TECNOLÓGICO
DE MONTERREY

Hacemos constar que en la Ciudad de México, el día 30 de julio de 2007 el alumno:

Jesús Muñoz Bauza

sustentó el examen oral en defensa de la Tesis titulada:

Modelo basado en reglas para la representación de conocimiento de negocios

Presentada como requisito final para la obtención del Grado de:

Doctor en Administración

Ante la evidencia presentada en el trabajo de tesis y en este examen, El Comité Examinador, presidido por el Dr. Cuauhtémoc Olmedo Bustillos, ha tomado la siguiente resolución:

APROBADO POR UNANIMIDAD

Dr. José Martín Molina Espinosa
Director de Tesis

Dr. Rafael Lozano Espinosa
Lector

Dr. Cesar Augusto Coutiño Gómez
Lector

Dr. Cuauhtémoc Olmedo Bustillo
Director del Programa Doctoral

Campus Ciudad de México
Calle del Puente 222, Col. Ejidos de Huipulco
14380 Tlalpan, México, D. F. México
Tel: (52/55) 5483 2020 Fax: (52/55) 5673 2500

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS CIUDAD DE MÉXICO

MODELO BASADO EN REGLAS PARA LA REPRESENTACIÓN DE CONOCIMIENTO DE NEGOCIOS

DOCTORADO EN ADMINISTRACIÓN

TESIS PRESENTADA POR

JESÚS MUÑOZ BAUZA

ASESOR

DR. JOSÉ MARTÍN MOLINA ESPINOSA

JULIO 2007

DEDICATORIA

A mi madre Dora Alicia Bauza Pérez

Gracias por tu apoyo mamá, eres un ejemplo de perseverancia y superación.

A la memoria de mi muy respetable padre Ernesto Muñoz Trinidad

A mis hermanos Ernesto, Liliana, Milagros y Rocío por el apoyo y respeto que nos hemos dado en las buena y en las malas.

AGRADECIMIENTOS

A mi Director de tesis Dr. José Martín Molina Espinosa:

Por su aprecio y acertada orientación en el desarrollo de toda la tesis.

A mi lector y gran amigo Dr. Cesar Augusto Coutiño Gómez y su esposa Brenda Villareal Antelo:

Por todo su apoyo en la obtención de mi grado y sobretodo por su amistad brindada desde que los conozco.

A mi lector y amigo Dr. Rafael Lozano Espinosa:

Por su aprecio y guía en la terminación de la tesis.

Al Dr. Emmanuel Moya Anica

Por todo el apoyo y tiempo otorgado para la conclusión de mi grado

A mis amigos y compadres de mi tierra Carlos Rivera, Ulises Olea y Valdemar Aranda:

Por esa gran amistad a lo largo de muchos años.

Resumen

La modelación con flujos de trabajo Workflow (WF) en la administración de procesos de negocio es un área de investigación clave en la economía actual basada en el servicio. Toda empresa requiere, como ventaja competitiva, responder en tiempo real a las necesidades de los clientes, proveedores y otros actores de la cadena de valor y cumplir, como objetivo de negocio, con una plataforma de servicios ad hoc a las necesidades de clientes y proveedores. La carencia de infraestructura, cuellos de botella en la cadena de valor, la ausencia de Tecnologías de Información (TI) y la falta de conocimiento en negocios son los mayores obstáculos para cumplir con este objetivo. En esta tesis se proponen las bases para alcanzar dicho objetivo a través de un modelo para la Construcción Dinámica de Workflows (DCW) que genera redes Workflow en tiempo real para modelar procesos de forma genérica, consistente y formal, y sean la base para la representación de procesos en forma dinámica. La modelación se realiza en tiempo real de tal manera que se puedan crear los nodos del WF de manera dinámica de acuerdo a las necesidades de una PYME e integrado con un sistema de motor de reglas de negocios permite adaptar el WF de acuerdo a la red que se esté representando. Los WF generados son consistentes y correctos de acuerdo a cinco patrones básicos, cuatro propiedades y un proceso general de construcción dinámico desarrollados con base en dos formalismos, la modelación visual con redes de Petri y la modelación textual con gramáticas de contexto libre. Para implantar el modelo se utiliza un cuestionario integrado en un sistema de información denominado Diagnósticador Interactivo Asesor y Consultor de Negocios Electrónicos (DIACNE). El cuestionario se basa en un conjunto de factores críticos para la detección de áreas de oportunidad en las PYMEs para su incursión en los negocios electrónicos. Cada WF construido es específico a las respuestas de cada empresario.

Tabla de contenido

Lista de Figuras

Lista de Tablas

Capítulo 1 Introducción

1.1 Contexto

El conocimiento completo de los procesos de negocio de una empresa permite su mejora continua en su planeación y toma de decisiones. Los sistemas basados en Workflow son herramientas clave para la administración de dichos procesos e involucra el conocimiento de expertos en el área de tecnologías de información y de expertos en el dominio de procesos de negocio. La conjunción de estas dos áreas permite generar soluciones eficientes en la administración de procesos y por lo mismo es indispensable conocerlas y dominarlas. En este capítulo se da una introducción a la administración WF y a los conceptos clave pertenecientes a las áreas de conocimiento con los que está relacionado directamente en la actualidad.

1.1.1 Workflow

Tradicionalmente los negocios y específicamente los procesos internos de los negocios, se diseñan y controlan mediante flujos de trabajo. El flujo de trabajo denominado Workflow (WF) está enfocado en la automatización de procedimientos dónde documentos, información o tareas son transferidos entre participantes de acuerdo a un conjunto predefinido de reglas para alcanzar o contribuir a un objetivo de negocio (WfMC 2006). Dicho objetivo se considera como el resultado que la capacidad de negocio quiere cumplir con respecto a su misión, por ejemplo: mantener satisfechos a sus clientes (Ross, 2003). Actualmente existe una organización sin fines de lucro que desarrolla y publica estándares WF, incluyendo un modelo de referencia WF y las especificaciones para sus interfaces. Dicha organización denominada Coalición para la administración WF (WfMC por sus siglas en inglés) es un grupo de investigadores participantes encabezado por Jon Pyke y que tiene más de 300 organizaciones miembro, incluyendo BEA, FileNet, Fujitsu, Hitachi, IBM, NECSoft, Oracle, Sun, TIBCO (Staffware), Toshiba, Vigente, Vitria y WebMethods. La Coalición tiene como objetivo identificar áreas funcionales y desarrollar especificaciones apropiadas para la implantación de productos WF, también, producen bastante literatura sobre la Modelación de Procesos de Negocio denominada BPM por sus siglas en inglés. La

Coalición (WfMC-URL 2007) define WF como *la automatización computarizada de una parte o completa de un proceso de negocio*.

Los WF son clasificados por la WfMC como colaborativos, administrativos y orientados a producción. Los WF colaborativos centralizan la información, son descritos como herramientas de trabajo cooperativo soportado por computadora (CSCW por sus siglas en Inglés), software para trabajar en grupo (*groupware*) y otras herramientas de compartición de datos. Los WF administrativos involucran procesos administrativos tales como órdenes de compra, hojas de tiempos y movimientos, reportes de gastos, cambios de órdenes, reportes de calidad y muchas otras actividades que traspasan las barreras departamentales e inclusive de la empresa misma. Los WF de producción son de naturaleza altamente repetitiva, muy estructurados y sirven para definir y administrar líneas de producción de productos. Los WF pueden ser dinámicos o estáticos, los estáticos son de naturaleza repetitiva y no es posible realizar cambios en su esquema o debe ser fuera de línea sin instancias en operación, en cambio los dinámicos deben soportar el problema de administrar instancias modificadas con respecto al esquema y/o el problema de modificar el esquema y migrar todas las instancias que se están ejecutando. En los dos casos se requieren sistemas que administren los WF (WFMS) y es importante en el caso de los dinámicos que dicho sistema proporcione los mecanismos para adaptar correctamente las instancias evitando en lo posible abortarlas. Los sistemas que administran los WF proporcionan la automatización procedural de un proceso de negocio mediante la administración de la secuencia de actividades de trabajo y la invocación de recursos de Tecnologías de Información (TI) y/o humanos apropiados asociados con los pasos de las actividades. Un WFMS, es definido por la coalición, como *un sistema que completamente define, administra y ejecuta WF a través de la ejecución de SW cuyo orden de ejecución es administrada por una computadora a través de la representación de la lógica del WF*.

1.1.2 Procesos de negocio

Los procesos de negocio consisten de un número de actividades que requieren llevarse a cabo y un conjunto de condiciones que determinan el orden de las tareas. Un proceso de negocio se puede definir como el flujo o progresión de actividades –dónde

cada una de las cuáles representa el trabajo de una persona, un sistema interno o el proceso de una compañía socia- hacia un objetivo común. En el contexto computacional un proceso de negocio es un algoritmo para alcanzar un objetivo de negocio y cada paso del proceso se denomina actividad o tarea. Las actividades pueden ser automatizadas o manuales, una automatizada es directamente ejecutada por el motor de ejecución WF de un WFMS y una manual es ejecutada por una persona participante en el proceso. Lo ideal es que la ejecución de todas las actividades de un proceso sea automatizada. Formalmente un proceso de negocio es: una capacidad de negocio que a partir de materias primas en un cierto estado como entrada se transforman en alguna forma con valor agregado como salida. Las entradas y salidas pueden ser tangibles (por ejemplo, productos o recursos físicos) o intangibles (por ejemplo, información) (Ross 2003). Dentro del contexto de negocios, la salida producida o modificada del proceso se denomina caso o instancia del proceso, de este modo, un caso o instancia puede ser tangible o intangible, tiene un inicio y un fin en un proceso, y puede distinguirse de los otros casos.

La transformación de casos, en un proceso de negocio, se clasifica típicamente como:

- Físicas, por ejemplo, de materias primas a un producto final.
- De localidad, por ejemplo, servicio de transporte de una aerolínea.
- Transaccional, por ejemplo, dinero en acciones.
- Información, por ejemplo, datos financieros en información financiera.

Los procesos de negocio se pueden jerarquizar en:

- Individuales: son llevados a cabo de forma individual no grupal ni departamental.
- Funcionales o verticales contenidos dentro de una unidad o departamento.
- Inter-funcional u horizontal que incluye varias unidades funcionales.
- Inter-organizacional que incluyen varias organizaciones.

Un proceso de negocio puede tener un ciclo de vida desde minutos a días, incluso meses, dependiendo de su complejidad y de la duración de las actividades que lo

constituyen. Los WFMS que automatizan procesos de negocio son esencialmente conjuntos de herramientas de desarrollo que definen, administran y ejecutan WF, y por lo mismo, se pueden implantar de distintas formas, usar una amplia variedad de infraestructuras y tecnologías de información, y operar en ambientes desde pequeños locales a inter-empresariales. A pesar de esta variedad, todos los WFMS tienen características comunes que proporcionan bases para desarrollar capacidades de integración y de interoperabilidad entre los distintos productos. El modelo de referencia de la coalición describe un modelo común para la construcción de WFMS, ver capítulo 2 y (WfMC-URL 2007).

1.1.3 BPM

En la actualidad muchos vendedores de herramientas WF y ahora exponentes líderes de modelación de procesos de negocio (BPM) utilizan intercambiablemente los términos WF y BPM (Pyke 2006). BPM es una disciplina de administración basada en un conjunto de tecnologías, sus conceptos habilitan a los usuarios a definir estrategias, establecer metas y objetivos para mejorar los procesos operacionales de forma correcta a través de la organización. La idea detrás de BPM es que aunque las teorías de administración de procesos son maduras, su aplicación y uso de la tecnología no lo es (Pyke 2006), BPM atañe a aplicaciones orientadas a procesos, tal y como sus siglas significan, concierne a la modelación de procesos de negocio usando representaciones estándar gráficas y en XML (Extensible Markup Language), como un flujo de actividades. Gracias a estos estándares, la semántica del flujo puede ser definida como un lenguaje de programación, por ejemplo, sin inconsistencias, flujos que no terminan, redundancia. BPM, denominado también administración de procesos de negocio, se utiliza para el diseño y ejecución de procesos. Los procesos se construyen para interactuar como servicios, coordinándolos a nivel de una organización (orquestación) o a nivel colaborativo con otros procesos de negocio de otras compañías (coreografía). La tabla 1.1 (Havey 2005) presenta una lista de referencia de estándares de BPM, y muestra cómo engloba la tecnología WF y las arquitecturas orientadas a servicio (SOA). En la primera columna se indica el estándar, la segunda la organización que lo representa, la tercera la descripción del estándar y la última para qué está orientado cada uno.

Workflow Reference Model	WfMC (http://www.wfmc.org)	Describe los componentes e interfaces de una arquitectura WF/BPM.	Modelo de la Arquitectura WF.
Workflow API (WAPI)	WfMC	Es un API con definiciones en los lenguajes C, CORBA IDL y COM Automation (Havey 2005)	Administración y monitoreo
XML Process Definition Language (XPDL)	WfMC	Lenguaje de definición de procesos en XML. Competencia de BPEL.	Lenguaje de ejecución
Workflow XML (WfXML)	WfMC	Lenguaje XML para comunicación basada en servicios web entre motores WF en ejecución.	Coreografía
Web Services Choreography Interface (WSCI)	W3C (http://www.w3.org)	Lenguaje XML para coreografía de servicios Web.	Coreografía
Web Services Choreography Description Language (WS-CDL)	W3C	Lenguaje oficial de W3C para coreografía de servicios Web.	Coreografía
Web Services Conversation Language (WSCL)	W3C	Lenguaje básico XML con un enfoque elegante para coreografía de servicios Web. Una conversación es un diálogo entre participantes basado en servicios Web.	Coreografía
Business Process Execution Language (BPEL)	OASIS (http://www.oasis-open.org)	Lenguaje XML, más popular, para la definición y ejecución de procesos.	Lenguaje de ejecución
Business Process Modeling Language (BPML)	Business Process Modeling Initiative (BPMI) (http://bpmi.org)	Lenguaje XML de definición de procesos, similar a BPEL, que describe la representación estructural de un proceso y la semántica de su ejecución.	Lenguaje de ejecución
Business Process Modeling Notation (BPMN)	BPMI	Lenguaje gráfico destinado a analistas de negocios y desarrolladores para construir diagramas de procesos de negocio. Es para diseño gráfico mientras que BPML y BPEL son para ejecución.	Lenguaje de notación
Business Process Definition Metamodel (BPDM)	Object Oriented Group (OMG) (http://www.omg.org)	Proporciona un modelo abstracto para definir procesos de negocio usando MDA (Model Driven Architecture).	Lenguaje de ejecución y/o notación
Business Process Runtime Interface (BPRI)	OMG	Es un API en MDA de interfaces independientes de plataforma para que los procesos de negocio sean iniciados, monitoreados y controlados.	Administración y monitoreo
XLANG	Microsoft (http://www.microsoft.com)	Es uno de los primeros lenguajes XML de definición de procesos. Influyó en el diseño de BPEL.	Lenguaje de ejecución
Web Services Flow Language (WSFL)	IBM (http://www.ibm.com)	Lenguaje primario XML de definición de procesos que también influyó en el diseño de BPEL. Es un enfoque de grafos dirigidos.	Lenguaje de ejecución
Business Process Specification (BPSS)	OASIS	ebXML BPSS: lenguaje de definición de procesos para colaboración negocio-a-negocio.	Coreografía

Tabla 1.1 Lista de estándares de BPM.

1.1.4 WF y la cadena de valor

La tecnología WF es un estándar que está evolucionando (Weske 2001) (Aalst 2004) (WfMC-URL 2007), muchos procesos de negocio dependen de ella y de Internet, y se han transformado y diversificado en disciplinas de red (Marinescu 2002), como consecuencia, el alcance de la administración WF se ha ampliado, sus fundamentos y

tecnologías para la automatización de procesos de negocio virtualmente abarcan todas las áreas desde la ciencia e ingeniería hasta el entretenimiento (Marinescu 2002) (Girault and Valk 2003) (Kalakota and Robinson 2003) (Shafiq et al. 2005). El comercio y los negocios electrónicos son probablemente los ejemplos más representativos y tradicionales de aplicaciones de red que requieren alguna forma de administración WF, permiten que los clientes de las empresas ordenen sus productos por medio de Internet y que los desarrollen bajo demanda. Para que esos productos sean diferenciados y de bajo costo se deben ejecutar una serie de actividades, las diferentes funciones que se realizan para cumplir con dichas actividades se denominan cadena de valor de la empresa y puede representarse por una red WF. La cadena de valor categoriza las actividades que producen valor agregado en una organización y fue descrita y popularizada por Michael Porter en su "best-seller" de 1986: "Competitive Advantage: Creating and Sustaining Superior Performance. New York, NY The Free Press". El objetivo es maximizar la creación de valor para el cliente minimizando los costos incurridos a través del uso de TI.

1.1.5 Integración de aplicaciones empresariales (EAI)

Inicialmente el gasto en TI fue usado para procurar que las aplicaciones empresariales administren la información relacionada con la cadena de valor, dichas aplicaciones empresariales se categorizan por los tipos de información e interacción que ellos administran, por ejemplo, Customer Relationship Management (CRM), Suply Chain Management (SCM) and Enterprise Resource Planning (ERP) son las categorías tradicionales de aplicaciones empresariales (Kalakota and Robinson 1999, 2003) (Tsai 2003) (Laguna and Marlund 2004). Dichas aplicaciones empresariales ya posicionadas, la TI invirtió y las conjuntó en iniciativas de integración de aplicaciones (EAI) las cuáles, permiten a los negocios incrementar su inversión en SW empresarial proporcionando la infraestructura para permitir el compartir los datos a través de la organización, sistemas y aplicaciones, además, mejoran la visibilidad y el flujo de información dentro de la organización, mejorando la respuesta (*responsiveness*) de la empresa a las demandas del mercado con respecto a los sistemas anteriores. A pesar de esto, EAI tiene desventajas para administrar la cadena de valor, tales como el costo de la infraestructura, su complejidad por integrar distintas plataformas, no es flexible y es propietaria de los

vendedores (Marks and Bell 2006), lo que consecuentemente inhibe la efectividad en el negocio y en la TI.

Actualmente el concepto de cadena de valor se ha extendido más allá de las organizaciones individuales y también se aplica a cadenas de suministro completas, por ejemplo, red de instalaciones y medios de distribución que tiene por función la obtención de materiales, transformación de dichos materiales en productos intermedios y productos terminados y distribución de estos productos terminados a los consumidores. La puesta a disposición de productos y servicios al consumidor moviliza distintos actores económicos, cada uno de los cuales administra su cadena de valor. La interacción sincronizada de esas cadenas de valor crea una cadena de valor ampliada que puede llegar a ser global, el objetivo, adoptado por muchos estrategas en la administración, es capturar el valor generado a lo largo de la cadena. Las compañías pueden intentar superar los intermediarios creando nuevos modelos de negocios mediante la explotación de información que se dirige hacia arriba y hacia abajo dentro de la cadena, esto implica, que la visión tradicional de integración en los negocios electrónicos está cambiando de vincular una infraestructura EAI costosa, rígida y propietaria, a la visión emergente de integración que es una composición flexible de procesos, aplicaciones e infraestructura denominada arquitectura orientada a servicios (SOA) y forma la base de los portales empresariales actualmente.

1.1.6 SOA

Una SOA es un enfoque de cómo la funcionalidad de TI puede ser planeada, diseñada y entregada como servicios electrónicos de negocios en forma modular para alcanzar beneficios de negocios específicos vía herramientas modernas y estandarizadas (Marks and Bell 2006). Un servicio electrónico es el componente primario de un SOA, es una pieza de cómputo empaquetada como un ingrediente re-usable tal como un servicio Web o uno ofrecido por la empresa, y cada servicio puede usar otros servicios, encapsular una actividad individual, un subproceso o la lógica entera de un proceso de negocio completo. De modo inverso, un proceso de negocio puede verse como una serie de servicios orquestados dentro del flujo de trabajo del proceso. Ambos casos pueden representarse por un WF, dónde el modelo de servicios puede consistir de una cadena de

valor y asegurar características de interoperabilidad, reusabilidad, flexibilidad e integración entre todos los procesos de negocio y todas las plataformas tecnológicas, incluyendo sistemas tradicionales (legacy), gracias a su estandarización aceptada por todos los vendedores (Erl 2005) (Marks and Bell 2006).

SOA no es un concepto nuevo, resurgió con el advenimiento de los servicios Web, sus características facilitan la expansión de negocios, posibilitan integrar aplicaciones y sistemas, sin importar la plataforma en que operen, aprovechando las ventajas del Web y el uso de estándares abiertos a fin de crear macro sistemas en los que la información aparezca de manera consistente, los procesos sean flexibles y la organización tenga la habilidad para crear nuevos servicios a la medida a partir del cúmulo de información de que dispone. SOA parte de la premisa que a los usuarios no les importa la tecnología, si no los servicios, el costo y su uso efectivo, y en una organización sus beneficios clave son poder cambiar la infraestructura tecnológica sin cambiar los servicios y/o poder cambiar los procesos de negocio sin cambiar los sistemas TI. Actualmente, el mercado ya está anticipando su tendencia, ver tabla 1.2 (Kalakota and Robinson 2003) (Ross 2003) (Albert t. al. 2005) (Erl 2005) (Marks and Bell 2006).

Compañía	Servicios
IBM	Web-Sphere and e-business on Demand
Sun Microsystems	Sun ONE and Services on Demand
Microsoft	.NET Application Servers and Biztalk Server
SAP	NetWeaver and xApps
Siebel	UAN
PeopleSoft	AppConnect
BEA	WebLogic Integration
Oracle	Oracle Service-Oriented Architecture Suite and Oracle BPEL

Tabla 1.2 Vendedores actuales de SOA y servicios que ofrecen.

