Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

**Departamento de Ingeniería Eléctrica**

**Sección de Computación**

# Modelado de workflow con redes de Petri coloreadas condicionales

**Tesis que presenta**

**Samuel Garrido Daniel**

**para obtener el Grado de**

**Maestro en Ciencias**

**en la Especialidad de**

**Ingeniería Eléctrica**

**Director de la Tesis**

**Dra. Xiaoou Li Zhang**

**México, D.F.** **Diciembre 2005**

# Índice general

# Dedicatoria

A ti papá, estos dos años de mi estancia en el CINVESTAV han sido gracias al apoyo incondicional y a la formación que me has dado con tu ejemplo, el resultado de este trabajo es tan tuyo como mío, al igual que este trabajo es tuyo mamá por todo lo que eres para mi.

También, dedico este trabajo a mis hermanos y a todos mis amigos y amigas con los cuales he compartido diferentes etapas de mi vida.

# Agradecimientos

A Dios, por darme la vida y darme el camino que hasta hoy día he recorrido.

A mis padres, por todo el cariño, amor, educación, apoyo y ejemplo que me han brindado, con lo cuál he logrado todas las satisfacciones por las cuales hoy día, me siento orgulloso como persona. A mis hermanos, por estar a mi lado.

A mi asesora la Dra. Xiaoou Li, por éste trabajo de tesis, por ser una excelente persona y haberme brindado su apoyo en todo momento.

Al Dr. Joselito Medina Marin, por su amistad y ayuda valiosa en el desarrollo de mi trabajo de tesis.

Al Dr. Francisco Rodríguez Henríquez y al Dr. Carlos Coello Coello, por sus valiosas observaciones y comentarios respecto a la escritura de mi trabajo de tesis.

A Sofia Reza, que desde siempre y hasta hoy día, me ha brindado su apoyo y amistad.

A los Doctores de la Sección de Computación, de quienes obtuve conocimientos de sus cursos, en los que tuve la oportunidad de estar.

A todos mis compañeros de la Sección de Computación, en especial a mis Amigos: Dennis Bazan, Roberto Linares, Efrén Clemente, Mario Parra, Juan Pablo Flores y Francisco López , quienes compartieron conmigo situaciones difíciles, pero también muchos logros y momentos que quedarán en nuestras memorias de este capítulo de nuestras vidas, que fué estudiar nuestra Maestría.

Al CONACYT por apoyarme con la beca de maestría y al CINVESTAV.

# Resumen

El modelado de workflow es un área de investigación que ha sido explorada por más de una década. Actualmente existen diversos trabajos en los que se emplean diferentes técnicas de modelado como son UML, IDEF0, redes de Petri, álgebras de procesos entre otras más; sin embargo actualmente no existe una metodología o técnica estándar. Un proceso de workflow es un proceso genérico de negocios, como ejemplos de estos procesos podemos citar la compra en línea de un libro, la inscripción de un alumno al nuevo semestre escolar, la requisición de productos en un departamento de compras, entre otros más. Es evidente que los procesos de workflow son procesos comunes que se encuentran en la vida diaria de toda organización o empresa.

El atractivo de la tecnología Workflow, es la ejecución de manera automatizada o semi-automatizada de los procesos de negocio de una empresa, además de permitir su administración y monitorización. Para estas tareas se utilizan los sistemas administradores de workflows, los cuales a partir de la definición del proceso a través de alguna técnica de modelado, pueden ejecutar dichos procesos en combinación con otras tecnologías de información y del personal humano involucrado.

Es así como la correcta especificación del proceso asegurará en mayor medida la ejecución correcta del mismo. Las redes de Petri son una herramienta gráfica y matemática para modelado de sistemas. En la presente tesis se propone una metodología basada en redes de Petri para modelación y simulación de procesos de workflow a través de un enfoque de reglas ECA (Evento-Condición-Acción).

# Abstract

Workflow modeling is a research area that has been explored by more than one decade. Currently, diverse research works have been proposed in which different techniques are used to model it, such as UML, IDEF0, Petri nets, algebras of processes, among others. Nevertheless, at the present time it does not exist a standard methodology or technique. A workflow process is a generic process of businesses. Examples of these processes are: the online purchase of a book, the registration of a student to the new school term, a product requisition in a purchase department, etc. It is evident that are processes that are present in the everyday life of every organization or company.

The attractiveness of the Workflow technology, is the execution of automated or semi-automated forms of the business processes of a company, besides allowing to its administration and to monitor. In order to do these tasks, Workflow Management Systems are used, which (from the definition of the process through some modeling technique) can execute these processes in combination with other information technologies and the involved human resources.

Thus, the correct specification of the process will ensure in greater measurement the correct process completion. Petri nets are a graphical and mathematical tool for systems modeling. In the present thesis a methodology based in Petri nets for modeling and simulation of workflow processes through an approach based on ECA (Event-Condition-Action) rules is proposed.

# Capítulo 1

# Introducción

Con la incorporación de la tecnología como ayuda en las actividades del ser humano, la forma de llevar a cabo éstas 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 el intento por automatizar las actividades cotidianas con ayuda de Tecnologías de Información (Information Technologies. IT's; por sus siglas en inglés). Es así como las organizaciones de diferentes sectores incorporan tecnología para la realización de sus procesos con el fin de obtener dicha automatización y mejorar los resultados en sus procesos y objetivos de negocio. Como consecuencia de lo anterior, las organizaciones actualmente poseen sistemas de información con el fin de organizar, automatizar y ejecutar sus procesos; de esta manera tenemos que la tecnología workflow proporciona una plataforma para lograr dicha automatización de procesos.

La tecnología workflow engloba diferentes conceptos, uno de los los principales es: "el proceso de workflow", a continuación se define con el fin de comprender de mejor manera lo que esta tecnología ofrece. Un proceso de workflow, es un proceso genérico dentro de una organización, como ejemplos de estos procesos podemos citar los siguientes: la compra en línea de un libro, la inscripción de un alumno a un nuevo semestre, la requisición de productos en un departamento de compras, etc. Es evidente que los procesos de workflow son procesos comunes que se encuentran en la vida diaria y en las organizaciones, independientemente del giro de éstas. A estos procesos comúnmente se les refiere como procesos de negocio; el atractivo de la tecnología workflow, es la ejecución de manera automatizada o semi automatizada de esos procesos de negocio en las organizaciones. Para dicha tarea se emplean los Sistemas Administradores de Workflow (Workflow Management Systems, WfMS), los cuales a partir de la definición del proceso de negocio realizada con alguna técnica de modelado, llevan

a cabo la ejecución del proceso conjuntamente a otras tecnologías de información y de las personas que colaboran en la empresa.

Dentro del ciclo de vida de un WfMS se encuentra al inicio la fase de creación y definición, seguida por las fases de verificación y de representación, concluyendo con la fase de ejecución y monitoreo del proceso. En esta tesis, el trabajo realizado se delimita dentro de la primera fase. La tarea más importante en esa primera fase es la modelación de los procesos de workflows que ejecutará el WfMS. El modelado de workflow consiste en la representación y definición de los procesos de workflow.

El modelado de workflow[1] es un área de investigación que ha sido explorada y de la cual existen diversos trabajos, en los que se emplean diferentes técnicas (metodologías y lenguajes de modelado) para modelar procesos de workflow, como las siguientes: UML, IDEF0, redes de Petri, álgebras de procesos, entre otras más; sin embargo en la actualidad no existe una técnica estándar para esta tarea.

En este trabajo de tesis se aborda el tema de modelado de workflow proponiendo una metodología basada en redes de Petri, así como un sistema de software que permita utilizar esta metodología para la definición de dichos procesos, así como su simulación a través del juego de tokens que proporcionan las redes de Petri. Actualmente existen diversos trabajos relacionados con el tema, tanto en la industria como en el área académica y de investigación. Estos trabajos son presentados a través de un estudio del estado del arte sobre modelado de workflow.

## 1.1. Estado del arte

En los años 90's surgió 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, organizaciones, entre otros elementos más, suficientes para que la Tecnología Workflow sea de gran importancia en la actualidad para la industria de software y organizaciones que hacen uso de ésta tecnología. A continuación se presenta el estado del arte sobre el modelado de procesos workflow.

[1] En este trabajo se emplea la palabra en inglés *workflow* para referirnos a procesos de negocios comúnmente conocidos como procesos de flujo de trabajo. Se usa la palabra en inglés debido a la aceptación que tiene ésta y su uso extendido en la mayor parte de la literatura sobre el tema.

La Coalición para la Gestión de Flujo de Trabajo (Workflow Management Coalition, WfMC) es la principal organización en lo que se refiere a workflow y sistemas administradores de Workflows, la cual está conformada por instituciones académicas y comerciales. Actualmente, enfoca sus esfuerzos hacia una estandarización que permita lograr interoperabilidad entre diferentes sistemas de workflow que son desarrollados, además de los que se encuentran actualmente en el mercado y que actualmente se utilizan. En su trabajo se han obtenido diversos resultados tales como el Modelo de Referencia de Workflow [1]. En este modelo se definen los conceptos y los elementos que debe poseer un sistema administrador de workflow. Todo trabajo relacionado con workflow debería guiarse por el contenido de este modelo de referencia. Además, en el sitio web [2] se presentan diferentes reportes técnicos en los que publican sus estándares tales como XPDL y WfXML.

La Iniciativa para la Gestión de Procesos de Negocio (Business Process Management Initiative, BPMI) es otra organización sin ánimo de lucroque tiene la misión de promover el desarrollo y uso de la administración de procesos de negocio a través del establecimiento de estándares para el diseño, distribución, ejecución, mantenimiento y optimización de procesos.

Existen otros portales en Internet dedicados a la administración de procesos de negocios como por ejemplo: BPMG [3] (The Global Business Process Management Community), ABPMP [4] (The Association of Business Process Management Professionals), el sitio web del SIGPAM (Special Interest Group on Process Automation and Management) perteneciente a la Asociación de Sistemas de Información. El portal e-Workflow [5]. Un buen sitio para comenzar a investigar acerca de Workflow es el Worfklow-Research [6] ya que además de proporcionar definiciones y conceptos, provee una sección bastante completa de direcciones web referentes a grupos de investigación, investigadores, proyectos y productos de workflow.

En la industria existen diversas compañías que mantienen bien posicionados sus WfMS (Workflow Management Systems). Ejemplos de éstas son: Workflow SAP R3, Lotus workflow, IBM MQSeries, Oracle Workflow, entre otros más. En software libre también existen WfMS que son muy utilizados, como por ejemplo: jBPM, wfmOpen, OfBiz, OpenWFE, ObjectWeb Bonita, Enhydra Shark.

.

Para el modelado de workflow existen dos enfoques principales: UML (principalmente sus diagramas de actividad) y redes de Petri. Con respecto a UML su principal desventaja es la falta de bases formales, es decir de una semántica formal que permita la verificación de los

modelos obtenidos a través de técnicas matemáticas. El grupo Precise UML Group (PUML[2]) trabaja en la clarificación y en dirección de obtener una semántica precisa para UML. Otros trabajos de investigación que existen se enfocan generalmente a desarrollar extensiones a los diagramas de actividad o a obtener algoritmos que transformen procesos modelados con diagramas de actividad en algún tipo de red de Petri con el objetivo de verificar el modelo con alguna técnica propia de redes de Petri. En [7], dos semánticas operacionales que utilizan transiciones etiquetadas son propuestas para el modelado de workflow utilizando diagramas de actividad. Sin embargo, ésta se encuentra desarrollada con la versión UML 1.4, esto es considerado como una desventaja al existir actualmente la versión 2.0 de UML, y además no considera la perspectiva de datos. En [8], [9] son descritas las limitantes que la nueva versión 2.0 de UML que aún presenta, a pesar de poseer bastantes mejoras, con lo que concluyen la no total conveniencia de usar UML para el modelado de workflow.

Por otro lado, algunos trabajos basados en extensiones o modificaciones de redes de Petri son los siguientes: Refinamiento de lugares con tokens de control usando redes predicado/transición (Token controlled place refinement using predicate/transition nets) [10], Redes con referencias (Reference Nets) [11], Redes Jerárquicas de tareas con estado (Hierarchical Task State Nets) [12], Redes workflow (WfNets) [13], Redes reactivas (Reactive nets) [14], entre otras más. Sin lugar a dudas el trabajo más completo es el realizado por Wil van der Aalst y sus colaboradores, en el cual definen un nuevo lenguaje de workflow llamado YAWL [9]. YAWL toma las WfNets como punto de partida y añade constructores que no son de redes de Petri. Para el desarrollo del presente trabajo fueron tomadas algunas ideas de YAWL, pero añadiendo elementos a las CCPN originales, es decir sin perder la estructura de red de Petri. Los patrones de workflow (Workflow Patterns, WP) es otra de las investigaciones del grupo de trabajo del profesor Aalst. Los WP son un conjunto de 20 patrones de flujo de control que proveen un benchmark para comparar lenguajes para modelado de workflow.

Los patrones de workflow más importantes que son identificados en la mayoría de sistemas workflow fueron reportados en el trabajo de tesis doctoral de Bartosz Kiepuszewsk [15] en la Universidad de Queensland, Australia en colaboración con W.V.D. Aalst. Estos 21 patrones se clasifican en 6 grupos: Patrones de control básico, Patrones de sincronización y ramificación avanzada, Patrones estructurales, Patrones que envuelven múltiples instancias, Patrones basados en estados y Patrones de cancelación. En este trabajo también se encuentra una comparativa mediante un caso de estudio de las siguientes herramientas de

[2] http://www.puml.org

Workflow que existen actualmente en el mercado: FileNet's Visual WorkFlo, Forté Conductor, Changengine, Staffware, Fujitsu i-Flow, MQSeries IBM Workflow, Verve, SAP R/3 Workflow. En esta comparativa se aprecia cómo en algunos casos es imposible representar ciertas instancias de un proceso, así como la diferencia en cuanto a simplicidad o complejidad en los modelos obtenidos con cada una de las herramientas. Lo más importante de esta comparativa es el resultado en cuanto a soporte de los 21 patrones de workflow, puesto que sólo entre 9 y 11 patrones soportan en promedio cada uno de los sistemas de Workflow analizados. Cabe mencionar que son diferentes los patrones que soportan cada uno de esos sistemas [15], [16].

En el sitio Web del grupo de investigación sobre Patrones de Workflow [16], se han publicado nuevos resultados del soporte que brindan otros productos de Workflow importantes como COSA, InConcert, Meteor, Mobile, entre otros, arrojando resultados similares a los mencionados anteriormente. El mismo trabajo fue realizado para lenguajes de descripción de procesos que se utilizan para el modelado de Workflow, tales como: XPDL, UML, BPEL (conocido también como BPEL4WS), XLANG, WSFL, BPML y WSCI soportando entre 11 y 14 patrones cada uno de estos lenguajes.

Sobre redes de Petri es importante mencionar el grupo: *Petri Nets World* [17]; este grupo está compuesto principalmente por investigadores de las Universidades de Aarhus en Dinamarca y de Hamburgo en Alemania además de investigadores de otras partes del mundo. Éstos se encargan de concentrar todas las publicaciones, herramientas de software, grupos de investigación, estándares y referencias bibliográficas sobre redes de Petri y temas relacionados a éstas. En cuanto a las CPN's, el grupo de redes de Petri Coloreadas [18] más importante se encuentra en la universidad de Aarhus, debido a que investigadores de esa universidad desarrollaron esta extensión de las redes de Petri. Su trabajo más importante es la herramienta de software *CPN-Tools* la cual posee buena aceptación en la industria para el modelado de sistemas.

## 1.2. Motivación

Un Workflow es un sistema reactivo por naturaleza, ya que mantiene interacción con su ambiente durante su ejecución [7]. Es decir un sistema reactivo corre en paralelo con el ambiente e intenta hacer cumplir ciertos efectos deseables en el ambiente. Esto lo hace por medio de reacción a cambios en su ambiente, llamados eventos. La respuesta del sistema

reactivo depende de su estado interno actual. Dada una respuesta, el estado del sistema reactivo puede cambiar [14].

El comportamiento de un sistema reactivo es modelado usualmente a través de reglas ECA (event-condition-action) [14]. Una regla E-C-A puede ser definida de la siguiente manera [7]:

Si el evento E ocurre y la condición C es verdadera, entonces realizar la acción A

Una regla ECA simple, con un solo evento puede ser modelada como una red de Petri Coloreada. Sin embargo para eventos compuestos los cuales son más complicados, los elementos básicos de una CPN no son suficientes [19]. Por esa razón, una modificación de la CPN ha sido propuesta en [19], [20] para representar estructuras de reglas ECA y su interrelación. Siendo así posible representar reglas ECA de manera más natural. El nombre de esta propuesta es *Redes de Petri Coloreadas Condicionales* (CCPN, por sus siglas en inglés).

En este trabajo se propone aplicar el enfoque de las CCPN para el modelado de procesos de workflow basados en el hecho de que son sistemas reactivos y su relación con las reglas ECA como enfoque para ser modelados. Para implementar la funcionalidad reactiva de los eventos se utilizan las reglas Evento–Condición–Acción (ECA), éstas poseen una sintaxis declarativa de alto nivel que permite una respuesta inmediata (reactiva). Las reglas ECA son un tipo de reglas muy utilizadas en la gestión de bases de datos activas, aunque no están limitadas a ese campo de investigación.

## 1.3. Planteamiento del problema

A pesar de que existen diversos enfoques para el modelado de procesos workflow, actualmente no existe una metodología gráfica estándar. Además a través de un marco de trabajo de evaluación conocido como patrones de workflow, han sido sometidos los lenguajes de modelado y herramientas de software para modelado de procesos workflow más utilizados e importantes en la actualidad. En investigaciones tales como [15], [9], [8] se presentan resultados y conclusiones que muestran que ninguno de los lenguajes y sistemas estudiados modela todas las posibles situaciones que pueden encontrarse en un proceso de workflow y que son representadas a través de patrones de workflow.

Una consecuencia de lo anterior, se presenta cuando el modelo de un sistema deja de ser un modelo pequeño, tiende a ser muy complejo mantener la perspectiva de lo modelado (principalmente de manera visual). Además cuando se trata de enfoques basados en gramáticas sin parte gráfica resulta menos intuitivo. La importancia de la definición del proceso de workflow radica en que en base al modelo obtenido se realiza la ejecución del proceso de manera automatizada con el WfMS. Es así como la correcta especificación del proceso asegurará en mayor medida la ejecución correcta de éste.

## 1.4. Objetivos

Los objetivos que se pretenden alcanzar con este trabajo se dividen en un objetivo general y en objetivos específicos.

### 1.4.1. Objetivo general

Proponer una metodología para modelado de procesos workflow basada en redes de Petri, con lo cual se tendrá una perspectiva formal y una gráfica del proceso modelado; la metodología propuesta tendrá un enfoque de reglas ECA al utilizar redes de Petri coloreadas condicionales (CCPN).

### 1.4.2. Objetivos específicos

- Plantear el modelo de CCPN que proporcione los elementos necesarios para modelar los conceptos que se encuentran presentes en la mayoría de los procesos de workflow.
- Mediante un marco de trabajo realizar una evaluación de la metodología propuesta y obtener los resultados correspondientes.
- Implementar un sistema de software que permita modelar y simular procesos de workflow utilizando la metodología obtenida.
- A través de un caso de estudio utilizar la metodología para modelar procesos de workflow utilizando el sistema de software implementado, de tal manera que permita obtener conclusiones sobre los beneficios y las limitantes, así como probar la viabilidad del trabajo realizado.

## 1.5. Descripción del trabajo

Para el desarrollo de este trabajo de tesis, se realizaron las siguientes actividades: Como primera tarea se realizó una investigación sobre el área de Workflow, la cual permitió conocer los conceptos y los temas relacionados a la tecnología Workflow; de igual forma se realizó el estudio sobre redes de Petri. Posteriormente se llevó a cabo el estudio sobre el modelado de workflow que permitió conocer los trabajos relacionados al tema que han sido reportados en la literatura. Con el conocimiento de los trabajos sobre modelado de workflow se obtuvo una amplia referencia utilizada al momento de plantear nuestra metodología. Además, se utilizó el estudio realizado con fines de evaluación y comparación.

Posteriormente, con el desarrollo de la metodología se obtuvo una modificación de las CCPN originales utilizadas en [20], [19]. Esta modificación se denominó Redes de Petri Coloreadas Condicionales para Workflow (Workflow Conditional Colored Petri Nets, WfCCPN por sus siglas en inglés) con el fin de diferenciar los trabajos realizados en el contexto de bases de datos activas. Con la modificación de algunas definiciones y la incorporación de nuevos elementos, se modelaron los conceptos de workflow que son definidos en el modelo de referencia de la WfMC en [21]. Con los conceptos modelados fue posible llevar a cabo la implementación de los veinte patrones de workflow definidos dentro del marco de evaluación (evaluation framework) conocido como patrones de workflow (workflow patterns[3]), obteniendo resultados satisfactorios ya que fue posible la implementación de todos ellos utilizando WfCCPN.

Al finalizarse las etapas anteriores fue posible iniciar el desarrollo del sistema de software. La implementación fue realizada tomando como base el sistema ECAPNSim desarrollado para el modelado de reglas ECA en bases de datos activas [19]. Se realizó una modificación y extensión en general de todo el sistema, añadiendo diversas características no consideradas por el ECAPNSim. El sistema desarrollado fue denominado: Simulador de redes de Petri-ECA para workflow (Workflow ECA Petri Net Simulator), el cual es referido a lo largo de este trabajo por sus siglas en inglés: *WfECAPNSim*.

Finalmente, se realizó la modelación de dos casos de estudio utilizando la metodología y el sistema de software obtenidos para efectos de verificación, validación, así como de comparación con trabajos relacionados. El trabajo realizado se reporta en el presente documento de tesis.

[3] http://www.workflowpatterns.com

### 1.5.1. Contribuciones

Las contribuciones que se ofrecen con el trabajo realizado son las siguientes:

- Un modelo extendido de las CCPN para el modelar procesos de workflow, denominado WfCCPN, ampliando así las áreas de utilización de las CCPN.
- Una metodología para la definición y simulación de procesos workflow utilizando las WfCCPN. Ésta ofrece elementos que permiten modelar aspectos que no son posibles de modelar con las redes de Petri coloreadas, La metodología se diseño de forma que ésta resultara lo más intuitiva y fácil de utilizar para usuarios con y sin experiencia en el uso de redes de Petri, esto se obtuvo gracias a los diversos tipos de elementos de la WfCCPN.
- Un sistema de software para el uso de las WfCCPN implementado en lenguaje Java el cual ofrece características que no son comunes de encontrar en sistemas de software para modelado de workflow gratuitos. El sistema WfECAPNSim puede ser considerado para futuros trabajos sobre representación y ejecución de workflow.

## 1.6. Organización de la tesis

La estructura de este trabajo fue dividida en 7 capítulos, los cuales se encuentran organizados de la siguiente manera:

En el capítulo 1, se presenta la investigación realizada, sobre la que se fundamenta esta tesis. Ésta consiste en un estudio sobre el estado del arte en cuanto al tema de modelado de workflow, también se proporciona la motivación para el desarrollo de este trabajo de tesis, y se plantea el problema sobre el cual se trabaja para obtener una propuesta de solución a éste. En el capítulo 2, se muestra lo que es la tecnología workflow y el modelado de workflow. En el capítulo 3, se proporciona la introducción a los conceptos básicos de redes de Petri, las redes de Petri Coloreadas su definición y utilidad, además de presentar la definición de una CCPN. En el capítulo 4 se expone la modificación a las CCPN denominada WfCCPN y se describe la manera en que los conceptos de workflow son modelados utilizando la metodología propuesta basada en WfCCPN. Además se muestra la implementación de los patrones de workflow utilizando dicha metodología. En el capítulo 5, se presenta el desarrollo de software realizado.

El sistema implementado, que fue denominado WfECAPNSim es descrito en cuanto a su arquitectura y diseño, así como en las características que posee. En el capítulo 6, se expone un caso de estudio modelado con la metodología propuesta en esta tesis. Finalmente, en el capítulo 7 se presentan las conclusiones sobre los resultados obtenidos, además del trabajo futuro que puede ser realizado al seguir sobre esta línea de investigación.

# Capítulo 2

# Tecnología Workflow

## 2.1. Introducción

En la década de los 90's surgió la tecnología Workflow[1] (Flujo de Trabajo[2]) en el contexto de Sistemas de Información, como una nueva solución para la administración o gestión de los procesos que se realizan dentro de una organización. Con el auge de Internet y de las Tecnologías de Información, el Workflow ha evolucionado de manera rápida y su uso se ha incrementado en los negocios de diversas industrias. Su característica principal es la automatización de procesos organizacionales (conocidos también como procesos de negocio), los cuales integran por lo general una combinación de personas y actividades basadas en máquinas, que además, en su realización interactúan con aplicaciones de software y tecnologías de información.

La evolución de los sistemas de información puede verse por etapas. En [13] se presenta la siguiente retrospectiva:

1975-1985 Administradores de bases de datos.

1985-1995 Administración de interfaces de usuario.

1995-2005 Administración de workflows.

En la primera etapa, con los sistemas *administradores de bases de datos,* se tuvo la existencia de diversos sistemas de información en los cuales se manejaba y administraba toda la

[1]En esta tesis, la tecnología Workflow será referida simplemente como Workflow.

[2]Flujo de trabajo es la traducción en español de workflow. Sin embargo, en esta tesis se utiliza la palabra en inglés debido a la aceptación que ésta tiene en la literatura.

información necesaria para llevar a cabo la producción en las empresas. Se lograron automatizar tareas, que antes eran realizadas manualmente. Sin embargo, en las siguientes etapas surgió la necesidad de mejorar el flujo de la información, y de obtener la información de manera rápida y eficiente, además de optimizar la productividad, acortar tiempos en la realización de los procesos, tener control sobre éstos, con el fin de reducir costos y mejorar la gestión en la organización. Como consecuencia a lo anterior, hubo un incremento en la competitividad organizacional, que actualmente también es resultado de la globalización que sucede en el mundo. En la segunda etapa, con la *administración de interfaces de usuario* se logró mayor facilidad de utilización de los sistemas para usuarios finales, gracias a interfaces de usuario cada vez más interactivas y menos susceptibles de aceptar errores humanos. Esto permitió la proliferación de los sistemas de información basados en computadoras, al ser más accesibles en cuanto a utilización para las personas en general. Finalmente, en la tercera etapa se tiene la *administración de workflows*, la cual ayuda en la administración del control y ejecución de procesos de negocio en las organizaciones.

Para entender mejor la tercer etapa en la evolución de sistemas de información, es necesario definir lo que significa un proceso de negocio, el Workflow Management Coalition (WfMC) principal organización en el mundo sobre el tema de Workflow lo define de la siguiente manera: *"un proceso de negocio es el conjunto de uno o más procedimientos o actividades directamente ligadas, que colectivamente realizan un objetivo del negocio, normalmente, lo anterior es dentro del contexto de una estructura organizacional que define roles funcionales y relaciones entre los mismos" [21].* También es necesario definir el término workflow, el WfMC lo define como: *"la automatización computarizada de un Proceso de Negocio, de manera total o parcial, en la cual documentos, información o tareas son pasadas de un participante a otro para efectos de su procesamiento. Lo anterior se realiza de acuerdo a un conjunto de reglas establecidas" [21].*

La administración de workflows no es una solución que se implementa únicamente con la instalación de un programa de software para después esperar la obtención de resultados de forma inmediata; por el contrario, una solución de este tipo representa el tener tecnologías de información, sistemas de software, políticas organizacionales, cooperación de los empleados de la empresa, entre otros elementos más que trabajan de acuerdo a la implementación necesaria para automatizar los procesos de negocio de la empresa; además, para la implementación se requiere de un análisis de la forma en que se realizan actualmente los procesos y en ciertos casos es necesario la aplicación de re-ingeniería en los procesos, lo que significa volver a

plantear la manera en que éstos se realizan con el fin de mejorar. Entre los sistemas de software necesarios, se encuentra una componente principal para la implementación de una solución basada en workflow, que es el Sistema Administrador de Workflow (Workflow Management System, WfMS por sus siglas en inglés), este tipo de sistemas son estudiados en la siguiente sección, antes de ello se remarcan a continuación las principales funciones y beneficios que ofrece la implementación del Workflow en una organización.

## 2.2. Workflow: Funciones y beneficios

Las funciones más comunes que realiza un workflow son:

- Asignación de tareas al personal.
- Aviso al personal de tareas pendientes.
- Permitir la colaboración en las tareas comunes.
- Optimización de recursos humanos y técnicos alineándolos a la estrategia de la empresa.
- Automatización de las secuencias de los procesos de negocio y optimización de las mismas.
- Agilización de los procesos de negocio y dando por resultado, un mejor servicio al cliente.
- Control y seguimiento de dichos procesos.
- Modificación de instancias de procesos sobre la marcha.

Las funciones citadas anteriormente reditúan en diversos beneficios a la organización, éstos han convertido al Workflow en una tecnología vital dentro los negocios de organizaciones actuales. Algunos de los beneficios más importantes son:

- Mejora en servicio y atención al cliente.
- Incremento en la ejecución de actividades en paralelo.
- Minimización en el tiempo para acceso a documentación, aplicaciones y bases de datos.

- Disminución del tiempo de transferencia de información entre tareas.
- Se asegura la completa colaboración del personal de trabajo.
- Permite tener conocimiento inmediato del estado de alguna tarea o proceso.
- Mejora en la gestión y optimización de procesos.

De lo anterior, lo más especifico de la tecnología de workflows es el Sistema Administrador de Workflows (WfMS). En la siguiente sección se proporciona una definición y las funciones de este tipo de sistemas, además de su arquitectura, la cual actualmente se encuentra estandarizada.

## 2.3. Conceptos básicos de Workflow

En el modelo de referencia del WfMC se establecen los conceptos que forman parte de la tecnología Workflow, tanto de un proceso de workflow como de los sistemas de workflow. Es preciso mencionar que en esta tesis nos referiremos al proceso de negocio como proceso de workflow y a los sistemas de workflow como *WfMS*. Los conceptos más importantes que forman un proceso de workflow son los siguientes: Caso, Proceso, Tarea, Enrutamiento, Datos, Personas, Roles, Grupos, Eventos, Plazos. Antes de presentar la arquitectura y funciones un WfMS, se proporciona una breve definición de los conceptos del proceso de workflow.

**Caso.** Es un elemento físico o abstracto el cual es producido o modificado por algún tipo de trabajo, los siguientes son sinónimos de un caso: Trabajo, Job, Producto, Servicio, Item. Un caso siempre tiene un inicio y un fin.

**Proceso.** Un proceso consiste de un número de tareas que necesitan ser realizadas y un conjunto de condiciones que determinan el orden de las tareas.

**Tarea.** Las tareas son unidades de trabajo formadas por un conjunto de acciones o actividades, que son realizadas por uno o más recursos (un recurso puede ser una persona o una máquina) en un intervalo de tiempo predeterminado. Una tarea es atómica, lo cuál quiere decir que no es divisible en tareas más pequeñas [22]. Es importante este concepto de tarea ya que para la fase de modelado es necesario tener previamente identificadas las tareas, esto se logra a través del análisis del proceso como un flujo de trabajo. Podemos considerar los

siguientes ejemplos de tareas: evaluar un reporte, calificar un examen, llenar un formato, contestar una llamada telefónica.

Las tareas pueden ser manuales, automáticas y semiautomáticas. Las tareas manuales son realizadas exclusivamente por personas. Por el contrario en una tarea automática en la que no existe participación alguna de personas. Mientras que, en las tareas semiautomáticas, se tiene una combinación de las dos anteriores.

**Enrutamiento.** En un proceso de workflow la información fluye a través de los distintos elementos que conforman el proceso. Es así como la información puede tomar diferentes rutas hasta llegar a completarse. Lo anterior es conocido como flujo de control del workflow; para representar esas posibles rutas se han definido los siguientes constructores básicos en el modelo de referencia de workflow de la WfMC.

**AND-Split.** Es un punto del workflow en donde un simple hilo de control[3] se divide en dos o más hilos, los cuales son ejecutados en paralelo dentro del workflow. Es decir, a través de cada uno de ellos puede fluir información en un mismo instante permitiendo que múltiples actividades sean realizadas simultáneamente. Cada AND-Split debe ser complementado con un AND-Join. En algunos sistemas un AND-Join recibirá hilos que pueden arribar de diferentes AND-Split [23].

**OR-Split.** Es un punto en el workflow donde al final de un único hilo de control se toma una decisión con la cual se bifurcará cuando se encuentra con múltiples alternativas.

**AND-Join.** Es un punto en el workflow donde dos o más actividades que se ejecutan en forma paralela convergen, entonces gracias a esa convergencia, otra actividad puede ser iniciada [23].

**OR-Join.** Es un punto en el workflow donde dos o más alternativas para continuar dentro del workflow re-convergen en una simple actividad como el siguiente paso a seguir en el workflow.

**Datos.** Los datos son los documentos, archivos, imágenes, registros de la Base de Datos, y otros utilizados como información para llevar a cabo el trabajo. Entre los datos manejados por el Workflow encontramos:

[3]La ruta que sigue la información de un elemento a otro se le conoce como hilo de control.

