Automaten" (Comunicación con autómatas), establece los fundamentos para el desarrollo teórico de los conceptos básicos de las RdP. Las RdP son consideradas una herramienta para el estudio de los sistemas. Con su ayuda podemos modelar el comportamiento y la estructura de un sistema, y llevar el modelo a condiciones límite, que en un sistema real son difíciles de lograr o muy costosas.

Para entender mejor que es una red de Petri observe la figura Figura 2.4.

Las RdP son un grafo bipartito formado por dos tipos de nodos: lugares y transiciones. Los nodos solo pueden ser conectados por arcos; se puede afirmar que el uso de las RdP ó PN se debe a:

- Herramienta matemática para modelar concurrencia, comunicación y sincronización.

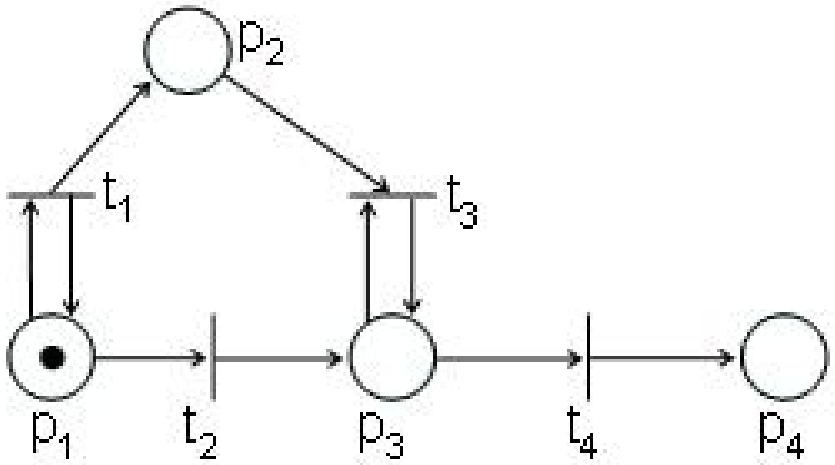

Figura 2.4: Ejemplo de una RdP

- Populares porque modelan de forma natural sincronización, eventos asíncronos, concurrencia, forma de compartir recursos, etc.
- Útiles en modelado y análisis de sistemas paralelos y concurrentes, de protocolos, evaluación de rendimiento y sistemas tolerantes a fallos.
- Multitud de variantes: coloreadas, con tiempo, orientadas a objetos, etc.

2.4.1. Red de Petri Estándar

La estructura de una Red de Petri Estándar es un grafo bipartito que incluye un conjunto de lugares P, un conjunto de transiciones T y un conjunto de arcos dirigidos E. Un lugar p es una entrada para una transición t si existe un arco incidente a la transición (p,t). Un lugar p es una salida de una transición t si existe un arco incidente de la transición al lugar (t,p). El conjunto de arcos puede ser particionado en el conjunto de arcos de entrada a transiciones Ei y el conjunto de arcos de salida de transiciones Eo.

Formalmente:

$RDP = (P, T, E)$

$P = (p_1, p_2, ...p_n)$, es el conjunto finito no vacío de lugares.

$T = (t_1, t_2, ...tn)$, es el conjunto finito no vacío de transiciones.

$E = E_i \cup E_o$, es el conjunto de arcos. Donde:

$E_i \subset (PxT)$: Arcos de entrada.

$E_o \subset (PxT)$: Arcos de salida.

2.4.2. Red de Petri Blanco y Negro (RDP B&W)

Las Redes de Petri pueden contener fichas, gráficamente dibujados como puntos negros. Una Red de Petri con fichas se llama Red de Petri con marcado. El estado de una Red de Petri con marcas está definido por el número m_i de fichas contenido en cada lugar p_i. El estado de una Red de Petri se llama usualmente "marcación de la red´´y se denota por un vector conocido como vector de marcación $[n_1 n_2 \dots n_m]^T$

Una Red de Petri marcada es una tupla:

$RDP = (P, T, E, X_0)$

$P = (p_1, p_2, ...p_n)$, es el conjunto finito no vacío de lugares.

$T = (t_1, t_2, ...tn)$, es el conjunto finito no vacío de transiciones.

$E = E_i \cup E_o$, es el conjunto de arcos.

Donde:

$E_i \subset (PxT)$: Arcos de entrada.

$E_o \subset (PxT)$: Arcos de salida.

$X : PN, X_0$ Marcado inicial (número de fichas en cada lugar, el estado de la red).

Es posible definir la ejecución de una Red de Petri con marcas. La misma se define a través de las siguientes reglas:

1. Una transición está habilitada cuando todos sus lugares de entrada contienen al menos una ficha.

2. Una transición habilitada se puede ejecutar o "disparar´´, removiéndose una fihca de cada lugar de entrada y colocando uno en cada lugar de salida.

3. Cada disparo de una transición modifica la distribución de las fichas, y por ello produce un nuevo marcado en la red.

De aquí en más notaremos m_{pi} al número de fichas en el lugar p_i, $\#(p_a, t_b)$ vale 1 si existe un arco desde el lugar p_a a la transición t_b y 0 en cualquier otro caso; de forma similar, $\#(t_a, p_b)$ vale 1 si existe un arco desde la transición t_a al lugar p_b y 0 en cualquier otro caso. Una transición t_b está habilitada si:

$$m_{pi} \geq \#(p_i, t_b) \forall p_i \in P$$

El resultado de la ejecucion de t_b es un nuevo marcado definido como:

$$m_{pi}^{'} = m_{pi} - \#(p_i, t_b) + \#(t_b, p_i) \forall p_i \in P$$

2.4.3. Propiedades De Las Redes de Petri

- **Alcanzable**

  Un marcado M' se dice alcanzable inmediatamente desde M si M' puede ser obtenido disparando una transición habilitada en M. La ejecución de una Red de Petri permite definir una secuencia de marcados $M_0, M_1, M_2, ...$ y una secuencia de transiciones $t_1, t_2, ...$

  Un marcado M" se dice alcanzable desde M si existe una secuencia de disparos de transiciones que mueve el estado de una Red de Petri desde M a M".

  Definimos el conjunto alcanzable $R(M_0)$ de una Red de Petri como el conjunto de todos los marcados que son alcanzables desde M_0. El conjunto alcanzable de una Red de Petri puede ser representado por un árbol, que se puede construir colocando el marcado inicial M_0 como raíz y como hijos todos los marcados inmediatamente alcanzables desde éste y repitiendo este procedimiento para cada marcado que se agrega. Existe la posibilidad de generar estructuras infinitas cuando a partir de un marcado M' se puede alcanzar un marcado M" y viceversa. En estos casos dejamos de expandir

el árbol apenas llegamos a un marcado agregado anteriormente. Llamaremos a estos marcados "marcados duplicados". Durante la construcción del árbol de alcance, podemos encontrar también "marcados muertos", aquellos en los que ninguna transición está habilitada. Los marcados duplicados y muertos constituyen las hojas el árbol.

- **Segura** Una Red de Petri con marcas se dice segura si el número de fichas en cada lugar es 1 para todos los marcados de $R(M_0)$. En otras palabras la red es segura si es 1-acotada, es decir si $k = 1$.

- **Acotamiento** Una Red de Petri se dice k-acotada si el número de fichas en cada lugar es $\leq k$ para todos los marcados de $R(M_0)$.

- **Conservativa** Una Red de Petri se dice estrictamente conservativa si, para todos los marcados de $R(M_0)$, la suma de fichas en una Red es constante, es decir, las fichas se mueven por la red sin ser creadas o destruidas.

$$\forall M_k \in R(M_0), \sum_{i=1}^{n} m_{k_i} = k$$

Esto implica que para cada transición el número de arcos de entrada es igual al número de arcos de salida. Una Red de Petri se dice conservativa con respecto a un vector de componentes no negativas $W = w_1, w_2, w_3, ..., w_n$ si:

$$\forall M_k \in R(M_0), \sum_{i=1}^{n} w_i m_{k_i} = k$$

Cuando una Red de Petri se utiliza para modelar sistemas reales, el concepto de conservación es de suma relevancia, pues generalmente las fichas son asociadas a recursos, los cuales, no pueden variar (p.e. el número de procesadores), y como consecuencia, su número no puede variar en una Red de Petri.

- **Vivacidad** Una transición se dice viva si, para todo $M \in R(M_0)$, hay un marcado M', alcanzable desde M, que habilita la transición t_1. Una Red de Petri se dice viva si todas sus transiciones t_i son vivas. El concepto de vida es muy importante en sistemas de computadoras, y usualmente es asociado con la ausencia de estados de bloqueos mutuos o "deadlocks".

  Dado un estado inicial M_0, una transición de una RDP está:

  - Viva a nivel L0 (muerta): Si nunca va a poder dispararse.
  - Viva a nivel L1: Si hay una secuencia desde M_0 tal que la transición puede dispararse al menos una vez.
  - Viva a nivel L2: Si la transición puede dispararse al menos k veces dado un entero positivo k.
  - Viva a nivel L3: Si existe alguna secuencia infinita en la que la transición aparece infinitas veces.
  - "Viva a nivel L4: Si la transición está viva a nivel L1 para cada estado alcanzable desde M_0.

- **Reiniciabilidad Y Estado Inicial** Una RDP es reinicializable si, para cada marcación M alcanzable desde M_0, M_0 es alcanzable desde M. Una marcación M' es un estado inicial si, para cada marcación M alcanzable desde M_0, M' es alcanzable desde M.

- **Cobertura** Una marcación M es cubierta si existe M' alcanzable desde M_0 tal que $M'(p) \geq M(p)$ para todo los lugares p. donde $M(p)$ es el número de tokens que tiene el lugar p.

- **Persistente** El subconjunto de las Redes de Petri que se obtiene exigiendo que, para todo par de transiciones t_1 y $t_2 \in T$, con $t_1 \neq t_2$, y para cualquier marcado alcanzable, el disparo de t_1 no puede deshabilitar el disparo de t_2. Estas redes son llamadas persistentes. Luego, si una transición está habilitada, permanece habilitada hasta que es disparada.

- **Distancia Sincrónica** La distancia sincrónica entre dos transiciones t_1 y t_2 en una RDP viene dada por:

$$d_{12} = max_{\sigma} \mid \sigma(t_1)\sigma(t_2) \mid$$

Donde es una secuencia de disparo comenzando en alguna marcación M alcanzable desde M_0 y σ_{ti} es el número de veces que la transición t_i, $i = 1, 2..$ dispara en σ

- **Imparcialidad**

  - **Imparcialidad acotada** Dos transiciones son imparcialmente acotadas si hay un número de veces que una transición puede dispararse mientras la otra no se dispara, está acotada.
  - **Imparcialidad incondicional** Una secuencia de disparo se dice que tiene imparcialidad incondicional si cada transición aparece infinitamente en la secuencia de disparo.

2.4.4. Extensiones de las Redes de Petri

Las extensiones fueron introducidas para aumentar el poder de modelado de la herramienta y de esta manera poder abordar todas las características de un sistema. En general las extensiones a la teoría de RdP dependen del modelo o la aplicación donde se estén usando, y existen en la actualidad varias extensiones siendo de interés en este proyecto las Redes de Petri coon tiempo.

- Redes de Petri Coloreadas (CRDP).
- Redes de Petri con Tiempo.
- Redes de Petri Estocásticas.

2.4.5. Redes de Petri con Tiempo

Las Redes de Petri Estándar no incluyen concepto alguno de tiempo, por ello, solamente es posible describir solo la estructura lógica de los sistemas y no su evolución temporal.

Existe una amplia variedad de extensiones adicionales de las RdP, que básicamente consisten en adicionar tiempos a las diferentes componentes del grafo bipartito que constituyen la red.