1.1.7 Planteamiento del problema

Como la tecnología provee sólo una ventaja temporal, se requiere constantemente evolucionar, los requerimientos de los usuarios así lo demandan y las compañías que tengan esta tendencia de servicios, no por ser las primeras significan que estarán ahí en el futuro. La tendencia de la digitalización está avanzando, primero el comercio electrónico después los negocios electrónicos y ahora los servicios electrónicos, las aplicaciones evolucionan en paralelo de aplicaciones empresariales a portales web y luego a plataformas de servicios. Para que una compañía conserve la ventaja competitiva deberá

tener en sus procesos, servicios electrónicos funcionales, consistentes, elegantes en su diseño, justos en precio y amigables de usar.

En la actualidad, la administración no está interesada en la tecnología que se ofrece si no en el valor y el futuro derivados de su inversión, esto converge completamente con el concepto de digitalización de servicios la cuál no está centrada en la tecnología sino en capturar su valor a través de su rendimiento y una productividad mejorada. El ambiente dinámico de la economía actual presiona a las empresas a tener necesidad de:

- Agilidad en los negocios: responder a cambios y amenazas de su entorno
- Flexibilidad en sus arquitecturas de TI: mayor independencia entre procesos de negocio-TI
- Soporte adaptivo en sus procesos de negocio que los permitan cambiar de forma correcta ante reingenierías de procesos.
- Tendencia en la composición de sus servicios, que les permitan compartir recursos, ahorrar costos y diferenciar los mismos, habilitados por la tecnología.

Estas necesidades permiten identificar un problema que se plantea con la siguiente pregunta en esta tesis:

- ¿Es posible modelar de manera formal los procesos de una empresa de tal forma que puedan adaptarse en forma consistente en un ambiente dinámico?

Las organizaciones actuales encaran los retos de la globalización y ambientes dinámicos que guían a niveles de competencia sin precedentes. En mercados globales muy competitivos las organizaciones necesitan siempre mejorar sus soluciones de negocios con estructuras confiables y flexibles. La mayoría de dicha infraestructura es administrada por sistemas de información que directa e indirectamente soportan sus procesos de negocio y son de importancia crucial para su crecimiento y supervivencia.

La administración eficiente de los procesos puede mejorar la ventaja competitiva mediante la reducción de costos, mejoras de producto y de servicios a sus clientes.

Toda empresa requiere que sus procesos de negocio se ejecuten de una manera correcta, es decir, que inicien, que terminen, que no existan bloqueos en sus actividades, saltos entre ellas, redundancias ni actividades invocadas inapropiadamente. Es de capital importancia resolver la pregunta anterior siendo tema de investigación para estos próximos años (Hamadi, and Benatallah 2003) (Kalakota and Robinson 2003) (Doshi et al. 2004) (Albert et al. 2005) (Pankratius and Stucky 2005) (Shafiq et al. 2005). El diseño e implantación de métodos de solución requiere de la integración de conocimiento de dos grupos disjuntos con necesidades de convergencia cada vez más altas: expertos en TI y expertos en el dominio de procesos de negocio, Internet es una red bastante amplia y heterogénea, dinámica, las empresas que las usan para el desarrollo de sus portales y sus servicios bajo demanda requieren que sea lo más consistente posible. En el caso de las PYME que no tienen enlaces dedicados requieren del uso de la infraestructura de Internet con sus respectivas ventajas, tendencia a bajo costo, globalizada, promueve la tecnología; y desventajas, principalmente no se tiene acceso a su infraestructura, lo cuál puede generar que las aplicaciones tales como portales, servicios, encuestas puedan pasmarse o no ejecutarse.

Actualmente, para un negocio, el refinamiento de sus procesos y servicios significan satisfacción del cliente, posicionamiento, respuesta rápida al mercado, diferenciación y bajos costos (Marks and Bell 2006). Por ello requerimos de herramientas formales que modelen y permitan representar procesos de forma dinámica que permitan adaptarse a cambios en el entorno de tal manera que la administración y los clientes no se ocupen de cómo se implantaron sino del valor que pueden obtener de ellos (Kalakota and Robinson 2003) (Erl 2005) (Marks and Bell 2006).

1.2 Objetivos de la tesis

- Diseñar un modelo que permita codificar procesos de manera formal y rigurosa para la construcción de WF dinámicos.
- Definir una gramática de contexto libre BNF con acciones semánticas para la representación de redes WF BNF.

- Construir WF en forma dinámica con adaptación de instancias mediante los formalismos visual y textual.
- Diseñar e integrar un sistema experto en el modelo de la construcción dinámica que permita representar el WF del proceso representado.

1.3 Descripción de los capítulos siguientes

En el capítulo 2 se detalla el flujo de actividades para procesos de negocio denominado Workflow, base de modelación de este trabajo de tesis, se presentan sus características y el modelo estándar actualmente propuesto. También se presenta el estado del arte actual de las áreas de conocimiento relacionadas a esta tesis, se incluye una comparación de los modelos y metodologías que existen en la actualidad.

En el capítulo 3 se detalla la selección del formalismo de modelación a utilizar en esta tesis con base en una comparación de los que existen actualmente y que fueron descritos en el capítulo anterior. Además, se detalla la metodología de investigación, justificación, las aportaciones y los resultados esperados del desarrollo de la presente tesis.

En el capítulo 4 se desarrolla el modelo para la construcción dinámica con los formalismos seleccionados en el capítulo 3. Dicho modelo denominado DCW se describe paso a paso mediante cinco patrones básicos, cuatro propiedades y un proceso general de construcción dinámica.

En el capítulo 5 se presenta la aplicación con la cuál se implanta el modelo, para ello se migra la aplicación a un sistema experto desarrollado con un entorno denominado *Jess* de modo que puede modelarse y ejecutarse con DCW.

Finalmente se presentan las conclusiones obtenidas con base en los objetivos planteados.

Capítulo 2 Workflow en la Actualidad

2.1 Introducción

El cambio en el mercado global es lo único constante y genera una clara e imperativa necesidad de mejorar continuamente los procesos fundamentales para el éxito de una organización. Las tecnologías de información que se enfocan en la administración y mejora de los procesos de negocio son herramientas importantes para que las organizaciones cumplan sus objetivos y mejoren sus posiciones competitivas. Los sistemas de administración WF (WfMS) son herramientas que administran y mejoran los procesos y si se utilizan adecuadamente pueden mostrar una imagen completa de lo que pasa en una organización. El conocimiento, la definición y la administración de los procesos permiten una mejora continua en la planeación y toma de decisiones. Los WfMS son una herramienta clave para las organizaciones que están motivadas en mejorar sus ventajas competitivas, servicios a clientes, productividad y su adhesión con estándares.

En este capítulo se describen los WfMS, sus usos y sus interacciones para la administración de los proceso de negocio de una organización. Posteriormente se da una descripción completa del estado del arte de los temas relacionados a esta tesis.

2.2 Flujos de actividades para proceso de negocio: Workflow (WF)

Un WfMS está compuesto de un conjunto de aplicaciones y herramientas que permiten la definición, creación y la administración de las actividades asociadas con procesos de negocio. Los WfMS están basados en el concepto de un WF definido como una abstracción de un proceso de negocio. Los WF pueden ser organizados manualmente o dentro del contexto de un sistema TI que proporcione automatización del proceso de negocio. Un WF normalmente consiste de un número de pasos lógicos denominados tareas, dependencias entre ellas, reglas de ruteo y participantes. Una tarea puede requerir la participación de personas o puede ejecutarse automáticamente por una aplicación TI. Un WfMS lee, automatiza, procesa y administra WF coordinando su información. Durante el procesamiento, las tareas, la información y los documentos son transferidos de

un participante a otro de acuerdo a un conjunto de reglas, rutas y roles, y la automatización incrementa la eficiencia de la ejecución de un caso en el proceso (Aalst and Hee 2002). Todo esto implica que la administración y el análisis de los casos de un WF proporcionan la oportunidad de medir parámetros del proceso de negocio con el fin de una mejora continua.

Las razones principales por las que las organizaciones utilizan sistemas WF son:

- Eficiencia y estandarización de sus procesos e implica una reducción de costos debido a que :
    - La estandarización de los procesos de una empresa permite un mayor conocimiento de los mismos y conduce a obtener una mejor calidad.
    - Control de los procesos.- es posible monitorear el estado de los componentes del proceso e identificar cuellos de botella.
- Asignación correcta de tareas mediante la definición de roles.
- Recursos disponibles cuando se requieran.
- Diseño de procesos de manera formal y correcta.

El sistema WF permite automatizar diferentes aspectos del flujo de información ruteando los casos en la secuencia correcta, administrar la ejecución de los mismos y proveyendo acceso a datos y documentos. Un sistema WF consta de distintos elementos, a continuación se da una descripción de los principales:

Caso

El objetivo primario de un sistema WF es la ejecución de instancias denominados casos, y se considera como el trabajo que se tiene que ejecutar cuando es creado. Ejemplos de casos son reclamo de un seguro, una deducción de impuestos o un paciente en un hospital. Cada caso tiene una identidad única, datos asociados y puede tener una trayectoria distinta a través de la organización prescrito por el WF. Todo caso tiene un estado y consiste de tres elementos: (1) atributos, usados para administrarlo, por ejemplo, que una tarea pueda omitirse si no se cumple un cierto valor, (2) condiciones, para saber el progreso de un caso, por ejemplo, qué tareas ya se llevaron a cabo y cuáles faltan para

ejecutarse (p.ej. orden aceptada, aplicación rechazada, etc..) y (3) contenido, se refiere a la información en documentos, archivo y bases de datos que no son administrados por el sistema WF. En general, el sistema WF no contiene detalles sobre el contenido de un caso, sólo de sus atributos y condiciones.

Tarea

Una tarea es una unidad lógica de trabajo, es indivisible y depende del contexto en el que está definida. La tarea se refiere a una pieza genérica de trabajo y puede ser manual, semi-automática o automática. Las tareas manuales las ejecuta una persona, las automáticas las ejecuta una aplicación TI y las semi-automáticas incluyen personas y aplicaciones. La combinación de una tarea y un caso se denomina elemento de trabajo (*work item*). El elemento de trabajo desaparece en el momento que inicia su ejecución.

Actividad

Una actividad es un paso/proceso discreto ejecutado por una persona, una aplicación o una máquina. Una actividad es la ejecución y el rendimiento de una tarea o servicio, a veces se considera como la ejecución de una o más tareas. Una tarea puede considerarse como estática y al momento de que un caso se crea, la ejecución de la tarea como parte del proceso de ese caso se considera como actividad. La figura 2.1 muestra la relación entre tarea, caso, elemento de trabajo y actividad (Aalst and Hee 2002).

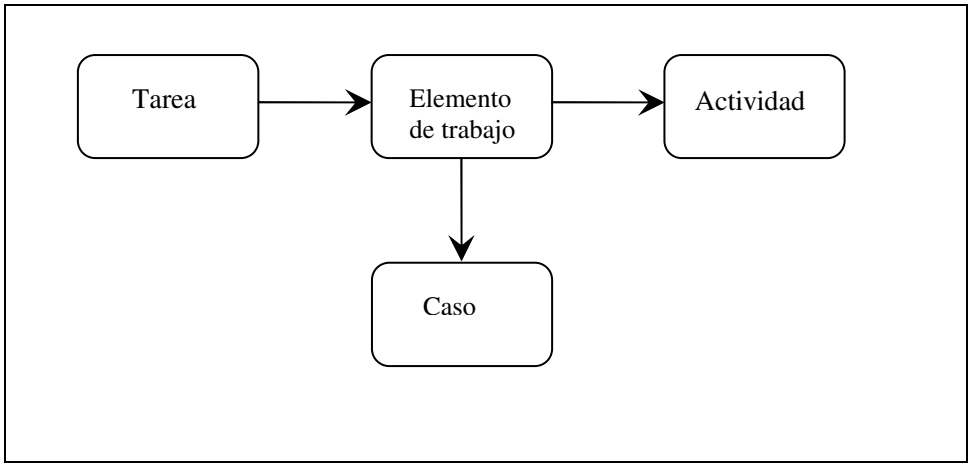


Figura 2.1 Relación entre tarea, caso, elemento de trabajo y actividad.

Proceso

El proceso indica la forma en que una categoría de casos deben llevarse a cabo, qué tareas deben ejecutarse y en qué orden. Un proceso también se denomina procedimiento para un conjunto de casos en particular, dichos casos pueden ser distintos entre ellos con base en sus atributos. En general, un proceso administra muchos casos, ya sea uno por uno o todos a la vez. El orden en el cuál las tareas de un proceso se ejecutan depende del caso y de las condiciones dentro del proceso, por lo mismo, un proceso generalmente es construido de tareas y condiciones. El ciclo de vida del caso esta regido por el proceso, dónde cada caso tiene un principio y una terminación. A su vez todo caso debe tener un ciclo de vida finito.

Ruteo

El ruteo determina qué tareas de un caso requieren ejecutarse y en qué orden. El ruteo de casos en un WF hace uso de cuatro construcciones básicas:

- Secuencia.- es la forma más simple de ruteo y consiste en la ejecución de una tarea después de otra. Usualmente existe dependencia entre ellas.
- Paralelo.- cuando dos o más tareas pueden ejecutarse simultáneamente o en cualquier orden sin que afecte a la otra.
- Selección.- cuando hay una selección entre dos o más tareas. La selección depende del caso.
- Iteración.- cuando se requiere ejecutar una o más tareas más de una vez.

La figura 2.2 muestra un ejemplo de un proceso para el ensamblaje de una computadora o una laptop según se requiera. En la figura 2.2a) se muestra un ejemplo de una trayectoria de selección cuando se pregunta si se va ensamblar una computadora o una laptop dependiendo de lo que se requiere se ejecuta cualquiera de los dos subprocesos *Ensamblar PC* o *Ensamblar laptop*. La figura 2.2b) muestra el subproceso para *Ensamblar laptop*, todas las actividades están en forma secuencial, los círculos mostrados a la derecha indican los estados del subproceso. Por ejemplo, en el estado *A* la tareas *Examinar orden* está lista para iniciar su ejecución como resultado de ello el

sistema se trasferirá al estado *B* cuando la tarea *Obtener componentes* se encuentre lista para ejecución.

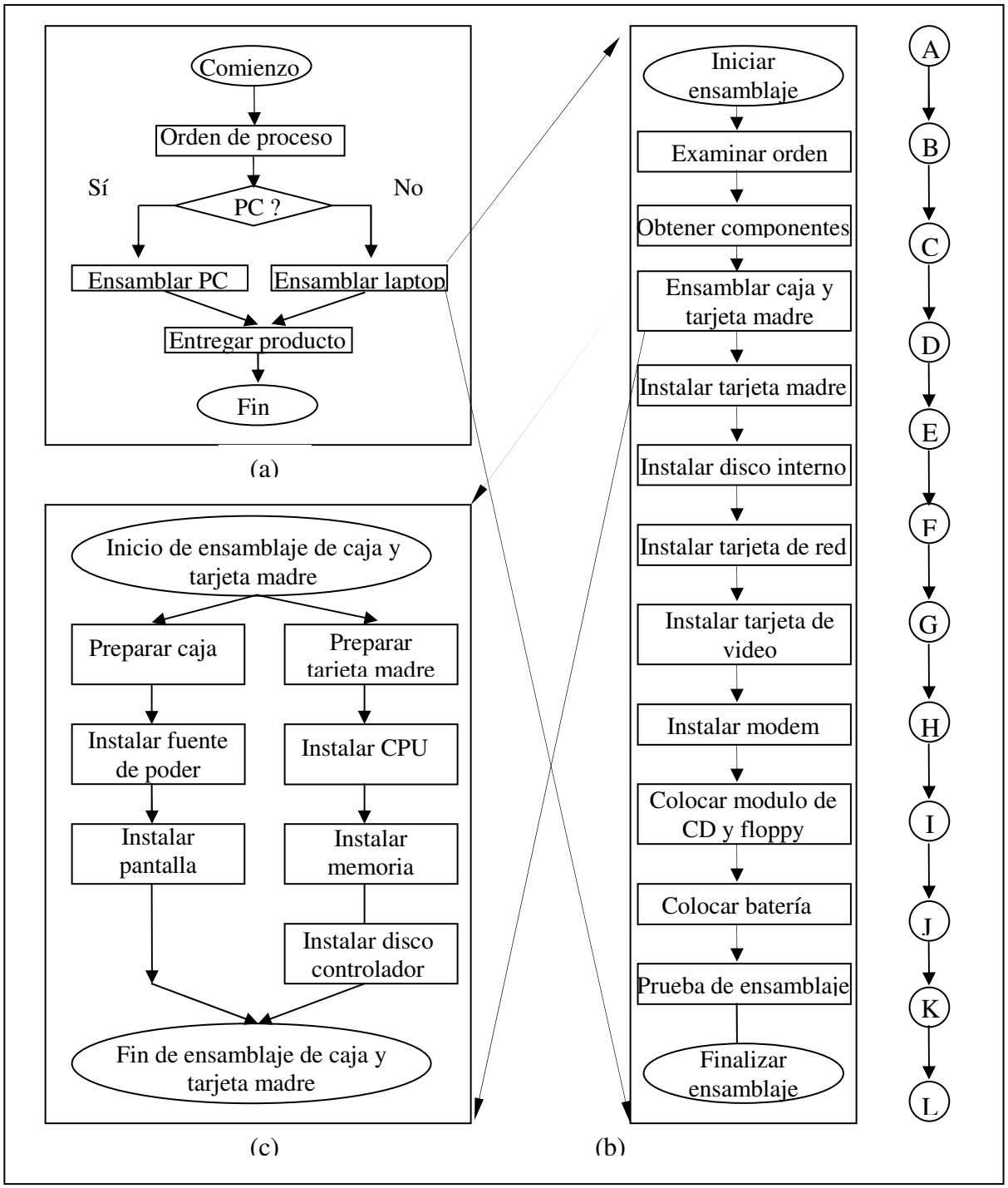


Figura 2.2 Proceso para el ensamblaje de una computadora o una laptop.

La figura 2.2c) muestra un ejemplo de ejecución de tareas en paralelo para el subproceso de ensamblaje de la consola y la tarjeta madre de la laptop. En cuanto inicia

el ensamblaje las tareas de las dos ramas se ejecutan simultáneamente, por ejemplo, *Preparar caja* y *Preparar tarjeta madre*, y así sucesivamente.

Ejemplos de ejecución iterativa se denominan ciclos y por su complejidad pocos sistemas WF los soportan, la figura 2.3 muestra un ejemplo de un proceso que usa ciclos para tratar las fallas técnicas de un producto. En el momento que llega un caso representado por un token en la plaza *falla* inicia el proceso, el patrón de selección (X-OR) *catalogar* se dispara y produce un token en la plaza *resuelto* o en la plaza *A reparar*, dependiendo del resultado y los datos del token con las propiedades relevantes de la falla. Si se requiere reparar, el token se transfiere a la plaza *A reparar* seguida por el disparo de la transición *reparar* enviando el token a la plaza *A probar* y activando y disparando el patrón de selección *prueba*. Dicho patrón dependiendo de los datos del token lo envía a una de las tres plazas de salida: (1) si la falla se resolvió a *resuelto*, (2) otra reparación si se requiere a *A reparar* o (3) si el componente debe reemplazarse a *reemplazar*.

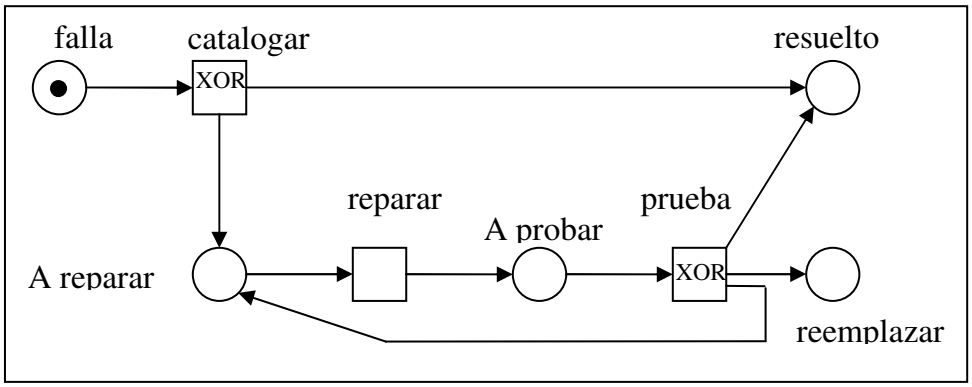


Figura 2.3 Proceso para tratar las fallas de un producto.