- Datos de Control: son los datos internos manejados por la lógica del sistema de Workflow.
- Datos Relevantes: son aquellos datos utilizados para determinar el ruteo de las distintas tareas del sistema.
- Datos de la Aplicación: estos datos son específicos de la aplicación, no son accedidos por la lógica del Workflow.

En cuanto a los datos, es muy utilizada la noción de documento como recipiente de información que se transmite de una tarea a otra. Por esta razón, cuando se hace referencia a datos manejados por el sistema, es posible referirse a estos datos como documentos. Existen ciertas propiedades que se le pueden asociar a un documento, tales como: la definición de los derechos de acceso a los mismos; las vistas definidas sobre ellos; manejo de accesos concurrentes (es decir, que dos personas o procesos puedan acceder al documento simultáneamente); también es posible definir formas de relacionar datos provenientes de fuentes externas al documento, tales como datos de la aplicación o de la base de datos.

**Personas.** Como se mencionó anteriormente las tareas son realizadas por recursos ya sean máquinas o seres humanos, es así como las personas dentro de una organización son parte importante en lo que a cumplimiento de los procesos se refiere. En muchos procesos las tareas son realizadas en un orden establecido previamente por ciertas personas de acuerdo a su capacidad o rol que desempeñe dentro de la empresa, además de realizarse en base a las condiciones y reglas de negocio. Una persona es conocida como un actor dentro del escenario de un workflow, sin embargo como actor también puede referirse a una máquina.

**Roles.** Cada rol define las distintas competencias potenciales que existen en el sistema. Se definen independientemente de las personas físicas a las cuales se les van a asignar dichos roles. Una persona puede tener más de un rol, siendo un rol un conjunto de características de un actor. En el caso de actores personas, un rol se puede referir a determinadas habilidades, responsabilidades o autoridad que ésta posea.

**Grupos** Es un conjunto de actores que realizan el mismo rol. Cuando se asignan tareas a un grupo de actores en vez de a un solo actor, se puede tener mayor flexibilidad en la selección de quien puede realizar la tarea, teniendo así una división de las labores más eficiente [7].

**Eventos.** Un evento es una interrupción que contiene información, éste tiene un origen y

uno o más destinatarios. La información contenida en el mensaje que se produjo por el evento puede ser implícita o dada por el usuario. Los eventos pueden ser disparados voluntariamente por el usuario; o en forma implícita durante un proceso según el estado de los datos o de decisiones tomadas por el usuario; o en forma automática. Por ejemplo, cuando un gerente de un banco hace una consulta sobre ciertos datos para hacer una auditoria, se dispara un evento que le devuelve la información sobre dicha consulta.

**Plazos.** Son los tiempos que se le asignan a ciertas tareas para que éstas sean realizadas o esten listas antes de tomar una determinación diferente a la normal, como podría ser su cancelación o la finalización del proceso mismo. Podemos considerar los siguientes ejemplos de plazos: el tiempo máximo que se le asigna a una tarea para que sea terminada, el tiempo máximo para recorrer una ruta; terminar una tarea antes de cierta fecha, entre otras más. A los plazos podemos asignarles eventos, de forma que con el incumplimiento del plazo se dispare dicho evento de forma automática.

## 2.4. Sistemas administradores de workflows

En las secciones anteriores se presentó el concepto de lo que es un proceso de workflow, así como los conceptos que lo conforman. También fue dada una breve definición de lo que es un sistema el cual ejecuta y administra procesos workflow, este tipo de sistemas se conocen como WfMS. En esta sección se describen las características y arquitectura de los WfMS de acuerdo al modelo de referencia de la WfMC.

### 2.4.1. Características

En su nivel más alto todos los WfMS proveen soporte en tres áreas funcionales:

- Las funciones en tiempo de diseño (Build-Time) relacionadas con la definición y modelado de cada proceso y actividades que le constituyen.

- Las funciones de control en tiempo de ejecución (Run-Time) relacionadas con la administración del proceso de workflow en un ambiente operacional y secuenciando las diversas actividades a ser mejoradas como parte de cada proceso.

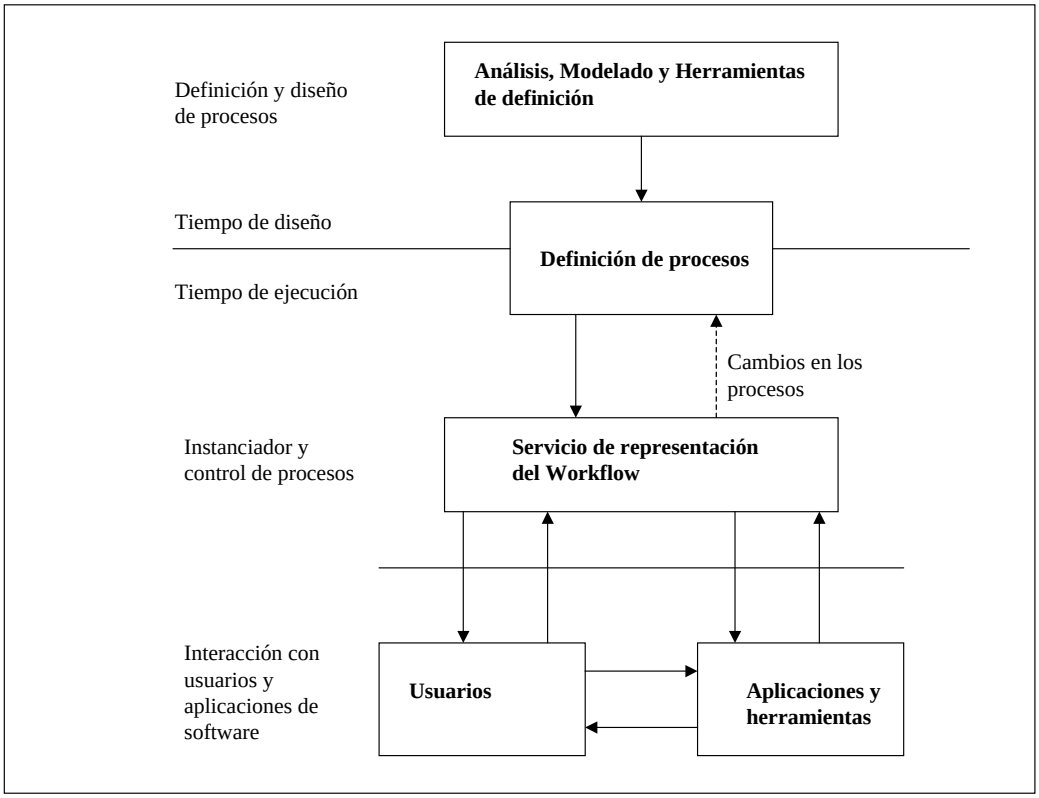


Figura 2.1: Características de un workflow

- Las interacciones en tiempo de ejecución entre personas, aplicaciones y tecnologías de información.

Estas características son reflejadas en los módulos o componentes que forman la arquitectura de un WfMS, las cuales se describen a continuación.

### 2.4.2. Modelo de Referencia de Workflow

La proliferación de los WfMS en el mercado de software y los diferentes enfoques por parte de cada una de las compañías desarrolladoras de este tipo de sistemas, además de una cada vez mayor necesidad de interactuar y compartir información entre organizaciones, develaron el problema de la interoperabilidad entre este tipo de sistemas. Es por ello que en 1992 se crea la Coalición para la administración de workflow (Workflow Managament Coalition, WfMC [2]), la cual se conforma por compañías de software, instituciones académicas y de investigación. Uno de sus primeros trabajos fue establecer una arquitectura de referencia para los WfMS, pensado para la adaptación de sistemas ya existentes, así también para los nuevos desarrollos, con el fin de lograr la interoperabilidad deseada. El Modelo de Referencia de Workflow (Workflow Reference Model) [21] fue presentado en su primera versión en 1994, habiendo muy pocas modificaciones a partir de su elaboración a la fecha. La figura 2.2 muestra la arquitectura para un WfMS sugerida en el modelo de referencia de workflow.

A continuación se describe la forma en cómo interactúan los elementos que componen un WfMS de acuerdo al modelo de referencia de la WfMC contenido en [21].

#### Herramienta para definición de procesos de workflow

Este componente y la *interfaz 1* se describen a continuación con más detalle que el resto de los componentes debido a que el proyecto realizado en esta tesis se enfoca en el modelado de procesos de workflow, actividad que se lleva a cabo gracias a este componente.

La herramienta para la definición de procesos, permite modelar, describir y documentar un proceso de workflow. Para esta tarea existen lenguajes y métodos de modelado para procesos de workflow, los cuales se basan en diferentes enfoques, como pueden ser: redes de Petri, UML (Unified Modeling Language), máquinas de estado finito, diagramas de flujo, entre otros más.

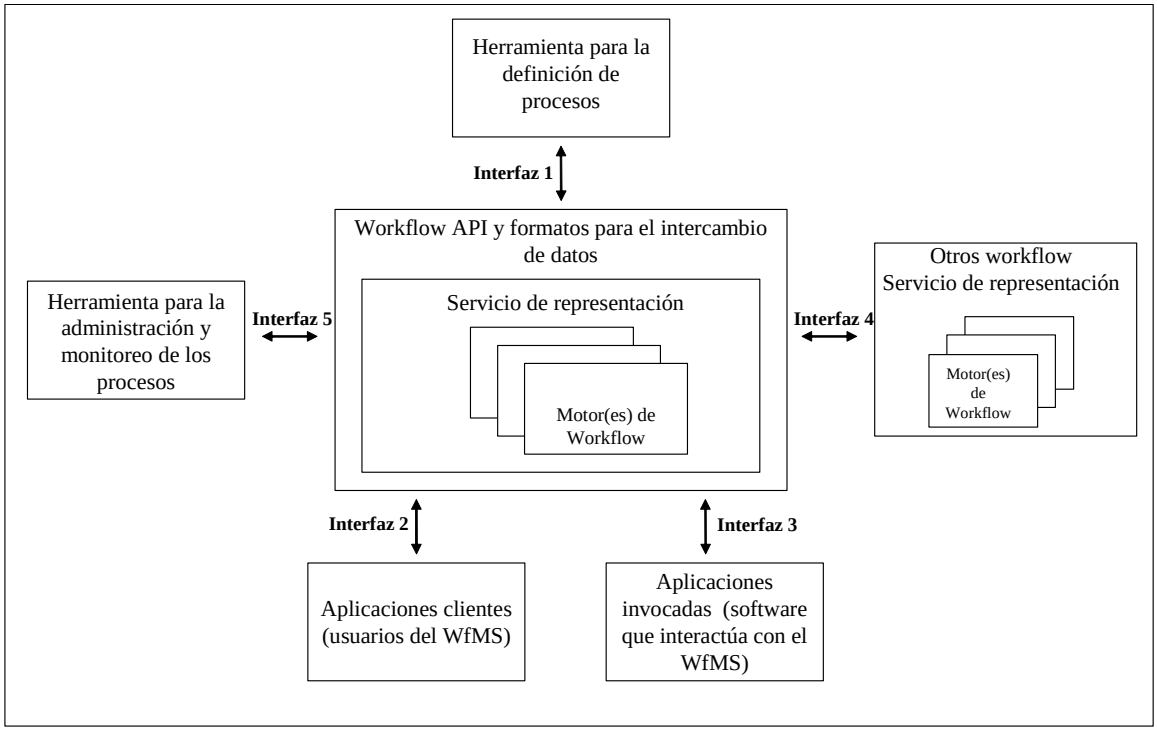


Figura 2.2: Modelo de referencia para un sistema administrador de workflow.

La definición de procesos se realiza modelando los conceptos de la vida real a través de los elementos que proporciona el lenguaje de modelado que sea utilizado. Preferentemente el proceso de workflow debe ser modelado de acuerdo a los conceptos que han sido definidos por el WfMC, los cuales se encuentran en [21].

La salida de este proceso de modelado y diseño es una "definición de procesos" la cual puede ser interpretada vía la *interfaz 1* en tiempo de ejecución por el *motor(es) de Workflow* a través del *Servicio de representación*. La *interfaz 1* permite la comunicación entre éste y el componente de representación de workflow.

### Servicio de representación

El *Servicio de representación de workflows* es el componente central encargado de interpretar la descripción de procesos realizada con la Herramienta para definición de procesos, además de controlar las diferentes instancias de procesos, administrar las secuencias de actividades, añadir elementos a la lista de trabajo de usuarios e invocar aplicaciones necesarias. Todas estas tareas son hechas por uno o más *motores de Workflow*, los cuales manejan la ejecución de las distintas instancias de los procesos. El motor de workflow es el software que

provee el control del ambiente de ejecución de una instancia de Workflow. En caso de estar en un entorno distribuido, pueden existir otros componentes de representación de workflow con sus respectivos motores de workflow. La comunicación con éstos se logra a través de la *interfaz 4*. El componente *Workflow WAPI y formatos para el intercambio*, permite la interacción del servicio de representación de workflow con el resto de los componentes, proporcionando datos entendibles para cada uno de los componentes.

### Aplicaciones cliente

Este componente representa los programas de software utilizados por el usuario final del WfMS en las actividades que requieren participación humana. La *interfaz 2* permite definir y manejar las listas de trabajo que se encuentran en los motores de workflow. Una lista de trabajo es una lista asignada que contiene actividades por hacer, trabajos pendientes, recordatorios entre otras, las cuales deben ser ejecutados por un usuario o grupo de usuarios.

### Aplicaciones invocadas

Este componente representa todo el software existente dentro de la organización, el cual es utilizado por el WfMS con el fin de interactuar y realizar ciertas actividades. Estas aplicaciones de software pueden encontrarse en cualquier lugar dentro de la red de trabajo, un ejemplo es el software administrador de correos electrónicos. La *interfaz 3* permite la comunicación entre este componente y el de representación de workflow a nivel de invocación, transformación y representación de datos, de manera que éstos sean entendibles para el motor de workflow.

### Herramienta para administración y monitoreo

El componente *Herramienta para administración y monitoreo* y la *interfaz 5* tienen como propósito permitir una vista completa del estado del workflow. Además, con éste es posible realizar auditorías sobre los datos del sistema. Esta herramienta es utilizada por el administrador del WfMS y por los altos mandos de la organización los cuales tienen la necesidad de tener información exacta del estado de sus procesos de negocio.

## 2.5. Modelado de Workflow

El modelado y especificación de procesos de workflow es un área que ha sido investigada por más de una década. En la actualidad sigue siendo un área de investigación en el campo académico, industrial y comercial. Se han reportado diversas propuestas para el modelado de procesos de workflow que poseen enfoques basados en extensiones de redes de Petri, en álgebras de procesos o en lenguaje de modelado unificado (UML). A pesar de la gran cantidad de propuestas de modelado, aún no existe una metodología gráfica estándar. A través de una extensión de las CCPN se propone una metodología gráfica para modelar los conceptos comunes que componen los procesos de workflow de acuerdo a [21]. Con este trabajo de tesis se pretende proporcionar una opción para modelar gráficamente, a través de una herramienta de software que sea fácil de comprender dirigida a usuarios que no tienen conocimientos sobre redes de Petri. Además, con este trabajo se extienden las aplicaciones en que tienen uso las redes de Petri coloreadas condicionales. En esta sección se presentan los aspectos relevantes en cuanto al modelado de workflow.

### 2.5.1. Perspectivas de modelado

Un proceso de workflow puede ser modelado desde diferentes perspectivas. Cada perspectiva representa de un proceso lo que considera más importante o más útil para trabajar con éste. De acuerdo a [15] la perspectiva de flujo de control (o de procesos) describe actividades y su ejecución, la perspectiva de datos delimita y describe el procesamiento de datos sobre la perspectiva de control. La perspectiva de recursos provee la estructura organizacional (personas y dispositivos que tienen un rol responsable de ejecutar actividades). La perspectiva organizacional describe las acciones elementales ejecutadas a través de actividades donde las acciones son mapeadas desde aplicaciones. A continuación se describe con más detalle cada una de éstas.

#### Flujo de control

Lo que concierne a esta perspectiva, es la ejecución de las tareas dentro de un tiempo específico(qué actividades han sido realizadas y en qué orden) [7]. En [23], se hace notar que es esencial que un lenguaje provea de manera adecuada esta perspectiva para definir workflows, así como en la mayoría de los trabajos realizados esta perspectiva es la que más importancia se le ha dado hasta el momento y es la que más ha sido estudiada. La perspectiva

de flujo de control podemos considerarla como la base de las siguientes perspectivas, ya que las restantes descansan en ésta.

### Flujo de datos

Representa los datos que viajan a lo largo de las rutas del flujo de control, proceso en el que los datos son consumidos y además son creados nuevos datos, resultado de la ejecución del proceso. Básicamente la unidad de datos más importante referida en workflow es el documento, teniendo en cuenta que puede éste encontrarse en cualquier medio, ya sea físico (p. ej. un memorandum, un recibo de banco, etc.) o electrónico (p. ej. almacenamiento en bases de datos, en formatos de procesadores de texto u hojas de cálculo). Estos documentos y las variables (datos de control) que deciden por dónde tienen que trazar su ruta dentro del workflow los propios documentos, son modelados dentro de esta perspectiva [23].

### Recursos

En la perspectiva de recursos, las características relevantes acerca de los actores (usuarios y empleados que laboran en la organización e inciden en el proceso que se esté automatizando) son modeladas. Es conocido que los actores son organizados en grupos de acuerdo a sus funciones o roles dentro de la organización [7]. En esta perspectiva se modela la jerarquía organizacional y la forma en como interactúan las personas que forman parte de la organización y que forman parte del workflow.

### Otras perspectivas

Algunas otras perspectivas de modelado para un workflow han sido investigadas y propuestas en diferentes trabajos; algunas de éstas son las siguientes:

**Excepciones**. Se encarga de modelar las posibles variaciones consideradas como situaciones excepcionales de manera que puedan ser anticipadas y monitoreadas en la ejecución de un workflow[24]. El modelado de excepciones se realiza en la mayoría de los casos utilizando el paradigma de reglas ECA. En [24] presentan una herramienta para la especificación de excepciones de workflow modelados como *"disparos"*[4] (estos son acciones que se ejecutan de manera automática en una base de datos, en respuesta a modificaciones que se realicen sobre

[4]Conocidos comúnmente como *triggers*, que es su traducción en inglés.

los datos que pertenecen a la base de datos). En ese trabajo se propone tener de manera predefinida un conjunto de estructuras de triggers genéricos almacenados en una herramienta de almacenamiento en forma de patrones. Al utilizar patrones pueden reutilizar determinados conjuntos de triggers.

### 2.5.2. Técnicas, lenguajes y herramientas de modelado de workflow

La modelación de workflow es también conocida como especificación de workflow. Para modelar un workflow es necesario un lenguaje o herramienta de modelado ya sea para modelado en general o para modelar específicamente workflows. De estos lenguajes o técnicas de modelado podemos obtener una clasificación que nos permita observar sus ventajas y desventajas. En primer lugar pueden ser clasificadas en dos grupos: Formales y No formales. A partir de dicha clasificación podemos hacer una distinción entre las técnicas que ofrecen un ambiente gráfico para el modelado y las que no lo ofrecen.

La *formalidad* es un principio que tradicionalmente no es bien acogido en el campo de los sistemas de información. Muchos lenguajes han sido propuestos sin tener fundamentos formales. Con el tiempo ha venido a ser claro que está no es una situación del todo deseable. Las técnicas de modelado formales, descansan sobre bases matemáticas que permiten que el modelo obtenido sea validado tanto sintácticamente como semánticamente, esta es la principal ventaja ya que permite que un modelo sea analizado y verificado, asegurando así en mayor grado la ausencia de errores. Su principal desventaja, es que son técnicas que pueden requerir un poco de más conocimiento o aprendizaje por parte de la persona que modela los procesos.

Ejemplos de este tipo de técnicas son:

- Redes de Petri
- Máquinas de estados
- Lógica temporal

Con las técnicas informales se tiene prácticamente lo contrario, es decir, el uso de éstas suele resultar de mayor facilidad y más comprensión al no estar atadas a restricciones matemáticas. Sin embargo, la principal desventaja es el no poder verificar los modelos obtenidos, con lo que es necesario esperar hasta tener el sistema desarrollado e implementado para verificar que todos sus módulos funcionan como se modelaron a través de alguna de

las técnicas informales. Lo anterior por supuesto, no es recomendable ya que incrementa los costos de desarrollo.

Ejemplos:

- UML
- Diagramas de flujo
- Diagramas IDEF

Los lenguajes sin una semántica formal fácilmente son susceptibles a ser inherentemente ambiguos. Una solución que en ocasiones es adoptada, es la de modelar los procesos utilizando una técnica informal para después mediante algún algoritmo de conversión generar la representación en alguna técnica formal, para de esa manera validar el modelo. Sin embargo, además de tener como resultado un trabajo extra, es muy difícil que el modelo obtenido represente de manera correcta el modelo original.

En [25] se muestra cómo la nueva versión de UML, la 2.0 (anteriormente 1.5) sigue careciendo de una semántica formal, y se muestra el trabajo de como con CPN son mapeados los modelos hechos con Diagramas de Actividad a estructuras de CPN para poder verificar semánticamente el modelo.

### Estándares para modelado

Los siguientes estándares utilizados para modelar procesos de workflow son los siguientes: XPDL, UML, BPEL, WSFL, BPML.

**XPDL** (XML Processing Description Language)**.** Es la forma estándar para intercambiar definiciones de workflows entre diferentes productos de software sin importar la plataforma o la metodología gráfica utilizada para definir los procesos, ya que éstos son mapeados a *XPDL* que está basado en XML como una representación intermedia que utilizan los diferentes productos de software WfMS (Workflow Managament Systems). Este estándar sólo se encuentra como gramática, no como estándar gráfico.

**UML** (Unified Modelling Language) es el lenguaje de modelado de sistemas de software más conocido en la actualidad, apoyado por la OMG (Object Management Group). UML cuenta con varios tipos de modelos, los cuales muestran diferentes aspectos del sistema modelado, particularmente para modelar workflows son utilizados los diagramas de actividad y en menor medida, los diagramas de secuencia.

**BPEL** (Business Process Execution Language) es un lenguaje común normalizado que tiene como objetivo principal optimizar la coordinación de procesos de negocios y servicios web; proporciona una especificación formal de los procesos y sus interacciones. Existe una extensión para modelar servicios web BPEL4WS.

**WSFL** (por sus siglas en inglés, Web Services Flow Language) es un lenguaje basado en XML propuesto por IBM para describir composición de servicios Web, sin embargo ha sido reemplazado por BPEL y BPEL4WS.

**BPML** (por sus siglas en inglés, Business Process Modeling Language) es un meta lenguaje para el modelado de procesos de negocio; así como XML, sólo es un meta lenguaje para el modelado de datos de negocio. BPML provee una ejecución abstracta del modelo para procesos colaborativos y transaccionales basados en el concepto de máquina transaccional de estados finitos.

#### Software existente

Las compañías de software y desarrollos académicos que se enfocan en la producción e investigación de sistemas administradores de Workflow las encontramos en gran cantidad. A continuación mencionamos los que de acuerdo a nuestra investigación resultan ser los más completos y/o conocidos.

En [15] se realizó un estado del arte sobre la industria, en donde se tiene un análisis no rigorista de 8 sistemas de software importantes, en donde es posible apreciar las diversas formas y las similitudes encontradas para modelar los diferentes conceptos de workflow. Estos productos son: FileNet's Visual WorkFlow, Forté Conductor, Changengine, Staffware, Fujitsu i-Flow, MQSeries Workflow, Verve, SAP R3 Workflow. Otros sistemas conocidos y que son referenciados en [16] son InConcert, Eastman, FLOWer, Domino, Meteor, Mobile y COSA este último basado en redes de Petri.

### 2.5.3. Redes de Petri para modelar Workflows

En base a diversos trabajos de investigación realizados desde inicios de los años 90's que han utilizado diversas modificaciones de redes de Petri para el modelado de Workflows a continuación mostramos una lista de ventajas que han sido identificadas desde siempre debido a las propiedades inherentes del modelo de red de Petri y su relación con los workflows. Así

mismo, mostramos algunos argumentos encontrados en la literatura que sugieren que las redes de Petri no son del todo buenas para modelar procesos Workflow.

### Ventajas y razones del modelado de procesos workflow con redes de Petri

**Semántica Formal.** Las bases matemáticas en las que descansa la teoría de redes de Petri proveen definiciones precisas y una semántica clara.

**Naturaleza gráfica.** La representación gráfica de las redes de Petri es bastante sencilla e intuitiva, fácil de aprender, y existen muchas aplicaciones de software que permiten de manera sencilla y genérica diseñar y modelar sistemas.

**Expresividad.** Las redes de Petri básicas pueden soportar primitivas funcionales necesarias para modelar sistemas de workflow de documentos sin tener detalle de qué contienen dichos documentos. Sin embargo podemos agregar expresividad con las extensiones de datos y de tiempo a nuestros modelos, por medio de las diferentes extensiones a las redes de Petri.

**Análisis.** Un conjunto completo de técnicas de análisis han sido desarrollados para verificar propiedades de un sistema modelado con redes de Petri: (safety, invariance, deadlock, etc.) y para calcular medidas de rendimiento en el modelo (response times, waiting times, occupation rates, etc.).

### Puntos en contra

Algunos autores como en [7] exponen las carencias y las desventajas de modelar con redes de Petri los procesos de workflow. Particularmente en ese trabajo son utilizados los diagramas de actividad de UML proveyéndoles de una semántica formal (ya que originalmente no la poseen, tampoco en su nueva versión UML 2.0). En [7] también se presenta uno de los argumentos más simples que pueden ser encontrados. Se menciona que las redes de Petri fueron inventadas antes de que lo fuesen los sistemas de administradores de Workflow. Por lo tanto no poseen las características necesarias. Sin embargo pensamos que por esa misma razón han surgido diversas modificaciones o extensiones para modelar workflows de una manera más natural. Otro argumento es que las redes de Petri sólo sirven para modelar sistemas cerrados y no sistemas reactivos, ya que en [7] se considera un workflow como un sistema reactivo, esto es por la naturaleza de disparo de las transiciones de las redes de Petri. En [9] se argumenta el

porqué las redes de Petri de alto nivel no proporcionan los elementos suficientes para modelar ciertas situaciones que pueden aparecer en un proceso de workflow.

## 2.6. Comentarios finales

En este capítulo se presentaron las características que posee un sistema de workflow, así como las ventajas de implementar un sistema administrador de workflows en una empresa. Se describió lo que representa y constituye un proceso de negocio o proceso de workflow. Los elementos que conforman un proceso de este tipo pueden ser modelados y representados por varias herramientas. El modelado de workflow es una de las grandes áreas de investigación respecto al tema workflow. En el siguiente capítulo se expone la teoría de redes de Petri que es utilizada para definir la propuesta de modelado para procesos de workflow que se presenta en este trabajo de tesis.

# Capítulo 3

# Redes de Petri

Este capítulo está dedicado a la teoría de redes de Petri con el fin de comprender de mejor manera las Redes de Petri Coloreadas Condicionales (Conditional Colored Petri Net, CCPN)[1]. Además aquí se revisan los conceptos básicos de redes de Petri coloreadas.

## 3.1. Introducción a las Redes de Petri

Las redes de Petri se deben a la tesis doctoral "Kommunikation mit Automaten[2]" del alemán Carl Adam Petri presentada en la facultad de matemáticas y física de la Universidad Técnica de Darmstadt en 1962.

Una Red de Petri (Petri Net, PN[3] por sus siglas en inglés) es una herramienta gráfica y matemática que ayuda a describir relaciones entre condiciones y eventos presentes en los sistemas del mundo real. Las características de un sistema que son modeladas utilizando PN son algunas de las siguientes: sincronización de procesos, eventos asíncronos, operaciones secuenciales, operaciones concurrentes, conflictos con recursos compartidos, procesos distribuidos, procesos paralelos, procesos no determinísticos y/o estocásticos.

Un sistema modelado a través de PN puede ser analizado de manera formal, obteniendo información de su comportamiento dinámico. Además es posible verificar propiedades que

[1] A lo largo de este trabajo nos referiremos a las redes de Petri coloreadas condicionales por sus siglas en inglés: CCPN.

[2] La traducción al inglés: Communication with Automata. New York: Griffiss Air Force Base. Tech. Rep. RADCTR-65-377, vol.1, Suppl. 1, 1966

[3] En esta tesis, una red de Petri es referida por sus siglas en inglés PN.

debe mantener el sistema. Por ejemplo, se puede verificar que no se encuentren estados inalcanzables en el sistema o verificar que no existan candados mortales (deadlocks) dentro del modelo.

Dos conceptos importantes en el modelado de sistemas dinámicos son los *eventos* y las *condiciones*. Los eventos son acciones que se presentan en el sistema y lo llevan a un estado. Es posible describir un estado como un conjunto de condiciones; para que cierto evento ocurra es necesario que ciertas condiciones se cumplan; a éstas se les llama precondiciones del evento, así mismo, la ocurrencia del evento puede llevar a otras condiciones; éstas se conocen como postcondiciones.

Al modelar con PN es necesario identificar las condiciones (pre y poscondiciones), así como los eventos que pueden suceder en él. De esta manera se traslada el sistema de la vida real al modelo en PN. La tarea de modelado es una actividad subjetiva. Es decir, el modelado de un sistema puede realizarse de diferentes maneras. Así cuando diferentes personas realizan el modelo de un mismo sistema, los modelo resultantes pueden diferir considerablemente; sin embargo ambos estarán modelando el comportamiento del sistema. Al tener las condiciones necesarias para que ciertos eventos del sistema sucedan, es posible modelar de forma modular, y de esta manera relacionar los modelos con otras condiciones; para esto es necesario conocer la estructura de la PN.

En [26] se presenta la tabla de la figura 3.1, en la que se muestra cómo una condición puede ser modelada a través de un lugar y un evento a través de una transición. De esta manera una precondición puede ser modelada empleando un lugar de entrada y una postcondición con un lugar de salida. En la tabla se muestran algunas maneras de interpretar los lugares y las transiciones.

Las redes de Petri poseen una parte matemática y una gráfica. Por lo tanto, pueden ser estudiadas formalmente a través de su estructura matemática y en su forma gráfica a través de su representación visual en forma de grafo dirigido.

### 3.1.1. Elementos

Una red de Petri es un grafo bipartito dirigido y se compone de los siguientes elementos:

- Un conjunto de lugares (nodos).

| **Lugares de entrada** | **Transiciones** | **Lugares de salida** |
|---|---|---|
| Precondiciones | Evento | Postcondiciones |
| Datos de entrada | Procesamiento | Datos de salida |
| Señales de entrada | Procesamiento de señales | Señales de salida |
| Requerimiento de recurso | Tarea | Liberación de recurso |
| Condiciones | Cláusula en lógica | Conclusiones |
| "Búfer" | Procesador | "Búfer" |

Figura 3.1: Posibles interpretaciones para los lugares y las transiciones en una red de Petri

- Un conjunto de transiciones.
- Una función de entrada
- Una función de salida.
- Marcado Inicial

Las funciones de entrada y de salida relacionan a los lugares con las transiciones y viceversa. La función de entrada es un mapeo de una transición $t_j$ a un conjunto de lugares conocidos como los nodos de entrada de una transición. La estructura de una PN es definida por los nodos, las transiciones, la función de entrada y la función de salida. A continuación se presenta la definición formal de una PN.

### 3.1.2. Definición formal

La estructura de una red de Petri es $PN = (P, T, F, W, M_0)$ donde:

$P = \{p_1, p_2, ..., p_m\}$ es un conjunto finito de lugares (nodos), donde $m \geq 0$.
$T = \{t_1, t_2, ..., t_n\}$ es un conjunto finito de transiciones, donde $n \geq 0$.
$F \subseteq (P \times T) \cup (T \times P)$ es un conjunto de arcos
$W : F \rightarrow \{1, 2, 3, ...\}$ es una función de peso
$M_0 : P \rightarrow \{0, 1, 2, 3, ...\}$ es la marca inicial

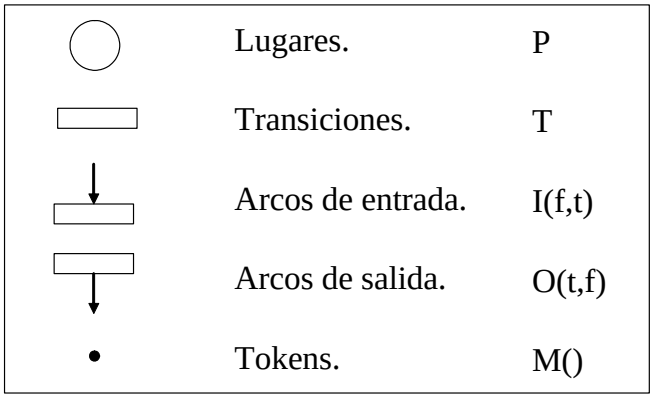


Figura 3.2: Representación gráfica de los elementos de una red de Petri.

Para definir el concepto de marca inicial es necesario referirse a los tokens. Un token es un concepto primitivo que forma parte de las PN, el cual es asignado a cada uno de los lugares, a esta asignación se le conoce como marca, un lugar puede contener $n$ tokens. Los tokens deciden la ejecución en la red de Petri. Por lo tanto la cantidad y la posición de los tokens cambia durante la ejecución. El paso de los tokens a través de las transiciones y los lugares de la PN, es utilizado comúnmente para interpretar la ejecución del sistema modelado.