Una Red de Petri con tiempo, puede ser definida como:

$$\forall TRdP = P, T, A, M_0,$$

Donde P, T, AyM_0 se definen como antes y Θ es el conjunto de retardos asociados a los elementos de la red. La introducción del tiempo en el modelo permite la descripción de comportamientos dinámicos de los sistemas, considerando la evolución de estados y la duración de cada acción tomada por el sistema. Hay múltiples formas diferentes de introducir el concepto de tiempo, entre estas formas se destacan:

- Tiempo En Los Estados.
- Tiempo En Los Fichas.
- Tiempo En Los Arcos.
- Tiempo En Las Transiciones.

2.4.6. Árbol de Alcance

Murata (?, ?) propone el un algoritmo para encontrar el árbol de alcance, que es una versión más de tallada del que propone Peterson (?, ?). Para hacer el árbol finito se introduce un símbolo ω, el cual puede considerarse como "infinito" y que tiene las siguientes propiedades para cada entero n, $\omega > n$, $\omega \pm n = \omega$ y $\omega \geq \omega$. El algoritmo es el siguiente:

Paso 1) Etiquete la marcación inicial M_0 como la raíz y llámelo *"nuevo"*.

Paso 2) Mientras existan marcaciones nuevas haga lo siguiente:

Paso 2.1) Seleccione una nueva marca.

Paso 2.2) Si M es idéntica a una marca en el camino desde la raíz hasta M, llame a M como *"viejo"* y vaya a otra nueva marcación.

Paso 2.3) Si no hay transiciones habilitadas en M, llame a M *"terminación muerta"*

Paso 2.4) Mientras existan transiciones habilitadas en M, haga lo siguiente:

Paso 2.4.1) Obtenga la marcación M' que resultan de disparar t en M.

Paso 2.4.2) Sobre el camino desde la raíz hacia M si existe una marcación M'' tal que $M'(p) \geq M''(p)$ para cada lugar p y $M' \neq M''$, M'' es cubierta, reemplace $M'(p)$ por ω para cada p tal que $M'(p) > M''(p)$.

Paso 2.4.3) Introduzca M como un nodo, dibuje un arco llamado t desde M hasta M', y llame a M' *"nuevo´´"*.

Este árbol de alcance es representado mediante un grafo en el que los nodos representan el vector de marcado y los arcos son las transiciones que al dispararse cambian el estado de la red.

Por medio de este grafo podemos analizar si la red alcanzará un cierto estado, y mediante qué secuencia de eventos. El problema es que el número de estados no siempre es finito (si la red no está acotada). Ver Figura 2.5

2.4.7. Matriz De Incidencia Y Ecuación De Estado

La matriz de incidencia y la ecuación de estado son técnicas de análisis formal que permiten entender el comportamiento de una RdP.

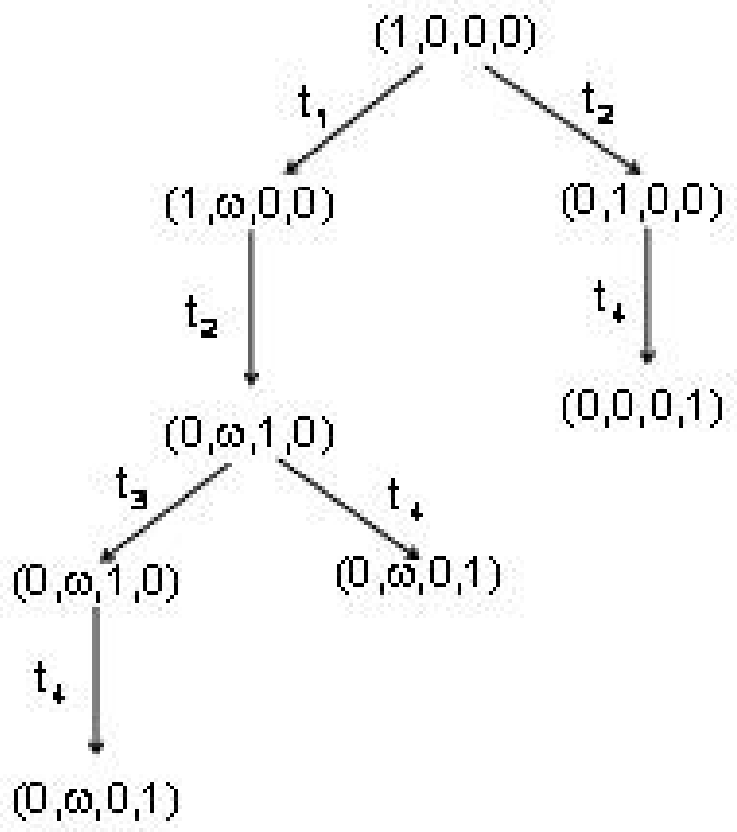

Figura 2.5: Arbol de alcance de la Figura 2.4

- **Matriz de Incidencia** Para una red RDP N con n transiciones y m lugares, la matriz de incidencia $A = [a_{ij}]$ es una matriz $n \times m$ de enteros donde cada elemento está dado por:

$$a_{ij} = a_{ij}^{+} - a_{ij}^{-}$$

Donde $a_{ij}^{+} = \omega(i,j)$ es el peso del arco de la transición i a su lugar de salida j y $a_{ij}^{-} = \omega(i,j)$ es el peso del arco a la transición i desde su lugar de entrada j. La transición i esta habilitada en una marcación M si y sólo si

$$a_{ij}^{-} \leq M(j), j = 1, 2, .., n$$

Ejemplo: ver la Figura 2.6.

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Figura 2.6: matriz de incidencia de la Figura 2.4

- **Ecuación de Estado** Se puede escribir una marcación M_k como un vector columna de $m \times 1$. El j-ésimo elemento de M_k denota el número de tokens en el lugar j inmediatamente después del k-ésimo disparo en cualquier secuencia de disparo. El k-ésimo disparo o vector de control u_k es un vector columna de $n - 1$ ceros y un elemento no cero, un 1 en la i-ésima posición indicando que la transición i se disparo en el k-esimo disparo.

  Ya que la i-ésima fila de la matriz de incidencia denota el cambio de la marcación como el resultado de la transición disparada i, se puede representar la siguiente ecuación de estado para una RdP.

$$M_k = M_{k-1} + A^t u_k, k = 1, 2, ...$$

Capítulo 3

Workflow

3.1. Automatización

La automatización, más que una rama de la ingeniería, es considerada como una ciencia con una gran profundidad de investigación. Se encarga de optimizar en forma general el manejo de la empresa, esto incluye su forma de operar, ofreciéndole incrementar la calidad de los procesos de negocio, disminuir los tiempos en la toma de decisiones y entregar a tiempo una respuesta permitiéndole permanecer dentro de la competitividad del mercado; entre otras factores ofrece disminuir las perdidas económicas, incrementar su rendimiento con una menor inversión de tiempo y esfuerzo.

El hecho de automatizar los procesos de negocio es dejar que un robot (mecánico ó programado) realice las tareas de estos procesos con o sin la supervisión del hombre, esto disminuye el riesgo de errores y se desarrollen los procesos de una forma ágil, permitiendo la reducción de tiempo en el retardo de la ejecución de tareas y por supuesto la disminución de costos.

3.2. Workflow o Flujos de Trabajos

3.2.1. Definición informal

Todo sistema tiene una dinámica de comportamiento, esta dinámica obedece a ciertos eventos y a su vez estos eventos depende de condiciones en el entorno para que puedan ocurrir; a esta dinámica es a lo que se le llamaría flujo.

Cada evento ocurre solo cuando las condiciones necesarias para poder ejecutarse están dadas; cuando este evento está predispuesto a suceder y sucede se realiza algo (una tarea, acción, etc.), el cual cambia el estado del sistema donde este ocurre o hace que se mantenga en el mismo estado. El hecho de cambiar o mantener el estado del sistema necesita de un ente que ejecute una acción en un determinada momento, a esta acción es lo que llamamos trabajo.

Entonces un flujo de trabajo o "Workflow´´ se puede definir como el conjunto de condiciones y eventos definidos que se ejecutan en una secuencia específica.

3.2.2. Definición formal

> " La automatización de un Proceso de Empresa , total o parcial, en la cual documentos, información o tareas son pasadas de un participante a otro a los efectos de su procesamiento, de acuerdo a un conjunto de reglas establecidas ".
>
> WFMC (WorkFlow Management Coalition)

3.3. Automatización de flujos de procesos de Negocio

No es más que la implementación de un sistema de software que ayuden al monitoreo y control de los procesos de negocio de una empresa, para mejorar el rendimiento y disminuir todo tipo de costos.

Para poder automatizar un proceso de la vida real (proceso físico o lógico) es necesario modelar dicho proceso, es decir llevarlo a un plano abstracto de donde se pueda estudiar y analizar para poder entenderlo lo mayor posible y luego poder optimizarlo para crear la representación lógica del proceso en un ambiente programado.

3.4. Evaluación de sistemas de Gestión Electrónica Documental en la empresa (SGED)

Esta pequeña evaluación fué tomada de REF ESGEDE y muestra claramente los obstáculos y beneficios reales de implantar un SGED.

3.4.1. Beneficios de la implantación de SGED

- Disminución del tiempo de localización y recuperación de los documentos al ser accesible desde el propio puesto de trabajo.
- Disminución del tiempo en tratamiento y gestión, el usuario no tiene que re-archivar cada documento al trabajar con él en pantalla.
- Disminución del coste de distribución; al estar los documentos accesibles en cualquier puesto, se eliminan los gastos de mensajería, fax, etc.

3.4.2. Disminución de costes administrativos

- Drástico recorte del espacio de almacenamiento y reaprovechamiento del mismo. Los originales en papel pueden enviarse a un espacio más barato y pequeño o un almacén de custodia. Dispositivos de almacenamiento electrónico.
- Eliminación de los documentos duplicados al estar accesibles en cualquier momento desde cualquier lugar.

- Drástica reducción en material de archivo al suprimirse los listados en papel y las copias.

3.4.3. Disminución de la pérdida de oportunidad

- Mayor control y seguridad; el acceso a los documentos puede restringirse a determinados usuarios definiendo niveles de confidencialidad que llegan a partes de un documento.
- No existen documentos extraviados o perdidos.
- Mejora de la calidad del servicio ofrecido; los clientes son respondidos "in situ" en sus demandas de documentos pudiendo recibir copia de los mismos en el acto.

3.4.4. Aumento de la productividad

- Rendimiento en la consulta, con multiplicidad de criterios de recuperación.
- Mejora de la gestión; la respuesta del sistema es más ágil y eficaz permitiendo una ventaja competitiva a la empresa.

3.4.5. Problemas asociados a SGED

- **Legalidad**; pese a existir jurisprudencia sobre el particular y evolucionar el marco jurídico para aceptar la legalidad tanto de la documentación electrónica (ley 30/1992, así como la aceptación de documentación en formato electrónico de carácter fiscal) como de la firma electrónica, lo cierto es que no existe aún ninguna norma que dé cobertura legal a los documentos en formato electrónico, si bien tampoco existe ley que manifieste lo contrario. En este sentido cabe reseñar los intentos de las diferentes administraciones Internacionales para buscar una solución, siendo una cuestión de tiempo la promulgación de legislación específica sobre el tema. En cualquier caso, y

mientras esto se produce, debe seguir existiendo el archivo tradicional en papel como prueba documental.

- **Cambio cultural**; quizás es el mayor obstáculo a vencer. El hábito del uso del papel, incluso para las operaciones más sencillas, no se elimina en 24 horas, si bien el contar con un sistema amigable para los usuarios, reduce este impacto considerablemente.

3.5. Integración del Sistema

La integración de sistemas es considerado como un factor muy importante para poder alcanzar los objetivos propuestos por la automatización, permitiendo la optimización en la forma de operar de la empresa, transformándola en la empresa integral del futuro. El concepto de integración del sistema involucra algunos requerimientos y conceptos los cuales a su vez tienen cada uno una funcionalidad bien definida la cual es contemplada en el momento del desarrollo:

- Plataforma de Conectividad.
- Modelo del Negocio.
- Estándar para Comunicación.