Recursos

Un recurso es un medio de producción. Incluye actores como aplicaciones, gente, máquinas, medios de transporte, departamentos y unidades de negocio. Los recursos son los que llevan a cabo las tareas, pueden ejecutar sólo un número limitado de ellas y son agrupados en una o más clases de recursos. En general, existen dos formas de clasificación de recursos: (1) basados en propiedades funcionales y (2) basados en la posición dentro de la organización. La clase de recursos basados en la funcionalidad se denomina rol y es un grupo de recursos dónde cada uno tiene un número específico de habilidades. Ligando una tarea a un rol correcto se asegura que la tarea se llevará a cabo

por un recurso calificado para hacerlo, por ejemplo, un rol puede ser un médico, un asesor, un contador, etc. La clase de recursos clasificados de acuerdo a la posición se basan en características organizacionales más que en funcionales y se denominan unidades organizacionales, por ejemplo, las unidades departamentales de computación, de adquisiciones, etc. Esta forma de clasificación se utiliza para asegurar que una tarea es llevada a cabo en el sitio correcto de la organización. Las dos clasificaciones no son excluyentes, por lo mismo, co-existen en toda organización y puede haber una dentro de otra o traslaparse. La asignación de tareas a los recursos se realiza en la definición de procesos especificando las condiciones que los recursos deben reunir para ejecutar una tarea dada. La figura 2.4 muestra un diagrama de clasificación de recursos. La figura muestra ocho clases de recursos, cuatro del tipo unidad organizacional que son *Atlanta*, *Denver*, *Departamento de adquisiciones* y *Departamento de ventas*; y cuatro del tipo funcional descritos como *Secretaria*, *Agente de ventas*, *Personal de oficina* y *Jefe de departamento*. La figura muestra que las clases de recursos pueden traslaparse e incluso un recurso puede ser un subconjunto de otro. Por ejemplo, el recurso *Agente de ventas* esta contenido dentro del recurso *Personal de oficina*. La relación entre una tarea en particular y un recurso puede generarse de la figura, por ejemplo si se requiere un agente de ventas ubicado en *Atlanta* sólo el recurso *Anita* lo cumple.

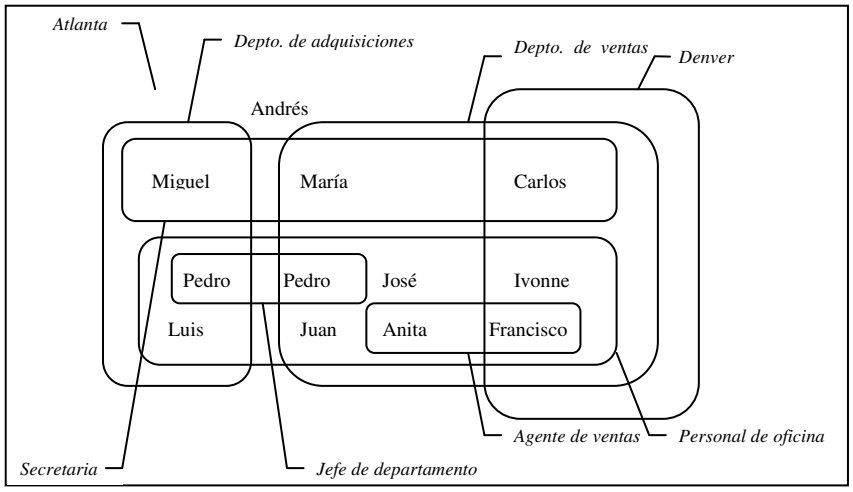


Figura 2.4 Clasificación de los recursos de un organización.

Asignación de actividades a recursos

Para que la asignación de actividades a los recursos sea ejecutada de manera correcta se define un principio de asignación en la definición de procesos, ver figura 2.5. Esta asignación especifica las precondiciones que los recursos deben reunir para realizar una actividad y en muchos casos la asignación especifica el rol que debe cumplir y la unidad organizacional. El recurso debe cumplir con la intersección entre esas dos clases de recursos y puede haber más de un recurso que pueda llevar a cabo una tarea. La asignación depende de los atributos de cada caso para la cuál las tareas deben llevarse a cabo y dependiendo de ellos se seleccionan las clases de recurso.

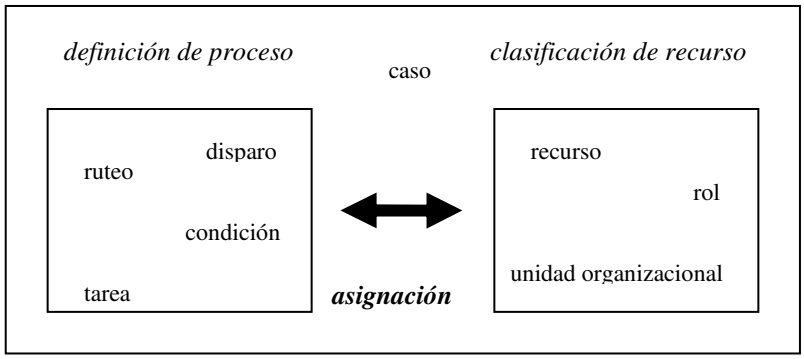


Figura 2.5 Principio de asignación que vincula la definición de procesos con la clasificación de recursos.

Las tres dimensiones del WF

La figura 2.6 presenta las tres dimensiones del WF (Aalst 2004) con respecto a las definiciones anteriores: (1) la dimensión del flujo de control, (2) la dimensión de recursos y (3) la dimensión de casos. La primera dimensión se enfoca en el orden de las tareas, por ejemplo, el proceso del flujo de trabajo. Se identifican las tareas que necesitan ejecutarse y se determina el ruteo de los casos a través de esas tareas como el ruteo secuencial, paralelo, condicional e iterativo. En la segunda dimensión, las tareas las ejecutan los recursos y éstos pueden ser humanos y/o no humanos (dispositivos, aplicaciones). Los recursos se clasifican por sus roles y unidades organizacionales. La tercera dimensión se enfoca en casos individuales que son ejecutados de acuerdo a la definición del proceso (primera dimensión) por los recursos apropiados (segunda dimensión). De las tres dimensiones las dos primeras son genéricas, es decir, no están hechas a la medida de un caso en específico.

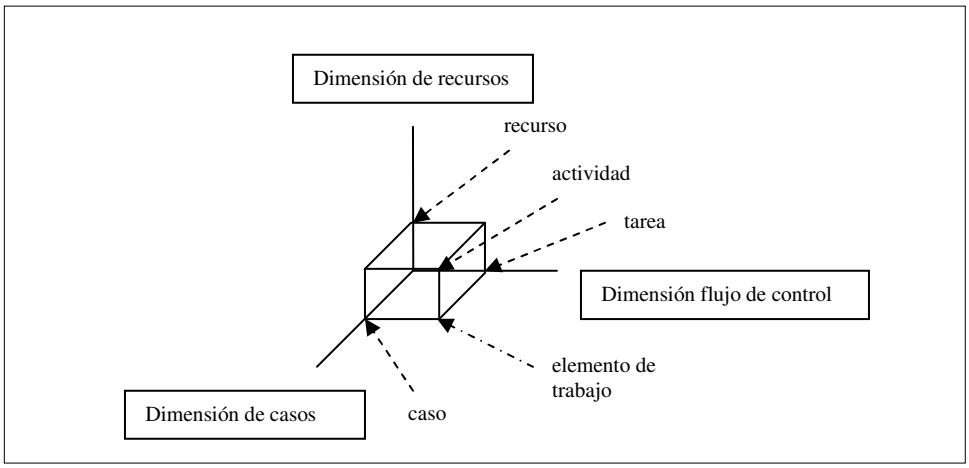


Figura 2.6 Las tres dimensiones del WF.

La dimensión de la perspectiva de flujo de control se considera relevante dentro de la modelación y ejecución WF debido a que se crea la definición de los procesos (esquemas WF) y se especifican las tareas y en qué orden deben de ejecutarse (ruteo). Como las definiciones de los procesos son instanciadas para casos específicos, es decir, un caso es una instancia de una definición, si se desarrollan mal las definiciones de procesos y/o se especifican ruteos incorrectos, por mucha eficiencia de los recursos no se podrán ejecutar los casos, por ejemplo, habrá ejecución de tareas redundantes, saltos de tareas, ciclos infinitos, bloqueos entre tareas (*deadlocks*) y actividades invocadas inapropiadamente. Un flujo de control correcto permite el desarrollo de más perspectivas tales como recursos, datos y operaciones (Aalst 2004). El enfoque de la tesis está dirigido a la perspectiva del flujo de control.

2.2.1 Componentes principales de un sistema WF genérico

Los componentes principales de un sistema WF genérico se muestran en la figura 2.7. El modelo genérico consta de tres tipos de componentes de:

- *Software*.- proporcionan soporte para varias funciones del sistema WF
- Datos y definición de procesos.- son usados por los componentes de *software*.
- Productos y datos externos.- no son parte de productos WF pero pueden invocarse.

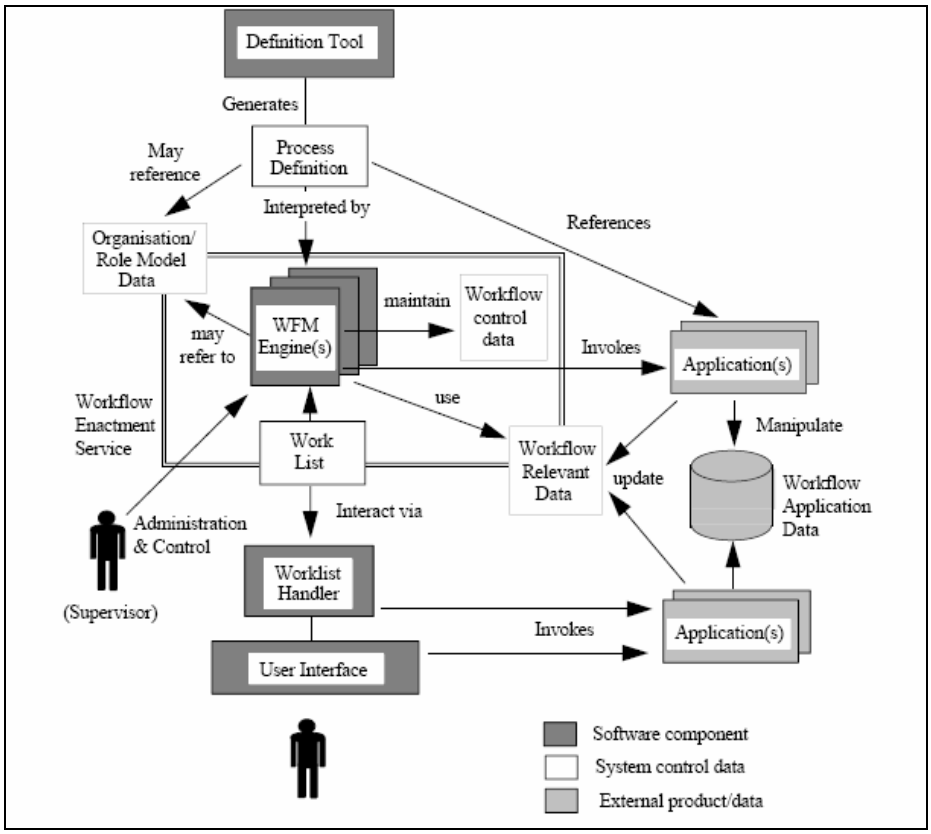


Figura 2.7 Modelo de un sistema WF genérico de la Coalición (WfMC-URL 2007).

A continuación se describen los componentes más importantes mostrados en la figura 2.7:

Herramientas de definición de procesos

Forma parte de los componentes de software y se utiliza para crear una descripción de los procesos en una forma procesable para una computadora. Existe gran variedad de herramientas para analizar, modelar, describir y documentar un proceso de negocio. El modelo WF no se centra con la naturaleza particular de tales herramientas o cómo interactúan en la ejecución. Dichas herramientas pueden suministrarse como parte de un producto WF o por separado. Los productos WF actuales tienen sus propias definiciones de procesos en una forma especializada al SW de administración WF para el cuál fueron diseñadas.

Definición de procesos

Parte del componente de datos y definición de procesos, contiene toda la información necesaria acerca de los procesos incluyendo comienzo de actividades, condiciones, reglas de navegación entre actividades, tareas de usuario y referencia a aplicaciones que pueden invocarse. La definición de procesos puede hacer referencias a modelos de roles/organizacionales dónde se almacena la estructura organizacional y los roles dentro de la organización. Esto significa que la definición de procesos se puede especificar en términos de entidades organizacionales y funciones de roles asociados con actividades particulares en lugar de participantes específicos.

Servicio WF Enactment

Este componente interpreta la descripción de procesos y controla las instancias de los procesos, la secuencia de actividades, añade elementos de trabajo a las listas de trabajo de los usuarios e invoca herramientas de aplicaciones tantas veces como sea necesario. Todo lo anterior es realizado por uno o más motores de administración WF cooperativos los cuáles administran la ejecución de instancias individuales de los diferentes procesos.

Listas de trabajo (Worklist)

Cuando es necesaria la interacción con el usuario en la ejecución de procesos, el motor WF coloca elementos en listas de trabajo administradas por un manejador de listas para controlar dicha interacción con los participantes. Este procedimiento puede ser visible o invisible al usuario dependiendo del sistema WF, muchas veces se deja que el usuario selecciones los elementos y los procese en forma individual.

Manejador de Listas de Trabajo

El manejador de listas de trabajo es un componente de *SW* que administra la interacción entre los participantes del WF y el servicio *WF enactment*. Es responsable por el progreso del trabajo que requiere la atención de usuarios e interactúa con el *SW* del WF *enactment* vía las listas de trabajo. En algunos sistemas puede ser una aplicación de escritorio proporcionando una simple bandeja de elementos de trabajo en la espera de la

atención de los usuarios. En otros sistemas puede ser más sofisticado, controlando la asignación de trabajo entre un conjunto de usuarios proporcionando facilidades como asignación y balanceo de carga de trabajo.

2.2.2 Modelo de referencia WF

El modelo de referencia se desarrolló a partir de una estructura de aplicación WF genérica identificando las interfaces para su estructura que permitan a los productos inter-operar en una variedad de niveles (WfMC-URL 2007). La figura 2.8 muestra los principales componentes y sus interfaces de la arquitectura del modelo de referencia.

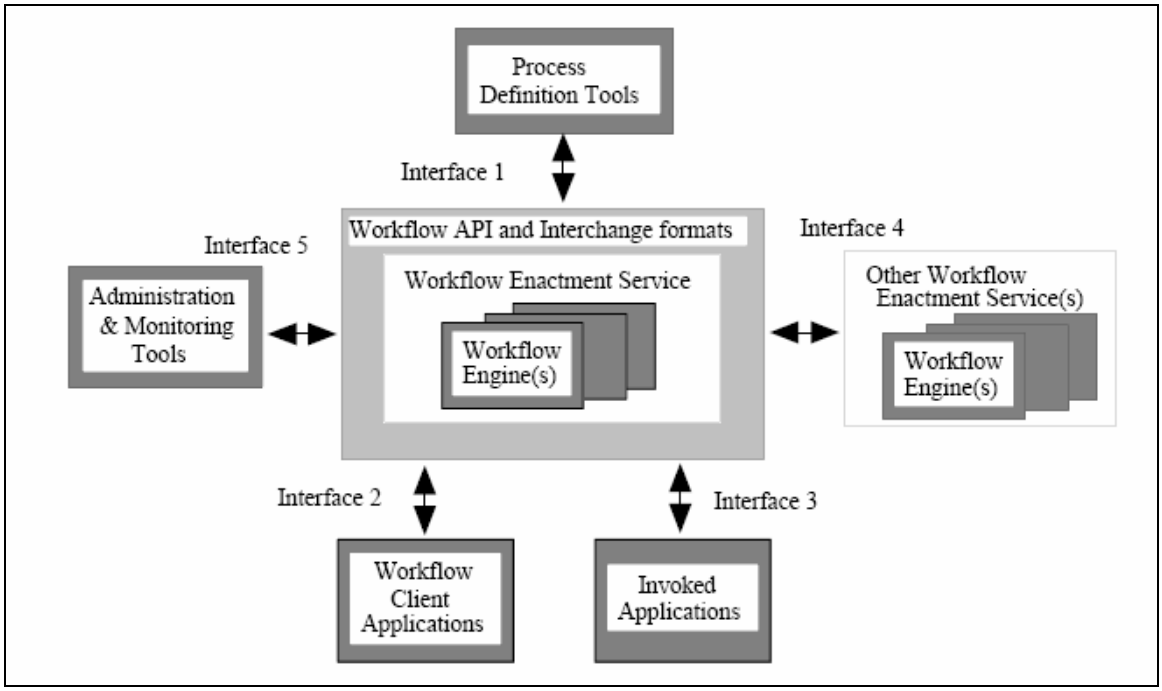


Figura 2.8 Modelo de referencia WF de la Coalición (WfMC-URL 2007).

Todos los sistemas WF contienen un número de componentes genéricos que interactúan en un conjunto definido de maneras; distintos productos exhibirán diferentes niveles de capacidad dentro de cada uno de sus componentes genéricos. Las interfaces son los medios de inter-operabilidad entre los componentes principales. El objetivo de la estandarización de las interfaces permite tener tres ventajas: primero, generalmente los estándares aceptados mejoran el intercambio de datos (compatibilidad) intra y entre sistemas WF. Segundo, es posible crear enlaces entre servidores *enactment* de diferentes fabricantes de una manera directa; los servidores *enactment* administran y ejecutan las

instancias de los procesos y la secuencia de actividades. Finalmente, los estándares habilitan el desarrollo de aplicaciones totalmente independientes del sistema WF seleccionado. Las interfaces alrededor del servicio *enactment* de Workflow se designa como Workflow API (Interfaces de Programación de Aplicaciones) y formatos de intercambio, y pueden ser considerados como un conjunto de constructos para acceder los servicios WF y para regular las interacciones entre el *SW* de control del WF y otros componentes del sistema.

Interacción entre los componentes del modelo

Servicios *enactment* de WF (*Workflow Enactment Service WES*)

El servicio *enactment* proporciona el ambiente para la instancia, activación y ejecución de procesos mediante uno o más motores de administración WF, responsables de activar parte o toda la definición del proceso e interactuar con los recursos necesarios para procesas las tareas. WES es definido por la coalición como un servicio de *SW* que puede consistir de uno o más motores con el fin de crear, administrar y ejecutar instancias WF. Las aplicaciones tienen comunicación a este servicio vía la interfase de programación de aplicaciones WF (WAPI). Existe una separación lógica entre el WES y las herramientas de aplicación y tareas de usuario que constituyen el procesamiento asociado con cada actividad.

Un motor WF es responsable del ambiente de control de ejecución dentro de un servicio *enactment*. La coalición lo define como un servicio de *SW* o *engine* que proporciona el ambiente de ejecución para una instancia WF. Típicamente ofrece facilidades como:

- Interpretación de la definición de un proceso.
- Control de instancias del proceso: creación, activación, suspensión, terminación, etc.
- Navegación entre las actividades del proceso.
- Entrada y salida de participantes específicos.
- Identificación de elementos de trabajo para atención de usuario y una interfase de soporte para interacción con usuarios.

- Mantenimiento de datos relevantes y de control WF, entre el WF y aplicaciones o usuarios.
- Una interfase para invocar aplicaciones externas y enlazar datos relevantes del WF.
- Acciones de supervisión para propósitos de control, administración y auditoria.

Un motor WF puede controlar la ejecución de un conjunto de procesos o subprocesos e instancias con un alcance definido. Un servicio *enactment* que consiste de múltiples motores, particiona la ejecución de procesos a través de los motores por tipo de proceso, por distribución funcional o algún otro mecanismo.

Interfase 1. Herramientas de definición de procesos (*Process Definition Tools*)

La interfase 1 entre el componente de definición de procesos y WES es denominada la interfase para exportar/importar definiciones de procesos. La naturaleza de la interfase es un formato y llamadas API que soporta el intercambio de información de definiciones de procesos completas o en subconjuntos, por ejemplo, cambios o atributos. Esta forma estandarizada permite dos beneficios principales: primero una separación entre el ambiente de definición y el de ejecución, permitiendo que una definición de proceso generada por una herramienta de modelación la puedan interpretar diferentes productos WF de ejecución. Claramente le da el beneficio al usuario de poder seleccionar de forma separada productos de ejecución y herramientas de modelación. Segundo, ofrece el potencial de exportar una definición de procesos a distintos productos WF para que puedan cooperar entre ellos proporcionando un servicio *enactment* de ejecución distribuido.

Esta interfase fue desarrollada para permitir el intercambio de datos de definición de procesos entre herramientas BPR, sistemas WF y repositorios de definición de procesos, habilitando a los usuarios a seleccionar la herramienta más apropiada para diferentes aspectos del ciclo de vida del proceso de negocio. Fue especificada como un Meta modelo de Definición de Procesos, definiendo los objetos proceso, sus atributos y relaciones para expresar la estructura de la definición de procesos y el contenido de su información. Posteriormente fue re-expresada como una definición de documento XML:

XPDL, e incluye actualmente especificaciones para el manejo de eventos y mensajes entre procesos.

Interfase 2. Funciones de cliente WF (*Workflow Client Applications*)

La interacción entre las aplicaciones del cliente y el motor WF se realiza a través de esta interfase mediante listas de trabajo, que consiste de una fila de elementos de trabajo asignados a un usuario particular por el motor WF. En el nivel más simple la lista de trabajo es accesible al motor WF para propósitos de asignación de elementos de trabajo y a un manejador de listas de trabajo para recuperación de dichos elementos y presentarlos al usuario para su procesamiento. El manejador es una entidad de SW que interactúa con el usuario en las actividades que requieren recursos humanos y generalmente es parte de un producto de administración WF.

La interfase fue desarrollada para facilitar la integración de aplicaciones de clientes con diferentes sistemas WF, en particular permitir el principio de portabilidad y re-uso de aplicaciones de cliente con diferentes sistemas de administración WF. Fue especificada como una serie de API WF para permitir el control de procesos, actividades y funciones para el manejo de listas de trabajo, y para la manipulación de objetos de definición de procesos y atributos. Inicialmente fueron definidas en el lenguaje "C" y subsecuentemente en IDL *Interface Definition Language* (OMG 2007) y OLE *Object Linking and Embedding* (Microsoft 2007).