Como herramienta matemática es posible obtener ecuaciones de estado, ecuaciones algebraicas y otros modelos matemáticos que describen el comportamiento de los sistemas. Mientras que como herramienta gráfica ésta permite obtener un concepto visual del sistema, a continuación se describe la representación gráfica.

### 3.1.3. Representación gráfica

La representación gráfica de una PN es importante ya que si se observa el modelo del sistema en forma gráfica y la manera en que cambia de un estado a otro, es posible mantener la atención y obtener una perspectiva más clara del análisis del problema. En la figura 3.2 se muestra la representación gráfica de los elementos que forman una red de Petri.

Los lugares son representados con círculos, las transiciones se representan mediante una barra o un rectángulo, los arcos dirigidos que conectan a los lugares con las transiciones y viceversa son representados con flechas. Los tokens son representados con pequeños círculos de color negro que van dentro de los lugares.

Un arco dirigido desde un lugar $p_j$ (transición $t_i$) a una transición $t_i$ (lugar $p_j$) define a

$p_j$ como un lugar de entrada (salida) de $t_i$, el cual describe como $w(p_j, t_i) = w(t_i, p_j) = 1$. Si $w(p_j, t_i) = k$( ó $w(t_i, p_j) = k$), entonces existen $k$ arcos dirigidos (paralelos) que conectan el lugar $p_j$ con la transición $t_i$ (o que conectan la transición $t_i$ con el lugar $p_j$). Generalmente en un esquema gráfico, los arcos paralelos que conectan a un lugar (transición) con una transición (lugar) se representan por un solo arco dirigido, etiquetado con el número de arcos que representa a su peso $k$. Un lugar que contiene un punto en su interior representa a un lugar que contiene un token. La capacidad de almacenamiento de tokens en cada lugar es infinita, y para denotar el numero de tokens o marcado de un lugar se utiliza M(p).

### 3.1.4. Regla de disparo

La ejecución de una PN es controlada por el número y distribución de los tokens. La ejecución de la red de Petri se activa disparando sus transiciones. Cuando una transición es disparada se remueven tokens de los lugares de entrada y se crean tokens en los lugares de salida. El número de tokens que son removidos es igual al peso del arco que conecta al lugar de entrada con la transición que fue disparada. De igual manera el número de tokens creados está dado por el peso del arco que conecta la transición con el lugar de salida.

Para que una transición pueda dispararse, requiere estar habilitada, Dicha habilitación depende de una condición conocida como regla de habilitación. Esta regla se describe a continuación:

Una transición $t$ se dice que será habilitada si cada lugar de entrada $p$ de $t$ contiene al menos un número de tokens igual al peso $k$ del arco dirigido que conecta a $p$ con $t$, es decir, que $M(p) \geq w(p, t)$ para algún $p$ en $P$. En la figura 3.3 se muestra la regla de disparo y algunos ejemplos.

La figura 3.3 a) muestra la condición de habilitación. Cuando se omite el peso del arco por convención el peso es 1, no sucede así con el número de tokens. Es decir, si no existen tokens el número de tokens es 0. Es así como en el ejemplo mostrado en la figura 3.3 b) la red no se habilitará nunca ya que el peso del arco es 1 y no es ni mayor ni igual a 0, que es el número de tokens en el lugar. Un caso similar se da en el ejemplo de la figura 3.3 c). Sin embargo, la figura 3.3 d) muestra un ejemplo en el que la transición sí se activará ya que el número de tokens en el lugar es mayor al peso de su transición.

Una transición activada puede o no dispararse, (dependiendo de la interpretación adicional que tenga la transición). Esencialmente el disparo de las transiciones y el cambio de tokens

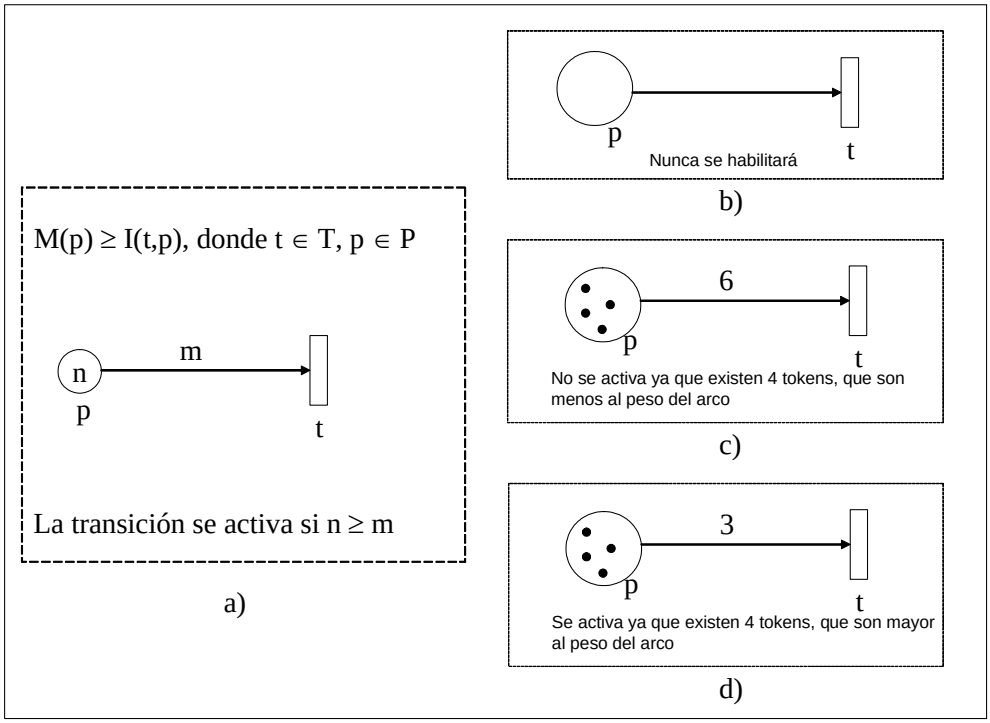


Figura 3.3: Habilitación de transiciones en una red de Petri

entre los lugares de la red, es lo que permite ver el comportamiento dinámico de un sistema. A lo anterior se le conoce como la ejecución del juego de tokens de la PN.

En la figura 3.4 se muestra la ejecución de una PN sencilla. En ella podemos verificar cómo el peso de los arcos decide el número de tokens que son removidos del lugar de entrada y el número de tokens creados en el lugar de salida. También podemos observar que llega un momento en que el lugar de salida queda sin tokens, esto es después del tercer disparo. Esto provoca la deshabilitación de la transición y la ausencia de posteriores disparos en el sistema, hecho que podríamos interpretar como la finalización de nuestro sistema, o el agotamiento de sus recursos, esto es dependiendo de la interpretación con que modelamos nuestro sistema.

### 3.1.5. Ejemplo

A continuación se presenta un ejemplo clásico [26] el cual a pesar de su sencillez ilustra claramente la idea de cómo utilizar las redes de Petri.

El ejemplo de la figura 3.5 muestra la bien conocida reacción química: $2H_2 + O_2 \rightarrow 2H_2O$. El marcado inicial de la red de Petri está dado por dos tokens en el lugar de entrada $H_2$ (se representan dos moléculas de $H_2$) y dos tokens en el lugar de entrada $O_2$ (se representan dos

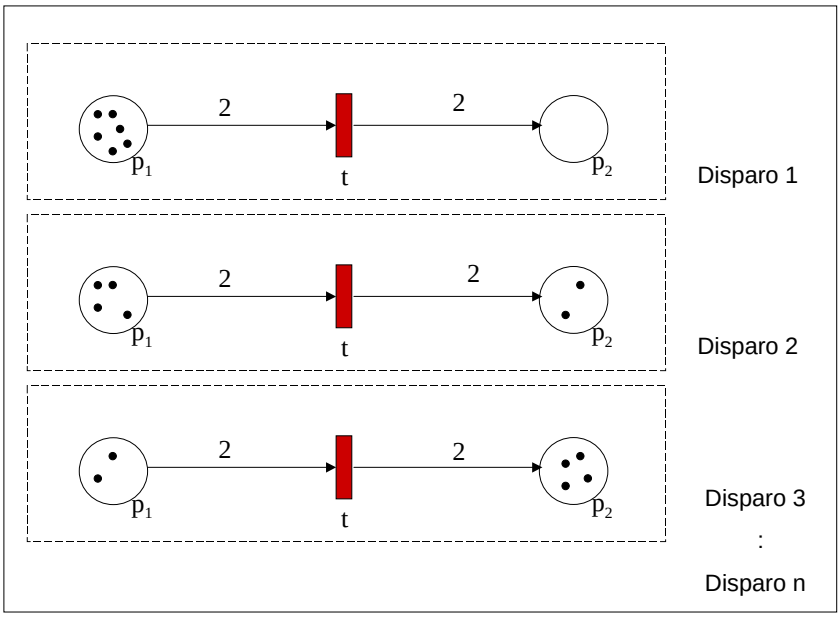


Figura 3.4: Ejemplo de una secuencia de disparos en una red de Petri.

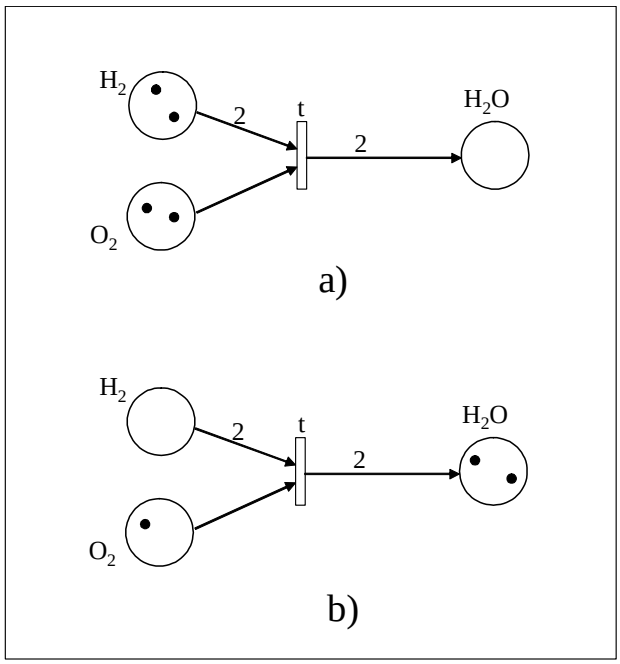


Figura 3.5: Ejemplo de una red de Petri.

moléculas de $O_2$). La transición se encuentra activada ya que en el primer caso el peso del arco es 2 y el número de tokens cumple la regla de activación $PesoArco <= NumTokens$ al tener más de un lugar de entrada la transición $t$ requiere que en todos los arcos se cumpla la regla de activación. El peso del arco del lugar $O_2$ también cumple al tener un valor menor al número de tokens del lugar con que conecta. Después del disparo de la transición $t$, la red de Petri toma el estado presentado en la figura 3.5 b). Claramente se observa que la transición $t$ nunca volverá a estar activa. El resultado son dos moléculas de $H_2O$.

### 3.1.6. Propiedades y métodos de análisis

Una de las mayores ventajas de utilizar técnicas de modelado que poseen semántica formal (tal es el caso de las redes de Petri) es que el modelo obtenido, puede ser analizado a través de técnicas matemáticas. Las PN poseen propiedades que permiten analizar la correctitud del modelo desde distintas perspectivas, principalmente en aspectos de concurrencia y paralelismo.

Dos tipos de propiedades pueden ser estudiadas en un modelo de red de Petri[26]:

- Las que dependen del marcado inicial (comúnmente referidas como propiedades de comportamiento).
- Las que son independientes del marcado inicial (propiedades estructurales).

A continuación se describe brevemente cada una de las propiedades de comportamiento ya que estas propiedades son de más interés en la tarea de modelar los procesos de tipo workflow que fueron presentados en el capítulo anterior. Esto es debido al interés de conocer el comportamiento y las propiedades estructurales en un proceso de negocios. En [26] es posible profundizar en el tema de las propiedades de una PN.

#### Propiedades de comportamiento

**Alcanzabilidad**. Una secuencia de disparos da lugar a una secuencia de marcados. Un marcado $M_n$ se dice que es alcanzable desde un marcado $M_0$ si existe una secuencia de disparos que transforma $M_0$ en una red $(N, M_n)$; se denota por $R(N, M_0)$. Un disparo o secuencia de ocurrencias es denotado por $\sigma = M_0\ t_1\ M_1\ t_2\ M_2\ ...tn\ Mn$ o simplemente $\sigma = t_1\ t_2\ ...t_3$. En este caso, $M_n$ es alcanzable desde $M_0$ a través de $\sigma$ por lo que se escribe

$M_0[\sigma > M_n]$. El conjunto de todos los posibles marcados que hacen alcanzable a $M_0$ en una red $(N, M_0)$ es denotado por $L(N, M_0)$ o simplemente por $L(M_0)$. Con lo anterior podemos definir el problema de la alcanzabilidad: Encontrar si $M_n \in R(M0)$ para un marcado $M_n$ dado en una red $(N, M_0)$.

**Acotamiento**. Una red $(N, M_0)$ se dice que está k-acotada o simplemente acotada si el número de tokens en cada lugar no excede un número finito $k$ para ningún marcado alcanzable desde $M_0$. Una red de Petri se dice que es segura si es 1-acotada [20]. El término *"segura"* se refiere al conocimiento del número de tokens $k$ que habrá siempre en los lugares de la red. Lo anterior, cierto tipo de sistemas lo requieren con el fin de que los elementos representados por los tokens no tiendan a infinito, con lo cual aseguran estabilidad en el sistema.

**Vivacidad**. Este concepto está relacionado con la ausencia total de candados mortales o *deadlocks* como son conocidos en el área de sistemas operativos. El concepto: *"viva"*, está estrechamente relacionado con la situación de candado mortal (deadlock), el cual ha sido situado en el ámbito de los sistemas operativos. Una red de Petri que modela un sistema y se encuentra libre de candados mortales es viva. Esto implica que para alguna marca alcanzable $M$, es posible disparar al menos una transición en la red a través de alguna secuencia de disparo. Sin embargo, este requisito debe ser bastante estricto para representar algún sistema real o escenarios donde se presente un comportamiento libre de candados mortales [20].

**Persistencia.** Una red de Petri se dice que es persistente si en el caso de tener dos transiciones cualesquiera habilitadas, el disparo de una no deshabilita a la otra transición. La noción de persistencia es utilizada en el contexto de programas paralelos [26].

### 3.1.7. Extensiones de las redes de Petri básicas

Las extensiones o modificaciones a las redes de Petri básicas fueron desarrolladas de acuerdo a necesidades que han surgido en el área de investigación referida a modelado de sistemas. Existen algunas extensiones de redes de Petri que presentan ciertas variaciones contra la estructura original. Estas variaciones fueron desarrolladas con la finalidad de cubrir características de sistemas que no manejan únicamente eventos y condiciones debido a que existen sistemas que necesitan incluir información en forma de datos o variables. En ciertos casos es necesario manejar intervalos de tiempo u otras propiedades con el fin de representar las características del sistema modelado.

Algunas extensiones de redes de Petri son las siguientes:

- Red de Petri Coloreada
- Red de Petri Difusa
- Red de Petri Estocástica
- Red de Petri con Tiempo
- Red de Petri Temporizada

En la siguiente sección se presenta una de estas extensiones: la red de Petri coloreada (Colored Petri Net, CPN por sus siglas en inglés) de la cual se presenta una breve descripción. Posteriormente se presentan brevemente las redes de Petri coloreadas condicionales (Conditional Colored Petri Net, CCPN por sus siglas en inglés) que fueron propuestas en [20], [19] para modelar bases de datos activas. En este trabajo de tesis se extienden sus características para modelar procesos de workflow.

## 3.2. Red de Petri Coloreada

Las Redes de Petri Coloreadas (CPN por sus siglas en inglés), fueron desarrolladas por el grupo de la Universidad de Aarhus [18] dirigidos por Kurt Jensen hace ya más de 20 años. *"El desarrollo de las redes de Petri coloreadas (CP-nets o CPN) fue dirigido por el deseo de desarrollar un lenguaje de modelado, que fuese teóricamente bien formado y lo suficientemente versátil para ser usado en sistemas prácticos, del tamaño y complejidad típicos en proyectos de la industria" [27].*

Una CPN combina el poder de modelado de una PN básica con la expresividad de un lenguaje de programación de alto nivel, ésto se logra al permitir la representación de diferentes tipos de datos en el modelo, utilizando para ese fin los tokens que circulan a través de la red. A diferencia de una PN en la que todos los tokens son de color negro, en una CPN se introduce el concepto de color del token o token coloreado, teniendo así tokens de diferentes colores, por lo tanto se tiene que *"un color = dato"*. Las características particulares de las CPN proveen un mayor potencial para el modelado de sistemas. Las expresiones de arco y las guardas de

las transiciones permiten restringir mejor las condiciones de disparo de las transiciones en una CPN con respecto a las PN clásicas. El empleo de colores y de expresiones de arco hace que el modelo que se obtenga con CPN sea más compacto que el modelo equivalente con PN clásicas.

Cada token puede transportar información cuyo tipo depende del lugar donde esté colocado, por tal razón en las CPN se incluye el concepto de dominio de color para cada lugar. Este dominio constituye el tipo de dato asociado a los tokens que pueden llegar a cada lugar.

Los cambios de estados en una CPN son llevados a cabo mediante la activación y disparo de las transiciones como en una PN básica. Sin embargo como ahora se tiene diferenciación entre los tokens de un lugar, es necesaria información adicional que permita conocer qué tipo de tokens deben de ser removidos de un lugar de salida y qué tipo de tokens deben ser generados en un lugar de entrada teniendo en cuenta los colores de éstos. A continuación se describe la definición formal de una CPN y los conceptos que permiten modelar utilizando esta extensión de PN.

### 3.2.1. Definición formal

La estructura de una red de Petri coloreada es $CPN = \{P, T, C^-, C^+, cd\}$ donde:

$P$ es el conjunto finito de lugares.
$T$ es el conjunto finito de transiciones.
$C$ es el conjunto finito de clases de colores para los tokens.
$cd : P \cup T \rightarrow C$ es una función que define el dominio de color de cada lugar a transición.
$C^-[p,t], C^+[t,p] : cd(t) \rightarrow Bag(cd(p))$ son las matrices de transición previa y posterior respectivamente.

Los estados en una red de Petri coloreada son representados por los lugares $p \in P$. Gráficamente éstos son representados por círculos y el nombre de cada lugar se escribe dentro de éstos. Cada lugar tiene asociado un tipo (color) el cual determina el tipo de dato que el lugar puede contener. Por convención, el nombre del tipo es escrito cerca del círculo con letras itálicas [28]. De esta manera, cuando se ejecute la CPN los tokens que lleguen a un lugar deberán ser del mismo tipo que tiene asociado el lugar, recordando que los tokens representan un tipo de dato según su color.

### 3.2.2. Marcado y regla de disparo

Formalmente el marcado de una CPN es un vector indexado con respecto a los lugares, el cual asigna a cada uno de los lugares $p$ un *multiconjunto* definido sobre $cd(p) : M[p] \in Bag(cd(p))$.

Se denota como $M[p, c]$ al número de tokens $c \in cd(p)$ que están presentes en $M[p]$. En las PN se tiene un marcado inicial, el cual es denotado como $M_0$. El marcado inicial en una CPN es definido utilizando una notación formal basada en sumas y se especifica de la siguiente manera:

$$M_0 = p_5(b_1 + ... + b_{nb}) + p_6(e_1 + ... + e_{nc}) + p_7(m_1 + ... + m_{nc}), .$$

Para que una transición $t$ en una CPN esté habilitada, debe disponer de un número suficiente de tokens en sus lugares de entrada (condición básica de una PN). Además, los valores de esos tokens deben corresponder a las expresiones de los arcos de entrada. Una transición $t$ opcionalmente puede tener una condición de guarda. De esta manera si está habilitada sólo se disparará si satisface su condición de guarda. En las expresiones de arco y en las condiciones de guarda se pueden utilizar variables, que corresponden a alguno de los tipos válidos, o bien constantes (colores).

El disparo de una transición coloca en los lugares de salida los valores correspondientes, en cuanto a color y multiplicidad, según la evaluación que se haga de las expresiones de los arcos de salida. Con esta funcionalidad es posible representar datos en el modelo del sistema que se representa con CPN's.

### 3.2.3. Ejemplo

En la figura 3.6 se muestra un breve ejemplo que ilustra la forma de uso de una red de Petri coloreada.

En el ejemplo se muestra que cada uno de los lugares tiene un tipo de dato asociado, este puede ser básico (*Integer* o *String* o *Float*, etc.) o también puede tratarse de una composición de tipos como por ejemplo: $Integer * Data$ el cual es un producto de tipos. El conjunto de colores o datos se muestra en la figura 3.7.

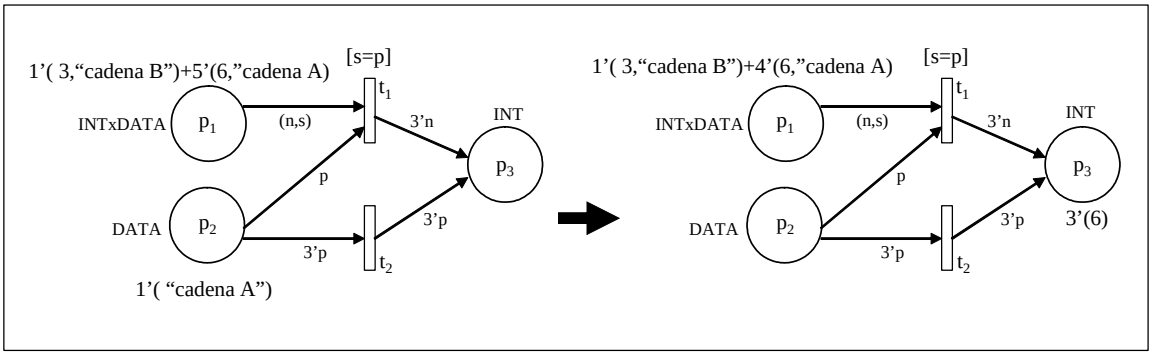


Figura 3.6: Ejemplo de una red de Petri coloreada

```
Type INT = Integer
Type DATA = String
Type INTxDATA = producto INT*DATA

Var n : INT
Var p, s : DATA
```

Figura 3.7: Definición de tipos (colores) para el ejemplo de la figura 3.6.

Cada color posee su multiplicidad, con lo que se indica cuantas copias del dato se encuentran en el lugar. En la CPN de la figura 3.6 el lugar $p_1$ posee 6 tokens, uno de estos tokens es un $INT * DATA$ con valor de $(3, \text{“}cadenaB\text{”})$, mientras que los otros 5 tokens poseen el valor $(6, \text{“}cadenaA\text{”})$.

Los arcos de entrada y de salida de las transiciones, poseen variables para restingir el paso de datos, es decir únicamente los datos que se especifiquen en el arco podrán transitar a través de él, debido a que la transición sólo será habilitada cuando datos de ese tipo se encuentren en el lugar de entrada.

En la CPN de la figura 3.6 la transición $t_1$ será habilitada ya que existen más de un tipo de dato ($INT * DATA$) en el lugar $p_1$ y en el lugar $p_2$ existe por lo menos un token de tipo $DATA$, es el tipo que se pide transite por el arco que conecta a $p_1$ con $t_1$ por lo tanto al cumplirse las dos condiciones en los arcos de entrada que conectan a la transición $t_1$ ésta se dispara. Con la transición $t_2$ se tiene el caso en que no será habilitada y por lo tanto no será disparada, esto es debido a que el peso del arco que conecta a $p_2$ con $t_2$ especifica que deben ser 3 tokens de tipo $p$, siendo que en el lugar $p_2$ únicamente se tiene un token el cual es $1'(\text{“}cadenaA\text{”})$ con lo que no se cumple ni siquiera la condición básica de una PN que establece que el número de tokens en el lugar de entrada debe ser mayor o igual al peso del arco que lo conecta con la transición en cuestión. Por último tenemos que en el lugar $p_3$ se crean 3 tokens de tipo $n$, es decir únicamente el tipo $INT$ se mantiene, desechándose el tipo $DATA$ en este caso la marca resultante es $3'(6)$.

A continuación se presenta brevemente lo que son las redes de Petri coloreadas condicionales, las cuales son una modificación de las redes de Petri coloreadas presentadas en esta sección.

## 3.3. Red de Petri Coloreada Condicional

Las redes de Petri coloreadas condicionales (Conditional Colored Petri Net, CCPN por sus siglas en inglés) fueron concebidas como una modificación a las CPN para modelar, analizar y ejecutar reglas ECA (Evento Condición Acción), aplicadas al estudio de bases de datos activas. En [20] se encuentran algunos resultados del trabajo realizado sobre modelado de bases de datos activas utilizando CCPN.

Como propósito y justificación de las CCPN es posible mencionar lo siguiente: *"La transición de una red de Petri no tiene la capacidad de realizar la evaluación de expresiones lógicas, solamente evalúa la presencia o ausencia de tokens en sus lugares de entrada. Pero, si agregamos a la transición la capacidad de evaluar expresiones lógicas entonces la parte condicional de la regla puede representarse como una transición de red de Petri" [20].*

La CCPN es una extensión de PN, la cual hereda algunos de sus atributos, como la regla de disparo de transiciones. Además, en la CCPN se toman conceptos que están presentes en la definición de la red de Petri coloreada (CPN), tales como la definición de tipos de datos, asignación de colores (valores) a los tokens, y la asignación de tipos de datos a los lugares. En CPN's la asignación de tipos de dato se hace hacia todos los lugares de la CPN, en el caso de la CCPN, la asignación de tipos de datos a lugares no es general, ya que en la CCPN se manejan lugares (lugares virtuales) los cuales poseen la capacidad de alojar tokens de diferentes tipos de datos.

En cuanto a la representación de reglas ECA con CCPN se considera lo siguiente: en una regla ECA se evalúa la condición de la regla. En la CCPN se utiliza una función que realiza la evaluación de la parte condicional de la regla ECA almacenada en una transición, mientras que a través de lugares de entrada de la transición se representan los eventos, en contraste, con los lugares de salida se representan las acciones. En [20], [19] se encuentra la definición de una CCPN para modelar conjuntos de reglas ECA presentes en bases de datos activas. A continuación se describen los elementos de una regla ECA.

### 3.3.1. Definición de Reglas ECA

Para comprender de manera más amplia lo que es una regla ECA y para lo que es utilizada, a continuación se presenta de forma breve su modelo de conocimiento. El modelo de conocimiento describe las características estructurales de las reglas ECA de forma individual. Cada elemento que conforma una regla ECA se describe a continuación:

**Evento.**

Un evento es algo que sucede en un instante de tiempo dado. Se utiliza un mecanismo de programación para permitir que un programa de aplicación señale la ocurrencia de un evento, por ejemplo, como respuesta a alguna información introducida por el usuario. Además de los

eventos comunes generados ya sea por el sistema de acuerdo a su flujo de ejecución o a tareas iniciadas por un usuario del sistema, se consideran las siguientes:

- Excepción. El evento sucede como resultado de una excepción, por ejemplo, intentar acceder a un documento que no existe.
- Reloj. El evento sucede en un instante de tiempo que puede ser absoluto, relativo a otro acontecimiento o periódico.
- Externo. El evento que sucede por una situación fuera del sistema, por ejemplo, una consulta de información de un sistema perteneciente a otra empresa.

Un evento puede ser de dos tipos: primitivo o compuesto [20]. El evento es primitivo cuando sucede por una ocurrencia de cualquiera de las fuentes descritas anteriormente.

Un evento es compuesto cuando sucede por una combinación de eventos primitivos o compuestos usando un rango de operadores que constituyen el álgebra de eventos. Los operadores de eventos más comunes son los siguientes:

1. Disyunción, $< E_1 \; o \; E_2 >$ sucede cuando cualquiera de los eventos $E_1$ o $E_2$ ocurre.
2. Conjunción, $< E_1 \; y \; E_2 >$ sucede cuando ambos eventos $E_1$ y $E_2$ ocurren en cualquier orden.
3. Secuencia, $< sec(E1, E2) >$ sucede cuando $E_1$ ocurre antes que $E_2$.
4. Cerradura, $< cerradura(E) \; en \; Int >$ sucede la primera vez que $E$ ocurre en el intervalo de tiempo $Int$, sin importar las posteriores ocurrencias de $E$;
5. Historia, $< veces(n, E) \; en \; Int >$ se señala cuando el evento $E$ ocurre $n$ veces durante el intervalo de tiempo $Int$.
6. Negación, $< negación(E) \; en \; Int >$ detecta la no ocurrencia del evento $E$ en el intervalo de tiempo $Int$.
7. Último, $< last(E) \; en \; Int >$, toma la última ocurrencia del evento E en el intervalo de tiempo $Int$.

8. Simultáneo, $< sim(e1, e2) >$ ocurre cuando suceden al mismo tiempo los eventos $E_1$ y $E_2$; alguno, $< any(E_1, E_2, ..., E_n, E_m) >$ ocurre cuando han sucedido $m$ eventos de $n$ posibles.

Estos eventos son los más comunes que se refieren en la literatura sobre algebras de eventos presentados en [20].

**Condición.**

La condición de una regla evalúa el contexto en el cual el evento tomó lugar. En las reglas ECA la condición puede ser opcional. Si la condición no se especifica, entonces se obtienen reglas de la forma evento - acción. Sin embargo, si el evento y la condición son opcionales, uno de ellos debe ser especificado.

**Acción.**

El rango de tareas que pueden ser llevadas a cabo por una acción se especifica como sus opciones. Las acciones pueden ser la modificación, creación, eliminación de un documento, o inicio de tareas dentro del proceso.

### 3.3.2. Representación de una regla ECA con CCPN

Las reglas ECA se modelan a través de la CCPN de acuerdo a la siguiente interpretación:

1. Los eventos se modelan como lugares de entrada de una transición
2. Las condiciones se modelan como condiciones agregadas a las transiciones.
3. Las acciones se definen en una transición y su resultado se refleja en lugares de salida de esa transición.
4. Las reglas ECA en sí mismas se mapean en transiciones.
5. El disparo de la regla corresponde al disparo de la transición.
6. La detección de eventos se modela con tokens iniciales.

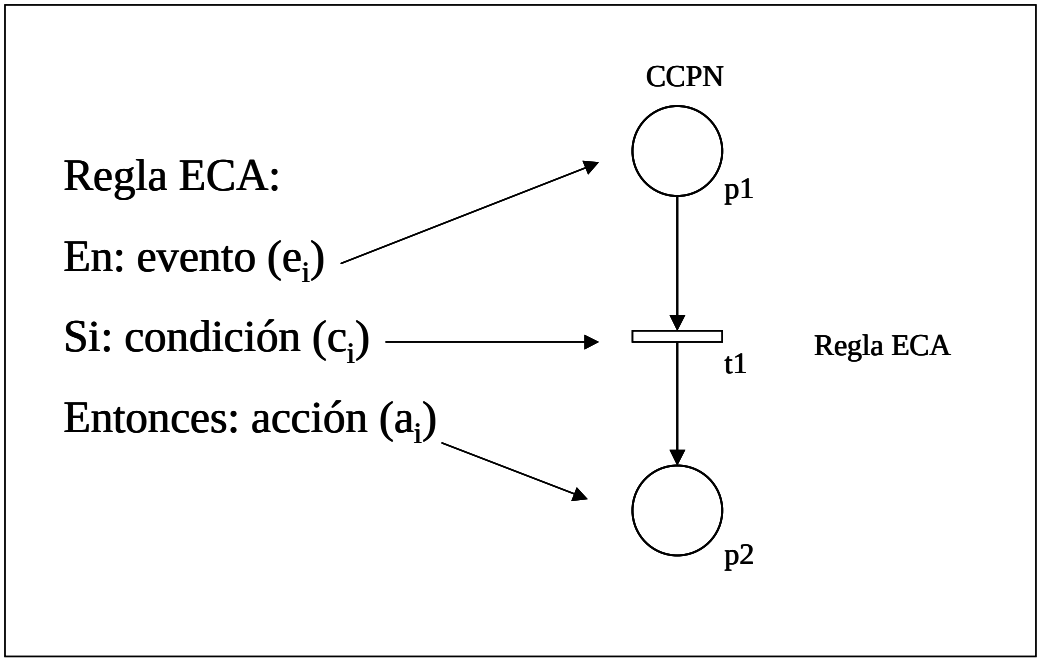


Figura 3.8: Representación con CCPN de una regla ECA

En el siguiente capítulo se presentan la definición y los elementos originales de la CCPN, además de ciertos elementos agregados con el fin de extender su definición para modelar los conceptos de workflow. En [20] se presenta la definición de las CCPN's utilizadas en el contexto de bases de datos activas.

Si bien, el uso de reglas ECA pudiese parecer propio de las bases de datos activas, existen trabajos en los que se proponen enfoques basados en reglas ECA para tratamiento de excepciones en workflows [24], [29] o para ejecución automática de procesos de workflow [30], [31]. En el siguiente capítulo presentamos una propuesta de modelado de procesos de workflow basada en el enfoque de reglas ECA utilizando una extensión de las redes de Petri coloreadas condicionales.