La integración del sistema contempla algunos conceptos de investigación que buscan la optimización en el desarrollo y desempeño del sistema computacional, cabe resaltar que este enfoque fortalece el Modelo del Sistema.

3.5.1. Plataforma de Conectividad

La plataforma de conectividad hace referencia a la conexión, interna como externa que posee la entidad ó empresa utilizando los diferentes medios de transmisión de datos como, cableado estructurado para el uso de redes LAN que pueden

permitir la conexión interna de la empresa; Internet para tener acceso a la información que se encuentra distribuida en diferentes áreas geográficas del mundo, VPN'S que permiten un camino privado seguro para el flujo de la información, medios inalámbricos de corto alcance que se pueden conectar a las estructuras de redes inalámbricas internas de la empresa, etc., esta plataforma permite la comunicación entre las diferentes entidades o componentes de una entidad mas general; por medio de ella transita un flujo de datos los cuales permiten mantener actualizada la información permitiendo que el acceso a la información sea más eficiente y seguro por la tanto las reacciones del sistema pueden ser más rápidas.

3.5.2. Introducción al Modelo de Negocio

El modelo del negocio es uno de los puntos más importantes contemplados dentro del concepto de integración de sistemas, ya que en este punto se da por manifiesto un modelo de cómo va a ser la dinámica del sistema cuando este se implemente. En este punto el enfoque que se aborda es el de entender las abstracción que aportan las técnicas de modelado, de manera que se tiene en cuenta el enfoque que se le quiere dar al sistema, en donde cada sistema tiene una forma de operar diferente ya que considera variables diferentes para desempeño y funcionamiento dentro de una organización.

El modelo del negocio considera la dinámica de los flujos de información entre diferentes componentes de una entidad los cuales juegan Roles que determinan sus acciones; teniendo en cuenta esto la lógica del negocio puede determinarse un grado ó nivel para el entendimiento entre las diferentes entidades, esto permite que el ambiente que se presente en la entidad sea integrado.

Este concepto es contemplado fuertemente por medio de Modelado mediante diferentes tipos de Redes de Petri (RdP), con el fin de entender la dinámica discreta que se puede presentar en el sistema, además del modelo que considera la interacción entre el usuario y el sistema.

3.5.3. Estándar Para la Comunicación

El estándar de comunicación es muy importante ya que es el medio usado para integrar una entidad, ya que el será quien transporte los datos entre los diferentes componentes de una entidad a otra, claro esta haciendo uso de la plataforma de comunicación la cual posee la entidad. El estándar de comunicación permite la forma de cómo hacer entender las diferentes entidades entre si, y esto es por que el estándar permite la creación de un lenguaje propio, es decir el estándar de comunicación soporta la creación de un nuevo lenguaje el cual es enriquecido con la semántica de los diferentes procesos. Es necesario hacer uso de un estándar que permita describir el funcionamiento de la entidad, haciendo que dicha descripción sea entendible por los demás de manera que permita reflejar la dinámica del sistema, para ello es necesario hacer una correcta manipulación del estándar. En este proyecto el estándar considerado es XML (Lenguaje de Marcas Extendidas).

3.6. Elementos que intervienen en un WorkFlow

Documento El documento es el elemento principal, representa una orden de trabajo la cual debe ser completada, y al paso por las distintas entidades de la empresa genera a su ve trabajos manuales o automáticos, que a su vez modifican el estado del documento. De ahí porque se concibe el documento como el elemento principal de este sistema dinámico. Un documento puede ser creado, modificado, rechazado, completado.

Usuario El usuario es el ente que realiza las tareas u órdenes de trabajo que le asigna un Documento. Solo puede realizar las tareas básicas que le son permitidas sobre un Documento determinado.

Rol Es el conjunto de privilegios que juega cada usuario dentro del sistema, la cual le permite el acceso a determinadas acciones dentro de un Documento y del Sistema mismo.

3.6.1. Relaciones entre los distintos elementos

Cada una de las ordenes de trabajo que genera un Documento necesitan ser completadas por un usuario que juegue determinado rol dentro del sistema, un Documento puede contener múltiples tareas y estas son asignadas automáticamente cada una a un usuario especifico según un rol especificado previamente. Dentro de un documento un rol es asignado a un único usuario, esto quiere decir que un rol dentro de un documento es jugado por un solo actor.

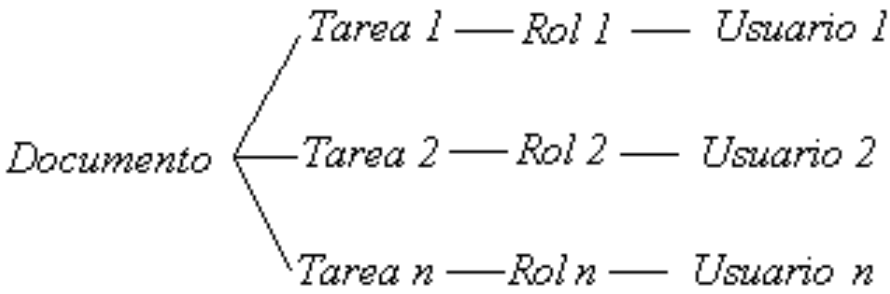

Figura 3.1: Relación entre los Elementos

Capítulo 4

Modelo del Sistema

Existen en el mercado actualmente una gran cantidad de de motores de "Workflow" que utilizan Redes de Petri, pero la gran mayoría son demasiado complicados para resolver tareas muy sencillas, en gran parte estos motores se encuentran desarrollados en Java, por ejemplo: OSWorflow, BPEL, BPML, etc. Tendríamos que pasar mucho tiempo entendiéndolos ya que son pocos intuitivos antes de pasar a ser productivos lo cual no representa un camino factible.

El modelo que se a tomado como referencia se esta desarrollando en la empresa Minotauro C.A. por el Ing. Gabriel Demián Gutiérrez (?, ?), para tener una manera de estandarizar el desarrollo de "Workflow" dentro de la empresa. Por medio de un motor propio que este bajo nuestro control y que se pueda moldear según las necesidades requeridas y ser mucho mas prácticos a la hora de realizar modificaciones sobre el motor para mejorar su comportamiento.

También se encuentra el hecho de que en PHP no son muchas las aplicaciones que se encuetran de este tipo y sólo son en su gran mayoría propuestas las que existen para desarrollar un motor de "Workflow" en PHP. Otro punto muy importante es que los motores existentes en Java ya tiene un gran respaldo y no valdría la pena desarrollar un motor en este lenguaje ya que la competencia es muy grande, caso contrario para PHP lo cual nos muestra un mejor futuro para el motor de "Workflow" en este lenguaje.

4.1. El Sistema

El sistema es una herramienta que permite llevar el control de los documentos que estén registrados en él; se encarga de asignar automáticamente las tareas a los usuarios correspondientes y es capaz de ejecutar tareas que le hallan sido programadas y/o asignadas.

Todo esto es posible gracias a las RdP que es sobre quien trabaja directamente el motor de "Workflow" ya que un Documento puede ser modelado por una RdP; para que esto sea posible se ha definido la construcción de una RdP de la siguiente manera:

1. Los estados del Documento están definidos por cada una de las marcaciones posibles de una RdP.
2. Todas las acciones a las que se encuentra expuesto el Documento están representadas por las transiciones de una RdP.
3. Cada transición solo debe ser ejecutada por un único usuario.
4. Se debe establecer un único estado inicial para cada Documento; debe existir una sola forma de iniciar un Documento.
5. Los estados finales deben poder ser alcanzables desde cualquier estado de la RdP, asegurando así la finalización del Documento.

Véa la definicion en XML de una Rdp en el apéndice A.1

4.2. Casos de Uso

Autentificación Identificación de usuario para la carga de roles, todos los usuarios deben ser autenticados antes de ingresar al sistema.

Iniciar Documento Solos Los usuarios con los roles adecuados podrán iniciar un Documento.

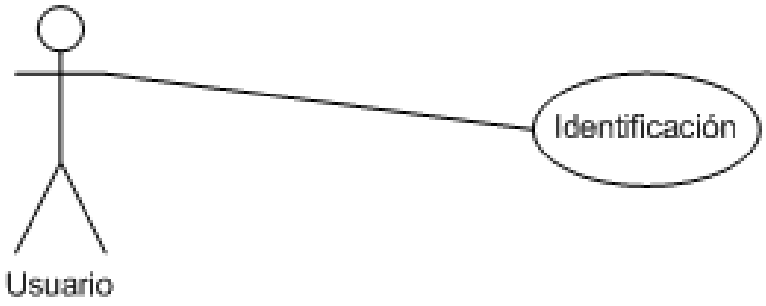

Figura 4.1: Caso de Uso: Autentificación

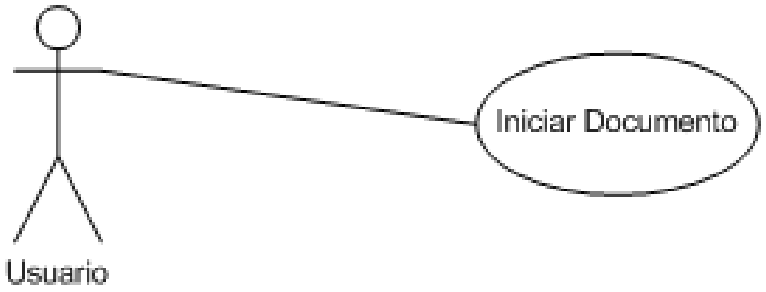

Figura 4.2: Caso de Uso: Iniciar Documento

1. Seleccionar el Documento deseado.
2. Introducir la información necesaria para iniciar el Documento.
3. Iniciar el Documento.

Ver Lista de Tareas (Buzón) Lista de tareas asignadas pendientes para el usuario actual.

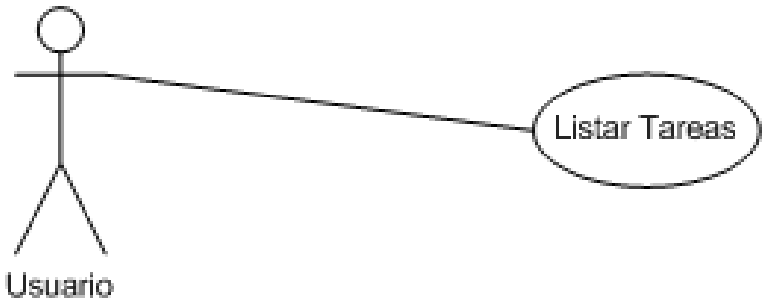

Figura 4.3: Caso de Uso: Buzón de tareas

Realizar Tarea Seleccionar una tarea pendiente para ejecutarla.

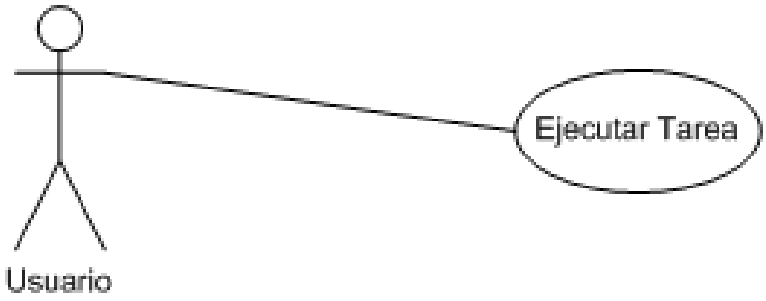

Figura 4.4: Caso de Uso: Ejecutar Tarea

Listar Documentos Ver los documentos del sistema.

Figura 4.5: Caso de Uso: Listar Documentos

Ver Documento Seleccionar un Documento para ver su estado y/o información necesaria.