Interfase 3. Funciones de invocación de aplicaciones (*Invoked Applications*)

Las aplicaciones son abiertas desde el sistema de administración WF a través de esta interfase. Fue extendida para proporcionar un entorno común para terceros para integrar otras aplicaciones y servicios de la industria, incluyendo soporte específico de interfaces de agentes para acceder aplicaciones de antaño o tradicionales (*legacy)*. Fue desarrollada como un conjunto de cinco llamadas API básicas, definidas dentro del documento WAPI para soportar un mecanismo común para conexión, desconexión y llamadas a una variedad de agentes u otros ambientes de terceros.

Interfase 4. Interoperabilidad WF (*Other Workflow Enactment Services*)

Fue desarrollada para facilitar la automatización de procesos a través de múltiples ambientes de implantación heterogéneos. Comprende un protocolo de intercambio que atañe cinco operaciones básicas, definidas en términos abstractos y con especificaciones denominadas vínculos concretos (*concrete binding*) separados a otras tecnologías. La versión inicial fue definida como un MIME para uso con correo electrónico; posteriores versiones se especificaron en XML (Wf-XML). Actualmente, la versión 2 comprende SOAP *Simple Object Access Protocol* (W3C 2007) y ASAP *Asynchronous Service Access Protocol* (OASIS 2007).

Interfase 5. Administración de sistemas (*Administration & Monitoring Tools*)

El propósito de esta interfase es permitir la administración y auditoria de casos WF a través de los sistemas, mediante la especificación de un modelo común para auditar datos, incluyendo identificación, formatos y registros. Fue inicialmente especificada en términos abstractos y posteriormente se desarrolló un conjunto común de API para la auditoria de datos. Actualmente está expresada como un conjunto de estructuras en XML.

Las cinco interfases

Aunque fueron concebidas como cinco interfases individuales su separación es aparente sólo cuando es vista en el contexto de objetivos de negocios establecidos. En realidad, existe mucha similitud de funciones entre las distintas interfases, por ejemplo, el disparo de la iniciación de un proceso para ejecución es fundamentalmente la misma acción si se realiza del lado del cliente (interfase 2) o del lado del servidor (interfase 4). La evolución de la especificación WAPI inició con aplicaciones de cliente y se ha expandido a un repertorio completo de llamadas API, similarmente Wf-XML fue desarrollado inicialmente para interacciones servidor-servidor y ya se usa para interacciones cliente-servidor.

El modelo de referencia intenta construir una vista abstracta de los procesos de negocio en términos de sus características, separadas de la tecnología. Esta separación de vista abstracta y concreta se aplicó en la diseño de las interfases, que fueron primero definidas en términos abstractos y posteriormente se desarrollaron los vínculos concretos,

de cada interfase abstracta a tecnologías de interfase específicas. El valor de este enfoque fue aparente sobre los años, a medida que la tecnología ha evolucionado, se han desarrollado un rengo de dichas especificaciones cada una apropiada a las tecnologías de ese tiempo, pero adhiriéndose al mismo modelo abstracto en la operación de la interfase.

Inicialmente las interfases reflejaron una programación de bajo nivel, típicamente las API estaban basadas en el lenguaje C. Posteriormente, conforme más tecnologías de interfase fueron aceptadas, fueron añadidas en la forma de IDL, CORBA (por la OMG) y correo MIME (para interoperabilidad de procesos). Actualmente las interfases más ampliamente usadas fueron redefinidas usando servicios Web y XML (XPDL y Wf-XML).

2.2.3 Categorías de soporte WF en la actualidad

El uso de sistemas WF es altamente relevante para cualquier organización, sin embargo, pocas organizaciones utilizan sistemas WF reales debido a que utilizan diferentes categorías de soporte WF. A continuación se describen las categorías (Aalst 2004):

- Sistemas de administración WF puros
    - Sistemas que soportan toda la funcionalidad de un sistema WF. Construidos para la automatización de procesos y administración de recursos.
- Componentes de administración WF añadidos en otros sistemas
    - Muchos paquetes de *SW* integran a un componente WF genérico cuya funcionalidad es comparable con los sistemas WFM puros. Por ejemplo, muchos sistemas ERP proporcionan un componente WF. SAP WebFlow (SAP 2007) es el componente WF de SAP ofreciendo toda la funcionalidad presente en productos tradicionales WFM puros.
- Soluciones WF hechas a la medida
    - Muchas organizaciones tales como bancos y compañías aseguradoras en lugar de adquirir construyen soluciones WF específicas a su organización. Dichas soluciones soportan sólo un subconjunto de la funcionalidad ofrecida por las

dos categorías anteriores, no obstante, soportan la definición y ejecución de distintos WF.

- Soluciones WF integradas en el código
    - Esta categoría se refiere a la situación en la que los procesos se codifican dentro de las aplicaciones, por ejemplo, no existe un soporte WF genérico pero las aplicaciones son acopladas del tal manera que soportan un proceso de negocio específico. La única manera de cambiar un proceso es cambiando las aplicaciones. No existe un nivel de orquestación.

Muchas organizaciones administran sus procesos en la tercera y cuarta categoría (Aalst 2004). Sin embargo, cada vez es mayor el porcentaje de soluciones que se encuentran en las primeras dos categorías y muchos desarrollos que se encuentran en las dos últimas integran conceptos y conocimiento provisto por las dos primeras. En este contexto, el área de los lenguajes de composición de servicios Web es muy similar a los lenguajes WF tradicionales.

2.3 Estado del Arte

El estado del arte escrito en esta sección se divide en dos partes, cada parte incluye un área de conocimiento estrechamente relacionada dónde existe una aportación importante de esta investigación. La primera parte de esta sección describe los enfoques actuales de los WF adaptivos, la selección de los grupos de estudio se hacen con base en las características principales que deben cumplir todo diseño de WF adaptivos y se presentan sus ventajas y desventajas. En la segunda parte se hace una crítica de los enfoques publicados actualmente de composición WF y se mencionan los lenguajes de modelación usados en los enfoques.

2.3.1 WF Adaptivos

En la administración WF, un aspecto clave de gran relevancia es la adaptabilidad de los WF. En el ambiente de negocios actual se tiene la necesidad de WF con capacidad de adaptarse a cambios dinámicos, administrando situaciones nuevas y siendo flexible a variaciones inesperadas en las tareas durante su ejecución, los WF que

soportan el cambio dinámico se denominan WF adaptivos (Ellis and Rozenberg 1995) (Kwan 1997) (Reichert and Dadam 1998) (Kradolfer 1999) (Agostini and DeMichelis 2000a, 2000b) (Sadiq 2000) (Sadiq et al. 2000) (Weske 2001) (Aalst and Basten 2002) (Aalst 2003b) (Doshi et al. 2004) (Rinderle et al.2004) (Reichert et al.2005) (Wang et al. 2005) (Yamaguchi et al. 2005). Básicamente, el cambio dinámico es transformar WF que están ejecutando casos o instancias en ese momento, y el problema surge en ¿Cómo administrar esos casos?, debido a que pueden surgir inconsistencias al adaptar los casos al cambio. Formalmente, el cambio dinámico ocurre cuando en la producción de un caso se tiene que migrar de una definición de proceso a otra (Ellis and Rozenberg 1995) (Aalst 2002a, 2002b), dicha definición afecta a todos los casos que se ejecutan en ese momento y a los que están por ingresar. Algunos cambios son fáciles de realizar otros son difíciles dependiendo si el cambio afecta a una solo caso o es estructural a todos los casos. El cambio dinámico está relacionado principalmente en el contexto de variantes y versiones de procesos WF. Los variantes de un proceso WF surgen en los denominados cambios ad-hoc de un WF, estos cambios, se administran caso por caso y afectan a sólo un caso/instancia o grupo de casos (Aalst and Basten 2002) (Aalst 2003b). El cambio resulta de un error y un ejemplo típico es el saltar la ejecución de una tarea. Las versiones son definiciones de procesos resultantes de cambios estructurales en sistemas WF, un cambio estructural (de esquema) cambia el WF para todos los casos que llegan al sistema y es el resultado, por ejemplo, de una nueva estrategia de negocios o una reingeniería de procesos (BPR) donde se rediseñan los procesos de negocio principales (Laguna and Marklund 2004). La dificultad de un cambio estructural se presenta principalmente en los casos que ya existen en el WF a la hora del cambio. La figura 2.9 presenta un ejemplo de un error de cambio dinámico de un proceso de negocio. El proceso está representado por una red de Petri dónde las barras representan actividades y los círculos los estados del proceso. El error puede surgir cuando transformamos la red de Petri paralela en una secuencial, por ejemplo, asumiendo el siguiente estado: un token en S1, antes de cobrar el producto y en S4, producto enviado. No hay un estado en la red serial que pueda representar dicho estado, un token en P1 significaría que no se ha enviado el producto violando S4 y un token en P2 significaría que ya se cobró el producto violando S1. El

error se debe a que en el cambio estructural se generaron menos estados que en la red original y por lo mismo habrá estados que no se puedan representar.

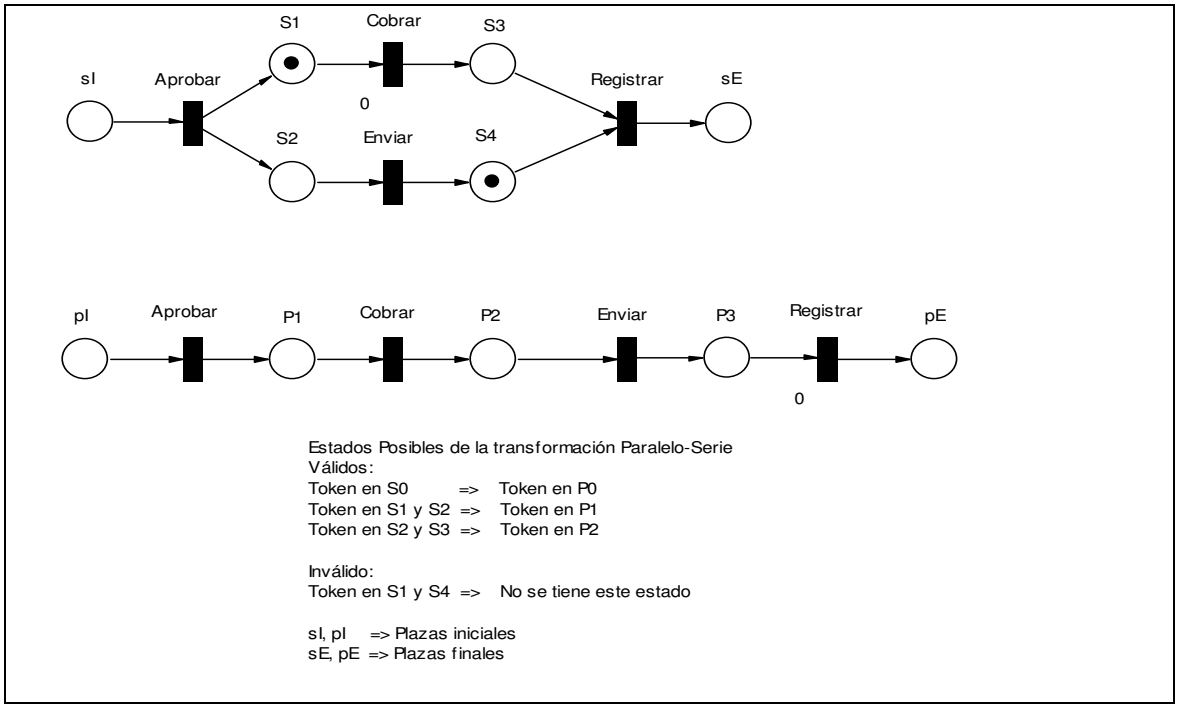


Figura 2.9 Estados posibles para la transformación de un proceso paralelo a serie.

Actualmente existen muchos enfoques que tratan de resolver el cambio estructural, las soluciones se pueden considerar parciales debido a que tienen restricciones a la hora de administrar los cambios en un WF en ejecución. No existe en la actualidad, una solución adaptiva que resuelva todos los casos que se puedan presentar en un cambio dinámico. En esta sección se presenta el análisis de los enfoques que existen hasta ahora, que se relacionan con el presente trabajo y que pueden clasificarse en los siguientes tres criterios indispensables que debe tener un WF adaptivo:

1. De integridad
    a. Debe tener flujos de control y/o de datos. Para propósitos prácticos debe proporcionar operaciones de cambio para insertar y/o borrar actividades así como dependencias de flujos de control y/o de datos entre nodos.

2. De correctes
    a. Cualquier enfoque adaptivo aspira a que los cambios dinámicos produzcan un WF correcto. Un WF es correcto si: todos los casos inician y terminan el WF, no existen bloqueos entre actividades (*deadlocks*), saltos de actividades, ejecución infinita de actividades, redundancia de actividades ni actividades invocadas inapropiadamente. Para que un WF sea correcto se necesitan adecuar criterios para verificar si una instancia I cumple con el esquema cambiado o no. Los criterios no deben ser muy restrictivos, por ejemplo, ninguna instancia debe ser excluida de ser adaptada a un cambio en el proceso.

3. De realización del cambio
    a. Asumiendo que el cambio puede ser propagado correctamente a una instancia I, por los dos criterios anteriores, debe ser posible migrar automáticamente la instancia I al nuevo esquema. En este contexto, un reto es adaptar eficiente y correctamente los estados de la instancia.

Se hizo un búsqueda exhaustiva de todos los enfoques en la literatura a partir del primer trabajo adaptivo publicado por Ellis y Rozenberg en 1995. La búsqueda se hizo con base en los tres criterios mencionados y se encontraron nueve enfoques que los cumplen total y/o parcialmente.

En la tabla 2.1 se muestra la clasificación de sus meta-modelos, la primera columna indica cómo funciona el formalismo, es decir, la semántica operacional, y las estrategias aplicadas para la ejecución de instancias WF (Kiepuszewski et al. 2000) y (Rinderle et al. 2004); la segunda columna indica los enfoques adaptivos que utilizan el formalismo descrito en la primera columna y la tercera los grupos de investigación y/o autores que los proponen.

Tipo de formalismo	Enfoques	Grupos y/o Autores
Semántica de un solo token: "True-semantics" Formalismo: Redes de Petri	WF Nets	Aalst, 2003b. Aalst, Weske and Wirtz, 2003a.
	Milano Nets	Agostini and DeMichelis, 2000a y 2000b.
	Flow Nets	Ellis and Rozenberg, 1995.
	Selective Shift	Yamaguchi and Mishima, 2005.
Semántica de tokens verdadero/falso V/F y uso de bitácoras: "True/false-semantics": 1. Sin marcaje inherente	WIDE	Casati Fabio, Castano Silvana, Fugini Mariagrazia, Mirbel Isabelle and Pernici Barbara, 2000.
	TRAMs	Kradolfer Markus and Geppert Andreas (1999).
2. Con marcaje inherente (marcajes especiales de nodos y arcos que representan una vista de la bitácora de ejecución)	ADEPT	Reichert M. and Dadam P., 1998. Reichert Manfred, Rinderle Stefanie, Kreher Ulrich and Dadam Peter, 2005.
	Breeze Activity Nets	Sadiq Shazia W., 2000. Sadiq Shazia W., Marjanovic Olivera and Orlowska M. E., 2000.
	WASA (Orientada a Objetos)	Weske, 2001.

Tabla 2.1 Enfoques de WF adaptivos en la actualidad.

La primera estrategia sólo utiliza un tipo de paso de "token" (elemento que representa un caso y que al trasferirse de un nodo a otro, significa el paso del caso de una actividad a otra) por cada instancia, denominada "True tokens". Los formalismos que lo usan son:

- Milano Nets
- Flow Nets
- WF Nets
- Selective Shift

y pertenecen a metamodelos basados en redes de Petri (Jensen 1997).

La segunda estrategia se basa en dos tipos de tokens: "True/False tokens", dónde los *True-tokens* se comportan de igual manera que la estrategia anterior, pero a diferencia de ella, se utilizan los "False-tokens" para indicar actividades saltadas o bifurcaciones. Además, los enfoques se pueden dividir de acuerdo a si utilizan marcaje inherente, que

consiste de marcajes especiales en nodos y arcos. Ambos enfoques administran una bitácora de ejecución.

Con marcaje inherente

- ADEPT
- Breeze
- $WASA_2$

Sin marcaje inherente

- WIDE
- TRAMs

y pertenecen a metamodelos basados en grafos.

Características principales y análisis de comparación de los enfoques adaptivos

En esta sección se presentan las características principales de los nueve enfoques mostrados en la tabla 2.1 y los criterios de correctes que utilizan, también, se presenta un análisis con respecto a diferentes modificaciones que pueden generar problemas en el cambio dinámico. Al final, se presenta el sumario comparativo de cada enfoque con respecto a su formalismo y a las modificaciones dinámicas.

La tabla 2.2 agrupa a grosso modo los enfoques con respecto a la forma en que migran y adaptan las instancias al nuevo esquema cuando ocurre un cambio dinámico. La tabla describe en la primera columna el método de equivalencia utilizado por cada enfoque para comparar los esquemas anterior y nuevo, y pueda evaluarse si la instancia en ejecución puede migrarse al nuevo esquema con base en el criterio de correctes mostrado en la tercera columna. La equivalencia de grafos puede realizarse de dos maneras, la primera la utiliza *WF Nets* y consiste comparar el esquema WF respectivo antes y después del cambio mediante métodos formales propios de las redes de Petri (Aalst 2002); la segunda consiste en transformar el grafo recorrido por la instancia *I* del WF al nuevo esquema cambiado WF, para ello se debe de tener la historia de ejecución de la instancia, lo usan *Milano Nets* y *$WASA_2$*. En ambos casos, la comparación o el mapeo indican si se puede aplicar el cambio. La equivalencia de ejecución, también denominada traza, se enfoca en el trabajo hecho por la instancia *I* hasta el momento del

cambio, si dicho trabajo puede lograrse en el nuevo esquema, la instancia puede migrarse. Este método requiere registrar en bitácoras la historia de ejecución de las instancias de tal manera de poder reproducirla en el nuevo esquema y pueda comprobarse si es posible el cambio, los enfoques que lo usan son *Flow Nets*, *WIDE*, *Breeze*, *TRAMs*, *ADEPT* y *Selective shift*.

Método	**Meta-modelo**	**Criterio de correctes**
Equivalencia de grafos/nodos: • La idea principal es comparar el esquema WF respectivo antes y después del cambio o mapear el grafo de la instancia al esquema WF cambiado.	WF Nets	Una instancia I en el esquema S correcto (representado por una red marcada) cumple con el esquema modificado S', si S y S' están vinculados mutuamente bajo relaciones de herencia definidas, por ejemplo, si S es una sub clase de S'. Un esquema S es correcto si es *sound* (Aalst, 2002).
	Milano Nets	Se basa en estados seguros/inseguros. Una instancia I cumple con un esquema cambiado si no está en un estado inseguro. Un estado de un esquema S es inseguro con respecto al esquema cambiado S' si el estado no está presente en S'.
	$WASA_2$	El criterio se basa en el concepto de mapeos válidos. Una instancia I cumple con un esquema S' si existe un mapeo válido de I a S'. Esto significa que cada actividad terminada de I está también contenida en S' y existen todas las dependencias de control y de datos.
Equivalencia de ejecución: • Esta equivalencia se enfoca en el trabajo hecho por la instancia I hasta el momento del cambio. Si dicho trabajo puede reproducirse en el nuevo esquema, la instancia I puede migrarse.	Flow Nets	La migración de las instancias se lleva a cabo sustituyendo la subred marcada N1, afectada por el cambio y previamente detectada, por otra sub red marcada N2, la cual refleja las modificaciones establecidas por el cambio. N1 es referida como la región de cambio anterior y N2 como la región de cambio nueva. Dependiendo del cambio pueden coexistir las dos regiones.
	WIDE	Introdujeron el criterio de correctes de cumplimiento (compliance) basado en historias de ejecución y ampliamente usado actualmente. Un cambio en un esquema S puede propagarse correctamente a una instancia I, si y sólo si la historia ejecución de I hasta ese momento puede reproducirse en el esquema cambiado S'.
	Breeze	Utiliza el criterio de cumplimiento propuesto por WIDE. En un cambio, agrupan las instancias con respecto al nivel de cumplimiento. Utilizan grafos de compensación (compensación de actividades) para instancias que no cumplen con el criterio de correctes y poder hacerlas cumplir.
	TRAMs	Utiliza el criterio de cumplimiento propuesto por WIDE. Sin embargo, considera muy ineficiente reproducir cada paso del histórico de una instancia en el esquema cambiado, sobre todo cuando se requieren migrar un número grande de instancias. Proporciona condiciones de migración para verificar de manera más eficiente (comparado con WIDE) el cumplimiento de instancias con respecto al esquema cambiado.
	ADEPT	Utiliza el criterio de cumplimiento propuesto por WIDE pero lo modifica de manera de soportar iteraciones repetitivas. Define una historia de ejecución reducida para verificar eficientemente el cumplimiento de una instancia con el esquema cambiado. La historia de ejecución reducida evita tener datos históricos voluminosos.
	Selective Shift	Similar a Flow Nets. Es más flexible ya que los casos, a diferencia de Flow Nets, se pueden administrar de manera individual a la hora de migrarlos a las regiones de cambio.

Tabla 2.2 Criterios de correctes de los meta-modelos.