## 3.4. Comentarios finales

En este capítulo se presentó la teoría relacionada a redes de Petri, se presentaron las condiciones básicas para habilitación y disparo de transiciones en una PN y en una CPN con lo que es posible generar el juego de tokens que visualmente proporciona la perspectiva de ejecución del sistema modelado. También permite obtener una perspectiva de los estados que

alcanza el sistema, además de que esta parte visual de las PN permite obtener modelos más reales de los sistemas representados. Adicionalmente, se tiene la ventaja de tener una semántica formal con lo que es posible validar y verificar los modelos obtenidos con la utilización de herramientas matemáticas. En la última sección se presentó una breve introducción de lo que son las CCPN como modificación de las redes de Petri coloreadas. En la siguiente sección se estudian con más detalle y se presenta su utilización para el modelado de procesos de workflow.

# Capítulo 4

# WfCCPN para modelado de procesos de workflow

En este capítulo se presenta la extensión de CCPN's la cual se propone como metodología para modelar procesos de workflow; se presenta la definición y los elementos de la CCPN extendida, así como también el enfoque sugerido para el modelado. Además se muestran las posibles formas de representar los conceptos de workflow presentados en el capítulo 2 y finalmente se presenta la implementación de veinte patrones de flujo de control identificados en [15]. Éstos son utilizados en la actualidad como marco de evaluación para lenguajes o metodologías utilizadas para modelar procesos de workflow.

## 4.1. ¿Por qué no utilizar CPN para modelar workflow?

Antes de presentar la definición y los elementos que conforman la extensión de CCPN propuesta, es conveniente responder a la siguiente pregunta: ¿Por quú no utilizar simplemente las redes de Petri Coloreadas para modelar procesos de workflow?

Las CPN's originales no son completamente convenientes para el modelado de procesos de workflow, como se menciona en [8], [9] debido a que las CPN han sido sometidas al marco de evaluación conocido como patrones de workflow, los resultados de lo anterior demuestran que con CPN no pueden ser implementados cinco patrones. En [9] se identifican los problemas que surgen al tratar de modelar esos patrones; en este capítulo se proporciona una posible solución utilizando la extensión de CCPN que al igual que las CPN son redes de Petri de alto nivel. Es decir consideran datos, tiempo y características de jerarquía. Además de lo anterior,

el enfoque sugerido en esta tesis se basa en representar el workflow desde un enfoque de reglas ECA, por lo cual es necesario representar las estructuras de reglas ECA relacionadas entre sí, al considerar esto con una CPN sus elementos resultan insuficientes [20], sobre todo al considerar reglas ECA con eventos compuestos.

Las diferencias más importantes entre una CPN y una CCPN son las siguientes: En primer lugar, en una red de Petri coloreada los eventos no se consideran [20], mientras que en una CCPN un evento es parte de una regla ECA; estos eventos son representados con lugares de entrada de una transición. Las condiciones en una CPN por lo general son representadas en los arcos, además de que las acciones son representadas a través de transiciones [28]. Es a través de las transiciones como en una CPN se modelan los estados que posee el sistema, mientras que en una CCPN los estados son modelados como en una PN básica, es decir utilizando lugares.

Otra diferencia es que en los arcos de la CCPN no se contemplan las expresiones que se utilizan en las CPN. Con esto se logra aumentar la flexibilidad en el modelo, sin embargo se pierde un poco el control de los datos que pueden circular en la red, dejando esta tarea a las condiciones utilizadas en las transiciones de una CCPN. Esto necesario al no tener restricción en los datos que pueden ser definidos al momento de modelar un proceso de workflow, en el que sus datos pueden ser de cualquier tipo y pueden pertenecer a cualquier tipo de contexto al ser procesos genéricos de negocios.

Por las razones anteriores, puede concluirse que las CPN no proporcionan todos los elementos que un lenguaje de modelado de procesos de workflow. Sin embargo, las CPN's son una herramienta que ha sido utilizada por varios años en el modelado de sistemas dentro de las áreas académicas así como en la industria, y perfectamente pueden modelar cualquier tipo de sistema. Sin embargo, en este trabajo nos enfocamos en detalles específicos del modelado de procesos de workflow que han sido expuestos en [8], [9], [20], trabajos que permiten sugerir una propuesta basada en una modificación de las CPN's, es decir las Redes de Petri Coloreadas Condicionales. A continuación se presenta la extensión realizada a las CCPN's que son utilizadas como propuesta de modelado en esta tesis.

## 4.2. Extensión a las redes de Petri coloreadas condicionales

En el capítulo 2 se presentaron las diferentes perspectivas en que se modela un proceso de workflow; éstas son las siguientes: la perspectiva del flujo de control, la perspectiva de datos, la perspectiva de recursos y la perspectiva operacional. El trabajo realizado en esta tesis se encuentra delimitado en la perspectiva de flujo de trabajo y de manera parcial en la perspectiva de datos. La perspectiva de datos se modela con la información que es posible transportar en los tokens de la CCPN; es así como el alcance y visibilidad de los datos manejados en los modelos obtenidos es únicamente local. Cabe recordar que este trabajo no contempla la representación y ejecución del workflow modelado. Por lo tanto, no es necesario contemplar por el momento datos externos al modelo que pudiesen incidir en el flujo de la información de éste. Por esta razón no se provee un soporte total para la perspectiva de datos, esto se explica con más detalle en las siguientes sub secciones. Con el fin de diferenciar el uso de las CCPN para modelar bases de datos activas y para modelar workflow, además de los elementos añadidos a la CCPN original, se consideró conveniente referirse de manera diferente a las CCPN en este trabajo, por lo tanto de aquí en adelante se agrega el prefijo "Wf", de tal modo que la extensión a las CCPN será referida como: WfCCPN (Workflow Conditional Colored Petri Nets).

Utilizando las WfCCPN, es posible modelar la perspectiva de flujo de control sin necesidad de abstraer los datos de control. Estos datos son los que deciden el flujo de la información en un proceso de workflow. En [13] se realiza la modelación abstrayendo datos de control, datos temporales y eventos externos. Actualmente son pocas las metodologías o lenguajes que permiten especificar más de una perspectiva dentro de un solo modelo. Uno de estos lenguajes es YAWL [9], el cual está basado en redes de Petri; con WfCCPN es posible modelar las dos sin perder la relación entre el flujo de control y los datos que viajan a través del flujo del proceso. Esto se logra principalmente por la interpretación de los elementos visuales de la WfCCPN, los cuales ayudan a mantener la perspectiva a pesar de que el modelo crezca considerablemente al ser más detallado el modelo del sistema, en este caso del proceso de workflow.

En la siguiente sección se presenta la definición y los elementos que componen una WfCCPN.

## 4.3. WfCCPN

### 4.3.1. Elementos

Los elementos que conforman una WfCCPN se muestran en la figura 4.1.

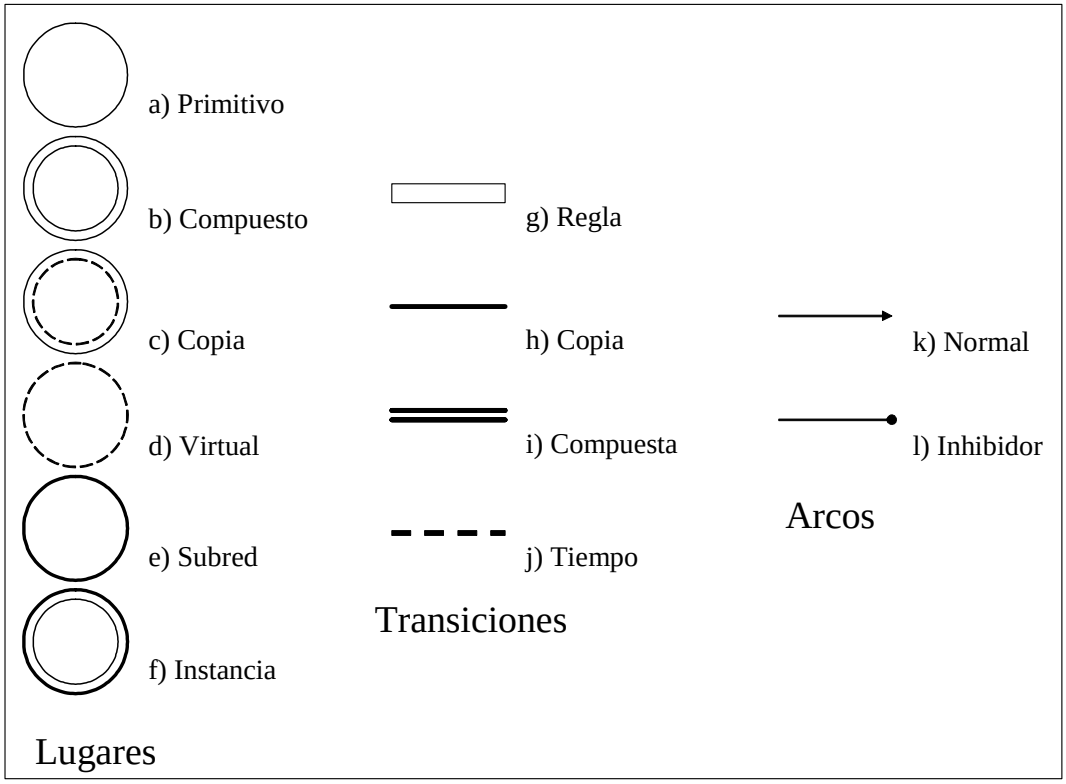


Figura 4.1: Elementos de una WfCCPN

**Lugares.**

Los lugares de una WfCCPN son clasificados en los siguientes tipos: *primitivo, virtual, copia, compuesto, subred* e *instancia*. En la figura 4.1 se muestra como se representan cada uno de éstos. Un lugar *primitivo* es similar a un lugar de una red de Petri básica aunque éste posee un dominio de color. Un lugar c*ompuesto* es utilizado para almacenar tokens con colores compuestos (i.e. crear un color a partir de atributos de diferentes colores) y para reflejar el resultado de una acción, éstos también se utilizan para verificar plazos temporales *(deadline, en inglés)*. Los lugares *copia* se utilizan cuando una tarea es activadora de más de una regla ECA. De esta manera, es posible la activación de varios eventos a partir de

una tarea o de una acción, lo anterior es similar a un evento de multidifusión *(broadcast, en inglés)*. Un lugar *virtual* tiene la capacidad de almacenar cualquier tipo de color. Un lugar *subred* se utiliza para crear el modelo de forma jerárquica permitiendo así tener un modelo con varios niveles de profundidad, se utiliza el enfoque de refinamiento de lugares en contraste con el refinamiento de transiciones que se utiliza en redes de Petri coloreadas [27]. Este tipo de lugares sólo posee una transición de entrada y únicamente una transición de salida. Finalmente un lugar *instancia* permite modelar un número $n$ de instancias de una sub red, permitiendo modelar sub-procesos. En WfCCPN una etiqueta es añadida a cada lugar, la cual permite conocer qué interpretación tiene con relación a la representación de una regla ECA.

**Transiciones.**

Las transiciones son: *regla, copia, virtual y tiempo*. La figura 4.1 muestra su representación gráfica. Una transición regla permite definir condiciones entre lugares que representan eventos, y lugares que representan acciones. Además en la transición *regla* se almacena la acción que se realizará si la regla se dispara, de la cual su resultado será representado por su lugar de salida, es decir, el lugar que representa la acción de la regla ECA. Las transiciones *compuestas* son utilizadas para generar eventos compuestos a partir de eventos primitivos, también permiten definir nuevos colores de tokens, y sirve también para verificar plazos temporales; Las transiciones *copia* se utilizan para conectar lugares *copia*. Finalmente, una transición *tiempo* permite definir intervalos temporales válidos para su activación o en su lugar permite definir un retardo de disparo.

**Arcos.**

Los arcos en una WfCCPN son dos: el arco básico de una PN y el arco inhibidor. Un arco inhibidor permite tener el siguiente comportamiento en la red: La transición que es conectada con un lugar de entrada a través de un arco inhibidor únicamente se activará cuando en el lugar no existan tokens, es decir si existe al menos un token la transición nunca será activada a pesar de que cumpla la condición básica de activación de la PN. Cabe mencionar que la función guardia de los arcos en una red de Petri coloreada que permiten restringir el paso de determinados colores a través de estos; en WfCCPN no están presentes.

**Tokens**

En este trabajo un token además de poseer un color (datos), contiene además dos estampas de tiempo, una para almacenar el momento en el tiempo en que es creado y la otra almacena el tiempo en que es modificado su color, ya sea su estructura o en la información de los atributos del token. La estampa de tiempo es de la forma: $año : mes : día - hora : minutos : segundos$

### 4.3.2. Reglas de disparo

En la WfCCPN los disparos se llevan a cabo de la siguiente manera: una transición *copia* se dispara si la condición básica de PN se cumple. Sin embargo, las demás transiciones requieren que una condición adicional sea evaluada a verdadero. Una transición *compuesta* requiere que su lógica y su condición temporal sea satisfecha en caso de que hayan sido definidas. Una transición *tiempo* requiere que el intervalo de tiempo sea satisfecho y finalmente una transición de tipo *regla* necesita que su condición lógica sea evaluada verdadero.

### 4.3.3. Definición formal

La definición formal se basa en el concepto matemático de multiconjunto, otros conceptos como son el tipo de elemento, tipo de variable Type(v), el tipo booleano ß, que son propios de las redes de Petri coloreadas y están presentes en las WfCCPN. Se presenta a continuación la estructura formal de la WfCCPN.

**Definición 4.1** *Para la definición se utiliza la siguiente notación:*

$B$=*una variable booleana con los valores* $\{verdadero, falso\}$

**Definición 4.2** *Una WfCCPN es una 11-tupla*

$$WfCCPN = \{\sum, P, T, A, N, C, Con, Accion, D, \tau, I\}$$

*donde*

*(1)* $\sum$ *es un conjunto no vacío de tipos, llamados conjunto de colores.*

*(2)* $P$ *es un conjunto finito de lugares. Para una mejor representación gráfica,* $P$ *es dividido en varios subconjuntos,*

$$P = P_{primitivo} \cup P_{compuesto} \cup P_{virtual} \cup P_{copia} \cup P_{subred} \cup P_{instancia}$$

$$y$$

$$P_{pre-etiquetados} = \{(P_{pre-etiquetados\_primitivos} \subset P_{pritmitivos}) \cup (P_{pre-etiquetados\_virtuales} \subset P_{virtuales})$$

*donde* $P_{primitivo}, P_{compuesto}, P_{virtual}$ *y* $P_{copia}$ *son los conjuntos de lugares primitivos, compuestos, virtuales y copia,* $P_{subred}$ *y* $P_{instancia}$ *son el conjunto de lugares subred e instancia que al ser refinables pueden contener una definición WfCCPN completa dentro de sí, además se tienen los subconjuntos de lugares pre-etiquetados* $P_{pre-etiquetados}$ *que son lugares con una función especial predefinida, pero con las mismas características de los lugares virtuales,* $P_{pre-etiquetados\_virtuales}$*, el mismo caso que los primitivos pre-etiquetados* $P_{pre-etiquetados\_primitivos}$*.*

*(3)* $T$ *es un conjunto finito de transiciones.* $T$ *es dividido en cuatro subconjuntos,*

$$T = T_{regla} \cup T_{copia} \cup T_{compuesta} \cup T_{tiempo}$$

*donde* $T_{regla}, T_{copia}, T_{compuesto}$ *y* $T_{tiempo}$ *son el conjunto de transiciones regla, copia, compuesta y tiempo.*

*(4)* $A$ *es un conjunto de arcos tales que*

$$P \cap T = P \cap A = T \cap A = \Phi$$

$A = A_{normal} \cup A_{inhibidor}$*, donde* $A_{normal}$ *y* $A_{inhibidor}$ *representan el conjunto de arcos normales e inhibidores respectivamente.*

*(5)* $N$ *es una función nodo. Ésta es definida de* $A$ *hacia* $P \times T \cup T \times P$*.*

*(6)* $C$ *es una función de color. Ésta es definida de* $P$ *hacia* $\sum$*.*

*(7)* $Con$ *es una función condición. Este es cualquiera de las transiciones* $T_{regla}$ *o* $T_{compuesta}$ *hacia expresiones tales que*

$$\forall t \in T_{rule} : [Type(Con(t)) = B]$$

*donde* $Con$ *es la función que evalúa la condición regla*

$$\forall t \in T_{comp} : [Type(Con(t)) = B]$$

*donde* $Con$ *es la función que evalúa la condición temporal*

*(8)* $Accion$ *es una función. Esta es definida de* $T_{rule}$ *y* $T_{compuesta}$ *hacia expresiones tales que:*

$$\forall t \in T_{regla} \cup T_{compuesta},\ p \in t : [Type(Accion(t)) = C(p)_{MS}]$$

*(9)* $D$ *es una función de tiempo.*

$$\forall t \in T_{tiempo} : [Type(Intervalo(t)) = B]$$

*Ésta es definida de* $T_{timepo}$ *hacia un intervalo de tiempo* $[d_1, d_2]$*; donde* $t \in T_{tiempo}$*, y donde se tiene un* $d_1, d_2$ *que son el intervalo inicial y el final respectivamente.*

$$\forall t \in T_{tiempo} : [Type(Retardo(t)) = B]$$

Esta es definida de $T_{tiempo}$ hacia un retardo de tiempo $[d]$; donde $t \in T_{tiempo}$, y donde $d$ es un retardo especificado en unidades de tiempo diferentes a 0.

(10) $\tau$ es una función de estampa de tiempo.

$\tau = \{\tau_{creacion} \cup \tau_{modificación}\}$

Éstas son definidas de M(p) hacia $\{0\} \cup \mathbb{R}^+$, las cuales se asignan a cada token en el lugar $p$. Una estampa de tiempo, ya sea de creación del token o de modificación de éste, se corresponde con el reloj natural de la forma: $año : mes : día - hora : minutos : segundos$

(11) $I$ es una función de inicialización. Ésta es definida de $P$ hacia expresiones cerradas tales que

$$\forall p \in P : [Type(I(p)) = C(p)_{MS}]$$

### 4.3.4. Reglas ECA modeladas con WfCCPN

En la definición de una conjunto de reglas ECA relacionadas a través de una WfCCPN, se toma a cada una de las reglas para convertir sus elementos en elementos de la WfCCPN. Dado un conjunto de reglas ECA $R = \{r_1, r_2, ..., r_n\}$, para cada regla $r_i(ei, ci, ai), i = 1, 2, ..., n$, el evento $e_i$ es modelado con una estructura de WfCCPN. En el caso de eventos primitivos se utilizan lugares $p \in P_{prim}$, y para el caso de eventos compuestos se generan estructuras correspondientes a cada evento compuesto. La condición de la regla $c_i$ se almacena dentro de una transición $t \in T_{regla} \cup T_{compuesta}$ la cual será evaluada durante el proceso de disparo de la transición $t$. La acción de la regla ECA es una operación que refleja la modificación de los datos que pasan a través de la transición (en la transición $t$ se adjunta la definición de la acción) y ésta es representada por un lugar $p \in P_{primitivo} \cup P_{compuesto}$ de la WfCCPN; esta operación puede ser cualquiera que se encuentra dentro de la fuente de eventos. En la figura 6.1, se ilustra la representación en WfCCPN del evento, la condición y la acción de una regla ECA. El lugar $p_1$ que representa al evento $e_i$ de la regla puede ser de cualquier tipo ($p_1 \in P_{prim} \cup P_{comp} \cup P_{virtual} \cup P_{copy}$). La condición $c_i$ se almacena dentro de una transición $t_1 \in T_{regla} \cup T_{compuesta}$. En el caso de tener un intervalo válido para la ejecución de la regla, éste es definidio utilizando $t_1 \in T_{tiempo}$.Para verificar fechas límites es posible definirlas utilizando una transición $t_1 \in T_{compuesta}$. Finalmente, la acción de la regla ECA $a_i$ se traduce en un lugar de salida $p_2 \in P_{prim}$ de $t$, en $t$ se define la función $T_{Accion}$. Podemos decir que la regla ECA está siendo representada por $t$, porque en $t$ se conocen los elementos de la regla: el evento es su lugar de entrada, la condición está almacenada en ella así como la acción a realizarse y el resultado de la acción es representada en su lugar de salida.

## 4.4. Representación de conceptos básicos de un workflow

En la sección 2.2 se presentó la definición de cada uno de los elementos (proceso, caso, tarea, usuario, etc.) que forman parte de un proceso de workflow. A continuación se muestra la manera de interpretarlos a través de WfCCPN. En su mayor parte las descripciones se apoyan de un ejemplo de workflow, el cual se refiere al proceso genérico que se lleva en la selección de artículos para publicación en congresos científicos.

### 4.4.1. Procesos y casos

El concepto principal en workflow es el proceso; un proceso es representado con una estructura WfCCPN. Los casos son instancias de un proceso, por lo tanto con la ejecución del proceso se tendrá un caso del proceso de workflow. El proceso engloba los conceptos que en las subsecuentes secciones se presentan. Ejemplo de un proceso es el siguiente: El proceso que sigue un artículo enviado por un autor para su publicación en un congreso científico. Un caso de ese proceso sería el envío en particular por cierto autor. De esta manera, por cada autor que envíe un artículo se tendrá un caso del proceso; en este trabajo el caso es referido como instancia del proceso.

### 4.4.2. Tareas

Un proceso puede ser dividido en tareas para su análisis, estas son atómicas porque no pueden ser divididas. Además estas tareas modifican los datos en el workflow. Con la ejecución de una tarea es posible que se provoque la activación y ejecución de otras. En redes de Petri, hay dos opciones para modelar una tarea: como una transición o como un lugar [7]. La mayoría de los enfoques basados en redes de Petri utilizan la primera opción. Algunos de éstos son: [13], [14], [9], [12], [11], [10]. Con WfCCPN una tarea se representa utilizando cualquier tipo de lugar excepto los lugares de tipo *sub_red* e *instancia*, se realiza de esta manera debido al enfoque de representación de una regla ECA mostrado anteriormente. De esta manera, el modelo obtiene temporalidad en las tareas. Es decir en su realización puesto que con la presencia de un token en un lugar su disparo no debe ocurrir inmediatamente, sino hasta que la condición lógica de la transición se cumpla, situación que no es posible tomando en cuenta únicamente la condición básica de una red de Petri. Además, mediante

transiciones de tipo tiempo, es posible retardar la realización o finalización de la tarea según la interpretación dada.

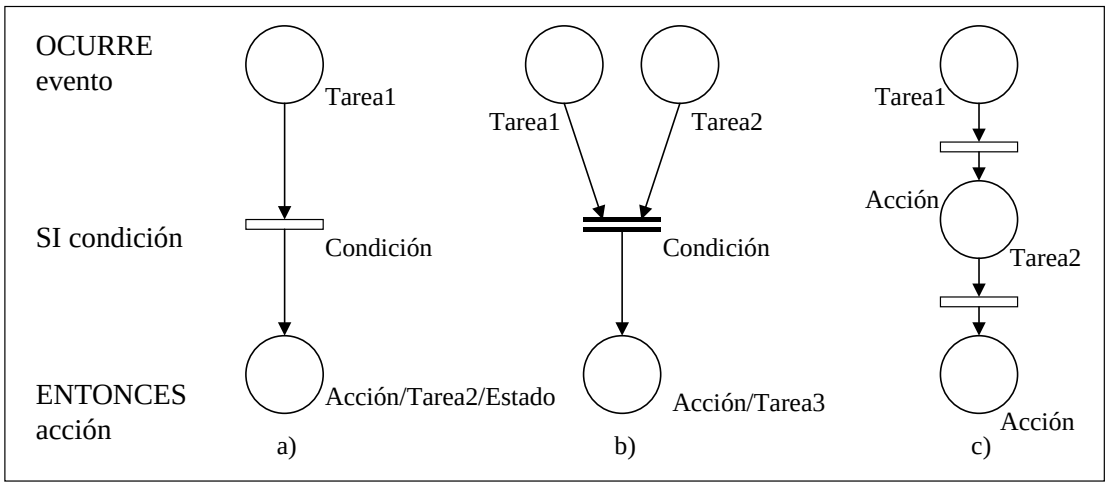


Figura 4.2: Representación de tareas

La figura 4.2 presenta las posibles representaciones para una tarea utilizando diferentes elementos de la WfCCPN. La figura. 4.2 a) muestra la situación más simple, la tarea es activada con la presencia de tokens en el lugar de entrada. Si la condición es satisfecha la tarea es realizada y la acción definida en la transición es reflejada en el lugar de salida, en ese punto la información del token pudo haber sido modificado por la acción. El lugar de salida puede representar una nueva tarea o un nuevo estado en el workflow. En la figura 4.2 b) se muestra la situación en que la finalización de dos tareas es necesaria para que se lleve a cabo la siguiente tarea o para que se realice una determinada acción. De igual manera que la situación anterior, en ésta, la información del token puede sufrir cambios. En reglas ECA existen relaciones entre reglas, eso puede verse en situaciones en que dos reglas compartan el mismo evento. También dos reglas están relacionas si una acción es la misma para dos eventos. Esta situación se muestra en la figura 4.2 c).

Ciertas tareas de un workflow pueden ser consideradas como imprescindibles, estas tareas son las que inicializan y finalizan el proceso. Para estas tareas se predefinieron lugares a los que se denomina lugares pre-etiquetados, los cuales se muestran en la figura 4.3.

En la figura 4.3 a) se muestran los lugares virtuales pre-etiquetados: *inicio, sub_inicio* y *diferido,* para iniciar el proceso principal y subprocesos del workflow respectivamente. Además con el lugar diferido, es posible colocar tokens en tiempo de ejecución con el fin

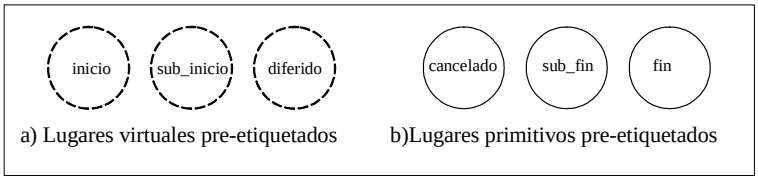


Figura 4.3: Lugares pre-etiquetados

de modelar eventos externos o acciones realizadas por usuarios del workflow. También son presentados tres lugares primitivos pre-etiquetados: *fin, sub_fin* y *cancelado* para acciones de finalización y cancelación del proceso de workflow. Éstos son mostrados en la figura 4.3 b). Los lugares *fin* y *sub_fin* se utilizan para finalizar el proceso principal y subprocesos respectivamente, mientras que *cancelado* que se utiliza en combinación con los dos anteriores para la cancelación del proceso completo o cancelación de subprocesos.

### 4.4.3. Constructores básicos de enrutamiento

La WfMC [21] define los constructores básicos para construir las posibles rutas que toma la información dentro de un proceso de workflow. Con estos constructores es posible representar la perspectiva de flujo de control descrita en el capítulo 2. Para este trabajo se definieron los constructores por medio de estructuras WfCCPN. Éstos permiten incluir información de control ya que pueden utilizarse tokens coloreados. Los constructores se muestran en la figura 4.4.

**Secuencia.** Este constructor modela la situación más natural y simple dentro de un proceso workflow, modela una tarea que se ejecuta sólo hasta que la tarea que le precede ha sido ejecutada. En la figura 4.4 a) se muestra la manera en que este constructor es implementado. La tarea o evento que se representa con el lugar B será ejecutada una vez que la tarea A haya sido realizada. En el caso de la PN, cuando la transición $t$ sea activada y cumpla con la condición lógica ésta será disparada colocando un token en el lugar B que representa la realización de su tarea asociada.

**AND-Split.** Con este constructor se modelan principalmente situaciones de paralelismo. La figura 4.4b) muestra la estructura en WfCCPN que lo implementa; cuando un token llega a la transición $t \in T_{copia}$ éste es replicado hacia los lugares de salida, los cuales pueden ser $p \in P_{copia}, P_{virtuales}$. De esta manera, se tendrá una copia del token original en cada lugar de

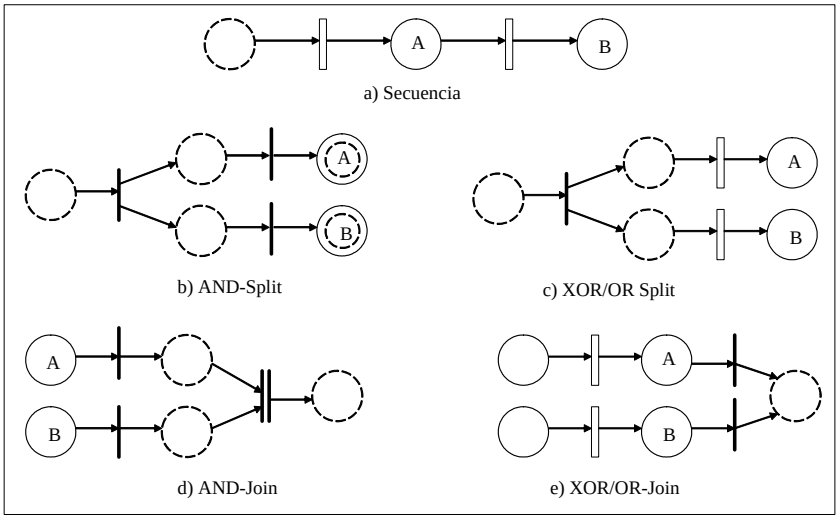


Figura 4.4: Implementación de los constructores de enrutamiento a través de WfCCPN

salida con su respectivo flujo de control. En la figura se muestra de que manera las tareas A y B se ejecutarán de manera paralela.

**AND-Join.** Cuando dos o más tareas se están ejecutando de manera paralela, llega un punto en el workflow en que convergen en un hilo de ejecución, es decir con el resultado o condición de haberse ejecutado previamente. Posteriormente es posible ejecutar otra tarea o bien, activar un evento o ejecutar una acción [15]. La figura 4.4 d) muestra la implementación de este constructor; la transición $t \in T_{compuesta}$ permite realizar una sincronización de los tokens que le llegan. Esta sincronización puede realizarse simplemente con permitir el paso a un determinado token, o la generación de tokens con nuevos colores los cuales pueden formarse a partir de los datos de los tokens que hubiesen llegado a la transición $t$.

**XOR/OR-Split**. En [15] el XOR se distingue del OR normal, siendo el XOR la situación en la que una y únicamente una ruta puede ser tomada en el flujo del workflow, por el contrario un OR es la situación en que más de una ruta o incluso ninguna puede ser elegida para seguirse en el proceso. Esta implementación se realiza de igual manera con WfCCPN, ya que con la condición en cada $t \in T_{regla}$ decide cuántas rutas son tomadas, por lo tanto es responsabilidad del modelador establecer si es un XOR o un OR el constructor.

**OR-Join**. La decisión que puede tomar una o varias rutas de entre un número elegible se realiza a través de las condiciones establecidas en cada una de las transiciones $t \in T_{regla}$. En la figura 4.4 e) se presenta este constructor.

### 4.4.4. Datos

En WfCCPN la representación de datos se realiza de igual manera que en CPN's es decir a través de colores. Para cada workflow es posible definir un conjunto $\sum$ de colores, el cual representa los datos del proceso modelado. Un color posee un nombre por el cual es identificado, y un conjunto de atributos que pueden ser de diferentes tipos alfanuméricos, numéricos, e incluso otros colores. Cada atributo contiene su respectivo valor y éstos pueden ser modificados a través del conjunto de *Acciones* de la WfCCPN. Además, a partir de estos atributos es posible definir las condiciones para el conjunto de transiciones $t \in T_{regla} \cup T_{compuesta}$.

### 4.4.5. Personas, Roles y Grupos

Las personas, roles y grupos son representados de manera parcial. Estos recursos son representados de manera completa en la perspectiva organizacional. Sin embargo, con WfCCPN es posible representar un usuario con un lugar virtual debido a la capacidad de almacenar tokens de cualquier color sin restricción alguna. Los roles y los grupos pueden ser representados a través de los atributos de los colores que pueden representar a una persona. Utilizando un lugar virtual es posible representar, por ejemplo, a los revisores del ejemplo.

### 4.4.6. Estados

Con WfCCPN un estado es representado a través de cualquiera de los siguientes tipos de lugares $p \in P_{primitivo} \cup P_{compuesto} \cup P_{virtual} \cup P_{copia}$. Utilizando el ejemplo del envío de artículos presentamos algunos posibles estados del workflow; recordando que un estado identifica la situación del sistema o de un documento en un momento dado, además de proveer información sobre el proceso desde alguna de las siguientes dos perspectivas:

- Los datos que contiene en ese momento algún lugar, por ejemplo un lugar que represente el estado del artículo enviado, podría ser representado con tres lugares uno para el estado aceptado, otro para el estado aceptado condicionalmente y otro más para los artículos con estado de rechazados.

- La etapa en que se encuentra el proceso. Por ejemplo si el proceso se encuentra en la etapa de recepción de artículos, o si ya concluyó esa fase, es posible que se encuentre

en la etapa de revisión de artículos, o en la etapa de recibir las versiones finales de los artículos aceptados.