Figura 4.6: Caso de Uso: Ver Documento

4.3. Diagrama de Estados de un Documento

En la Figura 4.7 se puede observar claramente que para cualquier documento los estados son secuenciales; en cualquier Documento existe en todo momento un único estado.

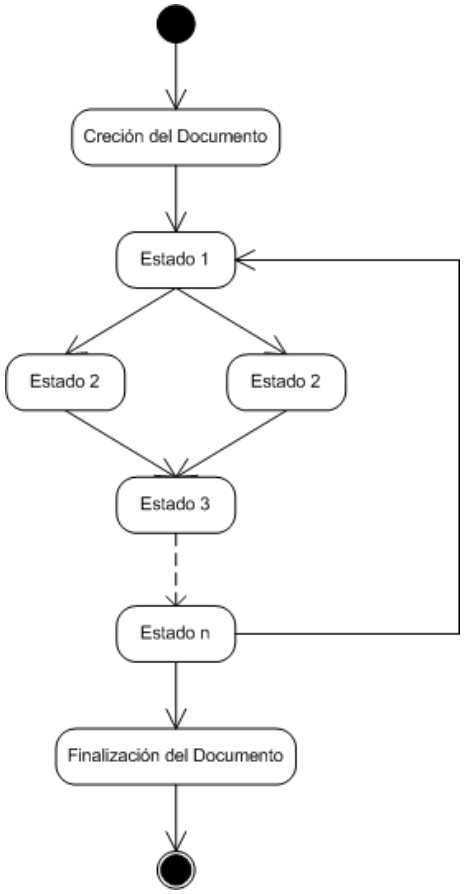

Figura 4.7: Diagrama de estados de un Documento

4.4. Definición del Modelo

4.4.1. Modelo Global del Sistema

En la figura Figura 4.8 representa el modelo global del sistema y muestra como es la interacción entre los objetos más importantes en el motor de "Workflow", mas adelante podrá observar en detalle el modelo.

4.4.2. Modelo de la Rdp

Una RdP está compuesta por lugares y transiciones, y a su vez estos se conectan a través de arcos quienes defines la dinámica del sistema.

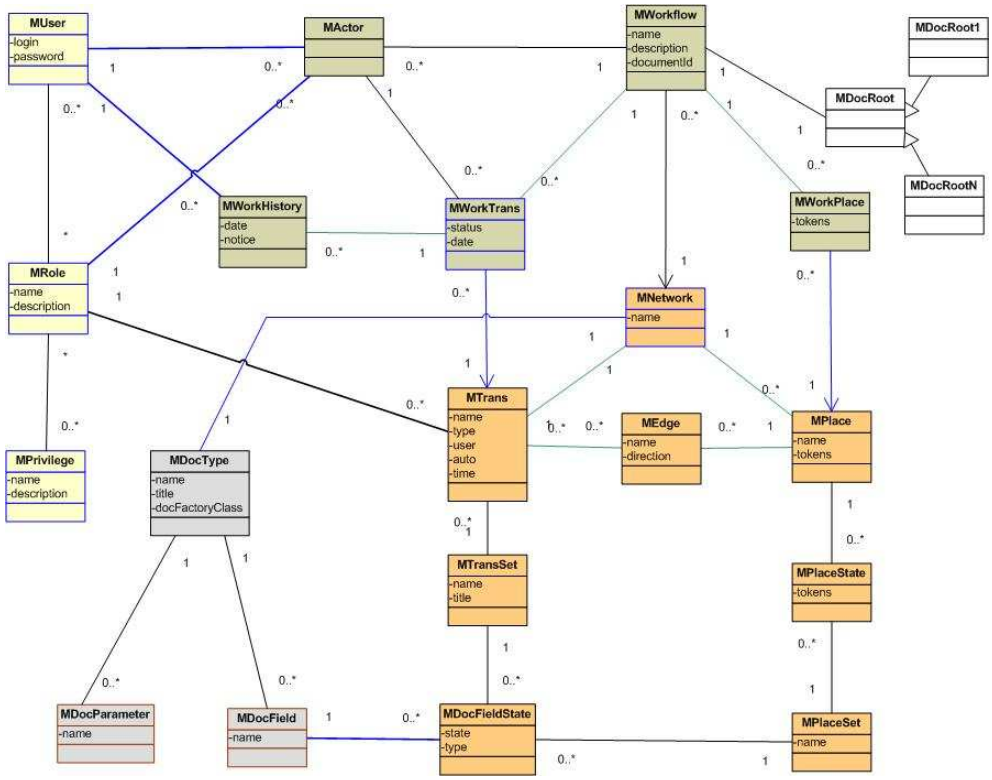

Figura 4.8: Modelo Global del Sistema

Los Lugares de una RdP son quienes definen el estado del documento, están compuestos por un nombre, nombre del estado del documento cuando este se encuentra en el estado que define.

Los lugares de una RdP son quienes habilitan las transiciones del sistema y las diferentes combinaciones de marcaciones de los lugares definen el estado actual de la red.

Las Transiciones tienen un nombre, que indica el nombre de la tarea o acción a ejecutar, un tipo que indica si es automática (disparado por sistema) o disparado por un usuario, una referencia a un rol, que es el rol del ente que la puede ejecutar.

El objeto MTransSet, agrupa las transiciones habilitadas por rol en un determinado estado del documento.

EL objeto MPlaceSet, da información de cual es el nombre del estado del documento para cierta distribución de fichas o marcación (MPlaceState) en la

RdP, ver Figura 4.9.

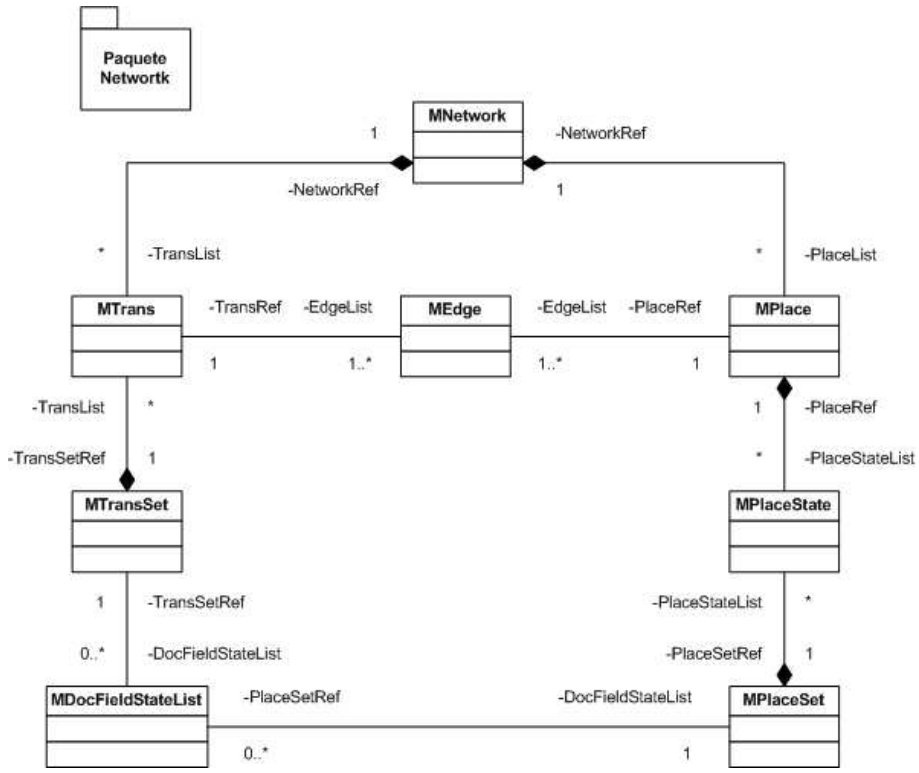

Figura 4.9: Modelo de una RdP, ver apéndice B.2

4.4.3. Modelo de un Documento

Un documento esta compuesto por campos que es donde se almacena la data con la cual esta relacionada el documento, y compuesto por parámetros que son los parámetros iniciales de un documento, ver Figura 4.10.

4.4.4. Modelo del Usuario del Sistema

El usuario del sistema está compuesto por nick o seudónimo y una contraseña; esta información se utiliza para la autentificación de los usuarios antes de entrar al sistema.

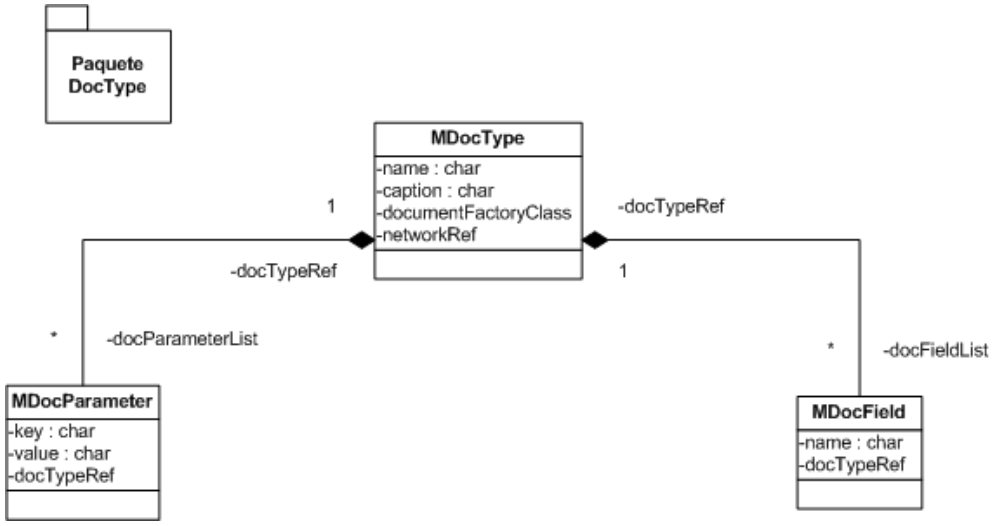

Figura 4.10: Modelo del Documento Asociado a una RdP

Un usuario tiene asociado los roles que juega en un determinado momento y estos roles contiene los privilegios del usuario, los cuales indican las tareas que el usuario puede realizar dentro del sistema, ver Figura 4.11.

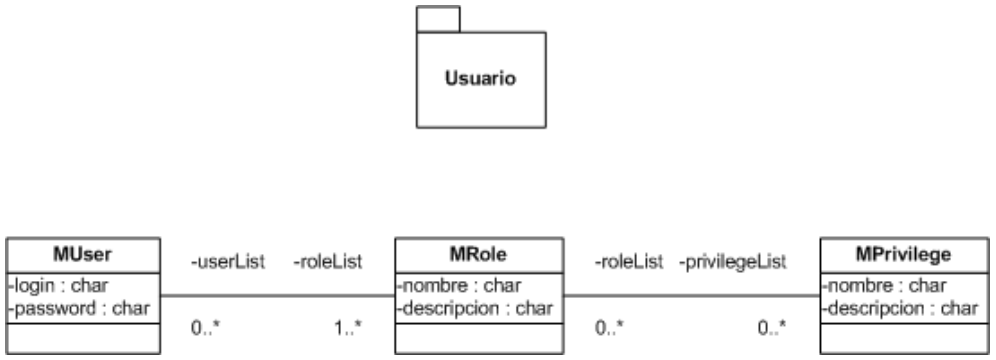

Figura 4.11: Modelo del Usuario Del Sistema

4.4.5. Modelo de una RdP en Ejecución

Este modelo representa la instancia de una RdP, cuando esta se ha puesto en ejecución, es decir, cuando un usuario a iniciado un documento determinado, en si es una instancia de una Red de Petri a la cual llamaremos “Workflow” ver

Figura 4.12.