Para una consulta del criterio de correctes formal de todos los enfoques ver (Ellis and Rozenberg 1995), (Reichert and Dadam 1998), (Kradolfer and Geppert 1999), (Agostini and DeMichelis 2000a, 2000b), (Casati et al. 2000), (Sadiq et al. 2000), (Weske 2001), (Aalst 2003b) y (Yamaguchi et al. 2005). Para el análisis comparativo de los enfoques, se utilizan cinco modificaciones, propuestas por Rinderle et al. (2004), que pueden generar problemas en el cambio dinámico:

1. **Cambios en el pasado (CP)**
   a. La figura 2.10 muestra los dos tipos de cambios en el pasado que pueden realizarse.
   b. El problema corresponde a la regla de dedo: no cambiar el pasado de una instancia. Rechazar esta regla podría guiar a estados de instancia inconsistentes, por ejemplo, actividades en ciclos infinitos (*livelocks*) o casos bloqueados que no se ejecutan (*deadlocks*) o pérdida de datos si existe dependencia en actividades que se cambian.
   c. La compartición de datos con cambios en el pasado y en el futuro permite aminorar y muchas veces resolver el problema de inconsistencia.

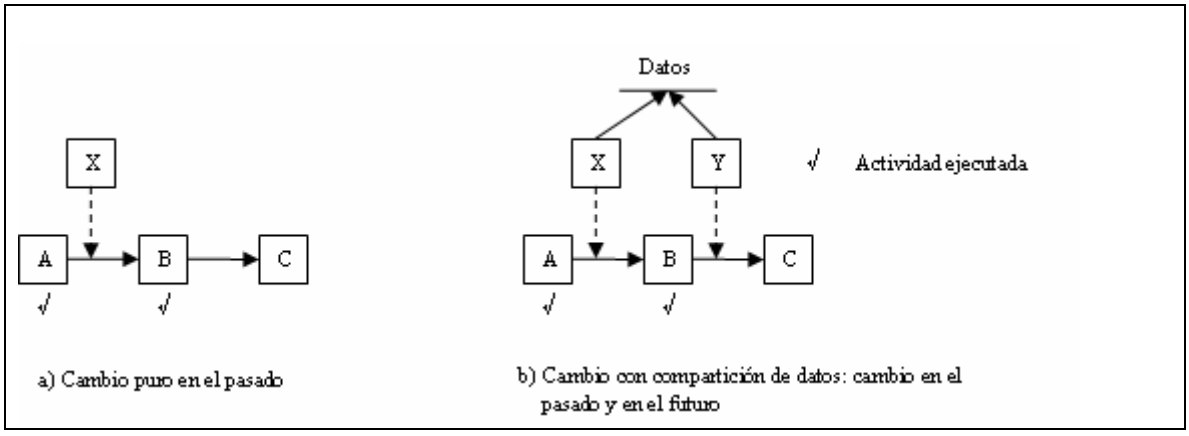


Figura 2.10 Cambios en el pasado.

2. **Tolerancia a ciclos (TC)**
   a. La figura 2.11 muestra el problema y se refiere a la habilidad del enfoque a tratar razonablemente y correctamente con cambios en la estructura de ciclos.

b. En particular, los enfoques no deben innecesariamente excluir instancias de migrarlas al nuevo esquema basado solamente del hecho de que los cambios respectivos conciernen a ciclos.

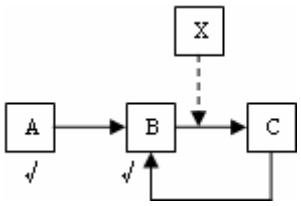


Figura 2.11 Tolerancia y cambio en ciclos.

3. Estado Actividades (EA)

a. La figura 2.12 muestra el problema, surge en conjunción con enfoques que no distinguen entre actividades activas y en ejecución. Como consecuencia muy a menudo tales enfoques no permiten el borrado de actividades activas (lo que es muy restrictivo) o permiten el borrado de actividades en ejecución, lo que guía a la pérdida de trabajo.

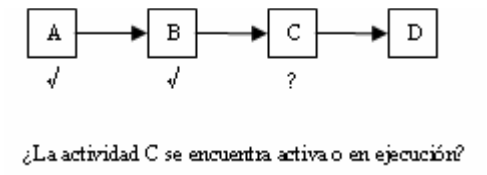


Figura 2.12 Identificación de los estados de las actividades.

4. Cambiando el orden (CO)

a. El problema se refiere a correctamente adaptar el marcaje de instancias cuando aplicamos operaciones de cambio de orden tales como concurrencia (hacer paralelas dos actividades), secuenciación e intercambio de actividades. La figura 2.13 muestra el ejemplo de hacer paralelas dos actividades.

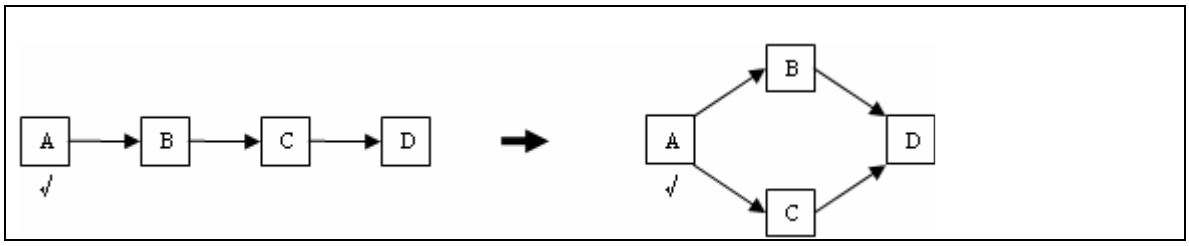


Figura 2.13 Haciendo paralelas dos actividades.

5. Inserción Paralela (IP)

a. El problema se refiere a correctamente adaptar el estado de las instancias cuando insertamos una rama paralela, ver figura 2.14.
b. Problemas OC e IP son fuertemente relacionados al error del cambio dinámico.

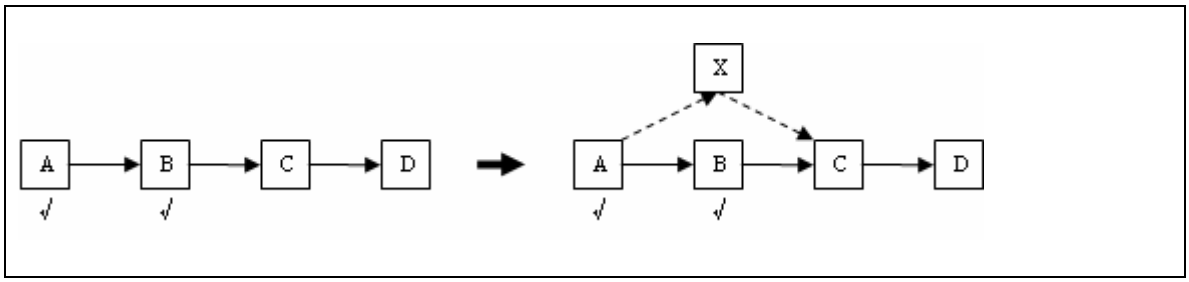


Figura 2.14 Inserción paralela.

El análisis de los enfoques consiste en lo siguiente:

1. Verificar si la modificación dinámica es permitida por el enfoque con base en el criterio de correctes.
2. Si la modificación dinámica es permitida por el enfoque, verificar si evita o resuelve el problema que se puede generar por la modificación.

La tabla 2.3 muestra en resumen la comparación de los grupos de la tabla 2.1, en el anexo A se muestran a detalle el análisis de los enfoques con respecto a las modificaciones y su criterio de correctes. Note que muchos metamodelos proponen como solución no permitir y/o evitar la modificación para no generar problemas con las consecuencias de soportarlo, por ejemplo, cambios en el pasado pueden generar problemas si existen dependencias entre las actividades modificadas y las futuras. La tabla muestra que ningún metamodelo soporta las cinco modificaciones y por ello, es una línea abierta de investigación en la actualidad.

Meta modelo	CP	TC	EA	CO	IP
WF Nets	Permitidos	Permitido	No distingue	No permitido	Permitido
	Las reglas de transformación simplifican posibles problemas.	No existe el problema.	Problema implícito en el modelo.	No hay una regla de transformación que lo permita.	No existe el problema.
Milano	Permitido	Evitado	No distingue	Permitido	Evitado
	La migración de instancias de estados inseguros se pospone hasta que estén seguros.	El modelo es acíclico	Problema implícito por el modelo.	No proporciona reglas para adaptar el marcaje.	Las reglas de cambio no lo permiten.
Flow Nets	Permitidos	Posible	No distingue	Permitido	Permitido
	Sólo en el pasado, no permite cambios al mismo tiempo en el pasado y en el futuro. Pueden surgir problemas.	Está muy restringido y depende del criterio de conformidad al nuevo esquema.	Problema implícito por el modelo.	No existe el problema. Utiliza dos regiones.	No proporciona marcaje.
Selective Shift	Permitidos	Evitado	No distingue	Permitido	Permitido
	Puede existir problemas	El modelo es acíclico	Problema implícito por el modelo.	No existe el problema. Utiliza dos regiones.	No existe el problema.
WIDE	No permitidos	Permitido	No distingue	Permitido	Permitido
	El criterio de conformidad al nuevo esquema no lo permite.	Soporta ciclos pero no permite cambios en los ciclos.	El histórico generado por este enfoque contiene sólo actividades terminadas.	No existe el problema.	No existe el problema.
TRAMs	No permitidos	Es evitado	Distingue	Permitido	Permitido
	Las condiciones de migración prohíben cambios en el pasado.	El modelo es acíclico.	No existe el problema.	No existe el problema.	No existe el problema.
ADEPT	No permitidos	Permitido	Distingue	Permitido	Permitido
	Las condiciones de migración y el criterio de tolerancia de ciclos prohíben cambios en el pasado.	El criterio de conformidad al nuevo esquema permite cambios en ciclos.	No existe el problema.	Las operaciones de cambio se administran por las condiciones de migración.	No existe el problema.
Breeze	No permitidos	Evitado	Distingue	Permitido	Permitido
	Su criterio de conformidad prohíbe cambios en el pasado.	El modelo es acíclico.	No existe el problema.	No existe el problema.	No existe el problema.
$WASA_2$	No permitidos	Evitado	Distingue	Permitido	Permitido
	No lo permiten los criterios de conformidad al nuevo esquema.	El modelo es acíclico.	No existe el problema.	El mapeo al nuevo esquema es muy restringido para las adaptaciones dinámicas, sólo se permiten las que cumplan con el mapeo.	No existe el problema.

Tabla 2.3 Comparación de los meta-modelos con respecto a las cinco modificaciones.

Sumario comparativo

Los enfoques son propuestas de solución al cambio dinámico y no todos se encuentran implantados. El enfoque *WF Nets* (Aalst and Basten 2002b) está basado en redes de Petri clásicas, utiliza propiedades definidas por Aalst para verificar la correctes del WF de tal manera que el caso que se esté ejecutando alcance el estado final de la red. El comportamiento de un caso en todas las redes de Petri es descrito por las reglas de disparo y por el marcaje de la red. Puede administrar múltiples versiones del esquema para casos que no pueden migrarse, permite secuencia, concurrencia, selección e iteración y, añadir y remover secuencias. Permite cambios en el pasado que pueden originar que una actividad insertada nunca se ejecute. El modelo no considera de forma explícita el flujo de datos, no distingue si una actividad está activa o en ejecución, ni soporta cambios de orden, por ejemplo, paralelizar secuencias.

La propuesta de *Milano Nets* (Agostini and DeMichelis 2000a, 2000b) utiliza redes acíclicas "free-choices" (Murata, 1989) y tienen como objetivo generar redes WF lo más simples posibles. Soporta secuencia, concurrencia e intercambio de actividades, pospone la migración de instancias para casos que no cumplan con el nuevo esquema, por lo que no administra múltiples versiones. La migración de instancias se realiza hasta que alcance un estado seguro y es una desventaja cuando se requiere un cambio instantáneo y obligatorio. No soporta la selección, no distingue entre actividades activas y en ejecución y además, no considera el flujo de datos de forma explícita.

Flow Nets (Ellis and Rozenberg 1995) fue el primer trabajo en la literatura matemáticamente articulado que trató sobre los aspectos de cambio dinámico en el WF. Los autores explicaron el tan denominado error de cambio dinámico y presentaron un enfoque de redes de Petri para la modelación del cambio dinámico, ésto es un gran mérito. Las redes de Petri que utiliza son de alto nivel, pueden contener más de un token, y la ejecución de las instancias está basada en la misma red, por lo que no administra múltiples versiones. Administra los casos colectivamente, la adaptación de una sola instancia no es posible, esto implica que no considera cambios ad-hoc de instancias. No

presenta un método automático para identificar regiones de cambio, la detección tiene que ser manual. La tolerancia a cambios en ciclos es posible y depende totalmente del criterio de correctes. Permite cambios en el pasado, no distingue entre actividades activas y en ejecución, y en el caso de inserciones paralelas no determinan el marcaje de la red cambiada para el estado resultante del caso.

Selective shift (Yamagushi and Mishima 2005) utiliza también redes de Petri de alto nivel y es una combinación de los enfoques *Flow Nets* y *Breeze*. Su aportación consiste en administrar los cambios de instancias al nuevo esquema de manera separada, es decir, no maneja las instancias de manera colectiva como el enfoque de *Flow Nets* pero administra las regiones de cambio para casos que no pueden migrarse al nuevo esquema de manera similar. Le añade temporalidad a su modelo y propone una medida para evaluar la eficiencia del cambio dinámico. El modelo es acíclico, permite cambios en el pasado y no distingue entre actividades activas y en ejecución,

En *WIDE* (Casati et al. 2000) se define por primera vez el criterio de cumplimiento (*compliance*). El enfoque trabaja con un modelo de ejecución basado en bitácoras, soporta secuencia, concurrencia, selección e iteración. Los cambios en el esquema se basan en un conjunto de primitivas de modificación cuya aplicación no viola el criterio de correctes, pero las excepciones deben ser conocidas en el momento de diseño. Se introducen criterios formales para determinar qué instancias pueden migrarse a la nueva versión e incluye soporte para administrar múltiples versiones de esquemas. El modelo no considera el flujo de datos en forma explícita, no permite cambios en el pasado ni en ciclos, y no distingue entre actividades activas y en ejecución. Los autores no proporcionan información detallada acerca de la adaptación de marcaje para instancias migradas y no especifican cómo verifican su criterio de correctes. No se tiene un prototipo operacional.

En contraste a los otros enfoques que usan grafos, en *TRAMS* (Kradolfer and Geppert 1999) el flujo de control no se representa por arcos. Los describen de una forma declarativa usando condiciones de inicio y terminación de actividades. La correctes se

preserva por invariantes, por ejemplo, condiciones relacionadas al esquema que deben cumplirse. Los invariantes se utilizan para migraciones válidas de instancias y se administran múltiples versiones del esquema por instancia en caso de que no puedan migrarse. El modelo es acíclico, no permite cambios en el pasado y debido a la definición del flujo de control declarativo y a la ausencia de arcos de control explícitos, la inserción de actividades es un cambio complejo.

ADEPT (Reichert and Dadam 1998) presenta primitivas de modificación que preservan la consistencia WF similar a la propiedad *sound* de las redes WF (Aalst and Kees 2002). Tiene reglas de marcaje bien definidas para garantizar el cumplimiento de las instancias, soporta secuencia, concurrencia, selección e iteración. Se concentra en cambios ad-hoc de instancias individuales, no proporciona modificación del esquema, el alcance de cada modificación es a nivel de instancias individuales, administrando múltiples versiones ad-hoc de instancias. El modelo no permite cambios en el pasado y no considera aspectos semánticos tales como: errores introducidos por intercambio de tareas, saltos de tareas o múltiples ejecuciones de tareas.

Breeze (Sadiq 2000) introduce el concepto de compensación de instancias usando grafos de compensación: crea esquemas compensados para aquellas instancias que no cumplen con el esquema cambiado, esto hace que las instancias que estén en concordancia con el esquema compensado puedan continuar ejecutándose, administrando múltiples versiones, y las que no puedan compensarse se abortan. El modelo soporta secuencia, concurrencia, selección e intercambio de tareas, es acíclico y no permite cambios en el pasado.

En *WASA*$_2$ (Weske 2001) una instancia del WF es descrito por un grafo cuyo estado es denotado por el marcaje de las actividades inherentes al modelo. Esto significa, que es explícito el marcaje, el estado de las actividades, el flujo de control y el flujo de datos. Dichas actividades pueden distinguirse en no activas, activas, en ejecución, terminadas y saltadas. Una instancia en este enfoque siempre es ligada a un esquema, administrando múltiples instancias y esquemas. El modelo soporta secuencia,

concurrencia y selección, es acíclico y no permite cambios en el pasado. No proporciona información detallada acerca de la adaptación de marcaje para instancias adaptadas.

WF comerciales

Los productos WF inicialmente fueron ofrecidos en los años 80: el primer vendedor fue File-NET con Worflo, seguido por IBM con Flowmark y por Bull con FlowPath. Actualmente hay cientos de productos WF ofrecidos y la industria experimenta un crecimiento de 20 a 30 % por año e indica un rol importante en el presente y en el futuro. Una lista de todos los productos estaría incompleta y lejana de la realidad, a continuación se mencionan algunos WF comerciales actuales y sus características con respecto al cambio dinámico.

WF comerciales de producción

Los WfMS Stafware de Staffware Plc, WebSphere MQ y Business Integration de IBM proporcionan funciones de soporte de procesos poderosas pero tienden a ser inflexibles. No soportan ni cambios estructurales ni cambios ad-hoc de instancias.

BEA ofrece productos y servicios para aplicaciones críticas de negocios, su actual plataforma BEA WebLogic® 9.2, que incluye BEA WebLogic Server® 9.2, BEA WebLogic Portal® 9.2, BEA WebLogic® Integration 9.2, BEA Workshop™ for WebLogic 9.2, y BEA JRockit® 5.0, es una plataforma sólida en la industria para aplicaciones industriales y servicios SOA. Sus aplicaciones son nativas en JAVA, soporta estándares de Internet y de servicios Web (BEA, 2006). BEA WebLogic® Integration 9.2 se utiliza principalmente para la administración de procesos de negocio, posee características para el diseño de procesos, automatización y monitoreo, soporta el manejo automático de excepciones y permite terminar, modificar o suspender instancias en ejecución mediante intervención humana. Permite el monitoreo a detalles de las instancias y soporta la administración de múltiples versiones.

BizTalk Server 2006 (Microsoft 2007) es un servidor de administración de procesos de negocio que habilita a las compañías a integrarlos, automatizarlos y optimizarlos, contiene herramientas para diseñar, implantar, monitorear en tiempo real y administrar procesos. La versión 2006 es la cuarta del producto (2000, 2002, y 2004) e incluye mejoras que permiten a un desarrollador crear soluciones más flexibles para integrar procesos de negocio, además, toda la funcionalidad de administración es concentrada en una consola que permite a los usuarios configurar, desplegar, iniciar y parar aplicaciones entre múltiples servidores. Tiene soporte para XSLT (Extensible Stylesheet Language Transformations), WS-I (Web Services-Interoperability) y BPEL (Business Process Execution Language). Permite la administración de múltiples versiones de casos y se recomienda mucho cuidado para ello, no soporta un cambio estructural ni la migración de instancias.

Oracle BPEL Process Manager (Oracle 2007) es parte de la solución de Oracle Business Integration y fácil de usar para crear, implantar y administrar procesos comerciales a través de aplicaciones tanto con pasos de WF manuales como automatizados, todo en una arquitectura orientada a servicios. Tiene soporte nativo para estándares como BPEL, XML, XSLT, XPATH, JMS, JCA y servicios Web, y aprovecha al máximo las características sofisticadas de sus plataforma subyacente de Oracle Application Server 10g como la seguridad, escalabilidad y alta disponibilidad. No soporta el cambio dinámico en la estructura, si se requiere una redefinición se administran múltiples versiones, las instancias en ejecución permanecen en el esquema anterior y las nuevas instancias se ejecutan en la nueva versión.

InConcert de TIBCO, SER Workflow Server de SER Solutions y FileNet Ensemble de FileNet Corporation permiten adaptaciones en tiempo real de instancias en progreso, sólo cambios ad-hoc de instancias no de estructura. Por ejemplo, los usuarios pueden insertar o borrar actividades dinámicamente de una instancia, pero no se permiten cambios en el pasado. El problema con estos motores es que las modificaciones dinámicas deben ser realizadas por expertos ya que se deben verificar los efectos

resultantes y colaterales del cambio, esto limita la utilización práctica para usuarios finales.

2.3.2 Composición WF

La composición WF es una propuesta para generar WF flexibles que puedan adaptarse al cambio dinámico y considerarse como una variante de solución. Consiste en generar WF complejos mediante módulos WF simples, estandarizados y re-usables, almacenados en un repositorio WF. El objetivo principal de que sean lo más flexibles posibles, es que en un cambio dinámico se reemplacen solamente los módulos simples afectados y no afecte a todo el WF, por lo mismo, es importante verificar si existen dependencias entre módulos.

La composición WF se enfoca en crear WF de aplicaciones, comercio electrónico, negocios electrónicos y servicios electrónicos, que permitan adaptarse en tiempo real a las necesidades de consumidores, proveedores y otros actores en la cadena de valor de una empresa (Juan et al. 1998) (Cardoso and Sheth 2002) (Aalst 2003a) (Aalst et al. 2003b) (Hamadi and Benatallah 2003) (Kalakota and Robinson 2003) (Laukkanen and Helin 2003) (Sykara 2003) (Kassam and Kontogiannis 2004) (Mnaouer et al. 2004) (Skogan 2004) (Albert 2005) (Pankratius and Stucky 2005) (Shafiq et al. 2005).

Actualmente la composición esta enfocada en los servicios Web y se denomina composición WF de servicios Web. El enfoque es un área en desarrollo y no esta directamente relacionado con el concepto propuesto por Aalst (2000) denominado composición de redes WF. El primero esta orientado a servicios Web mientras que el segundo esta orientado a la transformaciones de actividades en nuevas redes WF.