### 4.4.7. Condiciones

Las condiciones en una WfCCPN son representadas a través de transiciones de tipo regla y de tipo compuestas. Éstas permiten modificar el flujo natural de la información y llevar a ciertos estados que pueden ser deseados dentro del proceso. Además, las condiciones en el enfoque de modelado con reglas ECA es fundamental.

### 4.4.8. Plazos y Características temporales

En WfCCPN las transiciones *tiempo* y las *compuestas* proporcionan características de tiempo, y utilizándose con las estampas de tiempo de los tokens, pueden verificar condiciones de fechas límite (deadlines). Además con transiciones tiempo es posible definir intervalos válidos para la activación de la transición. Así, se pueden modelar tareas que necesitan de algún tiempo antes de estar disponibles para su ejecución. Recurriendo al ejemplo del envío de artículos a un congreso, la primera condición temporal que puede ser modelada es la fecha límite para el envío de artículos. El token del color que representa el artículo es evaluado a través de su estampa de tiempo de creación para decidir si está dentro del tiempo especificado.

### 4.4.9. Modelos jerárquicos

En la vida real muchos procesos que se modelan son grandes y modelarlos con redes de Petri puede dar como resultado redes bastante grandes como para ser bien manejadas y apreciadas. Es por eso necesario descomponer el modelo en varios sub modelos. Esta idea en redes de Petri es adoptada de la programación estructurada y es posible representar con un lugar o con una transición otra red de Petri, a esto se le conoce como refinamiento de lugar o de transición. En redes de Petri coloreadas se utiliza el refinamiento de transiciones [27].

Con WfCCPN, se utiliza el refinamiento de lugares debido al enfoque de utilizar lugares para representar tareas. Por tal razón, se utilizan los lugares $_p \in P_{subred}$, éstos fueron presentados en la figura 4.1 e).

En la figura 4.5 se muestra de manera gráfica un lugar subred y el concepto que representa. Dentro del lugar $P_1$ que pertenece a una red WfCCPN se tiene definida otra red WfCCPN

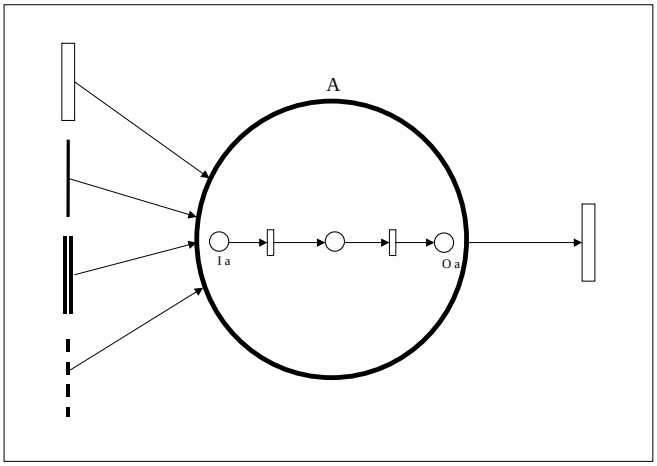


Figura 4.5: Sub-redes WfCCPN dentro de lugares de tipo sub-red

que cuenta con uno de entrada $P_{entrada}$ y únicamente un lugar de salida $P_{salida}$. Este tipo de lugares puede ser de salida de cualquier tipo de transición. Sin embargo, sólo puede ser lugar de entrada para una sola transición de tipo *regla* como se muestra en la figura.

### 4.4.10. Múltiples instancias

Uno de los principales requerimientos en un modelo de workflow así como para una herramienta de modelado es la definición de múltiples instancias de un caso. Es decir que un caso pueda ser ejecutado $n$ veces dentro del mismo proceso principal que es modelado. La capacidad de modelar múltiples instancias en WfCCPN se basó en [32], [9]. Con WfCCPN se utiliza un lugar especial llamado *instancia* éste fue mostrado en la figura 4.5 f). Este tipo de lugares contiene la definición de una red de petri de la misma forma que un lugar de tipo *subred*. Cada token que llega a un lugar instance creará y ejecutará una instancia por separado de la sub red. Es decir, si llegan cuatro tokens, tendremos cuatro instancias ejecutándose en paralelo. Este lugar únicamente puede tener una transición de entrada y una transición de salida.

## 4.5. Patrones de workflow

Los patrones de workflow son utilizados como framework de evaluación, éste ha sido bien adoptado por los vendedores de sistemas de workflow, evaluando sus productos con este

framework. En [15] se presentan estos patrones con la evaluación de algunos productos de software, se encuentra una breve evaluación de la expresividad de modelado de cada herramienta de software probado. Vale la pena mencionar que el término: "expresividad de modelado" no es lo suficientemente rigorista ya que puede ser subjetivo, así como subjetivos son los modelos obtenidos con cualquier herramienta de modelado, debido a una razón sencilla: "es posible que dos personas modelen de manera diferente el mismo proceso". Se utilizó este framework de evaluación debido a la aceptación que posee y los resultados que pueden encontrarse en [33], [9], [8], [32], [16], [32].

En [15], [9], [16] se presentan diversas evaluaciones a los sistemas administradores de workflow (WfMS, Workflow Managament Systems) más importantes al momento de escribir esta tesis. Los resultados muestran que los siguientes WfMS's: COSA, SAP/R3, InConcert, iFlow, MQ Series, Staffware, Visual Workflo, Domino workflow, Meteor y otros más únicamente soportan entre once y catorce patrones en promedio. El mismo resultado se tiene con estándares de modelado tales como: XPDL, BPEL4WS, BPML, WSFL, XLANG, WSCI y UML 2.0.

Los patrones de workflow en flujo de control son veinte, divididos en los siguientes seis grupos:

- Patrones básicos: (1) Secuencia, (2) Split paralelo, (3) Sincronización, (4) Elección exclusiva, (5) Mezcla simple.
- Sincronización y ramificaciones avanzadas: (6) Múltiple elección, (7) Mezcla con sincronización, (8) Mezcla múltiple (9) Discriminador, (9b) N-salen-de-M Joins.
- Patrones estructurales: (10) Ciclos arbitrarios, (11) Terminación implícita.
- Patrones de Múltiples Instancias: (12) MI sin sincronización, (13) MI con conocimiento en tiempo de diseño a priori, (14) MI con conocimiento en tiempo de ejecución a priori, (15) MI sin conocimiento en tiempo de ejecución a priori.
- Patrones basados en estados: (16) Elección diferida, (17) Enrutamiento paralelo de interllegada, (18) Entrega de producto.
- Patrones de cancelación: (19) Actividad cancelada, (20) Caso cancelado.

En [9] se presenta una perspectiva de los problemas que surgen al intentar modelar estos patrones con redes de Petri de alto nivel (CPN). WfCCPN son redes de Petri de alto

nivel; por esta razón se prestó especial atención en verificar dichos problemas y en realizar su implementación a través de estructuras WfCCPN. Los resultados obtenidos permitieron demostrar que ciertos patrones pueden ser modelados a través de las características básicas de la WfCCPN tal es el caso de los patrones 5 y 6. Sin embargo, en otros patrones el problema se resolvió con los elementos que fueron añadidos a la CCPN original. La tabla de la figura 4.6 que fue publicada en [9] muestra una comparación del estándar XPDL de la WfMC para modelar de manera gramatical (no gráfica), también la nueva versión UML 2.0. Un estudio sobre éste se encuentra publicado en [8]. Además el estudio de redes de Petri de alto nivel y del lenguaje YAWL están presentes en esa tabla. La tabla original fue modificada para anexar los resultados obtenidos con WfCCPN.

| Patrón | XPDL | UML 2.0 | High-level Petri nets CPN) | YAWL | WfCCPN |
|---|---|---|---|---|---|
| 1(secuencia) | + | + | + | + | + |
| 2(split-paralelo) | + | + | + | + | + |
| 3(sincronización) | + | + | + | + | + |
| 4(elección-múltiple) | + | + | + | + | + |
| 5(mezcla-simple) | + | + | + | + | + |
| 6(elección múltiple) | + | + | + | + | + |
| 7(sincronización múlt) | + | - | - | + | + |
| 8(mezcla múltiple) | - | + | + | + | + |
| 9(discriminador) | - | + | - | + | + |
| 10(ciclos arbitrarios) | + | + | + | + | + |
| 11(terminación implicita) | + | + | - | - | + |
| 12(mi-sin sincronización) | + | + | + | + | + |
| 13(mi-priori-t-diseño) | + | + | + | + | + |
| 14(mi-priori-t-ejecución) | - | + | - | + | + |
| 15(mi-sin-c-priori) | - | - | - | + | + |
| 16(elección diferida) | - | + | + | + | + |
| 17(enrut-paral-inter) | - | - | + | + | + |
| 18(entregables) | - | - | + | + | + |
| 19(cancela-actividades) | - | + | +/- | + | + |
| 20(cancela-casos) | - | + | - | + | + |

Figura 4.6: Patrones de flujo de control en diferentes metodologías de modelado de workflow

Cabe mencionar que no se presenta una definición en extenso de cada patrón. Sin embargo

en [33], [16] es posible ampliar la información sobre éstos. A continuación se muestra la implementación de cada patrón utilizando elementos de una WfCCPN.

### 4.5.1. Patrones de flujo de control básicos

Estos patrones son implementados de manera natural por todos los sistemas de workflow y lenguajes de modelado. Por esta razón se presenta de manera breve la implementación utilizando WfCCPN. No se muestra de manera gráfica ya que con los constructores básicos de enrutamiento definidos por la WfMC son suficientes para modelar estos patrones básicos. Los constructores básicos fueron mostrados en la figura 4.4.

**Secuencia.**

Este patrón aparece cuando es necesario que una actividad sea completada una para que otra actividad, evento o acción sea iniciada. Con WfCCPN se implementa a través de cualquier tipo de lugares utilizando cualquier tipo de transiciones que se conecten de manera secuencial, en la figura 4.4 a) se muestra la implementación de este patrón.

**Paralelización**

Este patrón permite ejecutar tareas en paralelo, la implementación con WfCCPN es similar al $fork$ que se utiliza en los diagramas de actividad de UML. Utilizando el constructor And-Split se obtiene el comportamiento de este patrón, que se muestra en la figura 4.4 b).

**Sincronización.**

Este patrón sincroniza dos o más hilos de ejecución, que se vienen ejecutando de forma paralela. Desde la perspectiva de flujo de control el constructor AND-Join ya sea implícito o explicito es suficiente. Sin embargo, con WfCCPN además de lograr la sincronización en el flujo del workflow, es posible añadir sincronización de datos al utilizar alguna de las Acciones que puede añadirse a las transiciones compuestas. En la figura 4.4 d) se muestra la implementación de este patrón.

**Elección exclusiva.**

Se elige una ruta de entre muchas que se tienen como alternativas. Con WfCCPN se utiliza una transición de tipo regla con su condición asociada para cada una de las rutas alternativas. Se definen las condiciones con el fin de que un único hilo de ejecución sea activado como resultado de la evaluación de las condiciones; el constructor XOR-Split es suficiente para modelar este patrón, el cual se muestra en la figura 4.4 c).

**Mezcla simple.**

Este patrón mezcla dos rutas alternativas de ejecución, y es implementado con el constructor XOR-Join de la figura 4.4 e). Con WfCCPN se utiliza un lugar virtual para almacenar tokens de cualquier color. Debido a que del OR-Join pueden provenir tokens de diferentes colores, en las diferentes rutas que llegan, nuevamente es necesario definir en la transición regla las condiciones que permitan que la información arribe a un determinado hilo de ejecución.

### 4.5.2. Ramificaciones y sincronización avanzada

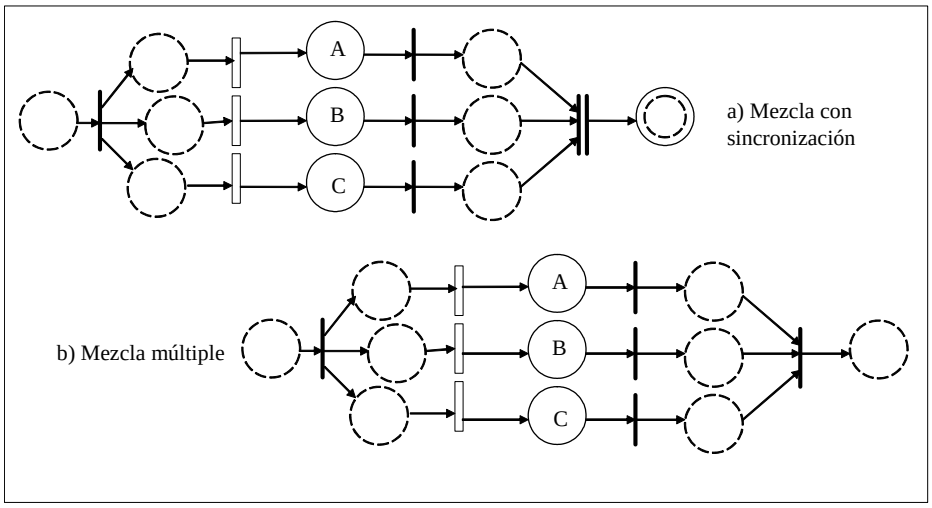


Figura 4.7: Implementación de los patrones: mezcla con sincronización y mezcla múltiple.

**Elección múltiple.**

Con este patrón es posible elegir $n$ rutas de ejecución de entre $m$ alternativas (donde $m > n$) [33]. Este patrón es implementado a través del constructor OR-Split. En la figura 4.4

e), se muestra que cada posible ruta posee su propia transición de tipo *regla* y éstas permiten seleccionar las rutas que deben seguir ejecutándose.

**Mezcla con sincronización.**

Este patrón considera la sincronización de varias rutas de ejecución hacia un único hilo de ejecución [33]. En la figura 4.7 a) se muestra la implementación mediante WfCCPN. En primer lugar se realiza la elección múltiple, en la figura 4.7 a) se representan tres tareas las cuales convergen hacia una transición *compuesta*, para posteriormente llegar a un lugar *compuesto*. De esta manera es posible sincronizar la información de las tres tareas, utilizando una función Acción que puede ser la creación de un color nuevo con información relevante de cada una de las tareas, o con la modificación de los atributos en los colores de los tokens. En [9] se menciona la siguiente razón como el problema al implementar este patrón con CPN: *"La mezcla con sincronización no es soportada porque el diseñador tiene que mantener la referencia del número de hilos en paralelo y decidir sí sincroniza o mezcla el flujo"*. Con WfCCPN esto queda resuelto al contar con los lugares compuestos, ya que de esta manera se implementa la sincronización, mientras que con transiciones *copia* es implementada la mezcla múltiple, es decir se tiene una implementación en particular para cada patrón. Ciertamente, con CPN's sólo es posible tener el comportamiento que con transiciones *copia* se tiene en WfCCPN.

**Mezcla múltiple.**

Este patrón es mencionado en el problema de modelado con el patrón anterior, En este patrón no existe sincronización, es decir puede pasar más de una tarea hacia el hilo de ejecución simple en que convergen, debido a que se emplea una transición de tipo *copia*. Este patrón se muestra en la figura 4.7 b).

**Discriminador.**

Mezcla de varias rutas de ejecución sin sincronización y ejecución de la subsiguiente actividad una sola vez. En [9] la descripción del problema identificado con respecto a este patrón con redes de alto nivel es la siguiente: *"El discriminador no es soportado porque el diseñador necesita mantener el número de hilos de ejecución y el número de hilos de ejecución completados, además tiene que reiniciar el constructor explícitamente a través de remover los tokens correspondientes a la iteración"*. Con WfCCPN es posible modelarlo con

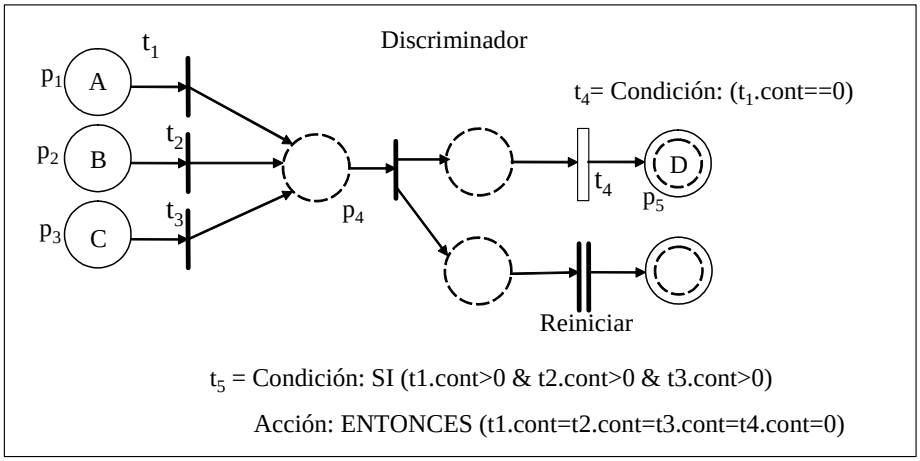


Figura 4.8: Implementación del patrón *discriminador.*

ayuda de transiciones compuestas y los contadores en las transiciones. En la figura 4.8 se muestra de manera gráfica que la implementación es similar a tener el patrón de mezcla con sincronización. Se utiliza el contador de la transición regla el cual permite saber cuantos tokens han llegado. Así, después de que el primer token ha llegado todos los demás son ignorados por la definición formal de una transición compuesta ya que cuando no cumplen la condición son desechados inmediatamente los tokens que llegan a ésta. Además podemos establecer una condición de reinicio del contador, utilizando una transición compuesta auxiliar que en su acción reinicia los contadores de las transiciones de las tareas involucradas, con esto el patrón vuelve a aceptar la llegada de la primer tarea de otro conjunto de tareas.

**N-salen-de-M que convergen.**

Este patrón es una generalización del patrón anterior, pero en vez de una salida, se tienen $n$ salidas. Con WfCCPN es necesario modificar la condición en la transición regla aumentando el valor del contador que es comparado, el cual es el contador de esa transición.

### 4.5.3. Patrones estructurales

Estos patrones son fácilmente implementados con redes de Petri, debido a la noción de estados que dan los lugares.

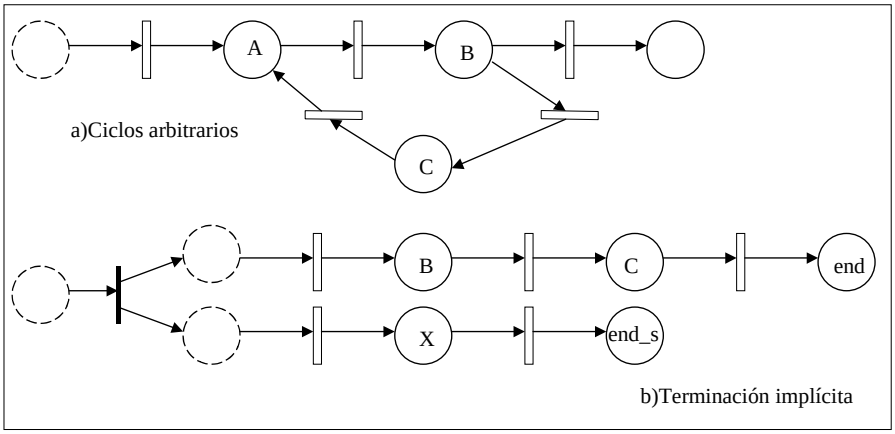


Figura 4.9: Implementación de los patrones de tipo estructural (basados en estados).

**Ciclos arbitrarios**

Ejecución de ciclos sin restricciones estructurales. Este patrón se implementa a través de la estructura XOR. En determinado punto del workflow es posible decidir por medio de una condición lógica si el flujo continúa de manera normal o si es necesario regresar a una de las tareas o eventos previos. En la figura 4.9 a) se muestra cómo a través de las condiciones en las transiciones regla puede ser enviado el token hacia la tarea A para que vuelva a ser ejecutada, o que siga su flujo normal con la ejecución de la tarea C.

**Terminación implícita**

Un subproceso debe ser terminado si no tiene nada más que hacer [33]. Con WfCCPN se adopta una solución similar a la de UML 2.0 presentada en [8], a través de dos tipos de lugares primitivos pre-etiquetados *fin* y *sub_fin* los cuales distinguen la finalización de un subproceso y la finalización del proceso principal permitiendo así tener diferentes puntos de finalización en el proceso de workflow modelado. El caso contrario se presenta en YAWL [9], ya que no conciben más que un solo punto de finalización para un proceso workflow; es por tal motivo que no soportan este patrón.

### 4.5.4. Patrones de múltiples instancias

Con este grupo de patrones se manipulan múltiples ejecuciones de un sub-proceso en el workflow, es fácil para la mayoría de los sistemas de workflow o lenguajes de modelado

iniciar la ejecución de múltiples instancias, sin embargo el problema se presenta al tratar con su ejecución, sincronización y terminación.

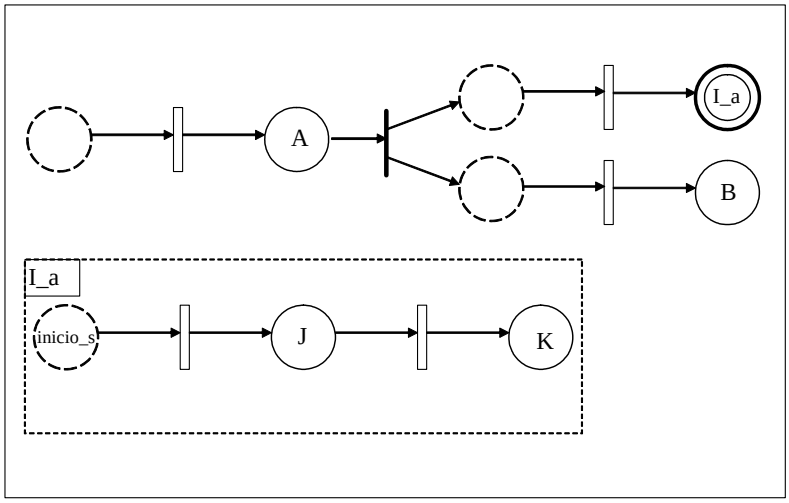


Figura 4.10: Implementación del patrón múltiples instancias sin sincronización.

### Múltiples instancias sin sincronización

Este patrón considera la generación de diferentes instancias de una actividad sin tener sincronización de algún tipo entre ellas [33], este patrón se modela a través de la siguiente interpretación: para tener $n$ instancias en ejecución es necesario colocar $n$ tokens en un lugar de tipo *instancia*.

### Múltiples instancias con conocimiento a priori en tiempo de diseño

Este patrón considera generar varias instancias de un sub-proceso cuando el número de esas instancias es conocido en tiempo de diseño, no es necesaria la sincronización entre instancias [33]. Para este patrón se implementa de una forma bastante simple ya que únicamente es necesario establecer con el peso del arco de entrada del lugar instancia el número $n$ de instancias que se desea ejecutar.

### Múltiples instancias con un conocimiento a priori en tiempo de ejecución

Con este patrón se generan varias instancias de una actividad, sin embargo aquí el número de instancias a ejecutarse no se conoce al momento de diseñar el proceso, ya que esto se

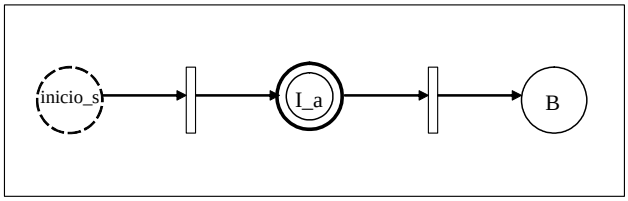


Figura 4.11: Implementación del patrón Múltiples instancias sin conocimiento en tiempo de ejecución.

decide en un punto del proceso en tiempo de ejecución, esto es similar a tener un ciclo For pero en paralelo [33]. Este patrón y el que se presenta en la siguiente sección no pueden ser implementados con CPN, de acuerdo a [9], debido a que es posible ejecutar un número variable de instancias de forma paralela, sin embargo es necesario para este patrón mantener la identidad del sub-proceso ejecutado y el número de instancias que se encuentren en ejecución. En este trabajo se utiliza un lugar especial llamado *instancia* el cual se propone con el fin de implementar este y el siguiente patrón. Este patrón es una generalización del patrón presentado en la siguiente sección [33], por tal razón la implementación se presenta con el patrón: Múltiples instancias sin conocimiento a priori en tiempo de ejecución.

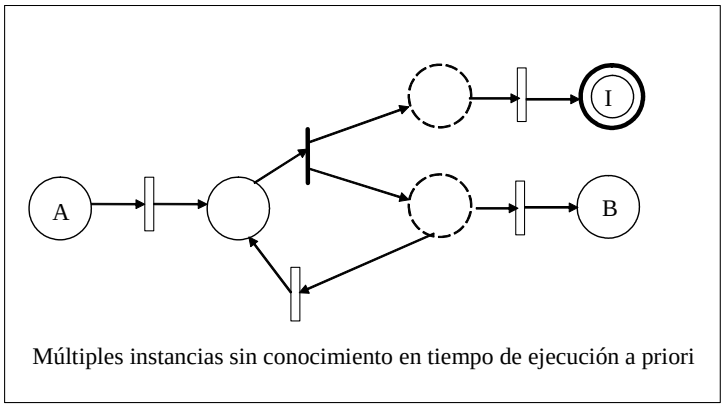


Figura 4.12: Implementación de patrones a través de lugares de tipo instancia

### Múltiples instancias sin conocimiento a priori en tiempo de ejecución

Con este patrón se generan varias instancias de un sub-proceso sin embargo, el número de instancias no puede ser determinado, esto es similar a tener un ciclo While pero en paralelo [33]. Por definición un lugar instancia requiere que todas las instancias sean terminadas, una

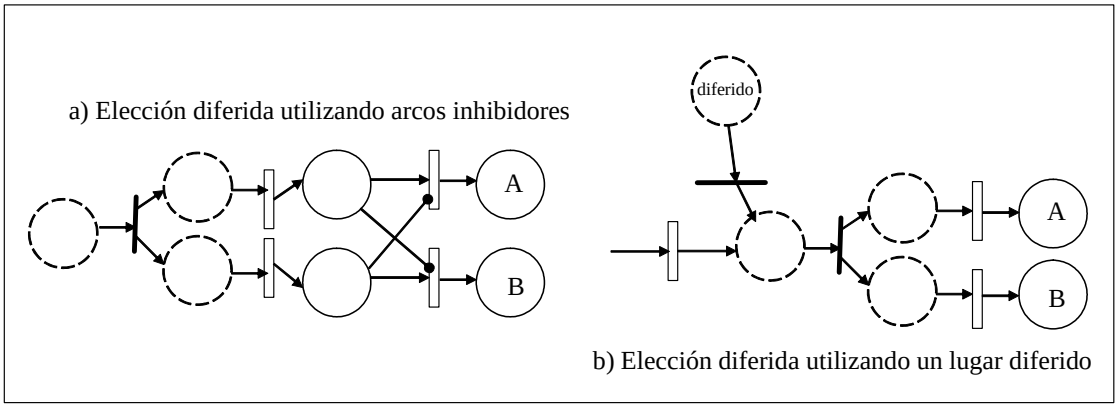


Figura 4.13: Implementación del patrón: elección diferida.

vez que estas son finalizadas, se continúa el flujo normal en el workflow, de esta manera este patrón es implementado a través de utilizar un ciclo de modo que el lugar instancia pueda ser alcanzado tantas veces como sea deseado sin necesidad de definir el número de instancias a ejecutarse en diferentes tiempos, esta implementación se muestra en la figura 4.11.

### 4.5.5. Patrones basados en estados

Estos patrones son bastante sencillos de implementar cuando la metodología de modelado posee la noción de estados, tal es el caso de todo enfoque que utilice redes de Petri. En este trabajo se mantiene la noción de estados a través de los lugares de la WfCCPN. Los arcos inhibidores también se utilizan en la implementación de estos patrones.

#### Elección diferida

Este patrón considera un punto en el workflow en que una de varias rutas de ejecución es elegida, se trata de un XOR split implícito. La implementación con WfCCPN se muestra en la figura 4.4 c) esta permite de manera implícita elegir una tarea para su ejecución sólo ejecutándose esa, debido al uso de arcos inhibidores los cuales mientras exista un token en la tarea, no permitirá que las demás sean activadas Otra implementación se realiza con lugares de tipo *diferido* que además permite almacenar tokens de cualquier color, con estos se representan eventos externos.

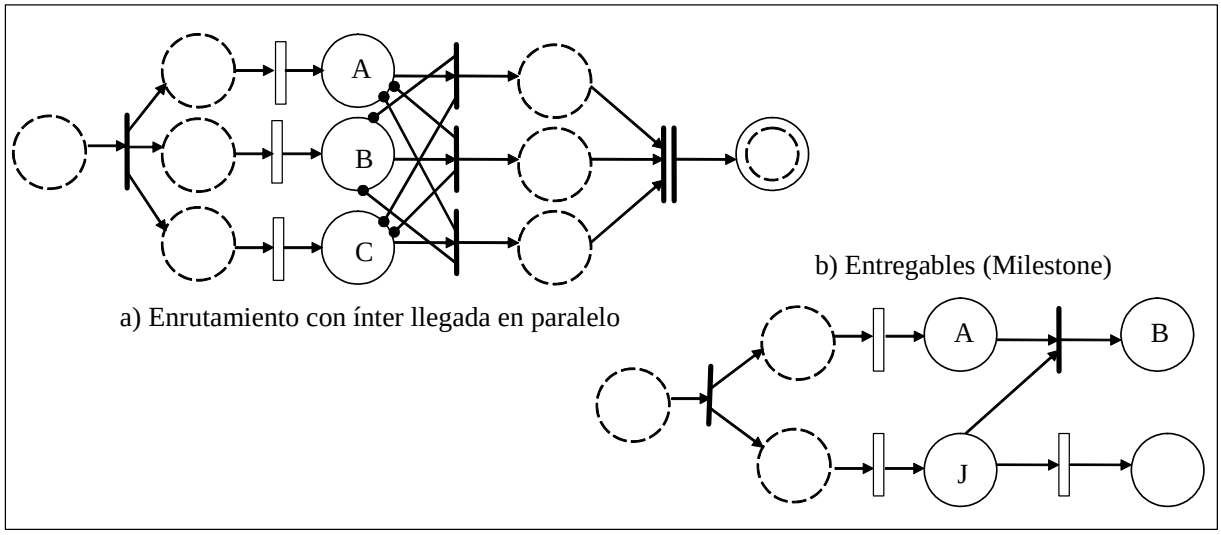


Figura 4.14: Implementación de los patrones basados en estados

**Enrutamiento paralelo con interllegada**

Con este patrón se ejecutan dos o más tareas sin importar el orden de ejecución entre ellas, el orden de ejecución es determinado en tiempo de ejecución con la restricción de que no puede ser ejecutadas más de una tarea en un mismo instante, es decir no es posible ejecutarlas en paralelo [33]. La implementación que se propone es utilizar arcos inhibidores para lograr la restricción de ejecutar una tarea a la vez.

**Entregables**

Este patrón es muy fácil de implementar con redes de Petri, ya que únicamente se utiliza un arco inhibidor como se muestra en la figura 4.14 b), en la figura se muestra como la tarea J

### 4.5.6. Patrones de cancelación

Los patrones de cancelación difícilmente se encuentran soportados de manera gráfica por las metodologías graficas que existen. La mayoría de los sistemas de workflow implementan este patrón a través de scripts o invocación de comandos directamente con el workflow engine para terminar el sub-proceso o proceso principal según sea el caso. De manera explicita con WfCCPN se tienen lugares pre-etiquetados que representan la finalización y cancelación de

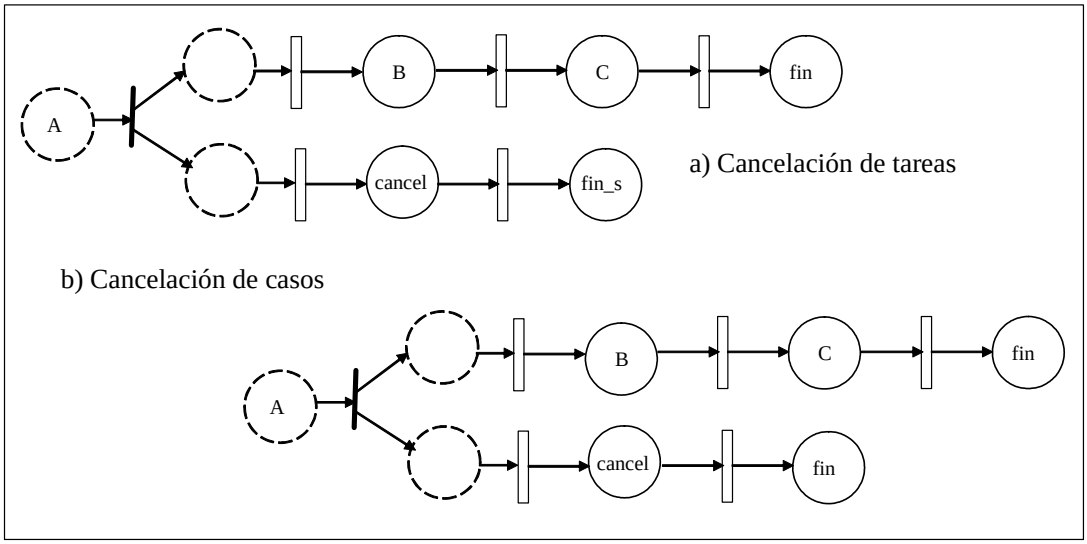


Figura 4.15: Implementación de los patrones de cancelación.

procesos y sub-procesos de workflow. Los lugares *cancelado* pueden ser alcanzados en el flujo del proceso de manera explicita ó de manera implícita. Por la primera forma se entiende que es en base a la información de control que es acarreada en el modelo a través de los tokens, que con las condiciones establecidas por ejemplo una transición *regla* puede decidir cancelar el proceso. Un ejemplo de cancelación implícita es el incumplimiento en algún intervalo de tiempo, verificado a través de la estampa de tiempo del token, esto puede ser motivo para disparar tokens hacia el lugar de cancelación del proceso.