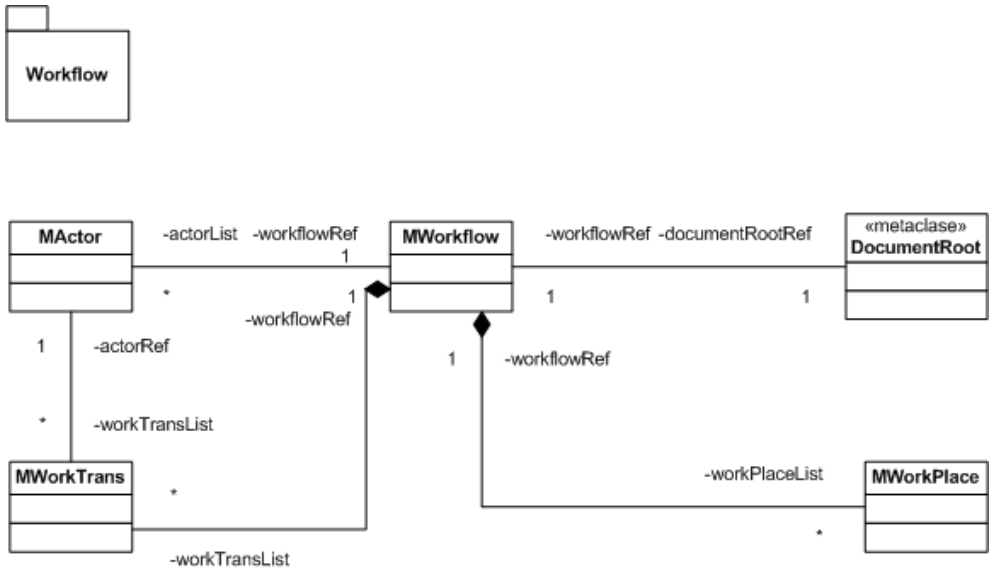

Figura 4.12: Modelo de RdP en Ejecución, ver apéndice B.1

Capítulo 5

Implementación

Para la implementación de este sistema se ha escogido trabajar bajo la arquitectura MVC, el lenguaje de programación PHP, un manejador de base de datos MySQL y una interfaz Web bajo un servidor Apache. La arquitectura se escogió para que el sistema sea de fácil entendimiento a la hora de depuración y para que siguientes investigadores que deseen retomar el proyecto; es de acotar que las tecnologías utilizadas están basadas en la ideología del software libre.

5.1. Tecnologías utilizadas

- Para el desarrollo de este sistema se utilizó un marco de trabajo llamado Mojavi para PHP y se utilizo versión 2.0, que es la última versión estable, esta librería se puede descargar gratis en http://www.mojavi.org.
- Para el manejo de persistencia de objetos se utilizó una librería llamada Ezpdo que en dicho momento se encontraba en la versión 1.1.3 y esta se puede descargar gratis en http://www.ezpdo.net.
- Para le definición del modelo estándar de una RdP se utilizó el lenguaje XML debido a su excelente forma de estructurar documentos.
- De PHP e utilizó la versión 5.0.5 y esta puede descargarse gratis en http://www.php.net.

- De MySQL se utilizó la versión 4.1.23 y esta puede descargarse gratis en http://www.mysql.com.

5.2. Esquema de implementación

5.2.1. Conversión de la especificación de comportamiento a MOJAVI

Relación de las páginas en el navegador con la especificación de la RED

Mojavi permite a través de cada acción especificar que privilegio debe tener el usuario actual para ejecutarla, definir la acción misma y enviar la vista adecuada.

Las acciones básicas que se pueden realizar sobre un Documento son, crear, modificar, rechazar y completar con lo cual podemos definir de forma genérica dichas acciones y en casos particulares solo será necesario redefinir acciones específicas para Documentos específicos a través del objeto Action de Mojavi.

Los métodos del objeto Action son:

execute Ejecutado si la acción viene de un método de envió de información a través del Request (GET o POST).

getDefaultView Método por defecto de una acción.

getPrivilege Retorna los privilegios que requiere el usuario para ejecutar la acción.

getRequestMethods Validar el método de envió de datos a través del Request.

handleError Manejo de Errores y enviar a la vista correspondiente.

isSecure Retorna si el usuario debe estar identificado para realizar esta acción.

validate Valida los datos que vienen en el Request introducidos por el usuario.

En cada una de las acciones del sistema de cada uno de los Documentos registrados a través de la función getPrivileges del objeto Action de Mojavi es donde se valida que los usuarios tengan los roles correspondientes para ejecutar las tareas de dicho Documento.

La objeto de edición de documentos llamado *EditarWorkflowAction* la cual realiza la tarea de carga del Documento deseado para ejecutar la tarea pendiente del "Workflow´´ seleccionado y poder modificar el Documento.

Para poder cargar el documento deseado se envía a través del Request el identificador único del documento (id) y así poder cargar la información del documento que se encuentra en Base de Datos.

- En la función getDefaultView se carga el Documento deseado.
- En la función validate se validan los campos del Documento que el usuario tiene permiso de modificar, la visibilidad de los campos esta definida en el archivo XML donde se realizo la definición de la red para el Documento.
- En la función execute se realiza las operaciones con Base de Datos necesarias.

En esta acción es donde se modifica, completa o rechaza un documento, la acción de crear un Documento debe ser definida por el programador ya que el nombre del Documento a crear no se conoce y no se puede determinar en tiempo de ejecución.

5.2.2. Almacenamiento de la dinámica del sistema en la base de datos

Una vez inicializado un "Workflow´´ (creado un documento) de un modelo determinado se crea una instancia de la red que lo define y esta instancia es

almacenada en Base de Datos, esta instancia es el objeto MWorkflow.
En esta instancia es almacenado el estado del Documento y las modificaciones que se le han realizado, como se puede ver en Figura 4.12 el objeto MWorkflow tiene objetos MWorkTrans asociados, que son las instancias de las transiciones de la red modelo que se van a ejecutar y en ellas se almacena cual es la transición pendiente según el estado del Documento que se conoce a través del objeto MWorkPlace con el cual se puede obtener la marcación actual de una instancia de una RdP.

Ejemplo de la red almacenada

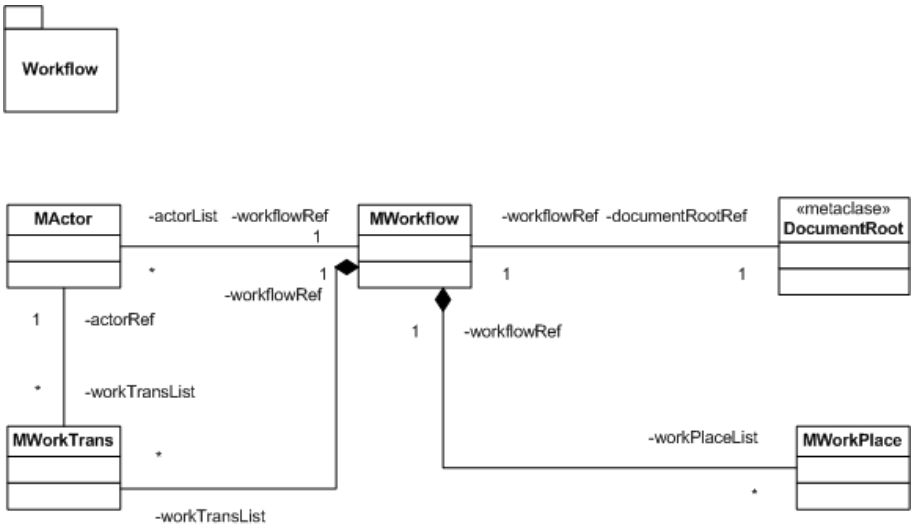

Figura 5.1: Recuperación del estado del Documento, ver apéndice B.1

5.3. Prueba del Sistema

Para esta prueba se ha utilizado un documento para crear un usuario nuevo dentro del sistema definido en el apéndice A.2

Entrar Al sistema ver Figura 5.2 y Figura 5.3.

Selección de Documento a Iniciar Iniciar el documento Demostración del sistema, ver Figura 5.4.

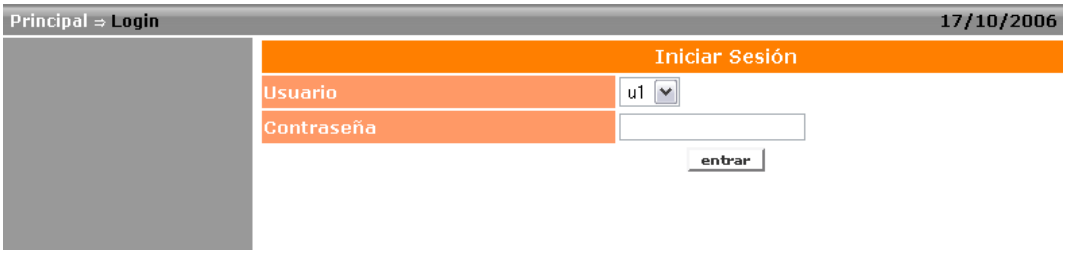

Figura 5.2: Identificación de Usuario

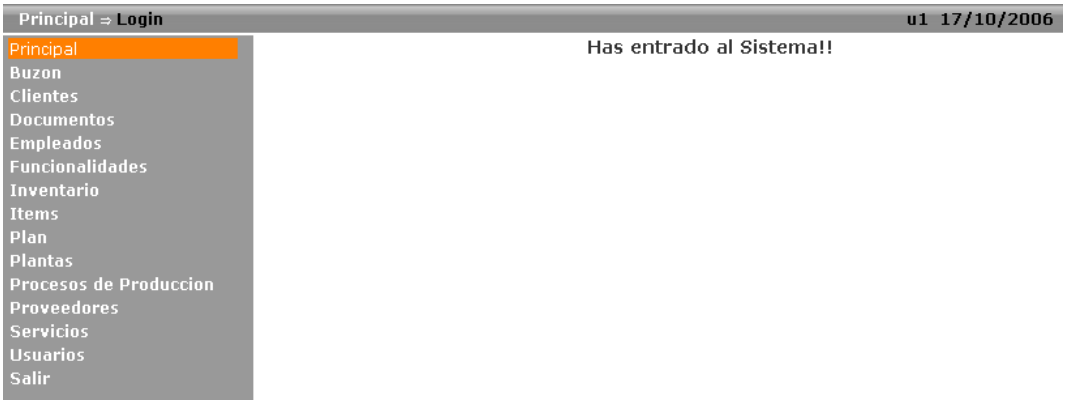

Figura 5.3: Bienvenido al Sistema

Iniciar Documento ver Figura 5.5, Figura 5.6 y apéndice C.1.

Ver asignación de tareas automáticas Según la definición del documento en el apéndice A.2, la tarea es asignada al un usuario que tenga el rol de prueba "rol1"; este rol lo tiene asignado el usuario "u2", se ingresa al sistema con el usuario 2 y revisamos su "Buzón", ver Figura 5.7, Figura 5.8 y apéndice C.2.

Ejecución de Tareas Para ejecutar una tarea el usuario debe seleccionar una de las tareas pendientes que aparecen en el Buzón de Tareas y realizar las operaciones adecuadas, en este caso "Aprobamos" el documento, ver Figura 5.9 y Figura 5.10 y apéndice C.3.