Con respecto a la composición WF de servicios Web, Cardoso y Seth (2002) se enfocan en dos aspectos principales: el descubrimiento de servicios Web y en su interoperabilidad a la hora de integrarlo. La composición la realiza el sistema determinando y evaluando el mejor mapeo entre las entradas y salidas del servicio Web y

su integración en WF. Un aspecto importante es que no aborda la correctes del WF a la hora de integrar un servicio Web.

Zhuge (2003), propone una de las primeras metodologías para desarrollo de sistemas WF basados en componentes. El enfoque incluye las partes fundamentales de la composición de WF actual: definiciones de componentes WF, métodos para la composición de componentes WF, re-uso y estrategias de similaridad entre ellos. La composición se realiza a través de cuatro reglas básicas y asume que la correctes de los componentes ya fue verificada por expertos o simulación. Aunque el enfoque presenta ventajas sobre los WF tradiciones, tales como una mayor adaptabilidad y menor complejidad para el desarrollo de WF, no incluye el aspecto dinámico en su composición, con instancias en ejecución, sino que es un enfoque de construcción estática.

Laukkanen y Helin (2003) proponen un modelo estableciendo cuatro pasos para la composición WF de servicios Web, se enfocan principalmente en el reemplazo de servicios Web no disponibles en WF, dejan para investigación futuro el aspecto dinámico que sería extender un WF existente o crear uno nuevo.

Kassam and Kontogiannis (2004) presentan una arquitectura de ingeniería de software para composición WF, mediante la satisfacción de restricciones, se basan en umbrales de tolerancia para minimizar el costo de restricciones violadas.

Albert et al. (2005) presentan un meta modelo para composición WF utilizando UML2 (Lenguaje de Modelación de Objetos) y OCL (Lenguaje de Restricción de Objetos) para las restricciones (Poole at al. 2003). El metamodelo no está completo en el sentido de que no soporta muchas construcciones WF (Aalst et al. 2003c).

Doshi et al. (2004) presenta un enfoque bastante interesante y más relacionado con esta investigación, utilizan procesos de Markov y un modelo de aprendizaje bayesiano para composición WF de servicios Web. Su algoritmo genera WF que son robustos a servicios Web con comportamientos no determinísticos y adaptivos a

ambientes dinámicos. El modelo se encuentra en etapa de pruebas con respecto a WF en el mundo real.

Pankratius y Stucky (2005) presentan una base formal para composición WF utilizando redes de Petri, se enfocan en tres aspectos relacionados con la composición: álgebra de WF para crear uno a partir de otros modelos, concepto de vistas WF y normalización WF. En este último aspecto analizan dependencias de flujo en las redes de Petri y establecen criterios para una buena estructura WF totalmente correcto sin redundancias. La propuesta es interesante pero sólo se enfoca en el análisis de la estructura de los WF, no en su dinámica (con casos).

La composición WF de servicios Web es un área en desarrollo con ventajas de flexibilidad pero con limitaciones en cuánto a restricciones en el cambio dinámico, mapeos entre módulos – proceso de negocio y principalmente en la verificación de procesos de negocio.

2.4 Conclusiones del capítulo

En este capítulo se presentó el concepto principal de la tesis: Workflow. Se presenta en qué consiste un WF, los componentes que incluye un sistema de administración WF y el modelo de referencia para su implantación.

El modelo de referencia WF se utiliza para construir una vista abstracta de los procesos de negocio en términos de sus características principales separando las tecnologías que podrían usarse en el mundo real. La especificación de las interfases permite la interoperabilidad del modelo abstracto con el de los fabricantes. Este enfoque ha tenido gran valor sobre los años debido principalmente a que la tecnología ha evolucionado y por lo mismo diferentes especificaciones de interfases se han definido conforme a las tecnologías de ese tiempo, sin la necesidad de cambiar el modelo de referencia.

La ventaja de la modelación WF viene de la posibilidad de tener un control automático de todo proceso de negocio. Esto es simple cuando el ambiente del proceso es estático pero se complica cuando el ambiente se torna dinámico que requiera un cambio

estructural en el WF. Dicha complicación se debe principalmente a que puede surgir el tan denominado error de cambio dinámico y requiere que se resuelvan los siguientes problemas:

- ¿Cómo garantizar la correctes estructural después de un cambio en el WF?
- ¿Cómo migrar los casos cuando ocurre un cambio estructural?
- ¿Cómo crear WF dinámicos que soporten los cambios?

En la actualidad existen muchos enfoques de investigación para la solución a estos problemas, todos tienen sus ventajas y sus desventajas. Aunque no existe una herramienta comercial que incluya dinamismo en sus WF, lo importante es que el problema se está abordando y cada enfoque que surge para resolverlo es un paso más hacia su solución completa.

Capítulo 3 Selección de la Formalización del Modelo

3.1 Requerimientos para la modelación

La tesis consiste en diseñar un modelo para construir dinámicamente WF que representen procesos de negocio, de manera correcta de una empresa. La construcción se realiza formando un WF mediante la inserción de patrones que representan actividades. Una actividad o un conjunto de actividades que pertenecen a un patrón pueden representar un servicio electrónico y depende del dominio de conocimiento del WF.

La construcción dinámica genera dos problemas que se deben solucionar:

1. Asegurar que la inserción de un patrón no genere nuevos problemas, es decir, el WF permanezca correcto.
2. Seleccionar el siguiente patrón a aplicarse en la construcción del WF.

El primer problema se puede resolver por medio de un proceso general de construcción que vaya construyendo la red WF a partir de patrones de red básicos con el fin de mantener la correctes a medida que se va construyendo el WF, es decir, que se vaya verificando en paralelo con la construcción. El proceso general de construcción genera propiedades de actualización de estado que aseguran el estado correcto del WF y pueden utilizarse como reglas para el cálculo del siguiente patrón.

El proceso general de construcción y los patrones de red básicos de inserción deben diseñarse con lenguajes de modelación formal que puedan representar la semántica del WF y den soporte de modelación visual y textual. La selección de los lenguajes es de capital importancia ya que ambos deben de poder representar la semántica del modelo de manera correcta y ser capaces de detectar inconsistencias. El uso de formalismos visual y textual es innovador en la modelación WF y permite que sea robusta, verificable y correcta garantizando la solución del primer problema.

El segundo problema consiste en seleccionar el siguiente patrón a aplicarse en el WF, y para su solución se requiere de un sistema que lo seleccione con base en el dominio de conocimiento de la red a partir de las actividades ejecutadas previamente. Es claro que en todo WF va haber una actividad de inicio a partir de la cuál se va construir la

red. El sistema para la selección del siguiente patrón deberá ser capaz de inferir con base en las actividades previas y cumpliendo con las propiedades de actualización de estado proporcionadas por el proceso general de construcción.

En el contexto de construcción dinámica, el conocimiento de negocios de una empresa consiste en la representación de sus procesos y capacidades de negocio mediante un conjunto de actividades y servicios, representados por las redes WF y seleccionados por las reglas de negocio (Debevoise 2005).

3.2 Selección de los formalismos de modelación

La construcción dinámica de WF implica realizar cambios dinámicos a medida que se va construyendo la red, por lo mismo, se requiere adaptar la instancia que se esté ejecutando al nuevo esquema. Para ello, se requieren de reglas y patrones que garanticen una construcción correcta del esquema WF y una adaptación consistente de la instancia en dicho esquema. La construcción de WF correctos es parte fundamental de esta tesis, por lo mismo se requieren de formalismos bien establecidos para la modelación, con base en esto, es importante tener representaciones del WF en construcción tanto visual como textual

Para la selección de los formalismos, se deben cumplir los siguientes cuatro aspectos fundamentales:

1. Soportar los patrones básicos de construcción: secuencia, selección y paralelismo.
2. Capacidad de modelar el aspecto estructural y dinámico
3. Soportar flexibilidad: cambio y migración de instancias
4. Verificación de la consistencia del WF a medida que se va construyendo.

En el caso de lenguajes de modelación visual WF, existen varios enfoques sustentados por grupos de investigación (Murata 1989) (Ellis and Rozenberg 1995) (Kiritsis, D. et al. 1998) (Agostini and DeMichelis 2000) (Casati et al. 2000) (Weske 2001) (Aalst and Kees 2002) (Marinescu 2002) (Dehnert and Aalst 2004) (Skogan et al.

2004) y comparaciones entre los enfoques (Janssen et al. 1997) (Aalst et al. 2003a) (Dehnert and Aalst 2004).

En (Aalst et al. 2003a) se presenta una comparación entre cinco enfoques, que resumen la mayoría de los por grupos de investigación de modelación visual WF: UML, OCoN, WF graphs, WF nets y Evolution WF. Se presentan las fortalezas y debilidades en cuanto a las perspectivas funcional, de procesos, organizacional, información y operacional; también de flexibilidad en el cambio y migración de instancias; y además de análisis de validación, verificación y rendimiento. El artículo es genérico y sólo se enfoca en la comparación de los cinco enfoques como una guía para el usuario en la selección de un lenguaje de modelación WF.

Evaluando los cinco enfoques con los cuatro aspectos fundamentales para la construcción dinámica resulta en lo siguiente: UML no cumple con los aspectos 2, 3 y 4; OCoN que es una combinación de UML y redes de Petri de alto nivel cumple con 3 y 4, aunque con extensiones cubre el 4; WF graphs no soporta el 4, Evolution WF con extensiones cumpliría con 4 y las redes WF con extensiones cumpliría con 3 (ver Tabla 3.1).

Aspecto\Enfoque	Patrones	Modelación	Flexibilidad	Verificación
UML	✓	X	X	X
OcoN	X	X	✓	-
WF graphs	✓	✓	✓	X
WF nets	✓	✓	-	✓
Evolution WF	✓	✓	✓	-

Tabla 3.1. Comparación de cinco enfoques de modelación actual.

En el caso de la composición WF, se utilizan muchos estándares y enfoques específicos de investigadores y vendedores para la modelación de servicios Web. La mayoría de los vendedores enfocan su solución en la facilidad de uso, cada uno propone su conjunto de símbolos y semántica similar al de los otros, pero sin la fortaleza del lenguaje o modelo formal. Havey (2006) discute que el flujo de control es pocas veces usado por los vendedores, quienes enfatizan más en la facilidad de programación que en la precisión semántica, propone que para construir soluciones satisfactorias se debe insistir en saber exactamente cómo un proceso representado por un WF se ejecutará, y concluye, que los lenguajes más robustos para el control de flujo, son los basados en

redes de Petri. Aunque actualmente existen muchos lenguajes estándar de composición de servicios Web – PDL, XPDL, BPSS, EDOC, BPML, WSDL, WSCI. BPMN y BPEL4WS- Aalst (2003a) (Aalst et al. 2003b) también discute, que no son lo bastante maduros para ser aplicados debido a que son usados por productos concretos y/o intereses comerciales. Propone el uso de técnicas de modelación de procesos comprobadas que combinan expresividad, simplicidad y semántica formal tal como redes de Petri y álgebra de procesos (Glance et al. 1996) (Aalst et al. 2003a) (Girault and Valk 2003) (Dehnert and Aalst 2004). Para una comparación detallada de los estándares ver (Aalst-URL, 2006).

Con base en lo anterior y en el contexto de la construcción dinámica dónde cada WF construido dinámicamente va a representar un caso, es decir, se requiere una red por caso, el enfoque de modelación visual que se adecua a las necesidades de la tesis son las redes WF – redes de Petri clásicas para la representación de WF (Murata 1989) (Aalst and Kees 2002) (Aalst 2004) (Dehnert and Aalst 2004) bajo los siguientes puntos que lo sustentan:

1. Permite los patrones básicos de construcción: secuencia, selección y paralelismo (Murata 1989) (Aalst and Kees 2002).
2. Contiene una semántica operacional para representar el aspecto estructural y dinámico de cada caso, ver tabla 2.1.
3. Soporta cambios y migración de cada instancia, ver tabla 2.3.
4. No requiere de bitácoras de ejecución de instancias para verificación ya que el WF se va construyendo verificado, es decir, el WF siempre es correcto a medida que se va construyendo, ver tabla 2.2.
5. El uso de una red por token, pueda verse como una limitación, se requiere de ello para modelar servicios de acuerdo a las necesidades de cada empresa. Por lo mismo no se requiere el uso de redes de Petri coloreadas.

Para el caso de formalismos textuales, aunque existe poco trabajo previo, se tienen gramáticas bien establecidas y usadas en la actualidad. Glance et al. (1996) propone una gramática textual para representación flexible de trabajo, el enfoque utiliza

restricciones y reglas para la definición de procesos. La flexibilidad en la especificación emerge de la negación de las restricciones –todo lo no excluido por las reglas gramaticales es permitido. El enfoque no aborda el aspecto dinámico, por ejemplo, cambiar la gramática durante la ejecución, o la posibilidad de construcción dinámica de WF verificados, y no hay referencias de que se haya implantado. Es perfectamente comprensible ya que la aproximación se diseño en 1996, cuando apenas se abordaban problemas de cambio dinámico en un proceso (Ellis and Rozenberg 1995).

BNF denominado Backus-Naur Form es una meta-sintaxis usada para expresar gramáticas contexto libre, un formalismo para describir lenguajes formales y fue diseñado para crear las reglas del lenguaje Algol 60. BNF se usa como una notación para las gramáticas de lenguajes de programación para las computadoras, protocolos de comunicación y para representar partes de gramáticas de lenguaje natural. Actualmente muchos libros de texto de teorías de lenguajes programación documentan el lenguaje en BNF incluyendo nuestra propia institución (Aho et al. 1998) (Giarratano and Riley 2001). También existen variantes tales como ABNF (*Augmented BNF*) que tiene su propia documentación y propósito. Su uso actual, su difusión y sus resultados comprobados como formalismo matemático, son aspectos clave para utilizarlo como lenguaje de modelación textual, complementando la modelación visual.

Las redes de Petri no sólo se han usado para construir nuevos lenguajes WF, también se usa para modelar semántica de control de flujos complejos y cadenas de procesos manejadas por eventos. Incluso estándares tales como BPEL, BPMN, WSFL y UML se basan en las redes de Petri para la modelación de flujos de control. En conjunto con BNF, permiten modelación robusta y verificable tanto visual como textualmente. Justificación suficiente y necesaria para usarlos como formalismos de modelación para la construcción dinámica.

La construcción dinámica de WF se enfoca en la estructura y la semántica de los WF, utiliza redes de Petri para el control de flujo y semántica visual, y BNF para la semántica textual. El uso de estos formalismos permite crear módulos WF que pueden servir para futura composición WF. Esto genera una ventaja, ya que es una base para la construcción de módulos de composición WF dónde cada módulo generado está debidamente verificado por los formalismos utilizados para su construcción.

3.3 Sistema para la selección del siguiente patrón

La construcción dinámica implica la inserción de patrones en la red que se esta construyendo a medida que se ejecuta la última transición de dicha red, es decir, la ejecución de la última transición va a disparar el cálculo del nuevo patrón a insertar. Cada patrón debe seleccionarse con base en las características del modelo de construcción que relaciona las actividades que se insertan con respecto a las ejecutadas anteriormente. Para ello se requiere un sistema que represente el área de conocimiento de la red que se está construyendo que relacione el siguiente patrón mediante enunciados no ambiguos denominados reglas.

Un sistema de negocios, es un sistema automatizado, desarrollado para soportar la operación de un negocio o de una capacidad de negocio. El motor para el uso y desarrollo de los sistemas de negocios siempre será la necesidad de negocio (Kalakota and Robinson 2003) (Ross 2003) (Thomson and Strickland 2004). Las personas de negocios deben concentrarse en esta necesidad de maneras muy concretas, en la práctica, se denomina análisis de negocio y consiste en aplicar las habilidades y técnicas necesarias para crear un modelo de negocios. El modelo de negocios proporciona un entorno completo y comprensivo en el cual un negocio planifica en cómo servir a sus clientes, generar ingresos y beneficios a través de estrategias y ventajas competitivas (Ross 2003) (Thomson and Strickland 2004). Un ingrediente fundamental en un modelo son las reglas de negocio y en conjunto con la necesidad de negocio, factor principal en un sistema de negocios, son la forma de encarar los retos del siglo XXI. Las reglas de negocio deciden con la información, si cambian las reglas, cambia la información, una regla de negocio puede ser una política, una restricción o un requisito regulatorio de ley.

El enfoque de las reglas de negocio es una técnica de diseño para formalizar las reglas de negocio críticas de una empresa en un lenguaje que el gerente y el ingeniero en TI entienden a la perfección. Las reglas crean un enunciado no ambiguo de lo que un negocio hace con información para decidir una proposición (Debevoise, 2005), el lenguaje utilizado debe ser formal para que sea legible a las TI, sus cláusulas incluyen términos, datos, procedimientos y procesos. Las reglas de negocio (también denominadas

lógica de negocio) son pasos hacia la innovación tecnológica enfocados en las capacidades de conocimiento de los negocios (Ross 2003).

En la actualidad, las barreras tecnológicas se han ido eliminado –sistemas monolíticos- y los servicios "*plug-in*" son cada vez más fáciles de incorporar – mediante plataformas de servicio. El conocimiento de negocios descrito en reglas puede incorporarse en plataformas de software. Es importante recalcar que las reglas de negocio tienen un amplio alcance y que no todas ellas pueden ejecutarse por un sistema, por ejemplo, reglas simples o comunes que requieren una interpretación por parte de las personas son difíciles de interpretar por una aplicación.

Ross (2003) menciona tres tendencias en que la tecnología comercial de reglas de negocio se ofrecerá:

1. Componentes independientes para incorporar en una arquitectura de cómputo apropiada.
2. Acoplada con un motor WF.- el resultado no sólo permite un control WF más sofisticado si no también toma de decisiones automatizada.
3. Incluidas en otros paquetes de SW.- muchos tipos de SW proporcionan nichos naturales para la tecnología de lógica de negocio tales como CRM y ERP.

El enfoque de las reglas de negocio es un área cuya base es el éxito en el negocio, no en la tecnología. Entre los problemas que las reglas de negocio resuelven se encuentran:

- Reglas ad-hoc.- pensar en reglas no en hechos, evitando confusión, contradicción e ineficiente operación.
- Mala comunicación.- los malos entendidos en los conceptos clave de negocios resultan en una mala comunicación.
- Diferenciación masiva.
- La necesidad de mantenerse actualizado.
- Administración del conocimiento.

Tradicionalmente los sistemas basados que administran reglas se denomina sistemas expertos y están enfocados en tratar de reemplazar el conocimiento (lo experto) humano en ciertas áreas y en automatizar o codificar prácticas de negocio u otras actividades que son parte de toda empresa. Tales sistemas también son usados cotidianamente para ordenar suministros, monitorear procesos industriales, rutear llamadas telefónicas y procesar formas Web. Actualmente están bastante difundidos (Giarratano 2001) (Friedman-Hill 2003) (Ross 2003) (Chisholm 2004), sus aplicaciones abarcan cualquier área desde filtraje de correo a configuración de órdenes y desde monitoreo de plantas químicas hasta diagnóstico de problemas médicos. Se enfocan principalmente en la solución de problemas que son difíciles de resolver usando métodos algorítmicos tradicionales y trabajan aún cuando los datos de entrada son incompletos, esto los hace la mejor opción para la toma decisiones de negocios (Giarratano, 2001) (Ross, 2003) (Debevoise, 2006).

Los sistemas expertos son herramientas importantes para representar el dominio de conocimiento de un proceso dada su capacidad de inferir con base a reglas establecidas. Facilitan la toma de decisiones de forma rápida y concisa gracias a su motor de inferencias lo que permite seleccionar y/o disparar los siguientes nodos con base en anteriores, aspecto necesario para la solución del segundo problema de la construcción dinámica.

Como el enfoque de las reglas de negocios es formalizar las reglas importantes de una empresa en un lenguaje sin confusiones, la combinación de una representación formal de WF con redes de Petri y BNF con un sistema no ambiguo para representar el conocimiento de una empresa, son las herramientas ideales de modelación por su carácter formal y sin imprecisiones a la hora de representar los procesos de negocio de una organización (Ross 2003).

En esta tesis para la implantación del modelo se utilizará el cuestionario de un sistema denominado DIACNE (capítulo 5), para ello, el proceso de implantación requerirá migrarlo a un sistema experto que permita representar el conocimiento mediante reglas y poder construir con el modelo de construcción dinámica WF correctos que resuelvan las desventajas que en la actualidad tiene DIACNE.

3.4 Metodología

- La metodología de investigación es la siguiente (Mauch and Birch 1993):
  - Teórica
    - Desarrollo del modelo para la construcción dinámica de WF
  - De diseño y demostración
    - Construcción de un sistema experto para la implantación del modelo.
- Validación
  - La validación del modelo se realiza de manera formal.

El desarrollo de la tesis consiste en dos etapas: la primera consiste en el desarrollo del modelo genérico para la construcción dinámica de WF y en la segunda etapa se definirá e implantará el modelo.

El desarrollo del modelo consiste en la realización de los siguientes puntos:

1. Definición de los patrones de construcción básicos, dichos patrones se refieren a la trayectoria que las instancias pueden seguir a los largo del WF.
2. Definición y validación del proceso general de construcción dinámica.
3. Definición de las propiedades básicas de actualización de estado para la construcción dinámica.
4. Definición de la gramática de contexto libre para la representación de WF
5. Formulación de un mapeo entre los formalismos de redes WF y BNF.
6. Implantación del modelo mediante un sistema experto para representar el conocimiento de negocios.