### Cancelación de actividades ó tareas

Este patrón presenta la situación en que es necesario finalizar la ejecución de alguna tarea que esta ejecutándose en el proceso de workflow y que no implique la finalización del proceso de workflow por completo [33]. En la figura 4.15 a) se muestra un ejemplo en que la cancelación de la tarea B no implica que se termine con la ejecución de la tarea X y las tareas subsiguientes. En la implementación se definió a través de un lugar primitivo pre-etiquetado *sub_final*, el cual consume cualquier token que llegue a éste.

### Cancelación de casos

Este patrón se refiere a la cancelación o deshabilitación del proceso de workflow [33]. Haciendo uso del lugar pre-etiquetado *cancel* seguido de un lugar pre-etiquetado *end* es posible

terminar el proceso y registrar la finalización del proceso como cancelación ó terminación anormal. Como se mencionó anteriormente por medio de la simulación es posible terminar todo el proceso con un lugar *end*. En la figura 4.15 b), se muestra un workflow en que la correcta ejecución de la tarea B seguida de la tarea C, darán como resultado la finalización correcta del proceso, sin embargo mediante la decisión tomada en el XOR a través de la transición regla es posible cancelar la ejecución del proceso.

### 4.5.7. Patrones de workflow en la perspectiva de datos

En [8] se describen los patrones de workflow que han sido identificados en la perspectiva de datos en lo que respecta al modelado de workflow, en este trabajo no son considerados para la evaluación de la metodología propuesta debido a que es únicamente de manera parcial en que se brindan opciones para modelar esta perspectiva, a través de las estructuras de datos que son representadas a través de colores en los tokens de la WfCCPN, además a través de esta investigación se considera que estos patrones son más apropiados para herramientas que integran el editor de workflows y el engine que permite su ejecución.

## 4.6. Comentarios finales

En este capítulo se mostraron las WfCCPN y los elementos que le componen y con los que se representan los conceptos de workflow en la metodología que se propone en esta tesis. Con éstas es posible modelar la perspectiva de flujo de control sin necesidad de abstraer datos de control como es realizado en [13]. A las CCPN originales fueron agregados elementos extras a partir de lo estudiado en trabajos como [15], [32], [9], [10], estos elementos permiten modelar todos los patrones de flujo de control que han sido propuestos, esto permitió evaluar la capacidad de modelado de las WfCCPN en la perspectiva de modelado, en ese capítulo fueron mostrados los resultados obtenidos.

# Capítulo 5

# Implementación

En este capítulo se expone la implementación de software realizada, la cual permite utilizar la metodología para modelar procesos de workflow propuesta en esta tesis. El sistema de software se denomina WfECAPNSim (Workflow ECA Petri Net Simulator, por sus siglas en inglés); el capítulo se encuentra dividido de la siguiente forma: la sección 5.1 presenta los antecedentes y una perspectiva de comparación entre las características y modo de uso con respecto al sistema ECAPNSim[1], la sección 5.2 muestra el diseño y arquitectura del sistema, la sección 5.3 describe las principales características que posee el sistema desarrollado, en la sección 5.4 se describe de forma general la forma de uso del sistema y finalmente la sección 5.5 presenta algunos sistemas de software relacionados, además de una breve comparación con respecto al WfECAPNSim.

## 5.1. WfECAPNSim

En [20], [19] se presenta el desarrollo del sistema ECAPNSim, el cual ofrece un medio gráfico y visual para representar bases de reglas ECA a través de las CCPN, además permite realizar la simulación del comportamiento de una base de datos activa, lo que se traduce a la simulación de su base de reglas ECA. El modo de funcionamiento del ECAPNSim es el siguiente: a partir de una definición de reglas ECA especificadas en un archivo de texto acorde a un formato predefinido, el sistema construye el modelo en CCPN el cual puede ser ejecutado a través del juego de tokens de la CCPN. El sistema permite realizar la ejecución de las reglas ECA en modo simulación, es decir únicamente se ejecuta el disparo de los tokens en

[1]Sistema desarrollado en [20] con el fin de utilizar las CCPN en el contexto de bases de datos activas

| Característica | ECAPNSim | WfECAPNsim |
|---|---|---|
| Edición | No necesaria | Edición de WfCCPN |
| Simulación | Juego de tokens de la CCPN | Juego de tokens de la WfCCPN |
| Verificación y validación | Mediante matriz de incidencia | Validación en tiempo de diseño de una WfCCPN bien formada |
| Manejo de colores | Colores definidos (de acuerdo a las sentencias SQL, i.e. color Insert, color Delete, etc.) | Creación y manipulación de colores a través de un editor. |
| Característica para modelado jerárquico | - | Manejo de subredes mediante lugares refinables. |
| Característica de múltiple instanciación | No necesaria | Implementada a través de la característica de jerarquía |
| Características temporales | - | Transiciones temporizadas, intervalos de tiempo. |
| Conexión con bases de datos | A diferentes sistemas de bases de datos (PostgreSQL, Oracle, Access, MySQL). | A través de serialización de objetos |

Figura 5.1: Características presentes en los dos sistemas.

la red de forma gráfica, ó en modo real a través de la conexión a una base de datos realizando las acciones establecidas en las reglas ECA afectando de manera real los registros de la base de datos, además de mostrarlo gráficamente a través de la ejecución de la CCPN.

Para la implementación de software realizada en esta tesis se utilizó como base de desarrollo el sistema ECAPNSim. Debido a la naturaleza y modo de uso del ECAPNSim este no posee la característica de edición de CCPN ya que no es necesaria para este trabajo, sin embargo el WfECAPNSim no puede prescindir de dicha característica, es así como este tipo de diferencias dieron como resultado el que ciertas *clases Java* del sistema ECAPNSim fueran utilizadas o modificadas, mientras que otra parte del desarrollo fue excluida. El desarrollo del WfECAPNSim es considerado una modificación y extensión al sistema ECAPNSim, ya que existen algunas diferencias en cuanto a características y forma de uso, estas diferencias se muestran en la figura 5.1.

El diseño y arquitectura del WfECAPNSim así como las características presentadas en la tabla de la figura 5.1 se describen en las siguientes secciones.

## 5.2. Diseño y arquitectura

La arquitectura del WfECAPNSim se muestra en la figura 5.2. El sistema se encuentra dividido en diferentes módulos los cuales cumplen una función específica, estos módulos se agrupan en subsistemas los cuales se comunican e interactúan entre sí para proporcionar la

funcionalidad deseada. Los componentes y sus sub-sistemas se describen a continuación.

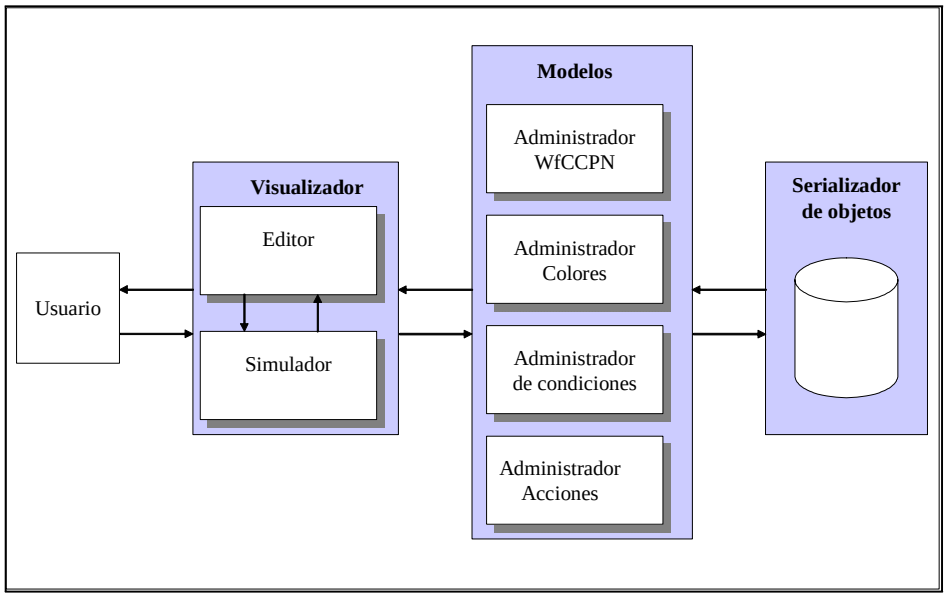


Figura 5.2: Arquitectura del WfECAPNSim

### 5.2.1. El componente visualizador

El componente visualizador es la interfaz con el usuario, está constituido por el lienzo (área de dibujo ó área de edición), está permite dibujar el modelo y observar la simulación de la WfCCPN. Cuando una WfCCPN es creada el usuario es libre de distribuir la red de Petri de la manera más adecuada con el fin de que al momento de ejecutar la simulación ésta sea percibida por el modelador sin problemas. Este componente posee dos subsistemas, que son los siguientes.

#### El subsistema Editor

Este subsistema controla todo lo referente a la presentación en pantalla de la WfCCPN, utilizando la pantalla de edición para mostrar la construcción del modelo. También proporciona los controles (botones, menús, entre otros más) necesarios para insertar cada uno de los elementos de una WfCCPN, así también para modificar y establecer los valores correspondientes, en este modulo se definen todos los controles y eventos de Java Swing con los que está construida la interfaz gráfica del sistema.

#### El subsistema Simulador

Este subsistema utiliza la pantalla de edición para mostrar la ejecución del juego de tokens de la WfCCPN, a través de este módulo se presenta la simulación gráfica del proceso de workflow. La animación se realiza a través de los *Timers* que proporciona el lenguaje Java, estos funcionan en base a la escala de tiempo configurada para la simulación.

### 5.2.2. El componente administrador de modelos

Este componente se encarga de mantener la relación lógica entre cada uno de los elementos que componen el modelo, tales como: lugares, transiciones, arcos, sub redes, conjunto de colores y los elementos que dan el enfoque de regla ECA, es decir el conjunto de Acciones y las Condiciones, además se mantiene un registro histórico de las ejecuciones, todo lo anterior se realiza para cada proceso de workflow modelado de manera independiente. A continuación se describen los subsistemas de este componente.

#### Administrador de WfCCPN

Este subsistema permite en tiempo de diseño administrar el control del grafo formado por los elementos de la WfCCPN manteniendo la relación entre índices de cada uno de estos elementos, por ejemplo la relación que existe entre una transición y sus diferentes arcos de entrada; en base a índices permite que en caso de ser eliminado algún elemento sea invocado un método, el cual mantiene la consistencia en la relación y estructura de la red de Petri, de esta manera al establecer alguna relación inválida entre elementos, el subsistema se encarga de anular la acción del usuario y enviar el mensaje correspondiente.

#### Administrador de colores

Este componente permite la gestión de tipos de datos (colores) para el proceso a modelar, la estructura de un color es similar a una estructura de datos propia del lenguaje C, a partir de la creación de un color, se definen sus atributos, los cuales son de un tipo de dato ya sea numérico o de tipo texto, incluso otro color, es decir colores formados por otros colores. Además es posible asociar un color RGB del sistema operativo a cada definición, esto para efectos de diferenciar visualmente los colores de tokens al momento de ejecutar la simulación.

#### Administrador de condiciones

Este subsistema permite la definición de condiciones lógicas y temporales en las transiciones, su evaluación se realiza a través de un analizador y evaluador de condiciones, el cual transforma la expresión a su forma postfija para ser evaluada, además en caso de que la condición haga referencia hacia atributos de color del token evaluado se obtiene el valor, cabe recordar que un token puede contener como atributo otro color que a su vez contendrá atributos, de este modo, el sistema tiene la capacidad de encontrar el atributo final al que se hace referencia en la condición. Por otro lado las condiciones temporales son de dos tipos: condición de cumplimiento de un plazo y condición de un intervalo válido para el disparo de transiciones, de estas condiciones posteriormente se presenta una explicación de como se utilizan en el sistema.

#### Administrador de acciones

Este subsistema permite la definición de acciones, para esta primera versión del software WfECAPNSim se implemento una interfaz tipo asistente de aplicaciones el cual permite definir 5 tipos básicos de acciones, esta interfaz tiene por objetivo mantener la consistencia en las acciones definidas con respecto a los elementos de la WfCCPN que posea en el momento de la definición. Por ejemplo, cuando un lugar es eliminado, la consistencia es verificada por este subsistema, alertando de los cambios y posibles errores que se pueden desencadenar debido a la eliminación de ese elemento.

### 5.2.3. Serializador de objetos

Cuando el usuario modela un proceso de workflow, necesitará almacenarlo y recuperarlo para posteriores sesiones de trabajo, este requerimiento básico en la mayoría de los sistemas de software, se provee en WfCCPN a través de la *serialización* de objetos proporcionada por el lenguaje Java, con esto se logra no tener dependencia de algún método de almacenamiento externo como pudiese ser una base de datos relacional. La serialización es un método que permite mantener los objetos construidos con Java, guardando su estructura y valores actuales, de esta manera es posible guardar la red generada con todos sus elementos (lugares, transiciones, arcos, tokens, páginas de instancias y jerarquía, conjunto de colores, además de las condiciones y acciones asociadas a las transiciones del modelo). De esta manera cuando se guarda un modelo se guarda toda la información asociada a este.

En el sitio web http://www.wfccpn.org es posible obtener información más detallada en cuanto al diseño e implementación (código fuente) del sistema WfECAPNSim. En la siguiente sección se muestra la implementación de los componentes anteriormente descritos y la forma en que se presentan al usuario a través de las interfaces del sistema.

## 5.3. Características

WfECAPNSim permite utilizar la metodología propuesta para modelar procesos de workflow a través de WfCCPN añadiendo características de workflow. La descripción de cada una de estas características se realiza en algunos casos mostrando las pantallas correspondientes del sistema. En la figura 5.3 se presenta la pantalla principal del sistema.

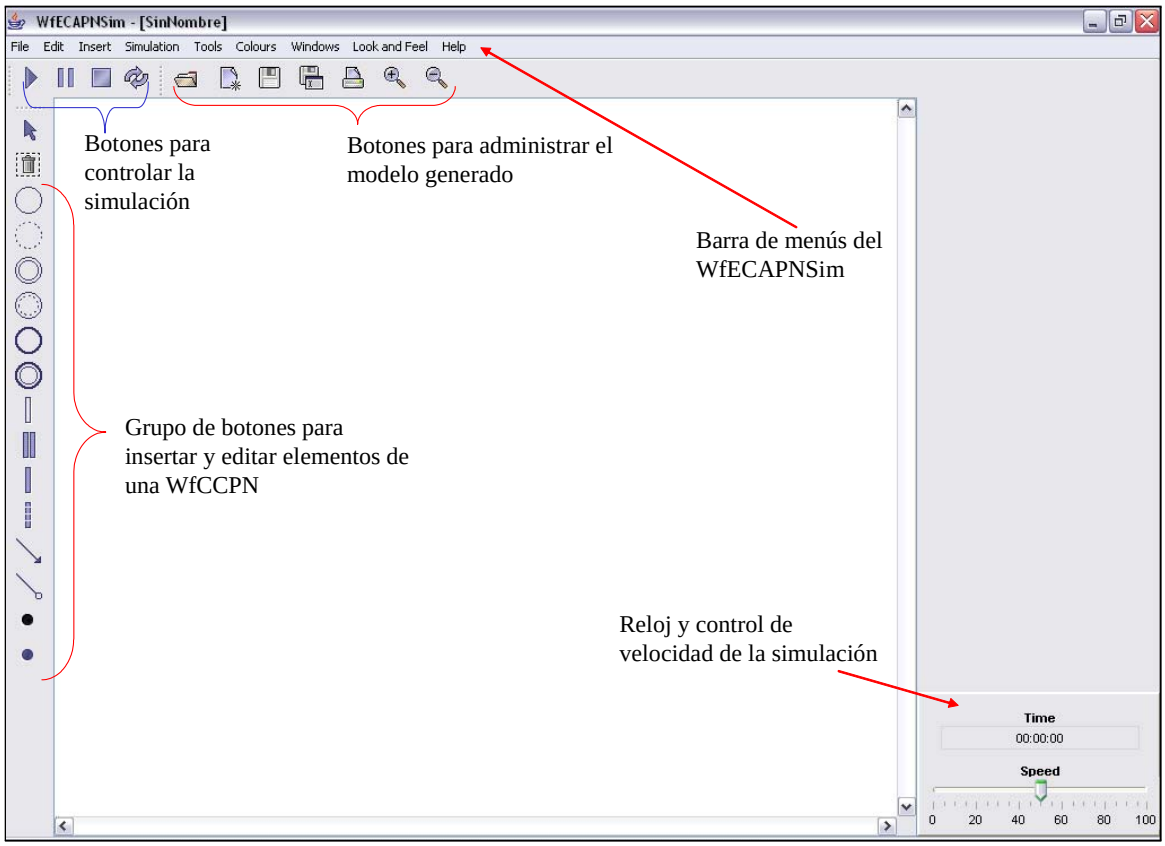


Figura 5.3: Pantalla principal del WfECAPNSim

### 5.3.1. Edición de WfCCPN

Esta es la primera característica con la que el usuario del sistema interactúa, la edición proporciona facilidades para un rápido diseño a través de controles y eventos necesarios. Se diseño de tal modo que los elementos de la pantalla principal sean suficientes para crear una WfCCPN de manera intuitiva. En la figura 5.3 se muestran las diferentes barras de botones y controles que posee el sistema. A continuación se muestran los detalles de edición para cada uno de los elementos de una WfCCPN.

#### Lugares

Cada tipo de lugar de una WfCCPN puede ser insertado con su respectivo botón que se encuentra en la barra de botones de la izquierda. Una vez insertado un lugar, es posible asignarle un color de dominio, asignarle tokens, establecer información descriptiva, además de tener la opción de eliminarlo ó desplazarlo dentro del lienzo.

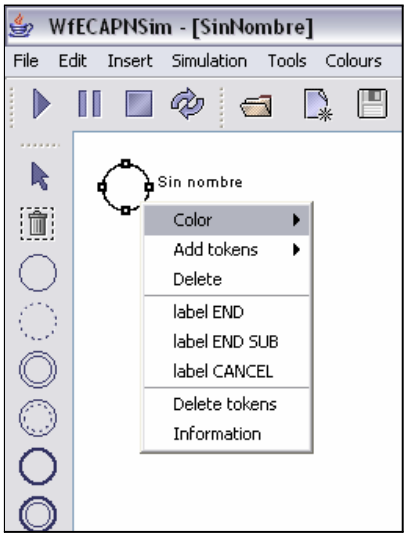


Figura 5.4: Edición de lugares

En la figura 5.4 se muestra el menú contextual y las opciones que corresponden a un lugar de tipo primitivo. Cuando se conoce el color de los tokens que llegaran a un determinado lugar o se desea restringir el color de los tokens que pueden llegaran al lugar, se utiliza el dominio de color, el dominio se establece con la asociación de uno de los colores definidos a través del editor de colores.

En cuanto a la asignación de tokens en los lugares virtuales se tiene la posibilidad de asignar tokens negros (neutros) y tokens de cualquier color (datos) . En los lugares primitivos únicamente es posible introducir tokens de color de su dominio, es preciso mencionar, que en caso de cambiar el color de domino cuando este ya hubiese sido establecido, entonces los tokens existentes (dentro de ese lugar) en ese momento son eliminados. En los restantes tipos de lugares no esta presente la opción de insertar tokens mediante edición del usuario.

La información descriptiva en los lugares consiste en especificar si el lugar representa cualquiera de las siguientes situaciones: evento de una regla, acción de una regla, evento de una regla y acción de otra, acción de una regla y evento de otra, la interfaz se muestra en la figura 5.5 b) y es acorde a la metodología sugerida en el capitulo 4, también se tiene un área de texto para introducir de manera libre la descripción de lo que representa ese lugar.

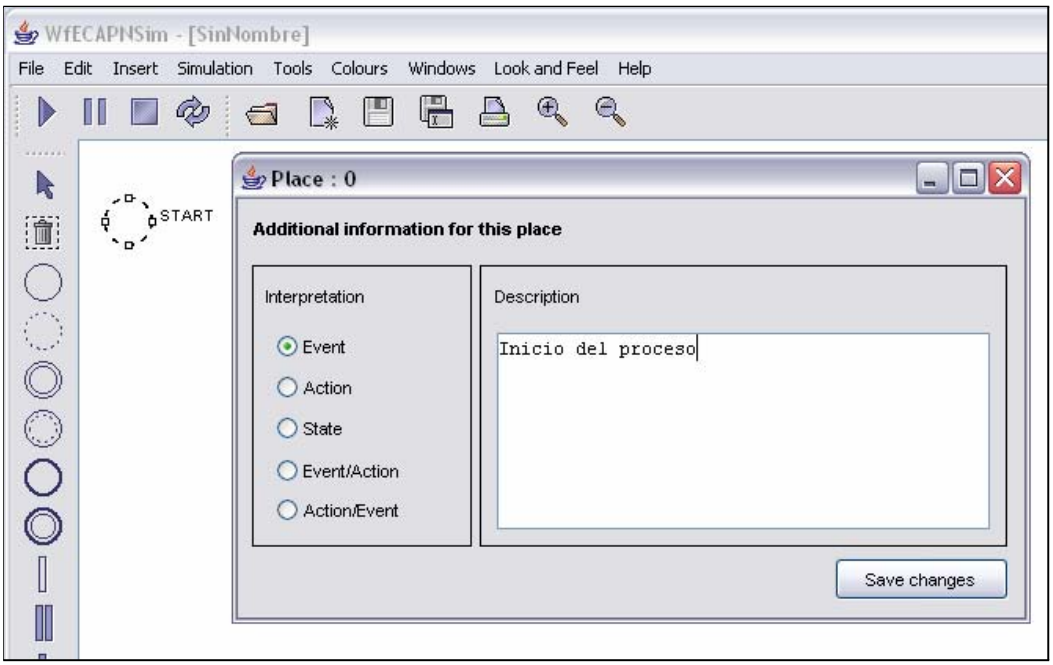


Figura 5.5: Edición de información adicional en los lugares

### Transiciones

Una transición de tipo *copia* no contempla datos adicionales para edición tal y como se muestra en la figura 5.6 b), sin embargo las transiciones de tipo regla y compuesta permiten la definición de una condición y una acción, la forma de definir una condición y una acción,

las figuras 5.6 c) y 5.6 a) se muestran los menús contextuales para estas transiciones respectivamente, por último las transiciones tiempo permiten dos comportamientos diferentes, ya sea como transición temporizada o con intervalos de tiempo válidos para su activación, en la figura 5.6 d) se muestra la opción dentro del menú que despliega la interfaz para especificar alguno de los comportamientos citados.

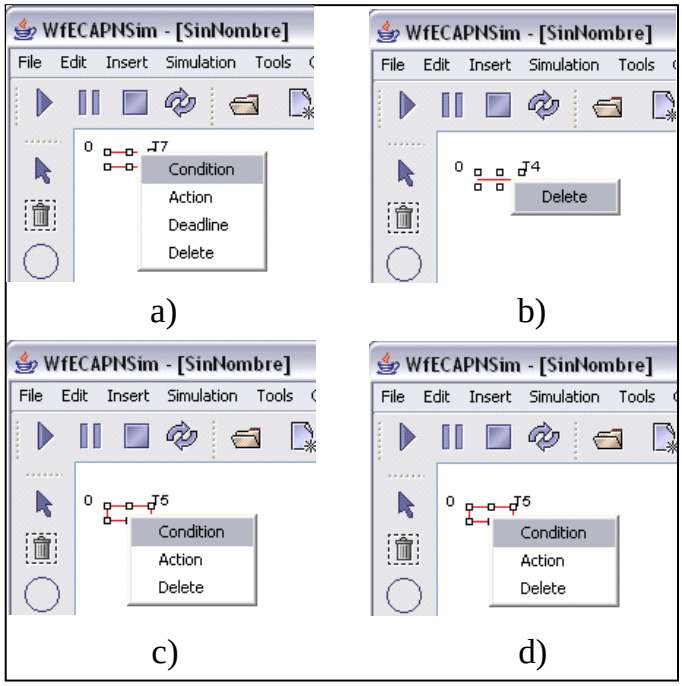


Figura 5.6: Edición de transiciones de una WfCCPN

Para intervalos de tiempo el usuario tiene la posibilidad de establecer el intervalo mínimo de activación sin establecer la fecha y la hora, con esto se indica que el intervalo mínimo puede ser cualquiera menor al intervalo máximo. También es posible especificar la fecha y la hora, o únicamente la fecha, o de igual manera únicamente establecer la hora; las mismas opciones se tienen para el intervalo máximo. Sin embargo es necesario que en el caso de las tres ultimas formas, estas se correspondan entre el intervalo mínimo y el máximo. La interfaz se muestra figura 5.7.

Para incluir retardos (*delays*) en el disparo de la transición *tiempo* se realiza con el segundo conjunto de opciones mostrado en la fig. 5.7. Es posible establecer retardos constantes en unidades de tiempo, recordando que la unidad de tiempo especificada en milisegundos en función de la configuración para la simulación que será mostrada en la sección 5.3.6. También

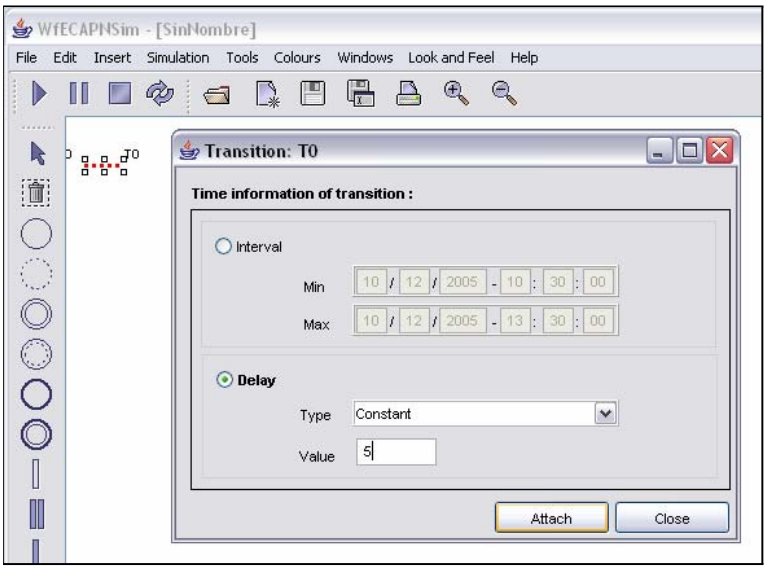


Figura 5.7: Transiciones tiempo

es posible establecer valores aleatorios con distribuciones discretas de probabilidad. Los valores generados están en el rango de 0 a 1, por lo que el usuario tiene la opción de establecer un factor de multiplicación y un intervalo de números aleatorios aceptados.

### 5.3.2. Editor de colores

La definición de los colores, (es decir los tipos de datos) realiza a través del editor de colores, siendo posible guardar y reutilizar el conjunto de colores en otros modelos. Los pasos para definir un conjunto de colores son los siguientes: Primero se define el nombre del color, utilizando el botón agregar, el color es guardado y de esta manera es posible agregar los atributos deseados que pueden ser de los siguientes tipos: Char, Varchar, Integer, Float, Numeric, Date, Time, Timestamp, Color.

Además es posible asignarle un color del sistema para representarlo de manera gráfica. En la figura se muestra el editor de colores.

### 5.3.3. Condiciones

Las condiciones lógicas se asignan a transiciones tipo *regla* y *compuestas*, mientras que las condiciones temporales únicamente se asignan a transiciones *compuestas*.

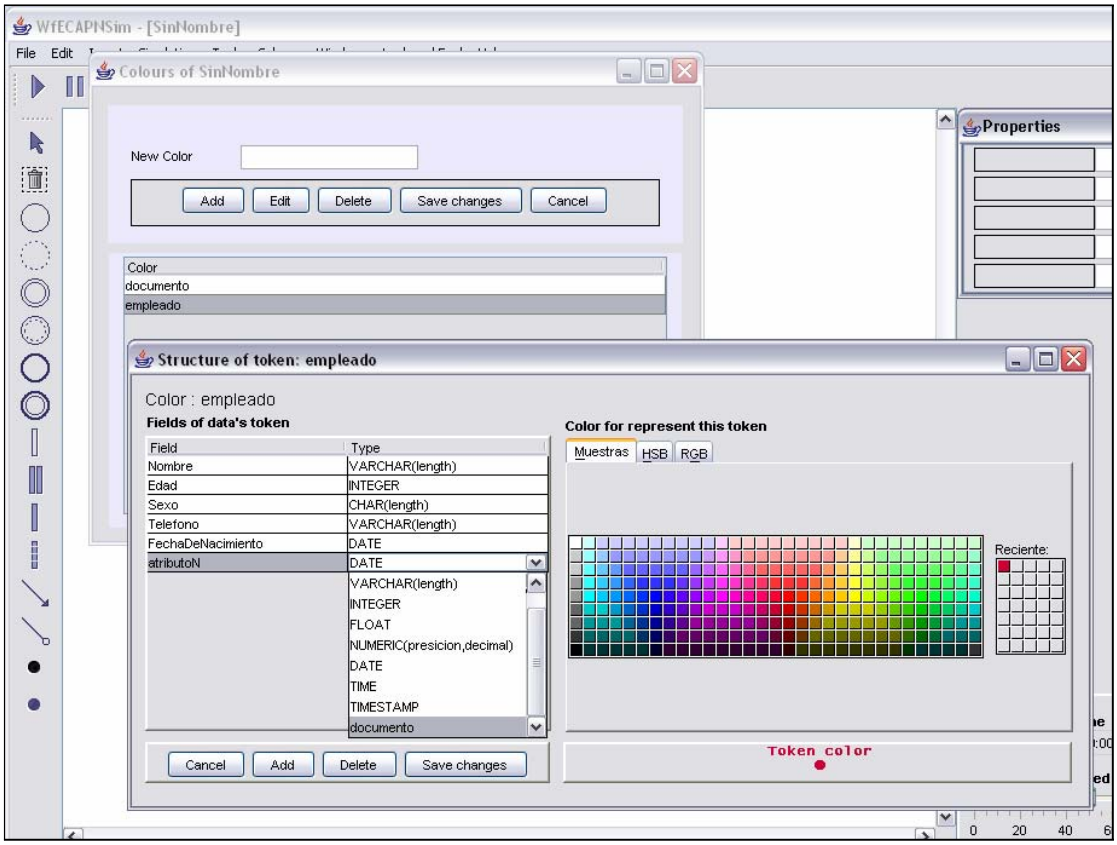


Figura 5.8: Administrador de colores del WfECAPNSim

#### Lógicas

La manera de asignar una condición lógica es a través de una cadena de texto definida a través de operadores lógicos. Los operadores utilizados son los siguientes:

Operadores de comparación: <, >, <=, >=, =, <>

Operadores de asociatividad: |, &, (, )

#### Temporales

Las condiciones temporales se asignan a transiciones *compuestas*, estas condiciones pueden comparar la fecha y hora del reloj de la simulación contra la estampa de tiempo de creación o de modificación en los tokens al momento de llegar a una transición compuesta.

### 5.3.4. Acciones

Si la condición establecida en la transición al ser evaluada resulta verdadera, entonces opcionalmente una acción puede ejecutarse, logrando de esta manera representar completamente el comportamiento de una regla ECA, para este trabajo se plantearon cinco acciones básicas, la definición de la acción se realiza a través de un editor de definición el cuál lleva paso a paso al usuario en la creación y configuración de la acción, reduciendo de esta manera la posibilidad de introducir errores en su definición. A continuación se describen las posibles acciones que pueden adjuntarse a una transición de tipo *regla* ó *compuesta*.

#### Modificar atributos en el color del token

Con esta acción es posible modificar el valor de cualquiera de los atributos del color referenciado en los tokens que son evaluados dentro de la transición que posee la acción asignada. Al establecer los atributos que se desean modificar es posible incluir operadores básicos para los atributos numéricos, ó valores constantes como cadenas de texto. El sistema mantiene la consistencia en el modelo a través de verificar que el dato asignado al atributo corresponda con su tipo definido en el color. En la figura 5.9 se muestra cómo el atributo revisado del color documento será modificado al asignársele el valor de tipo texto: “revisado”, al cumplirse la condición y ejecutarse la acción en la transición compuesta.