Ver nueva asignación de tareas automáticas Según la definición del documento en el apéndice A.2, la siguiente tarea es asignada al un usuario que

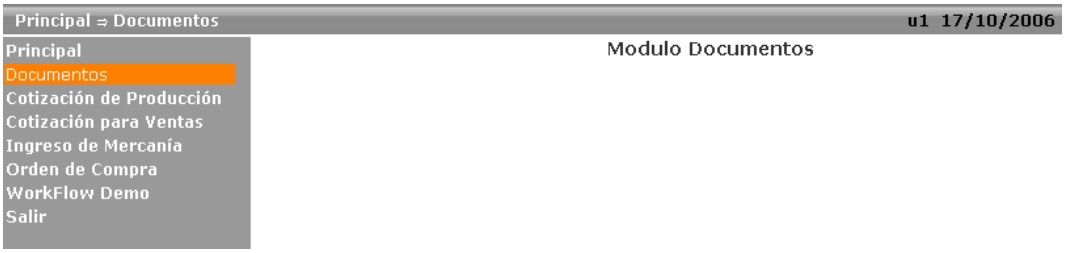

Figura 5.4: Seleccionar Documento

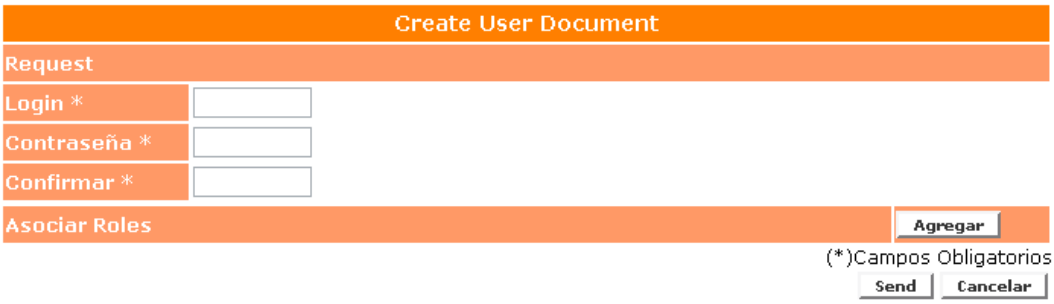

Figura 5.5: Crear Documento

tenga el rol de prueba "rol2"; este rol lo tiene asignado el usuario "u3", se ingresa al sistema con el usuario "u3" y revisamos su "Buzón", ver Figura 5.11 y Figura 5.12.

Ejecución de Tareas Para ejecutar una tarea el usuario debe seleccionar una de las tareas pendientes que aparecen en el Buzón de Tareas y realizar las operaciones adecuadas, en este caso este usuario tiene permiso de modificar la lista de roles del usuario solicitado, modificamos esta lista agregando un nuevo rol y se aprueba el documento, ver Figura 5.13 y Figura 5.14.

Ver nueva asignación de tareas automáticas (fin del documento) Según la definición del documento en el apéndice A.2, la siguiente tarea es asignada al un usuario que tenga el rol de prueba "rol0"; este rol lo tiene asignado el usuario "u1" quien es que crea el documento, se ingresa al sistema con el usuario "u1" y revisamos su "Buzón", ver Figura 5.15 y verificamos en la lista de usuarios el nuevo usuario creado Figura 5.16.

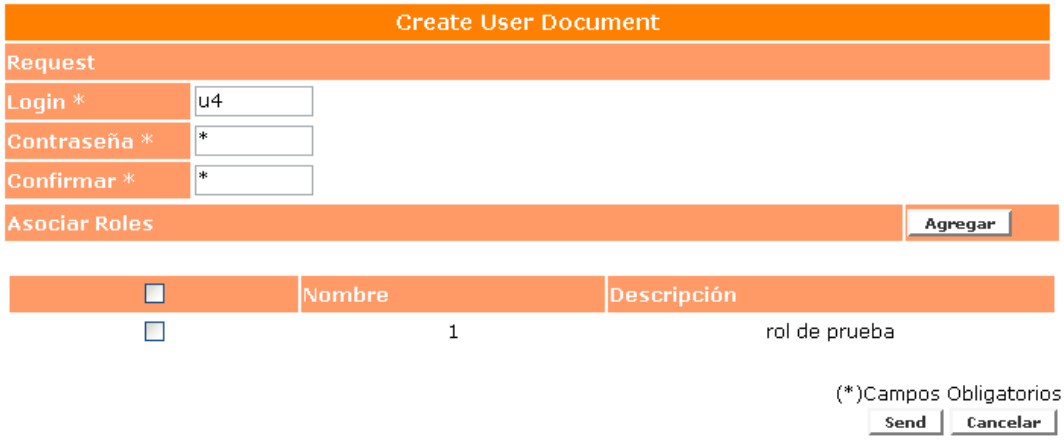

Figura 5.6: Rellenar Campos del Documento

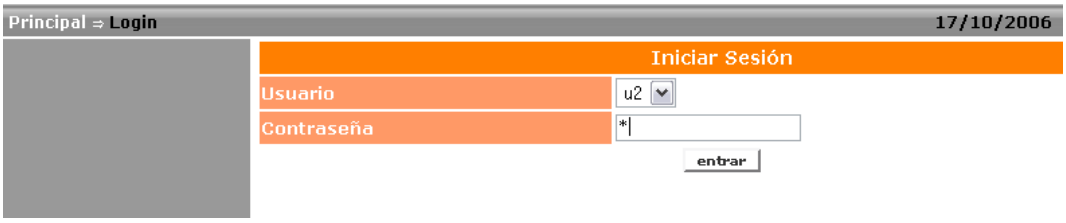

Figura 5.7: Iniciar Sesión con el usuario “u2”

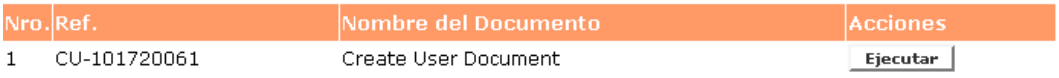

Figura 5.8: Buzón de Tareas del usuario “u2”

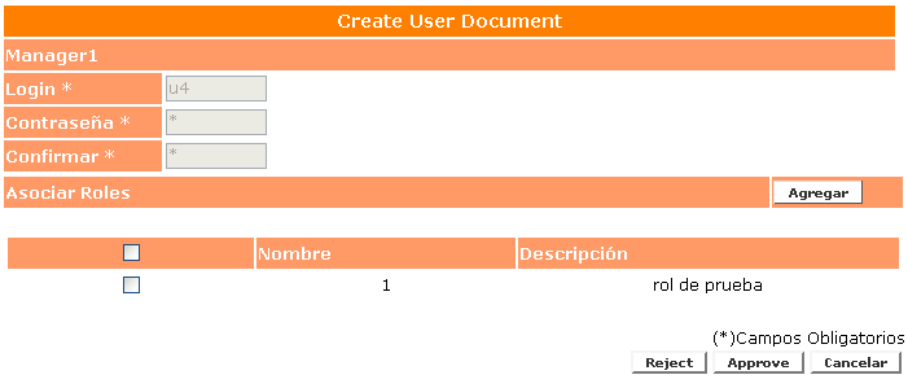

Figura 5.9: Ejecutar Tarea pendiente

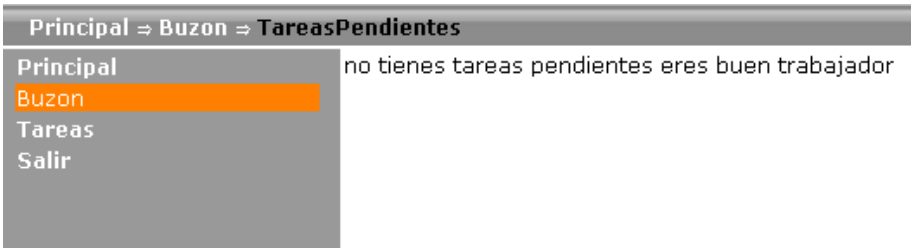

Figura 5.10: Ver Buzón de Tareas

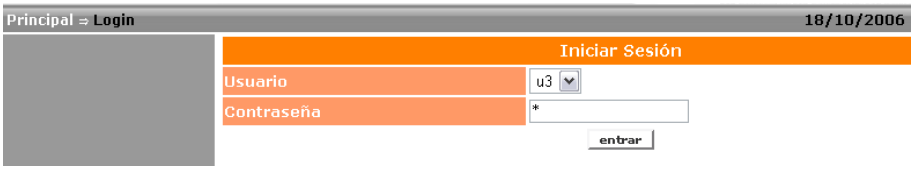

Figura 5.11: Iniciar Sesión con el usuario “u3”

Figura 5.12: Buzón de Tareas del usuario “u3”