3.5 Justificación

El propósito de esta investigación es permitir modelar a las PYMEs sus procesos de una manera segura y correcta, de tal manera que se enfoque más en el valor de sus servicios que en la tecnología para implantarlos. Dicha modelación tiene el beneficio clave de que la empresa puede modificar la arquitectura tecnológica sin

cambiar los servicios disponibles. El modelo sienta las bases para construir WF que representen el conocimiento de negocios mediante WF de una PYME de manera consistente y se adapte a las necesidades del mercado y al ambiente dinámico actual. La investigación beneficia además, a toda PYME en México que quiera incursionar en los negocios electrónicos y formalizar sus servicios a través de Internet.

3.6 Aportaciones

La investigación tiene las siguientes aportaciones:

- Representar procesos de negocio en forma dinámica mediante construcción dinámica.
- Aplicación de la construcción dinámica a la composición de redes WF (Aalst 1997a)
- Bases para la composición WF de servicios Web mediante la generación de patrones básicos de WF y la adaptación de las instancias.
- Innovación en la creación de un mapeo sintáctico y semántico entre el formalismo de modelación visual y el formalismo de modelación textual.
- Definición y validación de un proceso de construcción dinámica de WF.
- La representación de los WF en modelación textual permite poder verificar de manera automatizada su correctes, tal y como lo realizan los compiladores.
- El modelo está orientado tanto para el desarrollo de arquitecturas de procesos de negocio como de servicios Web. De esta manera se evita el peligro de ignorar a futuro aspectos organizacionales a favor de modelar recursos basados enteramente en los servicios Web.

3.7 Conclusiones del capítulo

En este capítulo se han presentado los requerimientos para el desarrollo del modelo. La selección de los formalismos de modelación se realizó con base en las necesidades que se presentan al momento de construir WF en forma dinámica. Las dos técnicas de modelación, redes de Petri y BNF, seleccionadas para la especificación de WF están basadas en teorías formales robustas y que pueden ser analizadas

matemáticamente. Ambas técnicas debido a su fundamento formal definen de forma exacta los estados de un proceso lo que facilita el uso de patrones para las trayectorias de un caso y permite a su vez su transferencia de una definición a otra y/o su cancelación. La construcción dinámica garantiza la creación de WF correctos de cualquier red de Petri cumpliendo con los cuatro aspectos fundamentales descritos en la sección 3.2.

También se presenta en le capítulo la selección del sistema que soluciona el segundo problema que toda construcción dinámica presenta. Los sistemas expertos actualmente son herramienta eficientes y con mucha rapidez para inferir. En el capítulo cinco se detalla el tipo sistema experto que se usó y en que consiste la aplicación que se migró al sistema experto para implantar el modelo de construcción. Por último, se presenta la metodología, la justificación de tesis y las aportaciones de valor teórico de la tesis, aspectos de capital importancia en el desarrollo de la investigación.

Capítulo 4 Modelo para la Construcción Dinámica de Workflows

4.1 Introducción

La modelación de la construcción dinámica es realizada mediante dos formalismos, el primero modela visualmente mediante redes WF basadas en redes de Petri clásicas, el segundo, modela textualmente mediante BNF. El diseño del modelo consiste en definir los patrones de construcción básicos, la definición de un Workflow BNF, el mapeo de BNF con redes de Petri y la definición del proceso general de construcción dinámica para que los WF construidos sean correctos sintácticamente y semánticamente. También se añaden cuatro propiedades para facilitar la actualización del estado de los WF y se definen las acciones semánticas para la actualización del estado en el WF BNF.

En este capítulo se detallan los formalismos para el modelo DCW desarrollado en esta tesis. Dicho modelo consiste de diez rubros separados en cinco patrones simples, cuatro propiedades y un proceso general de construcción que permiten de manera correcta todos juntos la construcción dinámica de WF también denominada construcción de WF dinámicos y que en este capítulo se usará indistintamente. En las primeras dos secciones se presentan los dos formalismos de modelación, en la tercera sección se describe el diseño del modelo de construcción dinámica denominado DCW e incluye la validación del proceso general de construcción, en las siguientes dos secciones se da un ejemplo de construcción y cómo puede implantarse la composición de redes WF a partir de la construcción. Posteriormente, se presenta la validación del modelo, su comparación con respecto al estado del arte y por último, se presentan las conclusiones del capítulo.

4.2 Redes WF

Las redes WF, están basadas en redes de Petri clásicas, son una herramienta de modelación matemática y gráfica aplicable a muchos sistemas caracterizados por ser concurrentes, asíncronos, distribuidos, paralelos, no determinísticos y/o estocásticos (Murata 1989). Su uso constituye un buen punto de inicio para un fundamento teórico

sólido y formal de la administración WF (Aalst et al. 2003a), debido a que previene ambigüedades, incertidumbres y contradicciones, aspectos relevantes en este modelo.

Redes de Petri

Una red de Petri clásica es un grafo dirigido bipartito con dos tipos de nodos denominados plazas y transiciones. Los nodos están conectados por arcos dirigidos y no se permiten conexiones entre nodos del mismo tipo. Las plazas se representan por círculos y las transiciones por rectángulos.

Formalmente[1]

Definición 1 (Red de Petri). Una red de Petri es una cuádrupla PN = (P, T, F, Mo) dónde

$P = \{p_1, p_2, p_3, \ldots, p_n\}$ es un conjunto finito de plazas

$T = \{t_1, t_2, t_3, \ldots, t_m\}$ es un conjunto finito de transiciones

$F \subseteq (P \times T) \cup (T \times P)$ es un conjunto de arcos que representan la relación de flujo

$Mo: P \rightarrow \{0, 1, 2, \ldots\}$ es el marcaje inicial denominado estado inicial.

dónde

$$\forall p \in P, \bullet p \cup p \bullet \neq 0$$

$$\forall t \in T, \bullet t \cup t \bullet \neq 0$$

La siguiente notación se utiliza para propósitos de definiciones conocidas de redes de Petri descritas más adelante:

$\bullet t = \{p | (p, t) \in F\}$= el conjunto de todas las plazas de entrada de t

$t \bullet = \{p | (t, p) \in F\}$= el conjunto de todas las plazas de salida de t

$\bullet p = \{t | (t, p) \in F\}$ = el conjunto de todas las transiciones de entrada de p

$p \bullet = \{t | (p, t) \in F\}$ = el conjunto de todas las transiciones de salida de p

[1] Las definiciones en letra itálica son referencias de autores con las cuáles se basa este trabajo. Las definiciones en negrillas son aportaciones de la tesis.

- *La plaza p se denomina plaza de entrada de una transición t si y sólo si existe un arco dirigido de p a t (p, t).*
- *La plaza p se denomina plaza de salida de una transición t si y sólo si existe un arco dirigido de t a p (t, p).*

En el contexto de WF, los arcos sólo pueden ser de 0 o 1 de peso ya que sólo corresponden a condiciones. El estado de una red se denomina marcaje y es la distribución de tokens sobre las plazas, toda plaza en cualquier momento puede tener cero o un token, dibujados como puntos negros. El vector de estados M = (m1, m2, m3, …, mn) indica por cada plaza si existe un token, de este modo, la notación m(pi) especifica el número de tokens en la plaza pi. Para dos estados M1 y M2, M1 ≤ M2 si y sólo si para todo p ∈ P: M1(p) ≤ M2(p).

Las transiciones son los componentes activos de una red de Petri y cambian el estado de la red de acuerdo a las siguientes reglas de disparo:

1. *Una transición t está habilitada si y sólo si cada plaza de entrada p de t contiene un token.*
2. *Una transición habilitada se puede disparar. Si una transición t se dispara, entonces consume un token de cada plaza de entrada p de t y produce un token por cada plaza de salida p de t.*

Dado un estado inicial M1 en una red de Petri

$M1 \xrightarrow{t} M2$: *la transición t habilitada y disparada en M1 resulta en el estado M2*

$M1 \rightarrow M2$: *existe una transición t tal que* $M1 \xrightarrow{t} M2$

$M1 \xrightarrow{\alpha} Mn$: *la secuencia de disparo* $\alpha = t_1t_2t_3\ldots t_{n-1}$ *guía del estado M1 al estado Mn*

Un estado Mn es denominado alcanzable desde M1 ($M1 \xrightarrow{*} Mn$*) si y sólo si existe una secuencia de disparo* $\alpha = t_1t_2\ldots t_{n-1}$ *tal que:*

$$M1 \xrightarrow{t1} M2 \xrightarrow{t2} M3 \xrightarrow{t3} \ldots \xrightarrow{tn-1} Mn$$

A continuación se definen propiedades de las redes de Petri relacionadas con sus propiedades estáticas y dinámicas (definiciones 2-11) obtenidas de (Murata, 1989) y (Aalst

1997a). Dada una red de Petri con un estado inicial M definido como (PN, M). Un estado M' es un estado alcanzable de (PN, M) si y sólo si :

$M \xrightarrow{*} M'$

Definición 2 (Red de Petri viva: Live). Una red de Petri (PN, M) está viva si y sólo si, para todo estado alcanzable M' y toda transición t existe un estado M'' alcanzable desde M' que habilita a t.

Una red de Petri está viva, cuando es posible disparar cualquier transición iniciando desde el marcaje inicial M.

Definición 3 (Red de Petri limitada y segura: Bounded and safe). Una red de Petri (PN, M) está limitada si y sólo si, para todo estado alcanzable y toda plaza p el número de tokens es limitado. La red es segura si y sólo si por cada plaza el número máximo de tokens no excede 1.

Una red de Petri está limitada si existe algún entero n con la propiedad de que en cualquier secuencia de disparo ningún lugar recibe más de n tokens, en este caso uno.

Definición 4 (Red de Petri bien formada : Well-formed). Una red de Petri está bien formada si y sólo si existe un estado M tal que (PN, M) está viva y limitada.

Definición 5 (Red de Petri fuertemente conectada: Strongly connected). Una red de Petri está fuertemente conectada si para cada par de nodos x, y existe una trayectoria dirigida de x a y.

Fuertemente conectada significa que hay una trayectoria dirigida de un elemento $x \in p \cup t$ a cualquier otro elemento $y \in p \cup t$. Ésta es una propiedad estática de la red.

Definición 6 (Red de Petri: Free-choice). Una red de Petri es free-choice si y sólo si por cada dos transiciones t1 y t2, $\bullet t1 \cap \bullet t2 \neq 0$ implica que $\bullet t1 = \bullet t2$.

Una red free-choice es una red tal que todo arco desde una plaza es un único arco de salida o un único arco de entrada a una transición:

Para todo $p \in P$, $|p\bullet| \leq 1$ or $\bullet(p\bullet) = \{p\}$; o en forma equivalente
Para todo $p1, p2 \in P$, $p1\bullet \cap p2\bullet \neq 0 \Rightarrow |p1\bullet| = |p2\bullet| = 1$.

Las redes de Petri free-choice son una generalización de las máquinas de estado y los grafos marcados (Murata 1989) lo que implica, que permiten la estructura de selección y la de paralelismo.

Redes Workflow

Una red de Petri que modela el flujo de control de un WF se denomina red WF. La representación de WF mediante redes de Petri se realiza a través de la asociación de tareas con transiciones, condiciones con plazas y casos con tokens. Una red WF siempre tiene dos plazas especiales, una plaza de inicio, pI que corresponde al estado inicial cuando el caso es aceptado para procesar y un estado final pE, que corresponde al estado cuando el procesamiento del caso ha finalizado exitosamente. También requiere que todo nodo (plaza o transición) esté localizado en una trayectoria de pI a pE. Todos estos requerimientos se traducen en las siguientes definiciones:

Definición 7 (Red WF). Una red de Petri PN = (P, T, F, Mo) es una red WF W si y sólo si:

1. *Sólo tiene una plaza inicial $\bullet pI \in P$ y una plaza final $pE\bullet \in P$ tal que $\bullet pI = 0$ y $pF\bullet = 0$.*
2. *Toda plaza o transición está en una trayectoria de pI a pF resultando en una red fuertemente conectada si añadimos una transición t* ($\bullet t^* = pE$ y $t^*\bullet = pI$) a W que conecte a pE con pI.*

El primer requerimiento implica que una red WF tiene una plaza de entrada pI y una de salida pE y que cualquier caso administrado por el proceso representado por la red es creado al introducirse al WfMS y suprimido una vez que lo ha finalizado, es decir, la red WF especifica el ciclo de vida de un caso. El segundo requerimiento de la definición 7 (la red de Petri extendida para que sea fuertemente conectada: definición 5) establece que por cada nodo debe haber una trayectoria dirigida de la plaza pI a pE vía el nodo. Este requerimiento evita tareas y/o condiciones saltadas o no ejecutadas.

Los dos requerimientos de la definición 7 están relacionados con la estructura de un WF, y requiere complementarse con el aspecto dinámico del WF, es decir, la ejecución de casos, Aalst (1997) describe que:

- Para todo caso, el proceso terminará eventualmente y en el momento que lo haga existirá un token en la plaza pE y todas las demás plazas estarán vacías.
- Además, todas las tareas deberán haber sido ejecutadas (no hay transiciones muertas) siguiendo la trayectoria apropiada de las red WF.

estos dos requerimientos adicionales corresponden a la propiedad *soundness*.

Definición 8 (Red WF Sound). Un proceso modelado por una red WF es sound si y sólo si:

8.a) Para todo estado M alcanzable desde el estado MpI, existe una secuencia de disparo que guía del estado M al estado MpE. Formalmente:

$$\forall_M (MpI \overset{*}{\rightarrow} M) \Rightarrow (M \overset{*}{\rightarrow} pE)$$

8.b) El estado MpE es el único estado alcanzable desde el estado pI con al menos un token en la plaza pE, m(pE) = 1. Formalmente:

$$\forall_M (MpI \overset{*}{\rightarrow} M \wedge M \geq MpE) \Rightarrow (M = MpE)$$

8.c) No hay transiciones muertas en (PN, MpI). Formalmente:

$$\forall_{t \in T} \exists_{M, M'} \; MpI \overset{*}{\rightarrow} M \overset{t}{\rightarrow} M'$$

MpI indica el estado con un solo token en la plaza pI, (PN, MpI). El primer requerimiento establece que empezando en el estado inicial siempre será posible alcanzar el estado final con un token en la plaza pE. El segundo requerimiento establece que al momento que un token es posicionado en la plaza pE, todas las demás plazas deben estar vacías. Estos dos requerimientos se denominan

terminación apropiada. El tercer requerimiento establece que no hay transiciones muertas en el estado inicial MpI.

Análisis de WF para verificar la propiedad *Soundness*

Para detectar si una red WF es *sound* se pueden usar propiedades estandarizadas de las redes de Petri. También puede usarse la técnica de grafo de cobertura (*coverability graph)* que consiste en detectar estados no limitados, lo cuál indica que una red WF no es *sound*, sin embargo, a medida que los WF son más complejos la verificación se hace intratable (Aalst 1997a). La propiedad sound tiene una correspondencia con dos propiedades de redes de Petri: viva y segura, a continuación se presenta la técnica para decidir si una red WF es *sound* basado en (Aalst 1997a).

Condiciones necesarias y suficientes para soundness

Para decidir si una red WF PN = (P, T, F) es *sound*, se define primero una red extendida $\overline{PN} = (\overline{P}, \overline{T}, \overline{F})$ que se obtiene al añadir una transición extra t* que conecta pI con pE:

$\overline{P} = P$

$\overline{T} = T \cup \{t^*\}$

$\overline{F} = F \cup \{(pE, t^*), (t^*, pI)\}$

Para cualquier red WF PN arbitraria y su correspondiente red extendida $\overline{PN}$, se probará que:

PN es sound si y sólo si ($\overline{PN}$, MpI) está viva y limitada.

Definición 9. Si ($\overline{PN}$, pI) está viva y limitada, entonces PN es una red WF sound.

Prueba.

($\overline{PN}$,MpI) está viva, por cada estado alcanzable M existe una secuencia de disparo que guía a un estado que habilita a t. Como pE es la plaza de entrada de t*, se encuentra que para cualquier estado alcanzable M desde el estado MpI es posible alcanzar un estado con al menos un token en la plaza pE. Considere un estado alcanzable arbitrario M'+pE, un estado con al menos un token en la plaza pE, en este estado t* es habilitado. Si t* se dispara entonces es alcanzado el estado M'+pI. Como ($\overline{PN}$, MpI) es también limitada, M' debe ser un estado vacío. De este modo, se mantienen los requerimiento 8.a) y 8.b) y se garantiza la terminación apropiada, el requerimiento c) se cumple por el hecho de que ($\overline{PN}$, MpI) es live. Por lo tanto, PN es una red WF sound.*

Definición 10. Si PN es sound entonces ($\overline{PN}$, MpI) es limitada.

Prueba

Asuma que PN es sound y (PN, MpI) no es limitada. Como PN no es limitada existen dos estados Mi y Mj tal que MpI $\rightarrow$ Mi, Mi $\rightarrow$ Mj y Mj > Mi. Sin embargo, como PN es sound hay una secuencia de disparo α tal que:

Mi $\overset{\alpha}{\rightarrow}$ MpE

además, existe un estado M tal que Mj $\overset{\alpha}{\rightarrow}$ M y M > MpE. Consecuentemente, no es posible que PN sea sound y no limitada. Así que, si PN es sound entonces (PN, MpI) es limitada. Del hecho de que PN es sound y (PN, MpI) es limitada, se deduce que ($\overline{PN}$, MpI) es limitada. Si la transición t se dispara en PN la red regresa al estado inicial MpI.*

Definición 11. Si PN es sound entonces ($\overline{PN}$, MpI) está viva.

Prueba

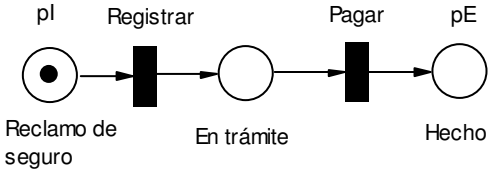
Asuma que PN es sound, por la definición 10 (PN, MpI) es limitada. Debido a que PN es sound entonces por cada estado alcanzable M' desde ($\overline{PN}$, MpI) es posible regresar al estado MpI. En la red (PN, MpI) es posible disparar una transición arbitraria t (requerimiento de la definición 8.c)) y también es el caso de la red extendida. De este modo, (PN, MpI) está viva porque para todo

estado M' alcanzable desde ($\overline{PN}$, MpI) es posible alcanzar un estado que habilita una transición arbitraria t.

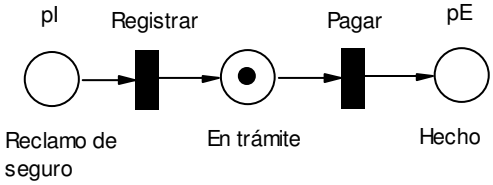
Con base en las definiciones 9, 10 y 11 una red WF es sound si y sólo su ($\overline{PN}$, MpI) está viva y limitada.

Definición 12 (Red WF correcta). Una red WF es correcta si es segura y tiene la propiedad *sound*.

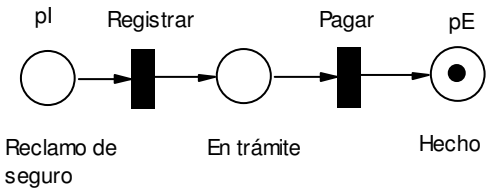
En el contexto de esta tesis, una red WF es correcta si cumple con la definición 12 y en esta tesis se le denomina correctes de un WF. La figura 4.1 muestra la ejecución de una red WF correcta. La red tiene una plaza inicial pI, una final pE, existe una secuencia de disparo que guía de pI a pE donde siempre existe un token en una plaza en todo momento y todas las transiciones se disparan. La figura 4.1a) muestra el estado inicial cuando arriba una caso (reclamo de un seguro), la figura 4.1b) muestra el estado después de registrar el caso y por último la figura 4.1c) muestra el estado final después de autorizar el pago. El proceso de negocio que representa esta red es muy simple, ya que paga todos los reclamos, pero ilustra el concepto de red WF correcta.



a) Estado inicial



b) Caso aceptado y registrado



c) Pago de seguro, caso finalizado

Figura 4.1 Ejemplo de una red WF correcta.

4.3 Gramáticas de contexto libre: BNF

En Lingüística e Informática, una gramática de contexto libre es una gramática formal en la que cada regla de producción es de la forma:

$$V \rightarrow w$$

dónde V es un símbolo no terminal y w es una cadena de terminales y/o no terminales. El término contexto libre se refiere al hecho de que el no terminal V puede siempre ser sustituido por w sin tener en cuenta el contexto en el que ocurra. Un lenguaje formal es de contexto libre si hay una gramática de contexto libre que lo genera.

Las gramáticas de contexto libre permiten describir la mayoría de los lenguajes de programación, de hecho, la sintaxis de la mayoría de lenguajes de programación está definida mediante gramáticas de contexto libre. Por otro lado, estas gramáticas son suficientemente simples como para permitir el diseño de eficientes algoritmos de análisis sintáctico que, para una cadena de caracteres dada determinen cómo puede ser generada desde la gramática.

La gramática describe de forma natural la estructura jerárquica de muchas construcciones de lenguajes de programación, por ejemplo, una proposición en el lenguaje C *if-else* tiene la siguiente forma:

if (expresión) proposición *else* proposición

puede expresarse por la siguiente producción

proposición → *if* (expresión) proposición *else* proposición

donde la palabra clave if y los paréntesis, se llaman componentes léxicos y la flecha se lee como "puede tener la forma". Las variables expresión y proposición representan conjuntos de componentes léxicos y se llaman no terminales.