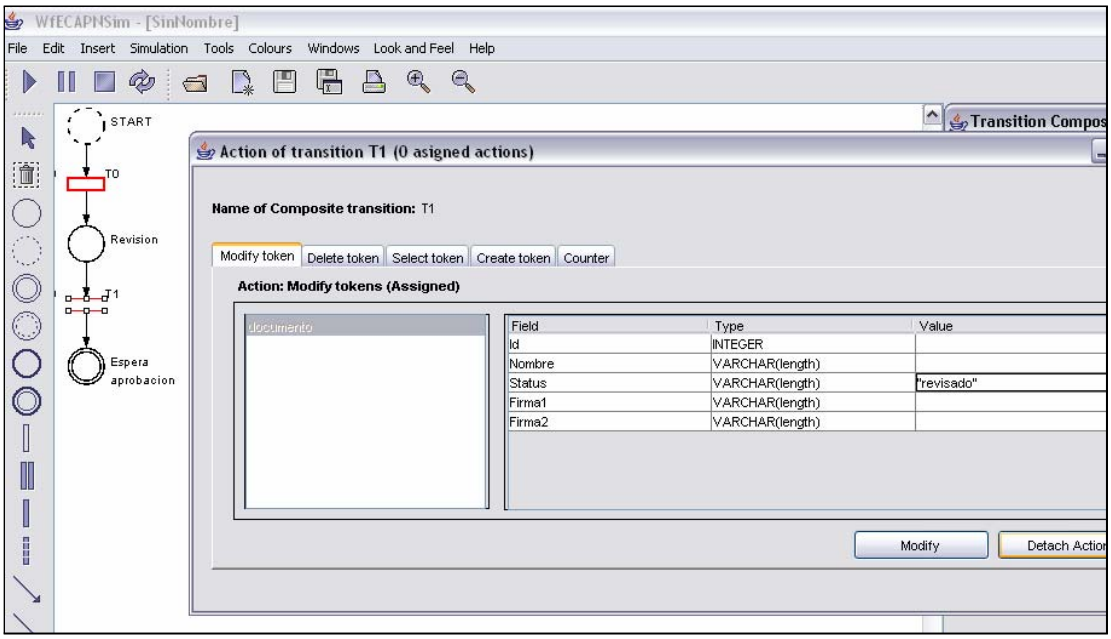


Figura 5.9: Editor de acciones - Modificación de atributos en colores de tokens

### Crear tokens con un color nuevo

Este tipo de acción permite obtener tokens con colores compuestos a través de la creación de nuevos colores en donde los atributos del nuevo color se toman de los colores de tokens que llegan desde los lugares de entrada de la transición que posee la acción. En tiempo de edición es conveniente definir los colores de los lugares de entrada de la transición asociada con la acción, ya que a partir de esos colores quedará definido el espacio de colores del cual será posible crear el nuevo color, cabe mencionar el caso en que uno de los lugares de entrada de la transición es de tipo virtual, entonces el espacio de colores será el conjunto total de colores del modelo.

Al crear el token con el color nuevo, será posible establecer los valores deseados en cualquiera de los atributos, sin embargo se tiene la posibilidad de asignar los valores que contienen los tokens formadores del token nuevo, esto se obtiene a través de mantener en blanco la celda correspondiente a la columna valor (value) de los atributos deseados.

### Seleccionar tokens

Esta acción permite establecer los tokens en base a su color, a los cuales se les permite pasar hacia el lugar de salida, los que no se especifiquen serán eliminados.

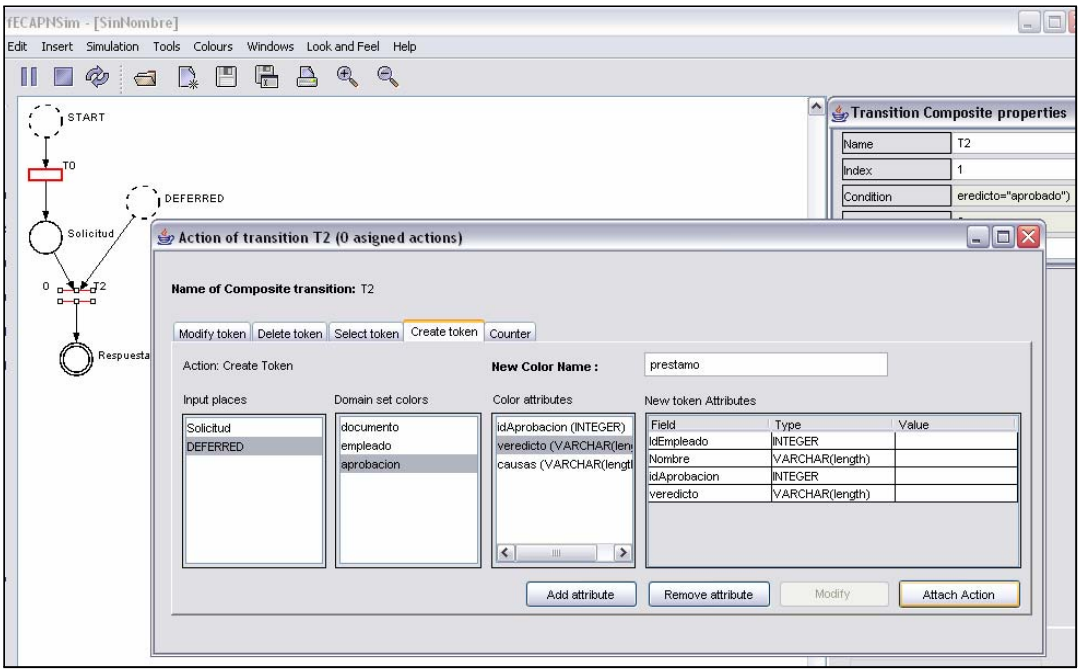


Figura 5.10: Administrador de acciones - Creación de colores para tokens

#### Eliminar tokens

Con esta acción es posible modelar situaciones de cancelación en un workflow, si la condición establecida en la transición se cumple, entonces los tokens existentes en los lugares seleccionados a través del asistente de acciones, serán eliminados.

#### Modificación del contador en las transiciones

Es posible manipular el valor del contador que poseen las transiciones, si una condición se cumple el contador puede ser reiniciado a 0, o puede ser inicializado al valor establecido a través del editor de acciones.

### 5.3.5. Jerarquía y múltiples instancias

Mediante la definición de sub-páginas es posible definir una sub-red. Una sub-página de un lugar de tipo *sub_red* o *instancia*, posee un lugar virtual para el inicio del sub-proceso y un lugar primitivo para su finalización de ese sub-proceso. El sistema permite la edición de una sola página a la vez, en caso de tener abiertas otras sub-páginas éstas serán desactivadas para su edición y el sistema las presenta en modo de inspección. En la figura 5.11 se muestra

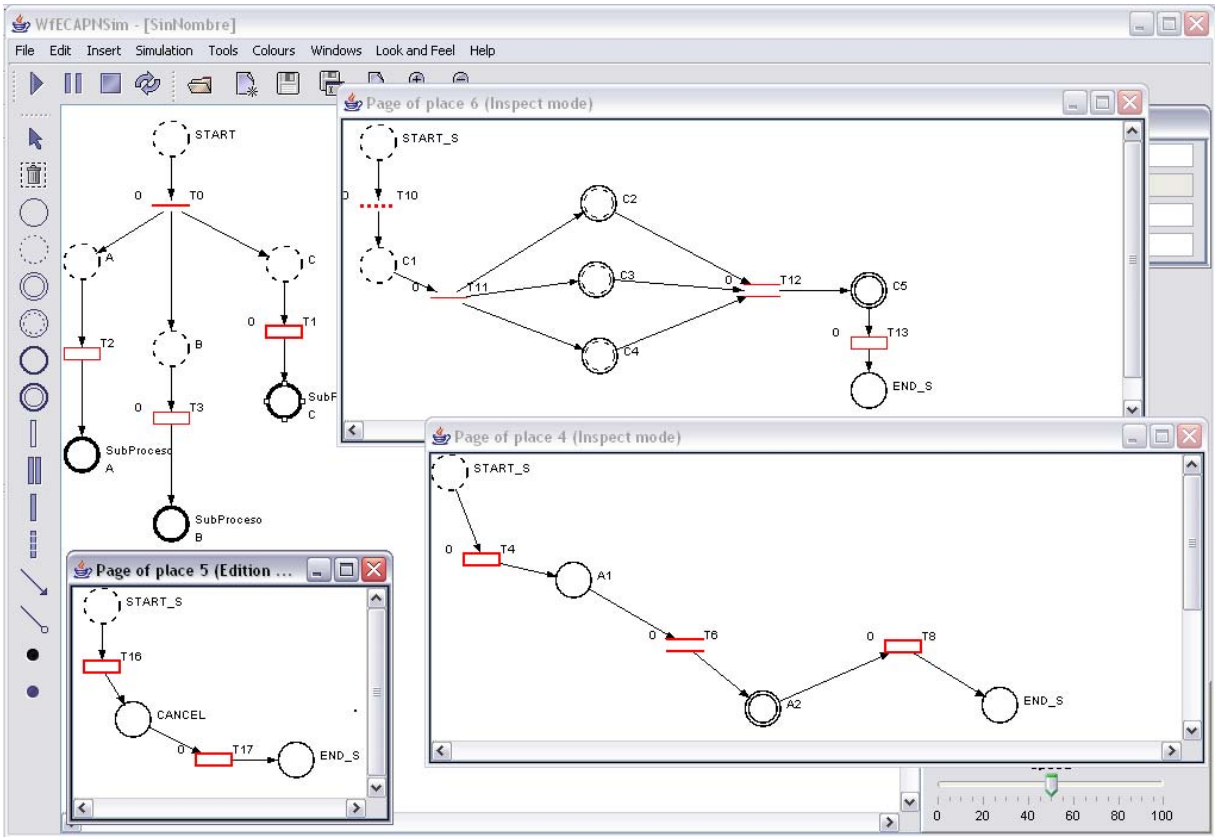


Figura 5.11: Administración de sub-páginas

la definición de dos sub-páginas, los controles de edición son los mismos que los de la pantalla principal del WfECAPNSim.

Es posible establecer el comportamiento en tiempo de ejecución para las sub-páginas de jerarquía y de instancia, a través de configurar el sistema para que la ejecución de todas las sub-páginas sean ó no, visualizadas; también es posible configurar el sistema de modo que este pregunte por cada sub-página si desea o no visualizar la ejecución.

### 5.3.6. Simulación

La simulación se realiza por medio del juego de tokens de la red WfCCPN, mediante temporizadores controlados por tiempos. El sistema permite iniciar, detener o pausar la simulación, de esta manera es posible observar los detalles y la información que va cambiando con el tiempo, para observar la información de un lugar basta con colocar el puntero del ratón sobre éste e inmediatamente se muestra dicha información a través de un "*tool tip text*"; a

través de la configuración general del sistema es posible definir la información que se desea visualizar, siendo posible conocer la información de los atributos del color en los tokens que se encuentren en ese momento dentro del lugar.

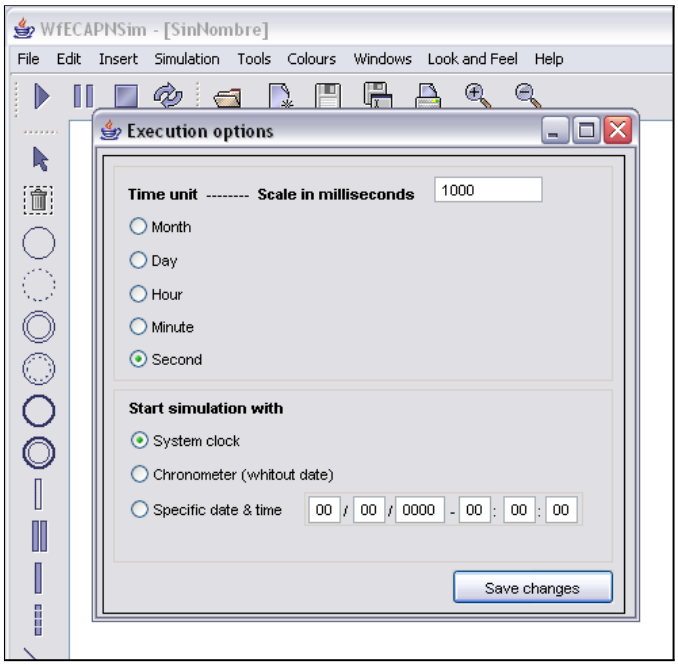


Figura 5.12: Configuración del tiempo para la ejecución de simulaciones.

### Tiempo

El sistema permite modelar procesos de workflow que tienen en cuenta el tiempo en que se realizan ciertas tareas, para esto se proporciona la posibilidad de tener unidades de tiempo escaladas en milisegundos, así como también es posible definir la fecha y hora de inicio de la simulación, en la figura 5.12 se muestra el panel de configuración.

Es posible que las unidades de tiempo sean segundos, minutos, horas, días o meses, por defecto el sistema está configurado para trabajar con un segundo como unidad de tiempo el cual se encuentra configurado para que su escala sea 1000 milisegundos, es decir, el tiempo real de un segundo. En la segunda sección de configuración es posible definir tres valores: Reloj del sistema, con esta opción la simulación se inicia con la fecha y hora actual del sistema operativo; Cronómetro con esta opción únicamente se visualizan las horas con sus minutos y segundos, iniciando desde cero; y por último es posible personalizar la fecha y hora que se

desea aparezca como inicio de simulación. Es en base a esta configuración de tiempo que son evaluadas las condiciones temporales en las transiciones de tipo *tiempo* y *compuestas*.

#### Reportes

Los reportes que proporciona el sistema son referentes a la interpretación dada de la red WfCCPN con relación al proceso de workflow modelado. Es posible configurar la información que se desea mostrar en los reportes, esta puede ser la siguiente: mostrar detalle de los eventos, condiciones y acciones ejecutadas, lo cual además contempla mostrar la fecha y hora en que ocurren es decir, el momento en que un token llega a un lugar o a una transición. también se pueden mostrar los resultados de las evaluaciones en las condiciones, otra información que puede mostrarse en los reportes se refiere a los tokens eliminados por no cumplirse alguna condición.

#### Validación de modelos

La validación del proceso de workflow se realiza en tiempo de diseño y únicamente comprueba que la red WfCCPN sea construida de forma válida, de esta manera se validan las conexiones entre elementos de la WfCCPN, por ejemplo un lugar de tipo virtual no puede tener una transición de entrada tipo regla, cuando el sistema detecta esta situación, envía un mensaje e ignora la edición del usuario.

## 5.4. Modo de uso

El sistema WfECAPNSim, puede utilizarse de la manera a la que más se adapte el usuario, sin embargo se consideró la presente sección para ilustrar de manera general como el usuario que modela procesos de workflow, puede realizar esta tarea utilizando el sistema presentado en este capítulo.

En la figura 5.13 a través de UML con un diagrama de secuencia se muestra un caso general de cómo el usuario construye el modelo de un proceso de workflow. A través del menú archivo la interfaz de usuario puede crearse un nuevo modelo y guardarse como un nombre de archivo y la extensión *.cpn*, una vez que el usuario ha creado el nuevo proyecto, es aconsejable crear el conjunto de colores identificado para el proceso de workflow, esto es la definición de los tipos de datos a través del administrador de colores. Con el conjunto

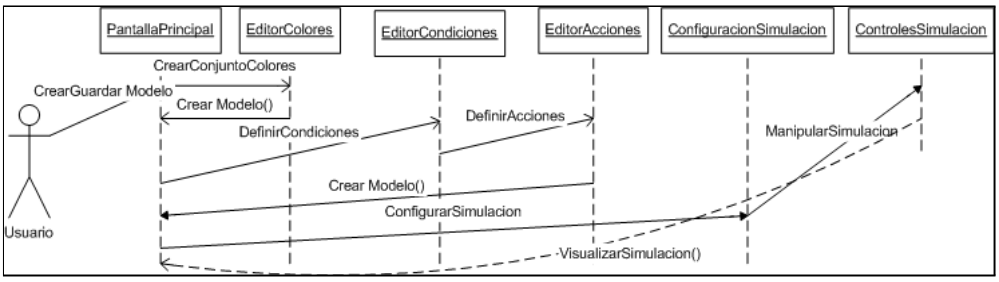


Figura 5.13: Diagrama de secuencia para describir el modo de uso del sistema WfECAPNSim

de colores creado es posible iniciar la construcción de la red WfCCPN representando las reglas ECA identificadas en el proceso de workflow, estableciendo la información relevante en cada elemento de la WfCCPN, al tener el grafo dibujado el cual se encuentra formado por los elementos WfCCPN, es posible definir las condiciones y las acciones que se adjuntan a las transiciones de tipo regla y compuestas, así como los detalles temporales a través de las transiciones tiempo. Una vez que se tiene el proceso modelado con todos sus detalles y datos asignados a los elementos WfCCPN, es posible configurar el tiempo y su escala para la ejecución de la simulación. Para posteriormente tener la posibilidad de ejecutar, pausar o detener la simulación. El registro de cada simulación puede obtenerse a través de los registros históricos que proporciona el sistema. De esta manera el modelador de procesos de workflow puede repetir el ciclo de diseño y modelación tantas veces sea considerado necesario, hasta obtener la mejor representación del proceso modelado en WfCCPN a juicio del usuario.

## 5.5. Sistemas de software relacionados

En capítulos anteriores han sido mencionados diversos sistemas administradores de workflow, algunos poseen licencias comerciales y otros más licencia libre[2], de estos sistemas una parte únicamente contempla la definición de procesos de workflow para su simulación y análisis, mientras que otros además contemplan la representación de las definiciones a través de un motor de workflow para su ejecución y monitoreo real del proceso, es decir son sistemas administradores de workflow completos. La gran cantidad de sistemas además de la diversidad en cuanto a tecnologías utilizadas y metodologías empleadas para modelar los procesos de workflow, convierte en prácticamente imposible la tarea de realizar una comparación com-

[2]En [34], [35], [17] es posible encontrar referencia a sistemas desarrollados en lenguaje Java, que son de licencia libre.

pleta y objetiva con cada uno de los sistemas existentes, por lo tanto de todas las opciones se eligieron tres sistemas los cuales permiten obtener un punto de comparación más justo con respecto al sistema WfECAPNSim.

Los sistemas COSA, YAWL, WoPeD, BOSSA, XRL/Lower, Renew, ExSpect y Flower son sistemas basados en redes de Petri y al igual que el WfECAPNSim son implementaciones realizadas con tecnología Java. Sin embargo la lista se reduce debido a que los sistemas COSA, ExSpect y Flower poseen licencia comercial, de estos tres sólo fue posible obtener una versión demo de ExSpect. Por otro lado XRL/Lower es un sistema basado en web, el cual su definición se basa en Workflow Nets, sin embargo no contempla la definición gráfica de los procesos de workflow, debido a que los procesos de definen a través de XML utilizando XRL (eXchangeable Routing Language), al no ser una herramienta gráfica, no se consideró para los objetivos de comparación.

| **Característica** | **YAWL** | **WoPeD** | **Bossa** | **WfECAPNSim** |
|---|---|---|---|---|
| Tipo de red de Petri que utiliza | Lenguaje basado en Wf-Nets | Wf-Nets | Redes de Petri básicas | Redes de Petri de alto nivel (WfCCPN) |
| Motor workflow | Si | No | Si | No |
| Patrones de workflow | Si | Si | No | Si |
| Perspectiva de datos | A través de definiciones XML | No | Definición de datos | Definición de Colores |
| Juego de tokens | Si | Si, pero es manual | No | Si |
| Características de tiempo | Si, a través de un modulo adicional | No | No | Si |
| Jerarquía | Si | No | No | Si |
| Múltiples instancias | Si | No | No | Si |
| Reportes | Si | No | Si | Si |
| Validaciones | Si | No | No | WfCCPN bien formada |

Figura 5.14: Comparativa entre sistemas de software relacionados

En la tabla de la figura 5.14 se muestran las características principales que poseen los sistemas comparados, los que a pesar de que tienen más tiempo y más recursos humanos en su desarrollo pueden comprarse con el WfECAPNSim. El sistema YAWL [9] es resultado de más de 10 años de trabajos de investigación por parte del doctor Wil Van der Aalst y su

equipo de colaboradores, este sistema hace uso de un lenguaje basado en las Wf-Nets que son un tipo de red de Petri, este sistema implementa todas las características tomadas en cuenta para la comparación realizada, el único detalle que puede ser mencionados sobre este sistema es sobre la definición de datos ya que se realiza de forma manual a través XML, sin embargo este es el software más importante actualmente y del cuál otros desarrollos toman ideas y conceptos, tal como fue hecho en este trabajo de tesis. Existen sistemas de software que hacen uso de las Wf-Nets para la definición de los proceso de workflow y el motor de YAWL a través de una interfaz, para la ejecución de los procesos, tal es el caso de WoPeD [36] el cual se considera como punto de referencia más justo con el sistema WfECAPNSim debido a que son herramientas para la definición de procesos y no motores de workflow, y como se muestra en la tabla comparativa son varias las ventajas de nuestro sistema sobre WoPeD. En el caso de Bossa [37] su mayor desventaja es la falta de apego al modelo de referencia de la WfMC, además de no existir alguna evaluación utilizando los patrones de workflow, si bien utilizan redes de Petri, no utilizan la característica de colores en estas.

## 5.6. Comentarios finales

La implementación de software desarrollada tiene su contribución como herramienta de modelado y simulación de procesos de workflow a través de redes de Petri; el código fuente del WfECAPNSim se distribuye bajo licencia GPL y se encuentra disponible para su uso o modificación de manera libre en la dirección de Internet: http://www.wfccpn.org. El sistema desarrollado ofrece características de redes de Petri las cuales por lo general únicamente se ofrecen en sistemas de software comerciales, es decir es necesario pagar cierta cantidad de dinero para poseer el programa de software, algunas de estas características son el manejo de colores, el diseño de modelos jerárquicos, y características temporales. Es bastante amplio el tema de workflow, por lo tanto las características que ofrece el WfECAPNSim pueden seguir desarrollándose, sin embargo se considera una buena aproximación como herramienta de modelado gráfico y de simulación para procesos de workflow, la cual se considera como una base sólida para explorar la construcción de un motor de workflow el cuál permita la ejecución de los procesos modelados.

# Capítulo 6

# Metodología y un caso de estudio

En el capítulo anterior se presentó la implementación de software realizada con el fin de utilizar las WfCCPN para modelado de procesos de workflow. En este capítulo se muestra la metodología propuesta en este trabajo de tesis, además a través de un ejemplo se describe la manera de modelar procesos de workflow a través de utilizar las WfCCPN y el sistema WfECAPNSim.

## 6.1. Metodología propuesta

La metodología para modelar procesos de workflow que se propone con este trabajo de tesis consiste en lo siguientes pasos.

1. Identificar las tareas y demás conceptos que contiene el proceso de workflow que se modelará.

2. Identificar las reglas ECA que componen al proceso de workflow.

3. Identificar los datos que tienen relevancia dentro del proceso de workflow.

4. Representar al proceso de workflow a través de reglas ECA mediante la sintaxis sugerida.

5. Generar la WfCCPN en base a las reglas ECA resultantes, utilizando el sistema WfE-CAPNSim.

6. Generar el conjunto de colores (datos) identificados en el paso 2.

7. Configurar la WfCCPN para su ejecución, añadiendo datos descriptivos y datos temporales en los elementos de la WfCCPN.

8. Guardar el archivo resultante y realizar las simulaciones del proceso modelado.

9. Ajustar el modelo, hasta obtener el comportamiento y resultados deseados.

### 6.1.1. Identificar el proceso de workflow

En la sección 4.4 se describen los elementos que componen a un proceso de workflow, y la manera en como se representan a través de los elementos de una WfCCPN. Mientras que, en la sección 4.3.4 se muestra la manera general de representar los elementos de una regla ECA a través de una WfCCPN. Al haber realizado los paso 1, 2 y 3 es posible representar al proceso a través de reglas ECA, las cuales serán descritas mediante la siguiente sintaxis.

### 6.1.2. Representación del proceso a través de reglas ECA

La representación del proceso a través de reglas ECA se refiere al paso 4 de la metodología. La sintaxis[1] para describir los elementos del workflow como reglas ECA se deriva de la utilizada en [20].

**Eventos.** Un evento se define utilizando la palabra ON, seguido del nombre del evento, sin embargo cuando se trata de un evento compuesto se utiliza la palabra que describa el tipo de evento, que pueden ser las siguientes: AND_SPLIT, AND_JOIN, OR_SPLIT, OR_JOIN, XOR_SPLIT. Estas representan las construcciones básicas de enrutamiento en un proceso de workflow. Además, los eventos compuestos definidos en [20] pueden ser implementados, estos son los siguientes: SEC (secuencia), SIM (simultáneo), TIMES (historia), NOT (negación), CLOSURE (cerradura), ANY (cualquiera).

Sintaxis: ON ( tipo_evento (evento$_1$:evento$_2$: ... :evento$_n$) | regla_eca )

**Condiciones.** Las condiciones se establecen comparando los tipos de datos primitivos permitidos y los colores definidos por el usuario, incluyendo los de tipo fecha.

Sintaxis: IF ( true | false | condición )

[1] Se utiliza el idioma inglés para la descripción, con el fin de mantener relación con los trabajos anteriores sobre CCPN's.

**Acciones.** Para las acciones se definió un conjunto básico. Éstas son las siguientes: ACTION_CREATE (permite crear colores compuestos en los tokens), ACTION_MODIFY (permite modificar atributos en los colores de los tokens), ACTION_PASS (permite discrimar colores), ACTION_DELETE (permite eliminar tokens), ACTION_TCOUNT (permiten modificar el valor del contador de una transición).

Sintaxis: THEN ( tipo_accion$_1$ .. tipo_accion$_n$ | regla_eca )

**Intervalos de Tiempo.** Para este trabajo se utiliza la dimensión de intervalos de tiempo válidos para las reglas ECA, con lo que es posible agregar una condición extra, la cual verifica que la ejecución de la regla ECA sea dentro de los intervalos de tiempo inicial y final especificados en esta parte de la regla.

Sintaxis: VALID TIME ( intervalo$_{inicial}$ - intervalo$_{final}$ )

### Representación de un conjunto de reglas ECA

Las siguientes reglas ECA se representan con WfCCPN en la figura 6.1.
Regla_1
**ON** ( $evento_1$)
**IF** ( $evento1.fecha_{creacion} < fechaPlazo_1$ )
**THEN** Regla_2

Regla_2
**ON** OR_SPLIT( $evento_2 : evento_3 : evento_4$ )
**IF** TRUE
**THEN** Regla_3

Regla_3
**ON** OR_JOIN( $evento_2 : evento_3 : evento_4$ )
**IF** TRUE
**THEN** Regla_4

Regla_4
**ON**( $evento_5$ )
**IF** ($evento_2.variable =$ “$aprobado$” | $evento_3.variable =$ “$aprobado$” | $evento_4.variable =$ “$aprobado$” )

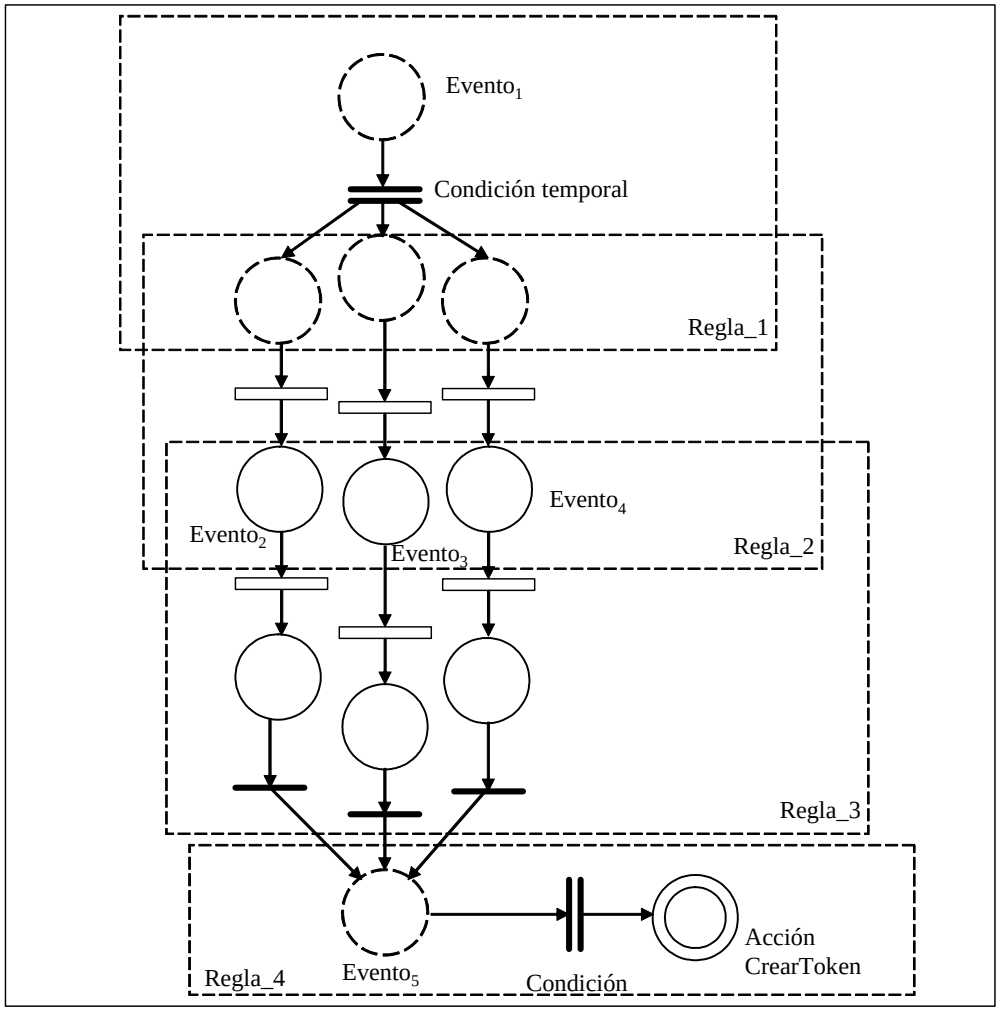


Figura 6.1: Representación de un conjunto de reglas ECA a través de WfCCPN.

**THEN** CREATE_TOKEN( $token\_final$ )

En la figura 6.1 se muestra un conjunto de cuatro reglas ECA y la forma en cómo éstas se relacionan al modelarlas con WfCCPN.

Una vez que se han identificado y representado las reglas ECA a través de la sintáxis sugerida, es posible realizar los pasos posteriores de la metodología.

### 6.1.3. Generación de la WfCCPN y uso del WfECAPNSim

Los pasos 5, 6, 7 y 8 consisten en generar (dibujar) la WfCCPN y ejecutar simulaciones de ésta a través del sistema WfECAPNSim, en el capítulo 5 se muestra de manera detallada la forma de uso del sistema, explicando la manera en como generar cada uno de los elementos que componen a una WfCCPN.

Actualmente, el sistema WfECAPNSim carece de implementaciones de software que permitan validar y verificar los modelos obtenidos.

A continuación se presenta un ejemplo, con el fin de mostrar de manera especifica el uso de la metodología.

## 6.2. Selección de artículos para un congreso científico

La organización de un congreso científico presenta un reto en cuanto a organización se refiere, son diversas las tareas que deben ser llevadas a cabo para su realización exitosa, de manera general es posible identificar los procesos principales que atañen a la organización de un evento de este tipo, a continuación se menciona una posible lista de los procesos realizados antes de la fecha del evento:

- Proceso de publicidad y difusión del evento.
- Proceso referente al aseguramiento de instalaciones y la logística.
- Proceso de selección de artículos, posters, talleres propuestos por los participantes al evento.

- Proceso de elaboración y realización del programa final del congreso.

Para este ejemplo se toma el tercer proceso para ser modelado a través de WfCCPN, se contempla únicamente el caso de selección de artículos sometidos para publicación en el congreso. A continuación se describe el proceso de workflow correspondiente.

### 6.2.1. Descripción del workflow

El proceso se inicia con la convocatoria *"call for papers"* dirigida a la comunidad científica relacionada con la temática del congreso, con el fin de que envíen en forma de publicación los resultados de sus investigaciones, en la convocatoria se estipula el formato para escribir el artículo, la fecha límite para el envió de de los artículos, la fecha para la notificación sobre aceptación o rechazo del articulo, fecha limite para envío de la versión final para publicación en caso de haber sido aceptado. También se estipula el procedimiento de envío del articulo, en ocasiones se realiza únicamente enviándolo vía correo electrónico, sin embargo generalmente es necesario enviar un formulario con datos necesarios para una mejor administración del proceso de recepción de artículos por parte de los organizadores.