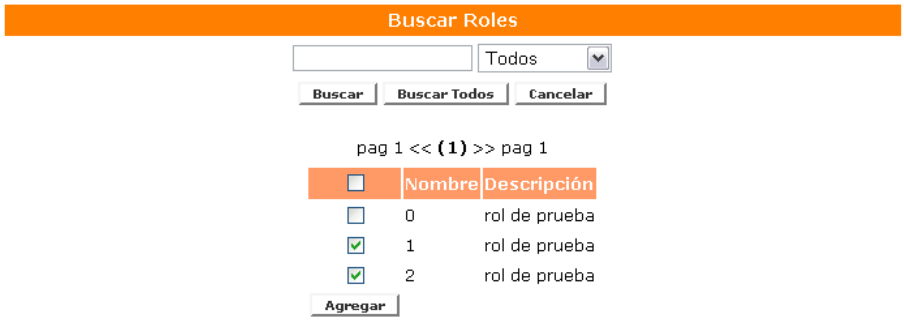

Figura 5.13: Ejecutar Tarea pendiente

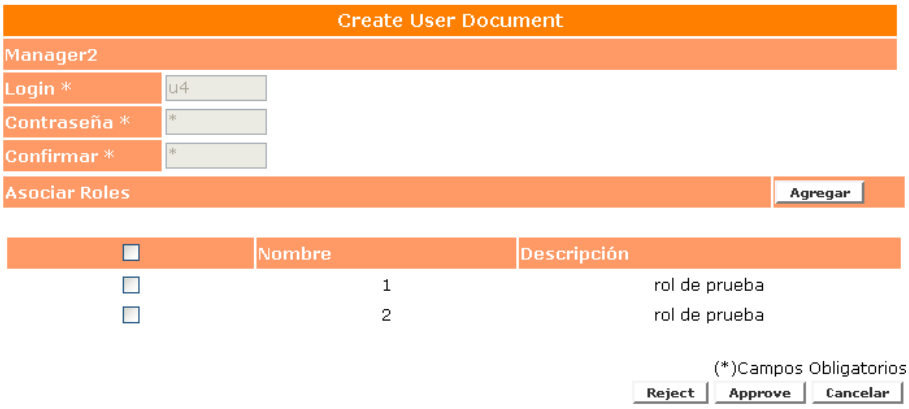

Figura 5.14: Ver Buzón de Tareas

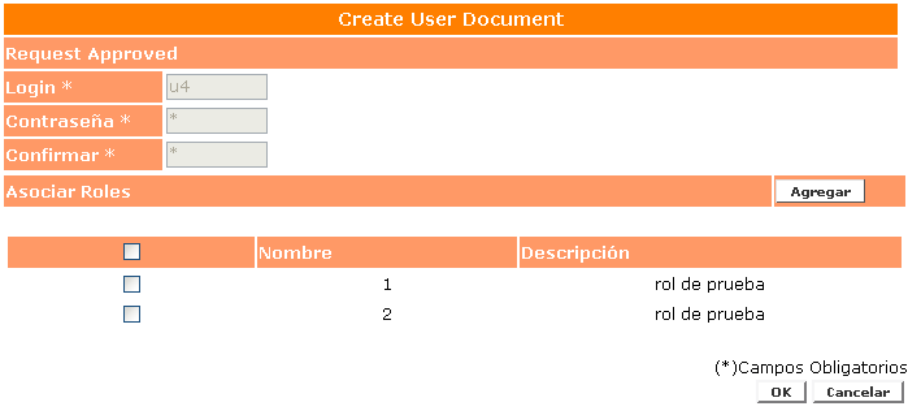

Figura 5.15: Finalización del Documento

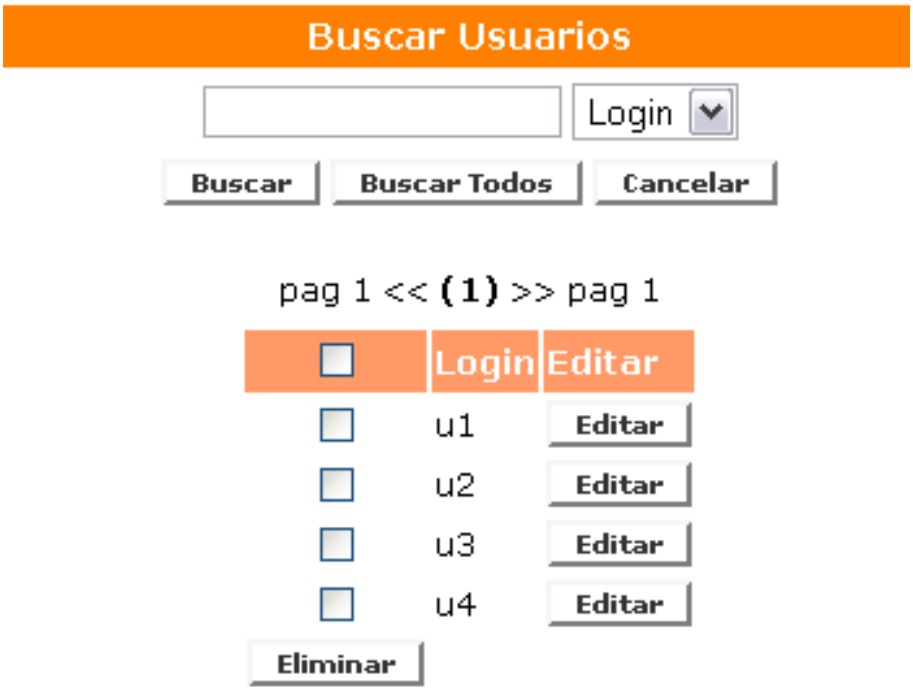

Figura 5.16: Ver Lista de Usuarios

Capítulo 6

Conclusiones y Recomendaciones

6.1. Conclusiones

La facilidad y gran utilidad de las RdP para modelar y desarrollar sistemas las coloca como una muy buena opción para ser utilizadas como herramientas de control y monitoreo de procesos de cualquier tipo. Durante el desarrollo de esta investigación se pudo notar que muchas situaciones de la vida real se pueden resolver modelándolas con RdP.

Con el motor que se ha desarrollado se pueden ejecutar procesos de diferentes tipos únicamente conociendo la dinámica de dicho proceso y realizando el modelado como hemos visto anteriormente. Aun cuando el motor resuelve y alcanza los objetivos deseados se puede afirmar que este se encuentra en una etapa inicial todavía y que puede seguir mejorándose.

Este proyecto desarrollado es de gran utilidad y representa una buena opción para utilizarlo como base de un sistema más grande, dejando de un lado la preocupación de la lógica de negocio y de otro la capa de presentación.

Uno de los aportes ha sido la mejora del modelo de la definición XML de una RdP hecho inicialmente Minotauro C.A., otro aporte es el del motor de ejecución de RdP desarrollado en PHP y MySQL y finalmente el sistema de prueba llamado Motor de "Workflow" para la ejecución de las RdP.

6.2. Ventajas y Desventajas

- Permite modelar casi la gran mayoría de procesos de negocio; no esta contemplado los procesos de producción.
- Sencillez de agregar nuevos documentos a la aplicación; es necesario saber programar.
- Se pueden automatizar gran número de tareas dentro de los procesos de negocio permitiendo así disminuir los errores humanos.
- Los agentes son quienes realizan la mayor parte de las tareas dentro de las órdenes de trabajo, por ejemplo enviar email's, notificaciones, creación de nuevos documentos, etc. Siendo en ellos donde se almacena la lógica de negocio y no en el motor.
- El problema de integración con los sistemas actuales no esta contemplado.

6.3. Recomendaciones

El motor desarrollado solo trabaja con RdP estándar y temporizadas, dejando abierta la posibilidad de trabajar con RdP de alto nivel.

A medida que es utilizado el sistema surgen grandes ideas para mejorarla y hacer de ella una herramienta más poderosa. La posibilidad de realizar un sistema de escritorio nunca ha sido descartado pero no se ha tratado en esta investigación, ya que la gran mayoría de aplicaciones actuales se están mudando lentamente al mundo de Internet debido a la facilidad de acceso desde cualquier lugar.

Apéndice A

Definicion en XML de un RdP

A.1. Modelo XML de una RdP

```
<?xml version="1.0" encoding="iso-8859-1"?>

<RdP>

    <!--
    ****************************************
    *                                      *
    * Doc Type                             *
    *                                      *
    ****************************************
    -->
    <docType name="">
        <caption></caption>

        <parameter key="">value
        </parameter>
        ...
```

```
    <docField>name</docField>
    ...
</docType>

<!--
****************************************
*                                      *
* Network                              *
*                                      *
****************************************
-->

<network name="">

    <!-- ************************************* -->
    <!-- Place List                            -->
    <!-- ************************************* -->

    <place name="" tokens=""/>
    ...

    <!-- ************************************* -->
    <!-- Trans List                            -->
    <!-- ************************************* -->

    <trans name="" type="" time="">
        <caption></caption>
        <role></role>
    </trans>
```

```
    ...

    <!-- ************************************* -->
    <!-- Edge List                             -->
    <!-- ************************************* -->

    <edge src="place" tgt="trans"/>
    <edge src="trans" tgt="place"/>

</network>

<!--
*****************************************
*                                       *
* Trans Set Visibility                  *
*                                       *
*****************************************
-->

<transSetList>

    <transSet name="">
        <caption></caption>
        <transList>trans1,trans2,...</transList>

        <visibleDocField>
            docField1,docField2,...
        </visibleDocField>
```

```
            <readonlyDocField>
                docField3,docField4,...
            </readonlyDocField>

        </transSet>

    </transSetList>

    <!--
    ****************************************
    *                                      *
    * Place Set Visibility                 *
    *                                      *
    ****************************************
    -->

    <placeSetList>

        <placeSet name="">
            <caption></caption>
            <placeState place="" tokens="1"/>

            <visibleDocField>
                docField1,docField2,...
            </visibleDocField>
        </placeSet>

    </placeSetList>

</RdP>
```

A.2. Documento para Crear Usuario (Definición de la RdP)

```
<?xml version="1.0" encoding="iso-8859-1"?>

<workflowDefinition>

<!--
****************************************
*                                      *
* Doc Type                             *
*                                      *
****************************************
-->
<docType name="CreateUser">
<caption>Create User Document</caption>

<parameter key="documentRootClass">
MCreateUser
</parameter>

<parameter key="documentEditUrl">
?module=Demo&amp;action=Editar
</parameter>

<parameter key="documentViewUrl">
?module=Demo&amp;action=Ver
</parameter>

<parameter key="documentEditForward">
```

```
TareasPendientes
</parameter>

<parameter key="documentViewForward">
TareasPendientes
</parameter>

<docField>login   </docField>
<docField>password</docField>
<docField>roleList</docField>
</docType>

<!--
****************************************
*                                      *
* Network                              *
*                                      *
****************************************
-->

<network name="Net 2">

<!-- ************************************** -->
<!-- Place List                             -->
<!-- ************************************** -->

<place name="p1" tokens="1"/>
<place name="p2" tokens="0"/>
<place name="p3" tokens="0"/>
<place name="p4" tokens="0"/>
```

```
<place name="p5" tokens="0"/>
<place name="p6" tokens="0"/>

<!-- ************************************* -->
<!-- Trans List                            -->
<!-- ************************************* -->

<trans name="t1" type="0" time="0">
<caption>Send</caption>
<role>0</role>
</trans>

<trans name="t2" type="0" time="0">
<caption>Reject</caption>
<role>1</role>
</trans>

<trans name="t3" type="0" time="0">
<caption>Approve</caption>
<role>1</role>
</trans>

<trans name="t4" type="0" time="0">
<caption>Reject</caption>
<role>2</role>
</trans>

<trans name="t5" type="0" time="0">
<caption>Approve</caption>
<methods>createUser</methods>
```

```
<role>2</role>
</trans>

<trans name="t6" type="0" time="0">
<caption>OK</caption>
<role>0</role>
</trans>

<trans name="t7" type="0" time="0">
<caption>OK</caption>
<role>0</role>
</trans>

<!-- ************************************* -->
<!-- Edge List                             -->
<!-- ************************************* -->

<!-- User -->
<edge src="p1" tgt="t1"/>
<edge src="t1" tgt="p2"/>

<!-- Manager1 -->
<edge src="p2" tgt="t2"/>
<edge src="t2" tgt="p3"/>

<edge src="p2" tgt="t3"/>
<edge src="t3" tgt="p4"/>

<!-- Manager2 -->
<edge src="p4" tgt="t4"/>
```

```
<edge src="t4" tgt="p3"/>

<edge src="p4" tgt="t5"/>
<edge src="t5" tgt="p5"/>

<!-- User -->
<edge src="p3" tgt="t6"/>
<edge src="t6" tgt="p6"/>

<edge src="p5" tgt="t7"/>
<edge src="t7" tgt="p6"/>

</network>

<!--
****************************************
*                                      *
* Trans Set Visibility                 *
*                                      *
****************************************
-->

<transSetList>
<transSet name="ts1">
<caption>Request</caption>
<transList>t1</transList>

<visibleDocField>
login,password,roleList
</visibleDocField>
```

```
</transSet>

<transSet name="ts2">
<caption>Manager1</caption>
<transList>t2,t3</transList>

<editNotice>
<caption>Reject Reason</caption>
<mandatory>t2</mandatory>
</editNotice>

<readonlyDocField>
login,password,roleList
</readonlyDocField>
</transSet>

<transSet name="ts3">
<caption>Manager2</caption>
<transList>t4,t5</transList>

<editNotice>
<caption>Reject Reason</caption>
<mandatory>t4</mandatory>
</editNotice>

<readonlyDocField>
login,password
</readonlyDocField>
<visibleDocField>
roleList
```

```
</visibleDocField>
</transSet>

<transSet name="ts4">
<caption>Request Denied</caption>
<transList>t6</transList>

<showNotice>t2,t4</showNotice>

<readonlyDocField>
login,password,roleList
</readonlyDocField>
</transSet>

<transSet name="ts5">
<caption>Request Approved</caption>
<transList>t7</transList>

<readonlyDocField>
login,password,roleList
</readonlyDocField>
</transSet>
</transSetList>

<!--
****************************************
*                                      *
* Place Set Visibility                 *
*                                      *
****************************************
```

```
-->

<placeSetList>
<placeSet name="ps1">
<caption>Request</caption>
<placeState place="p1" tokens="1"/>

<visibleDocField>
login,password,roleList
</visibleDocField>
</placeSet>

<placeSet name="ps2">
<caption>Manager1</caption>
<placeState place="p2" tokens="1"/>

<visibleDocField>
login,password,roleList
</visibleDocField>
</placeSet>

<placeSet name="ps3">
<caption>Manager2</caption>
<placeState place="p4" tokens="1"/>

<visibleDocField>
login,password,roleList
</visibleDocField>
</placeSet>
```

```
<placeSet name="ps4">
<caption>Request Denied</caption>
<placeState place="p3" tokens="1"/>

<visibleDocField>
login,password,roleList
</visibleDocField>
</placeSet>

<placeSet name="ps5">
<caption>Request Approved</caption>
<placeState place="p5" tokens="1"/>

<visibleDocField>
login,password,roleList
</visibleDocField>
</placeSet>

<placeSet name="ps5">
<caption>Closed</caption>
<placeState place="p6" tokens="1"/>

<visibleDocField>
login,password,roleList
</visibleDocField>
</placeSet>
</placeSetList>

</workflowDefinition>
```