Para este ejemplo se contempla el envío del registro y envío del articulo para completar el proceso de someter el articulo en su primera versión por parte del autor. La fecha límite para el envío de artículos decide si es aceptado o no el artículo para ser evaluado. Una vez llegada a esa fecha, el proceso recae en los organizadores del congreso, y se inicia la gestión para evaluar los trabajos recibidos con el fin de seleccionar los trabajos que cumplan con ciertos requisitos de calidad deseados por el comité de organizadores. Comúnmente se tiene un comité de revisores formado por personalidades con trayectorias sobresalientes dentro de la temática del congreso, de modo que expertos en los temas de los trabajos recibidos tienen la tarea de evaluarlos. Generalmente cada uno de los trabajos es enviado a tres revisores para su evaluación con el fin de que realicen los comentarios pertinentes y den su veredicto sobre si el trabajo debe ser aceptado, aceptado condicionalmente (siempre y cuando el autor realice las modificaciones sugeridas por los revisores) o rechazado. Este proceso de evaluación tiene una duración límite al estipulado como fecha de notificación de resultados a los autores. Por esta razón una vez que los tres revisores entregan sus evaluaciones, el comité organizador tiene la decisión de aceptar y rechazar artículos, en los siguientes casos el articulo es aceptado:

- al menos dos evaluaciones con resultado aceptado

- al menos una evaluación con resultado aceptado y una con resultado aceptado condicionalmente
- al menos se tienen dos resultados de aceptado condicionalmente.

Basta tener dos resultados de rechazo, para que el artículo sea descartado para su publicación en el congreso. A través de la información de contacto que se recibió en el formulario de registro, se envía la notificación a los autores, así como las observaciones realizadas por los revisores. Una vez que los autores a los cuales su artículo fue aceptado, tienen la tarea de enviar una versión final al comité organizador. Esta tarea da fin a este proceso de workflow.

### 6.2.2. Modelo del proceso de workflow

A continuación se presenta el modelo que se obtuvo a partir de la descripción del proceso genérico de selección de artículos para su publicación en congresos científicos, primero a partir de la descripción se identifican las reglas ECA, expresándolas a través de la sintaxis sugerida en el capitulo 4; una vez que se tienen las reglas se crea el conjunto de colores, es decir todos los datos que participan en el proceso de workflow, de esta manera es posible generar las estructuras de WfCCPN que permitan representar tanto las reglas ECA como los datos del proceso de workflow a través del sistema WfECAPNSim.

### 6.2.3. Conjunto de Reglas ECA

El conjunto de reglas ECA identificadas en el workflow descrito anteriormente se lista a continuación.

**Regla_1**
**ON** Inicio
**IF** TRUE
**THEN** Regla_2

**Regla_2**
**ON** AND_SPLIT(EnvioRegistro : EnvioArticulo)
**IF**
c1.timestampCreate < fechaLimite AND

c2.timestampCreate < fechaLimite

**THEN**

ACTION_CREATE(ArticuloRegistrado)

**Regla_3**
**ON** Recepción
**IF** TRUE
**THEN** AND_SPLIT(Regla_4 : Regla_5 : Regla6)

**Regla_4**
**ON** Revisor1
**IF** t.delay=true
**THEN** Regla_7

**Regla_5**
**ON** Revisor2
**IF** t.delay=true
**THEN** Regla_8

**Regla_6**
**ON** Revisor3
**IF** t.delay=true
**THEN** Regla_9

**Regla_7**
**ON** Evaluacion1
**IF** true
**THEN** ACTION_CREATE(calificacion1)

**Regla_8**
**ON** Evaluacion2
**IF** true
**THEN** ACTION_CREATE(calificacion2)

**Regla_9**
**ON** Evaluacion3
**IF** true

**THEN** ACTION_CREATE(calificacion3)

**Regla_10**
**ON**
AND_JOIN(calificacion1 : Calificacion2 : Calificacion3)
**IF**
calificacion.desicion=‘aceptado’
**THEN**
ACTION_COUNT(incremento)
ACTION_CREATE(Notificacion)

**Regla_11**
**ON** Notificacion
**IF** if ( contador.Transición.Regla_10 >= 2)
**THEN** Regla_12

**Regla_12**
**ON** OR_SPLIT (SUB_NET(Aceptado) : SUB_NET(Rechazado))
**IF** -
**THEN** -

**Regla_13**
**ON** SUBNET(Aceptado)
**IF** TRUE
**THEN** CANCEL

**Regla_14**
**ON** AND(Notificacion:Preparacion camera ready)
**IF** Articulo.timestamp_Create < Plazo_CameraReady
**THEN** Enviar Articulo al editor del congreso

**Regla_15**
**ON** Articulo final enviado al editor del congreso
**IF** TRUE
**THEN** END

**Regla_13**

**ON** SUBNET(Rechazado)
**IF** TRUE
**THEN** CANCEL

### 6.2.4. Conjunto de colores

Los datos necesarios para modelar el proceso se muestran en la figura 6.2 a través del conjunto $\sum$ de colores definidos para este proceso de workflow.

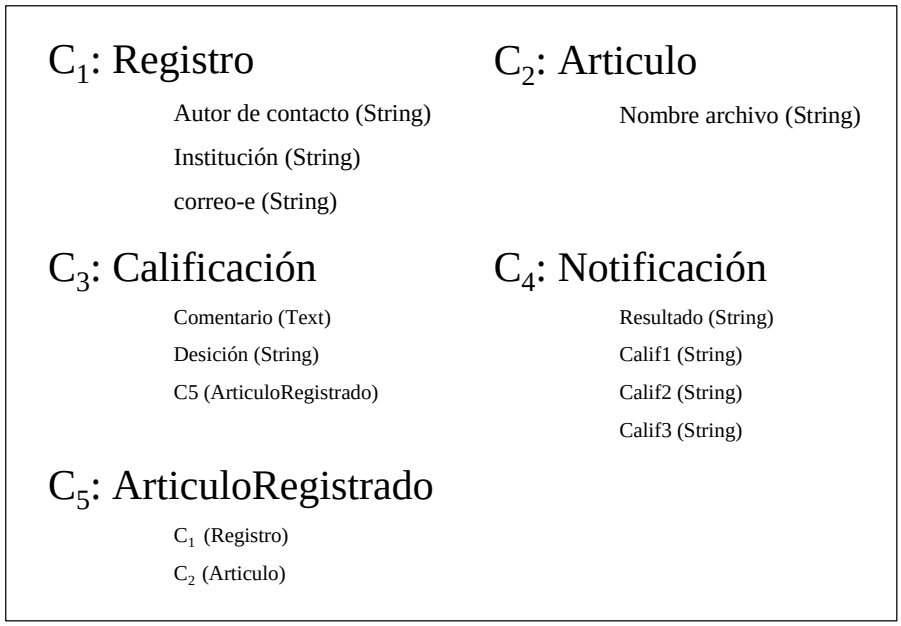


Figura 6.2: Conjunto de colores

### 6.2.5. WfCCPN obtenida

En la figura 6.3 se muestra el proceso principal modelado a través de WfCCPN. La marca inicial es:

$$M(P_0) = \{(C_1), (C_2)\}$$

El color de cada lugar se especifica dentro del círculo que representa cada lugar, mientras que una etiqueta $P_{indice}$ identifica la posición del lugar al cual nos referimos en la explicación, además una etiqueta descriptiva con formato de letras itálicas se adjunta a cada lugar y transición de la red que modela el proceso principal la cual se muestra en la figura 6.3. El proceso de workflow es iniciado a través de la regla ECA 1, la cual tiene un evento primitivo *inicio* el cual representa el momento en que un token de color $C_1$ y un token de color $C_2$ son depositados en el lugar virtual de inicio $P_0$, en donde la acción es el evento evento $AND\_SPLIT(EnvioRegistro : EnvioArticulo)$ de la regla 2, la cual a su vez posee la acción: $ACTION\_CREATE(ArticuloRegistrado)$ asociada a la transición compuesta $t_1$ será ejecutada si las estampas de tiempo de creación de los tokens que pasen a través de ella, son menores la fecha límite para el envío de artículos.

Si la acción es ejecutada el resultado será reflejado en el lugar compuesto $P_5$, es decir el token de color compuesto creado por la acción será depositado en $P_5$, representando la recepción del artículo para su evaluación por el comité revisor del congreso. La acción representada en el lugar $P_5$, es además el evento activador $AND\_SPLIT(Regla\_4 : Regla\_5 : Regla\_6)$ el cual se representa a través de la transición copia la cual envía una copia del artículo y la información de registro representadas a través del token de color $C_5$ recientemente creado; este evento representa el envío del articulo para su evaluación a tres revisores, a través de transiciones tiempo que poseen retardos para su disparo es posible modelar el tiempo[2] que le toma a cada revisor evaluar el artículo. Estas tres reglas se representan por los siguientes elementos: $Regla\_4 = \{P_6, t_3, p_9\}$, $Regla\_5 = \{P_7, t_4, p_{10}\}$ y $Regla\_6 = \{P_8, t_5, p_{11}\}$.

En los lugares compuestos $P_{12}, P_{13}, P_{14}$ se representan las tareas de revisión de los artículos y sus resultados, generados por las acciones de las transiciones de los cuales son lugares de salida, estas transiciones son $t_6$, $t_7$ y $t_8$ y corresponden a las reglas $7, 8$ y $9$ respectivamente, esas acciones crean los colores que contienen la información con el resultado de la evaluación, de esta manera los tokens son creados con el color $C_3$ que contienen la decisión de cada uno de los revisores en su atributo $C_3.decisión$.

Una vez que se ha modelado la respuesta de los revisores, es posible modelar la notificación a los autores de artículos sobre la aceptación o rechazo de sus propuestas. La regla ECA 10, representa la notificación, es decir el token de color $C_4$ que se deposita en el lugar $P_{15}$ el

[2]El tiempo que se toma cada revisor en evaluar un artículo probablemente no sea de gran relevancia para el proceso de workflow modelado, sin embargo el objetivo es mostrar la manera de modelar tareas que tienen un tiempo especifico o aleatorio para ser realizadas.

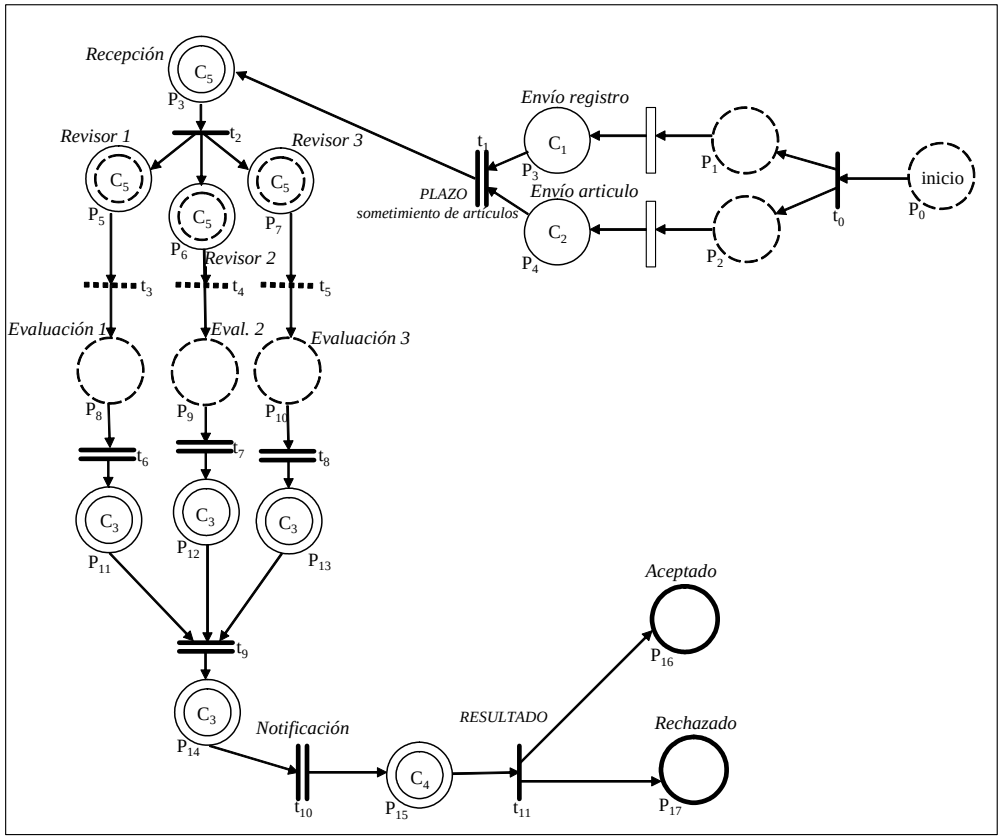


Figura 6.3: Proceso principal.

cual es creado a partir de los valores en los tokens de color $C_3$ y que llegan a la transición *compuesta* $t_9$.

El evento $OR\_SPLIT(SUB\_NET(Aceptado) : SUB\_NET(Rechazado))$ de la regla 12 representa el envío de la notificación al autor, a través de la transición *copia* $t_{11}$ se envía una copia del token con la notificación hacia los lugares de tipo *subred* $p_{16}$ y $p_{17}$ llamados *Aceptado* y *Rechazado*. Las redes en que se sigue el proceso de workflow contenidas en estos dos lugares *subred* se presentan a continuación

En la figura 6.3 se muestra el subproceso contenido dentro del lugar $P_{15}$, el cual representa la aceptación del artículo. La regla 13 representa la notificación de aceptación a través de la condición en la transición regla $t_1$, lo que provoca la espera del evento de la regla 14 el cual será activado hasta que el usuario deposite un token el cual representa la versión final del artículo, al tener un token el lugar $p_1$ y $p_2$ se disparará la regla 14, siempre y cuando la condición del plazo para el envío de las versiones finales sea cumplido por los tokens que

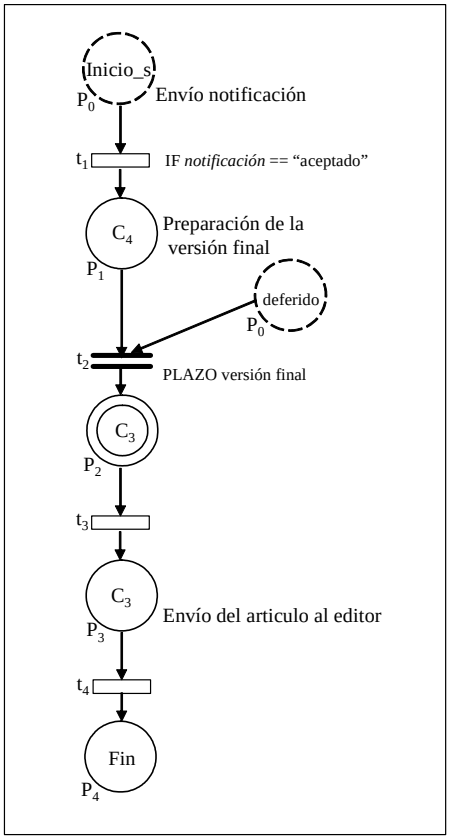


Figura 6.4: Sub proceso de aceptación.

llegan a la transición $t_2$, depositando un token de color $C_3$ el cual representa la acción de la recepción de la versión final del artículo. Una vez que se recibe el artículo se genera el evento de la regla 15, con lo cual el artículo es enviado al editor de las memorias del congreso en donde serán publicados todos los artículos, llegando así al final del proceso de workflow.

Por otro lado cuando el artículo es rechazado el subproceso Rechazado se ejecuta, simplemente enviando el mismo token hacia un lugar de cancelación y posteriormente a un lugar primitivo fin, con lo que el proceso de workflow se cancela.

En la figura 6.6 se muestra el modelo generado en WfECAPNSim.

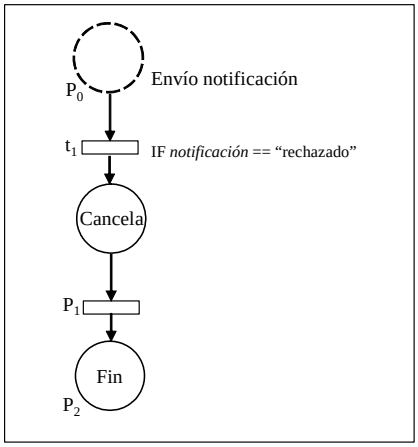


Figura 6.5: Sub proceso de cancelación del proceso de workflow

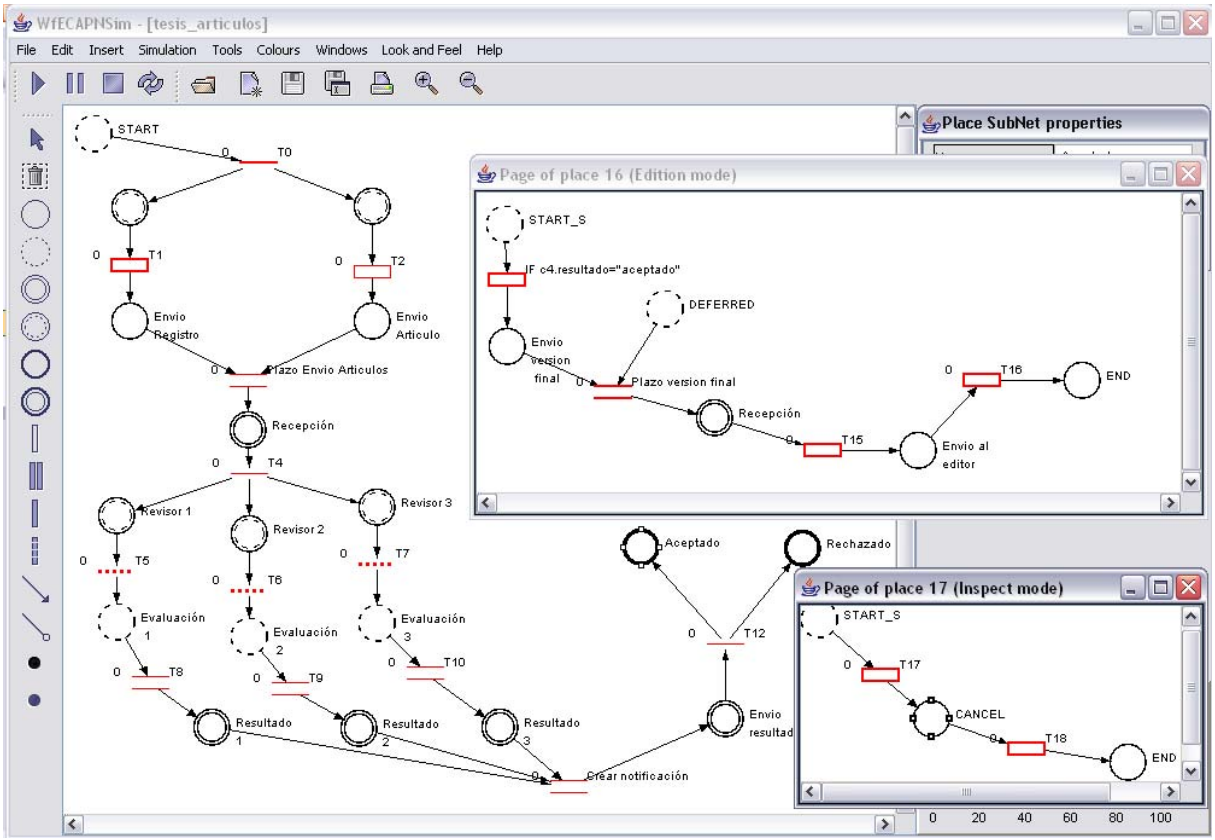


Figura 6.6: Proceso modelado en el sistema WfECAPNSim

## 6.3. Comentarios finales

Este capítulo presentó a través de un ejemplo el enfoque de modelado propuesto al utilizar las WfCCPN para modelar procesos de workflow. Como ejemplo se utilizó el conjunto de reglas ECA identificadas en el proceso general que se lleva a cabo en la selección de artículos científicos para su publicación en un congreso científico.

# Capítulo 7

# Conclusiones

En este capítulo se presenta de forma general los resultados obtenidos con la realización de este trabajo de tesis. También se presentan sugerencias referentes a posible trabajo futuro para mejorar y extender los resultados obtenidos. El capítulo se estructura de la siguiente forma: la sección 7.1 presenta los resultados obtenidos así como las aportaciones principales del trabajo realizado; la sección 7.2 expone las actividades propuestas para mejorar y extender las WfCCPN's y el sistema WfECAPNSim además los temas en que es posible continuar la investigación con respecto a la tecnología Workflow.

## 7.1. Resultados obtenidos

En este trabajo de tesis se presentó una propuesta para modelar y simular procesos de workflow a través de una extensión de redes de Petri coloreadas condicionales, con esta es posible modelar la perspectiva de flujo de control sin necesidad de abstraer datos de control como se hace en [13]. A las CCPN originales fueron agregados elementos extras a partir de lo estudiado en trabajos tales como [15], [32], [9], [10], con estos elementos se modelan los conceptos de workflow de acuerdo al modelo de referencia del WfMC (Workflow Management Coalition), además se modelan los patrones de flujo de control que han sido propuestos como marcos de trabajo que permiten evaluar metodologías de modelado para procesos de workflow, en este trabajo se implementaron dichos patrones y se presentaron en el capítulo 4, obteniendo como resultado, la posibilidad de modelar todos los patrones de flujo de control a través de WfCCPN. El hecho de que los estándares existentes y los principales WfMS que existen actualmente únicamente modelen entre once y catorce patrones (con excepción del Sistema

YAWL[1]), permite considerar como satisfactoria la aplicación de las WfCCPN en el modelado de este tipo de procesos, ya que además permite modelar de acuerdo a las definiciones de la WfMC y como parte adicional al objetivo del trabajo (modelar completamente la perspectiva de flujo de control) fue posible modelar de manera parcial la perspectiva de datos a través de los colores en los tokens de la WfCCPN. Se considera que no se tiene un soporte total para la perspectiva de datos debido a que no se utilizaron las variables de manera completa, ya que únicamente se utilizó una variable contador en cada transición de la WfCCPN, con el uso de un número n de variables en las transiciones se aumenta la flexibilidad de la WfCCPN e incluso la implementación de algunos patrones de flujo de control se realiza de manera más compacta.

Además de los constructores de enrutamiento los cuales se modelan como eventos de una regla ECA, es posible utilizar los eventos compuestos implementados con CCPN y presentados en [20], con lo que se obtiene mayor capacidad de modelado para los procesos de workflow, por ejemplo con el evento secuencia con el cual es posible establecer el orden necesario de ejecución de un conjunto de tareas en un proceso de workflow, no se contempla en los patrones de flujo de control propuestos hasta la fecha, así como tampoco se encontraron implementaciones similares en los sistemas de software estudiados.

La aplicación de reglas ECA en Workflow ha sido utilizada en el modelado, en la ejecución y en el manejo de excepciones de procesos de workflow, sin embargo son pocos los trabajos realizados a la fecha y como se menciona en [30] el enfoque de reglas ECA no ha sido muy utilizado en los WfMS comerciales debido a la dificultad que presenta el generar las reglas ECA de manera manual (gramaticalmente). Con WfECAPNSim la tarea de generar las reglas ECA que representan al proceso de workflow es realizada de manera gráfica a través de la extensión realizada a las CCPN las cuales fueron desarrolladas especialmente para la representación de reglas ECA, con lo que la dificultad de crear el conjunto de reglas ECA se facilita al tener una perspectiva gráfica en los elementos de la WfCCPN los cuáles permiten identificar de manera intuitiva las partes de una regla ECA.

A través de utilizar transiciones que toman en cuenta el tiempo, es posible retardar disparos y verificar intervalos de tiempo válidos con lo que la metodología propuesta con WfCCPN se extiende el uso de reglas ECA en su dimensión para validación de tiempos.

[1] El sistema YAWL modela 19 patrones, en esta tesis se presentó que debido a que YAWL sólo considera un punto de finalización del workflow, no es posible que este modele el patrón de terminación implícita descrito en el capítulo 4.

De los sistemas comerciales que aplican el enfoque de reglas ECA ninguno proporciona la capacidad de utilizar la dimensión de tiempos válidos de las reglas ECA [38].

Además de modelar procesos de workflow con características de tiempo, es posible modelarlos de manera jerárquica lo que proporciona la posibilidad de modelar procesos de workflow de un tamaño considerable, las dos características anteriores son necesarias para la simulación del proceso a través del juego de tokens de la red obtenida, sin embargo en cuanto a simulación puede resultar difícil la visualización de procesos grandes, debido a la restricción natural en cuanto a tamaño de visualización en una computadora.

Finalmente otro resultado y contribución del trabajo realizado es el software WfECAPNSim, el cual permite modelar procesos de workflow y ver su comportamiento a través de la simulación que ofrece el sistema, el WfECAPNSim provee características que difícilmente se encuentran en los sistemas de software gratuitos, tales como modelar características de jerarquía, de tiempo y de manejo de colores, además de simular el proceso modelado, en algunos casos se encuentra una u otra característica pero no todas, a excepción de YAWL el cual proporciona las características citadas. Sin embargo en sistemas de software comerciales es más común encontrar dichas características. De esta manera la implementación desarrollada contribuye con una herramienta de software gratuita la cual puede obtenerse y modificarse o extenderse de manera libre.

## 7.2. Trabajo futuro

El trabajo a futuro que es sugerido a partir del trabajo realizado y de los resultados obtenidos se presenta a continuación:

- Ampliar la metodología con el fin de modelar de manera total la perspectiva de datos.
- Investigar la factibilidad para que a través de WfCCPN se modele también la perspectiva de recursos (conocida también como perspectiva organizacional) y otras perspectivas de manera completa, como la de tratamiento de excepciones.
- Continuar el desarrollo del sistema WfECAPNSim con mejoras como las siguientes: Manejo de variables en las transiciones, Ampliar el conjunto de acciones, Reportes de la simulación a mayor detalle, e implementación de métodos matemáticos de análisis y validación más completos.

- Modelar procesos de workflow mucho más complejos con el fin de continuar mejorando la metodología propuesta y el sistema desarrollado.
- La investigación sobre Workflow y CCPN puede extenderse al desarrollo de un motor de workflow, el cual pueda representar y ejecutar los procesos modelados con el sistema WfECAPNSim, proponiendo así la ejecución automática de los procesos de workflow en base a la traducción de las reglas ECA, estas a través del sistema ECAPNSim con su módulo de conexión a base de datos y ejecución de reglas ECA pueden ser ejecutadas en una base de datos relacional.

La tecnología Workflow es bastante extensa, por lo que los puntos anteriores son sugeridos para su realización debido a que se consideran como factibles a partir del trabajo que ha sido desarrollado en la presente tesis y en otras tesis en las que se utilizan las CCPN's.

# Publicaciones

Artículo 1

Título: "Modelado de workflows utilizando redes de Petri coloreadas condicionales".

Congreso: 12vo. Congreso Internacional de Investigación en Ciencias Computacionales (CIICC'05), Monterrey, México, 15-17 de Noviembre de 2005.

Autores: Samuel Garrido-Daniel, Xiaoou Li, Joselito Medina-Marín.

Artículo 2

Título: "WfECAPNSim: Sistema para modelado y simulación de procesos workflow".

Congreso: Congreso nacional de Informática y Sistemas Computacionales (CONAIS 2005). Villahermosa, Tabasco, México 31 de Agosto, 1 y 2 de septiembre.

Autores: Samuel Garrido-Daniel, Xiaoou Li, Joselito Medina-Marín.

Artículo 3

Título: "Modelación y simulación de procesos workflow utilizando redes de Petri".

Congreso: I Congreso sobre Ingeniería e Investigación Científica (CONIIC 2005). Lima-Perú. 19 - 22 de Octubre.

Autores: Samuel Garrido-Daniel, Xiaoou Li, Joselito Medina-Marín.

# Bibliografía


[1] David Hollingsworth. *The Workflow Reference Model.*

[2] Sitio web del workflow management coalition. `www.wfmc.org`.

[3] The global business process management community. `www.bpmg.org`.

[4] The association of business process management professionals. `www.abpmp.org`.

[5] Portal workflow. `www.e-workflow.org`.

[6] Workflow research. `www.workflow-research.de`.

[7] Eshuis. Hendrik. *Semantics and Verification of UML Activity Diagrams for Workflow Modelling.* PhD thesis, University of Twente, Enschede, The Netherlands, 2002.

[8] Wohed. Petia, W. M. P van der Aalst, Dumas. Marlon, A. H. M. ter Hofstede, and Russell. Nick. Pattern-based analysis of uml activity diagrams. *In K. Jensen, editor, Proceedings of the Fourth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2002), volume 560 of DAIMI, pages 1–20, Aarhus, Denmark*, August 2002.

[9] van der Aalst. Wil M. P. and ter Hofstede. Arthur H. M. Yawl: Yet another workflow language. QUT Technical Report FIT-TR-2003-04, Queensland University of Technology, 2003.

[10] Stork. David and van Glabbeek. Rob. Token-controlled place refinement in hierarchical petri nets with application to active document workflow. In *ICATPN '02: Proceedings of the 23rd International Conference on Applications and Theory of Petri Nets*, pages 394–413, London, UK, 2002. Springer-Verlag.

[11] Thomas Jacob, Olaf Kummer, Daniel Moldt, and Ulrich Ultes-Nitsche. Implementation of Workflow Systems using Reference Nets – Security and Operability Aspects. In Kurt

Jensen, editor, *Fourth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, Ny Munkegade, Bldg. 540, DK-8000 Aarhus C, Denmark, August 2002. University of Aarhus, Department of Computer Science. DAIMI PB: Aarhus, Denmark, August 28–30, number 560.

[12] Zhu Li. Haiping and Zhang Guojun. Peigen. Htsn: A complex workflow model based on colored petri net. *Journal of Computer Science and Technology*, 4(1):7, April 2004.

[13] van der Aalst. Wil M. P. and van Hee. Kees. *Workflow Management. Models, Methods, and Systems.* MIT Press, Cambridge, MA, 2002.

[14] Eshuis. Hendrik and Dehnert. Juliane. Reactive petri nets for workflow modeling. In *ICATPN*, pages 296–315, 2003.

[15] Kiepuszewski. Bartosz. *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows.* PhD thesis, Queensland University of Technology, Brisbane, Australia, 2002.

[16] Workflow patterns group. The workflow patterns page. this site serves as a repository for workflow modeling patterns. `http://www.workflowpatterns.com`.

[17] Sitio web principal sobre redes de petri. `www.daimi.au.dk/PetriNets/`.

[18] Sitio web principal sobre redes de petri coloreadas. `www.daimi.au.dk/CPnets/`.

[19] Li. Xiaoou and Medina Marin. Joselito. Composite event specification in active database systems: A petri net approach. *In Fifth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools Aarhus, Denmark, DAIMI PB - 570, ISSN 0105-8517*, October 2004.

[20] Medina Marin, J. *Red de PetriColoreada Condicional (CCPN) y su Aplicación en Bases de Datos Activas.* PhD thesis, CINVESTAV-IPN, September 2002.

[21] Hollingsworth. David. The workflow reference model version 1.1. Technical Report WFMC-TC-1003, Workflow Management Coalition, January 19th 1995.

[22] G.J. Houben W.M.P. van der Aalst, K.M. van Hee. Modelling and analysing workflow using a petri-net based approach. 2(2):98–132, 1998.

[23] Kiepuszewski. Bartosz, Hofstede. Arthur, and van der Aalst. Wil. Fundamentals of control flow in workflows. *B. Kiepuszewski, A.H.M. ter Hofstede, and W.M.P. van der Aalst. Fundamentals of Control Flow in Workflows. QUT Technical report, FIT-TR-2002-03, Queensland University of Technology, Brisbane.*, 2002.

[24] Casati. Fernando, S. Ceri, B. Pernici, and G. Pozzi. An environment for designing exceptions in workflows. *Information Systems, vol. 24, no. 3, pp. 255-273,*, May 1999.

[25] Harald Störrle. Semantics and verification of data flow in uml 2.0 activities. *Elsevier*, 77(1):5, April 2004.

[26] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, Abril 1989.

[27] K. Jensen. An introduction to the practical use of coloured petri nets. In *Lectures on Petri Nets II: Applications, Lecture Notes in Computer Science*, volume 1492, pages 237–292. Springer Verlag, 1998.

[28] Lars Michael Kristensen, Soren Christensen, and Kurt Jensen. The practitioner's guide to coloured petri nets. *International Journal on Software Tools for Technology Transfer*, 2(2):98–132, 1998.

[29] Dickson K. W. Chiu, Qing Li, and Kamalakar Karlapalem. A meta modeling approach to workflow management systems supporting exception handling. *Information Systems*, 24(2):159–184, 1999.

[30] Bae. Joonsoo, Bae. Hyreim, Kang. Suk-Ho, and Yeongho. Kim. Automatic workflow processes using eca rules. *IEEE Transactions on knowledge and data engineering, Vol. 16, No. 8*, August 2004.

[31] Casati. Fernando, S. Ceri, B. Pernici, and G. Pozzi. Deriving active rules for workflow enactment. *Proc. Seventh Int'l Conf. Database and Expert Systems Applications, pp. 94-110*, 1996.

[32] Zhou. Jiantao, Shi. Meilin, and Ye, Xinming. On pattern-based modeling for multiple instances of activities in workflows. Agosto 2003.

[33] van der Aalst. Wil M. P., Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(3):5–51, 2003.

[34] Workflow research. www.manageability.org/blog/stuff/workflow$_i n_j ava/view$.

[35] Workflow research. java-source.net/open-source/workflow-engines.

[36] Sitio web del sistema woped. www.woped.org.

[37] Sitio web del sistema bossa. www.bossa.org.

[38] Robert Müller, Ulrike Greiner, and Erhard Rahm. Deriving active rules for workflow enactment. *Data Knowledge Engineering*, 14(3):5–51, 2004.