Apéndice B

Diagramas UML

B.1. Modelo UML de un Workflow

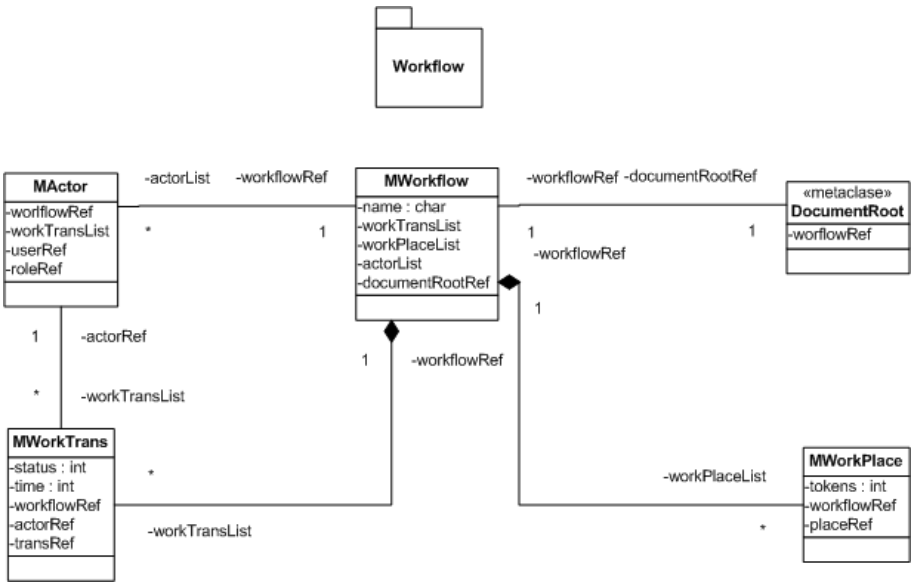

Figura B.1: Modelo de RdP en Ejecución

B.2. Modelo UML de una RdP

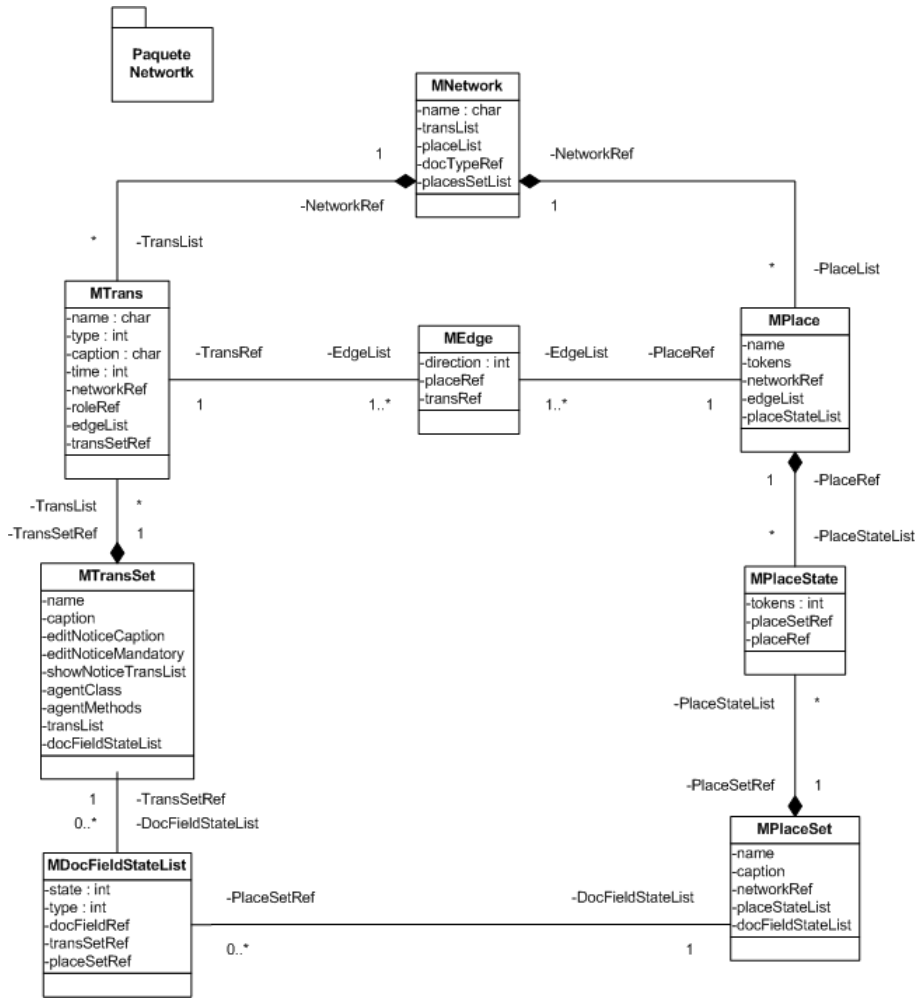

Figura B.2: Modelo de una RdP

Apéndice C

Definición de las Funciones mas importantes del Motor De Workflow

C.1. Inicializar

```
public static function & initWorkflow(& $session, & $documentRoot, & $role) {
$facade = & new EFacade();
$facade->session = & $session;
$facade->init($documentRoot, $role);
return $facade;
}
private function init(& $documentRoot, & $role) {
$this->documentRoot = & $documentRoot;

if ($role) {

$this->workflow = & $documentRoot->getWorkflowRef();
```

```
// ----------------------------------------
// inicializar wf
// ----------------------------------------

$this->engineNetwork->setCurrDate(time());
$network = & $this->workflow->getNetworkRef();
$this->engineNetwork->setModelNetwork($network);
$this->engineNetwork->initModelNetwork();

$this->engineNetwork->setModelWorkflow($this->workflow);
$this->engineNetwork->initModelWorkflow();

$this->transSet = & EFacade :: getCurrentTransSet($this->workflow, $role);
$this->placeSet = & EFacade :: getCurrentPlaceSet($this->workflow);
} else {
// ----------------------------------------
// cargar el documento
// ----------------------------------------

$query = 'FROM MDocType as x WHERE x.name = ?';
$docType = $this->session->find($query, $documentRoot->getDocumentName());

if (!$docType) {
throw new Exception('No se puede cargar el doctype con nombre ' .
$documentRoot->getDocumentName());
}

$docType = $docType[0];

// ----------------------------------------
```

```
// cargar red modelo
// ----------------------------------------

$modelNetwork = & $docType->getNetworkRef();

$this->workflow = $this->session->create('MWorkflow');
$this->workflow->setNetworkRef($modelNetwork);
$this->engineNetwork->setCurrDate(time());
$this->engineNetwork->setModelNetwork($modelNetwork);
$this->engineNetwork->initModelNetwork();

// ----------------------------------------
// Relaciones de los lugares
// ----------------------------------------

$enginePlaceList = & $this->engineNetwork->getEnginePlaceList();

foreach ($enginePlaceList as $key => & $value) {
$enginePlace = & $value;
$workPlace = & $enginePlace->getWorkPlace();

$workPlaceList = & $this->workflow->getWorkPlaceList();
$workPlace->setWorkflowRef($this->workflow);

$modelPlace = & $enginePlace->getModelPlace();
$workPlace->setPlaceRef($modelPlace);
$workPlaceList[] = $workPlace;
}

// ----------------------------------------
```

```
// Relaciones de las tranciciones
// ---------------------------------------

$engineTransList = & $this->engineNetwork->getEngineTransList();

foreach ($engineTransList as $key => & $value) {
$engineTrans = & $value;
$workTrans = & $engineTrans->getWorkTrans();

$workTransList = & $this->workflow->getWorkTransList();

$workTrans->setWorkflowRef($this->workflow);

$modelTrans = & $engineTrans->getModelTrans();
$workTrans->setTransRef($modelTrans);
$workTransList[] = $workTrans;
}

// ---------------------------------------
// obtener el transet
// ---------------------------------------

$this->transSet = & $this->getCurrentTransSet($this->workflow, $this->role);
$this->placeSet = & $this->getCurrentPlaceSet($this->workflow);
}
}
```

C.2. Lista de Tareas

```
/**
```

```
 * @param MWorkflow
 * @param MRole
 * @return MTransSet
 */
public static function & getCurrentTransSet(& $workflow, & $role) {
$ret = NULL;

$workTransList = & $workflow->getWorkTransList();

// ----------------------------------------
// revisar la lista de worktrans
// ----------------------------------------

foreach ($workTransList as $key => & $workTrans) {

if ($workTrans->getStatus() == MWorkTrans :: $STATUS_DISABLED) {
continue;
}

$modelTrans = & $workTrans->getTransRef();

// ----------------------------------------
// ignorar las que no tienen el rol del usuario actual
// ----------------------------------------

if ($role != NULL && $role != $modelTrans->getRoleRef()) {
continue;
}

$currTransSet = & $modelTrans->getTransSetRef();
```

```
// ----------------------------------------
// solo puede existir una tarea para cada usuario
// ----------------------------------------

if ($ret != NULL && $ret != $currTransSet) {
throw new Exception($ret->getName());
}

$ret = & $currTransSet;
}

return $ret;
}
```

C.3. Edicion de Documentos

```
class Editar_WorkflowAction extends Editar_Action {

public function getRequestMethods() {
return REQ_NONE;
}

public function getDefaultView(& $controller, & $request, & $user) {

if (!$this->loadObject($controller, $request, $user)) {
$request->setError('error', 'Accion Abortada');
$request->setMethod(REQ_POST);
$controller->forward($controller->getCurrentModule(),
$user->getAttribute('repeatAction', REDIRECT));
```

```
return VIEW_NONE;
}


$ezpdo_ = & getManager();
$doc = & $user->getAttribute('obj', $controller->getCurrentModule());
$usuario = & $user->getAttribute('usuario', 'Datos');

$query = "FROM MActor as a " .
"WHERE a.workflowRef.documentRootRef.numero = ? " .
"AND a.workTransList.contains(wt) " .
"AND wt.status = ? ";

$actor = $ezpdo_->find($query, $doc->getNumero(), 1);

if (count($actor) > 1 || count($actor) == 0 || $actor == FALSE) {
throw new Exception('No se pudo encontrar un
<b>único</b> actor para ejecutar esta tarea');
}

$role = & $actor[0]->getRoleRef();

$facade = EFacade :: initWorkflow($ezpdo_, $doc, $role);
$workflow = $facade->getWorkflow();
$workflow->epSetCommittable(FALSE, FALSE);
$docType = $workflow->getNetworkRef()->getDocTypeRef();
$transSet = EFacade :: getCurrentTransSet($workflow, $role);

$user->setAttribute('transSet', $transSet, $controller->getCurrentModule());
$user->setAttribute('docType', $docType, $controller->getCurrentModule());
```

```
$user->setAttributeByRef('role', $role, $controller->getCurrentModule());
$this->redirectLocation($controller, $request, $user);

return VIEW_INPUT;
}

public function initialize(& $controller, & $request, & $user) {
return TRUE;
}

public function redirectLocation(& $controller, & $request, & $user) {

parent :: redirectLocation ( $controller, $request, $user);
$obj = & $user->getAttribute('obj', $controller->getCurrentModule());
$docType = $obj->getWorkflowRef()->getNetworkRef()->getDocTypeRef();
$user->setAttribute('successAction',
$docType->getParameter('documentEditForward')->getValue(), REDIRECT);
$user->setAttribute('accion', 'Editar', REDIRECT);
}
}
```

Bibliografía

Aalst, G. H. Van der, K.M. van Hee. *Modelling and analysing workflow using a petri-net based approach w.m.p.*

Gutierrez, G. D. (2003-2004). *Motor de ejecución de redes de petri.*

Kerr, S. (2004-2006). *Mojavi.* http://www.mojavi.org.

Murata, T. (1989). *Petri nets: Properties, analysis and applications* (Vol. 7) (No. 4).

Peterson, J. L. (1981). *Petri net theory and the modeling of systems.*

The petri nets world. (2004). URL: http://www.daimi.au.dk/PetriNets/. (An online database of Petri net-related conferences, mailing lists, bibliographies, tool databases, newsletters, research groups, etc)

Team ezpdo. (2005). *Ezpdo (easy php data objects).* http://www.ezpdo.net.