Definición 13 (Gramáticas de contexto libre). Una gramática de contexto libre es una cuádrupla de terminales, no terminales, un símbolo inicial y producciones:

$$G = (T, N, I, P) \qquad (5.1)$$

dónde:

T = Terminales, son los símbolos básicos con que se forman las cadenas. "Componente léxico" es un sinónimo de terminal cuando se trata de gramáticas para lenguajes de programación.

N = No terminales, son variables sintácticas que denotan conjuntos de cadenas. Los no terminales definen conjuntos de cadenas que ayudan a definir el lenguaje generado por la gramática, también imponen una estructura jerárquica sobre el lenguaje que es útil para el análisis sintáctico.

I = Inicial, en una gramática, un no Terminal es considerado como el símbolo inicial y el conjunto de cadenas que representa es el lenguaje definido por la gramática.

P = Producciones, las producciones de una gramática especifican cómo se pueden combinar los terminales y no terminales para formar cadenas. Cada producción consta de un no terminal, seguido por una flecha (o el símbolo :==) seguida por una cadena de no terminales y terminales.

Se sigue la regla convencional de especificar las gramáticas dando una lista de sus producciones, donde las producciones del símbolo inicial se listan primero. Se asume que los dígitos, los signos como "<" y las palabras clave de los lenguajes como *while* son terminales.

Una gramática de contexto libre da una especificación sintáctica precisa y fácil de entender de un lenguaje de programación, esto implica, que cualquier notación representada por una gramática se puede generar sin ambigüedades sintácticas y puede ser verificada que sea correcta de forma automática.

La notación más frecuentemente utilizada para expresar gramáticas de contexto libre es BNF. El Backus-Naur form (BNF) (también conocido como Backus-Naur formalism, Backus normal form o Pānini-Backus Form) es una metasintaxis usada para expresar gramáticas de contexto libre, una manera formal de describir lenguajes formales. BNF se utiliza extensamente como notación para las gramáticas de los lenguajes de programación de la computadora, de los sistemas de comando y de los protocolos de comunicación, así como una notación para representar partes de las gramáticas de lenguaje natural. La mayoría de los libros de textos para la teoría y/o la semántica del lenguaje de programación documentan el lenguaje de programación en BNF. Algunas variantes, tales como la *augmented* Backus-Naur form (ABNF), tienen su propia documentación. La notación se llama Backus por el nombre del autor John Backus y a sugerencia de Donald Knuth, el nombre de Naur se añadió debido a que Peter Naur simplificó la notación minimizando el conjunto de caracteres usados. La Backus-Naur Form o las gramáticas de BNF tiene semejanzas significativas a las reglas de la gramática de Pānini. BNF fue creado como parte para crear las reglas de ALGOL 60.

Una especificación de BNF es un sistema de reglas de la derivación, escrito como

<símbolo> ::= <expresión>

donde <símbolo> es un no terminal, y la expresión consiste en secuencias de símbolos y/o secuencias separadas por la barra vertical, '|', indicando una opción, el conjunto de secuencias es una posible substitución para el símbolo de la izquierda. Los símbolos que nunca aparecen en un lado izquierdo son terminales. BNF permite que se inserten acciones semánticas en el lado derecho de las producciones y la posición en que se ejecutan se da entre llaves. Las acciones semánticas son una secuencia de proposiciones que permiten acceder cualquier componente de una producción.

A partir de la gramática se puede construir automáticamente un analizador sintáctico que determine si un programa o una cadena tienen una sintaxis correcta, es decir, cumple con las reglas de escritura y no contiene ambigüedades. Los métodos empleados generalmente se clasifican como descendentes y ascendentes. Como sus nombres indican, los analizadores sintácticos descendentes construyen árboles de análisis sintáctico desde la raíz hasta las hojas, mientras que los analizadores sintácticos ascendentes comienzan en las hojas y avanzan hacia la raíz. En ambos casos, se examina la entrada al analizador sintáctico de izquierda a derecha, un símbolo a la vez. La figura 4.2 muestra el ejemplo para la generación de un árbol de análisis sintáctico a partir de una gramática:

SEQ ::== cAd
A ::== ab | a

y una cadena de entrada p = cad. Primero se crea un árbol de análisis sintáctico para un solo nodo etiquetado por *SEQ*. Después se utiliza la primera producción de *SEQ* para expandir el árbol y obtener el de la figura 4.2a). Se empareja la hoja más a la izquierda con el primer símbolo de *p*. Después se expande *A* para emparejar el segundo símbolo de *p*, si la primera alternativa de A concuerda, se continua con el tercer símbolo de *p*, y no se toma en cuenta *b* ya que no empareja con ninguna cadena de *p*, figura 4.2b). Por último se empareja la hoja *d* con el tercer símbolo de *p*, resultando el árbol mostrado en la figura 4.2c).

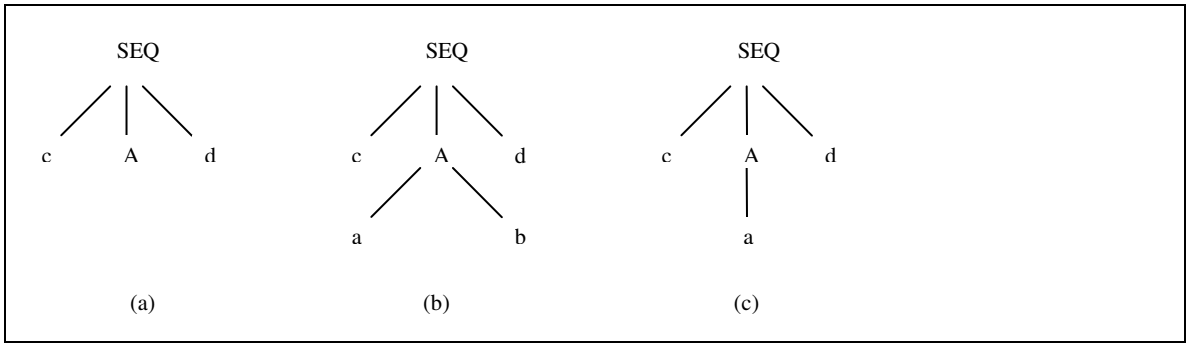


Figura 4.2 Generación del árbol sintáctico para p = cad.

En BNF el análisis sintáctico descendente se denomina derivación y da una descripción precisa de la construcción descendente de un árbol de análisis sintáctico. La idea central es que se considera una producción como una regla de reescritura, donde el no terminal de la izquierda es sustituido por la cadena del lado derecho de la producción, por ejemplo, dada la siguiente gramática:

WFK	::== SEQ
SEQ	::== P T P
T	::== T P T

La derivación consiste en construir el siguiente árbol que puede representar la estructura de la red de Petri mostrada en la figura 4.1

SEQ :== P T P
:== P T P T P
:== pI t1 p1 t2 pE

Dónde:

t1	representa la actividad *Registrar*
p2	representa el estado *En trámite*
t2	representa la actividad *Pagar*

Al análisis sintáctico ascendente se le denomina reducción y para construirse, en cada paso se sustituye una subcadena determinada que concuerde con el lado derecho de una producción por el símbolo del lado izquierdo de dicha producción y si en cada paso se elige correctamente la subcadena se traza una derivación por la derecha en sentido inverso, si se llega a la secuencia inicial se considera que la cadena es correcta sintácticamente, por ejemplo, dada la gramática del ejemplo anterior, la red resultante del ejemplo se puede reducir:

pI t1 p1 t2 pE

⇓

P T P T P

⇓

P T P

⇓

SEQ

Los procesos de derivación y de reducción se utilizarán en la sección 4.4.2 para representar y construir redes WF utilizando BNF.

4.4 Modelo para la construcción dinámica de Workflow DCW

En esta sección se define el modelo DCW para la construcción de WF dinámicos. Los pasos para el diseño del modelo son los siguientes:

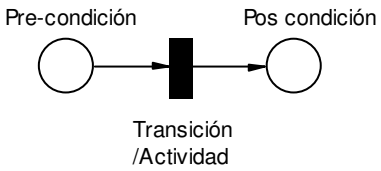
1. Especificación de los patrones básicos de construcción.
2. Diseño, especificación y validación del proceso general de construcción de WF correctos.
3. Definición de las propiedades de construcción a partir del punto anterior para actualizar el estado de un WF.

El diseño del modelo se realiza utilizando los dos formalismos de modelación, el textual con BNF y el visual con redes de Petri.

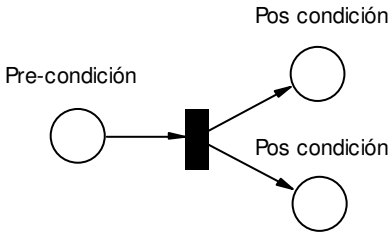
4.4.1 Patrones para la construcción dinámica

En un esquema WF todo proceso requiere soportar distintos patrones dependiendo del mismo. El término patrón se refiere a la relación temporal entre tareas. Para la construcción dinámica se utilizan cinco patrones simples de los propuestos por (Aalst 2003c) mostrados en la figura 4.3:

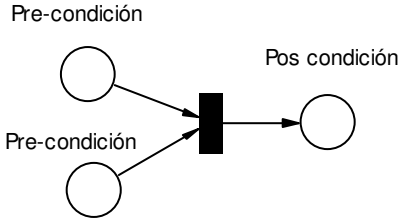
- Secuencia: una actividad en un proceso WF se habilita después de que haya terminado otra actividad del mismo proceso, figura 4.3a).
- AND split: un punto en el proceso WF dónde un solo hilo de control se divide en múltiples hilos de control que pueden ejecutarse en paralelo permitiendo que las actividades se ejecuten simultáneamente en cualquier orden figura 4.3b).
- AND join: un punto en el proceso WF dónde múltiples actividades paralelas convergen en un solo hilo de control, sincronizando múltiples hilos figura 4.3c).
- X-OR split.- un punto en el proceso WF dónde con base en una decisión o datos de control WF una de varias ramas/opciones es seleccionada figura 4.3d).
- X-OR join.- un punto en el proceso WF dónde dos o más ramas alternativas se unen sin sincronización. Sólo una rama se ejecuta, no hay ejecución paralela figura 4.3e).



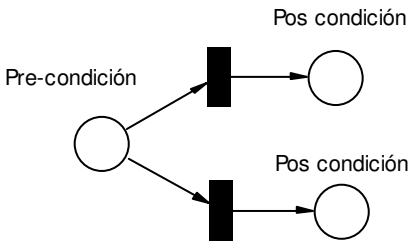
a) Secuencia básica



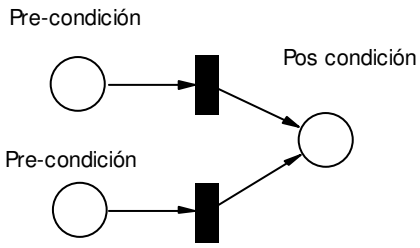
b) AND-split



c) AND-join



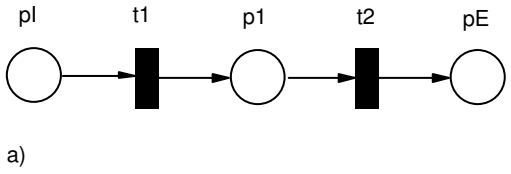
d) X-OR-split



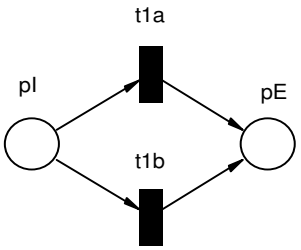
e) X-OR join

Figura 4.3 Patrones simples de relaciones entre tareas.

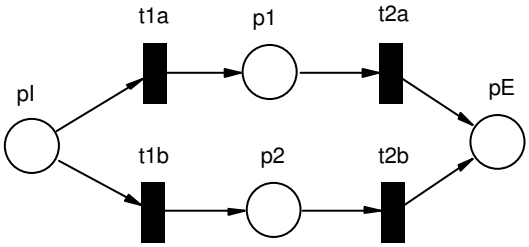
La combinación de los patrones simples genera cuatro patrones básicos de construcción y cada uno de ellos es una red WF correcta (sección 4.10), iniciando con una plaza pI y finalizando con una plaza pE. Dichos patrones básicos, permiten construir diferentes redes WF y sincronizar trayectorias de secuencia, de selección y/o de actividades en paralelo según se requiera. La figura 4.4 muestra los cuatro patrones básicos que se utilizan para construir dinámicamente redes WF:



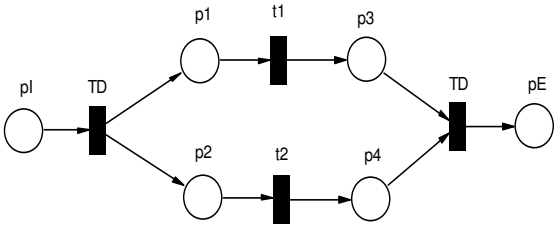
a) Patrón de secuencia



b) Patrón X-OR implícito



c) Patrón X-OR explícito



TD es una transición de sincronización.

d) Patrón AND de paralelismo

Figura 4.4 Patrones básicos de construcción.

El patrón de secuencia (p1 t2 pE) mostrado en la figura 4.4a) se usa cuando una actividad (t2) en un proceso tiene que habilitarse después de que la anterior haya terminado (t1). Los patrones de selección mostrados en las figuras 4.4b) y 4.4c) se utilizan cuando se requiere una decisión para habilitar una de varias actividades, en este caso, t1a o t1b. El patrón de paralelismo, figura 4.4d), se usa cuando varias actividades se requieren ejecutar en paralelo. La formación de los patrones básicos requieren de patrones simples y de sus complementarios, por ejemplo, el patrón AND *split* que permite ejecutar en paralelo varias actividades requiere de AND *join* para sincronizarlas de tal manera de que todas finalicen antes de iniciar otra actividad. El caso de los X-OR es similar, todo X-OR *split* requiere de un X-OR *join* que permita iniciar otra actividad cuando una de varias alternativas ha finalizado.

4.4.2 Representación de WF con BNF

Para la representación textual se define el WF en términos conceptuales de BNF. Primero se especifica una definición y después se describen los elementos de mapeo entre BNF y WF.

Definición 14 (WF BNF). Un Workflow definido con BNF es una cuádrupla:

BNWF = (T, N, I, P)

Dónde

T = {pi, ti, pI, pE} i = 1, 2, 3...

N = {WFK, TRANS, PLC, CONCSEQ}

I = {SEQ}

P =

WFK	**::== SEQ**	**(5.2)**
SEQ	**::== PLC TRANS PLC**	**(5.3)**
TRANS	**::== TRANS PLC TRANS**	**(5.4)**
TRANS	**::== [TRANS ↕ TRANS]**	**(5.5)**
TRANS	**::== TRANS CONCSEQ TRANS**	**(5.6)**
CONCSEQ	**::== [SEQ ↕ SEQ]**	**(5.7)**
CONCSEQ	**::== [SEQ ↕ CONCSEQ]**	**(5.8)**

↕ **representa la construcción paralela de dos secuencias.**

Las producciones permiten la modelación textual de cualquier WF que soporte patrones de secuencia, selección y paralelismo. Por ejemplo la figura 4.5, muestra la derivación para la representación en BNF de los patrones completos de construcción mostrados en la figura 4.4. La figura 4.5a) muestra la secuencia, las figura 4.5b) y 4.5c) muestran la selección implícita y explícita, y la figura 4.5d) el paralelismo:

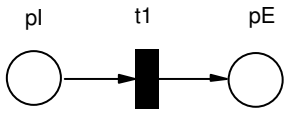
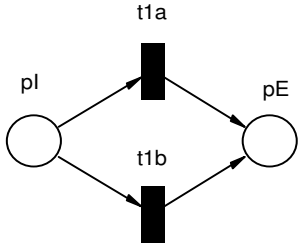
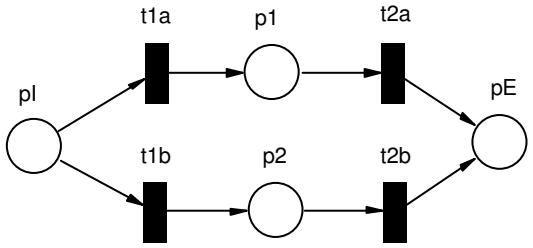
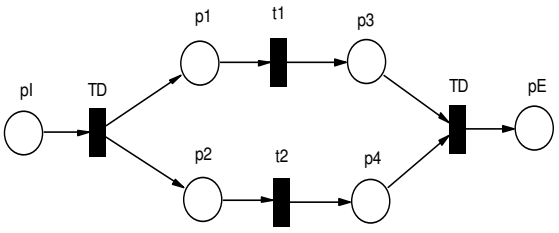
 a) Patrón de secuencia	WF → SEQ → PLC TRANS PLC → pI TRANS PLC → pI t1 PLC → pI t1 pE
 b) Patrón X-OR Split & X-OR Join implícito	WF → SEQ → PLC TRANS PLC → pI TRANS PLC → pI TRANS pE → pI [TRANS ↕ TRANS] pE → pI [t1a ↕ t1b] pE
 c) Patrón X-OR Split & X-OR Join explícito	WF → SEQ → PLC TRANS PLC → pI TRANS pE → pI [TRANS ↕ TRANS] pE → pI [TRANS PLC TRANS ↕ TRANS PLC TRANS] pE → pI [t1a p1 t2a ↕ t1b p2 t2b] pE
 TD es una transición de sincronización. d) Patrón AND Split & AND Join de paralelismo	WF → SEQ → PLC TRANS PLC → pI TRANS PLC → pI TRANS pE → pI TRANS CONCSEQ TRANS pE → pI TRANS [SEQ ↕ SEQ] TRANS pE → pI TD [PLC TRANS PLC ↕ PLC TRANS PLC] TD pE → pI TD [p1 TRANS p3 ↕ p2 TRANS p4] TD pE → pI TD [p1 t1 p3 ↕ p2 t2 p4] TD pE

Figura 4.5 Patrones básicos de construcción en redes de Petri y BNF.

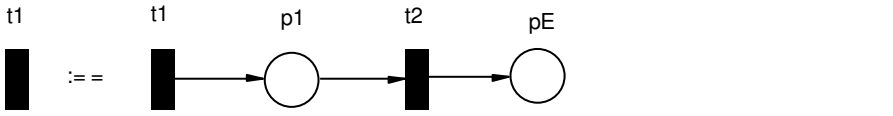
4.4.3 Proceso general de construcción dinámica

La construcción dinámica de una red WF requiere de la inserción de patrones y debe realizarse de tal manera que la red que se va construyendo sea en todo momento correcta. La construcción de la red WF requiere de dos aspectos de capital importancia:

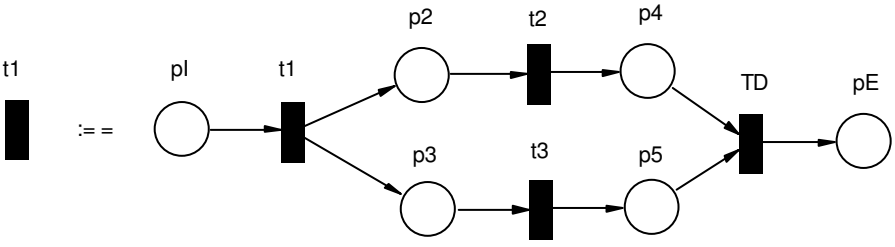
1. La inserción de una red básica correcta: aspecto sintáctico.
2. La adaptación del estado de la red: aspecto semántico.

Aspecto sintáctico

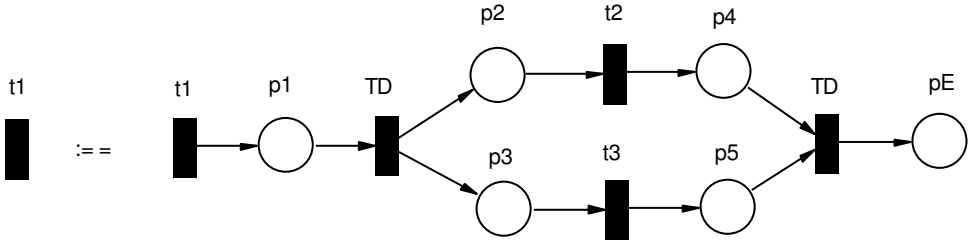
El proceso de construcción consiste en la inserción de patrones, considerados como redes WF básicas y correctas. Cada inserción de un patrón transforma la red original WF en una nueva red WF correcta y actualizada en su estado. La figura 4.6 muestra por separado el aspecto sintáctico de la inserción en una transición de cada patrón utilizado. El proceso resultante muestra cada transformación de la red por cada patrón insertado en la última transición de una red WF, se denota como p1 la plaza inicial de cada patrón. Por ejemplo, la figura 4.6a) muestra la inserción de una secuencia plaza transición, la figura 4.6b) y 4.6c) muestran dos opciones que pueden ser viables para la inserción de un patrón paralelo, en el primer caso se conectan las ramas directamente a la transición t1, en el segundo caso se conecta a t1 una red WF completa representando ejecución en paralelo y utilizando dos transiciones TD para sincronización, más adelante se muestra cuál es la mejor opción; y las figuras 4.6d) y 4.6e) muestran la inserción de un patrón de selección implícito y explícito, respectivamente.



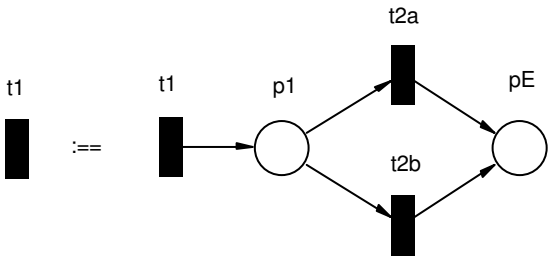
a) Transformación para una secuencia.



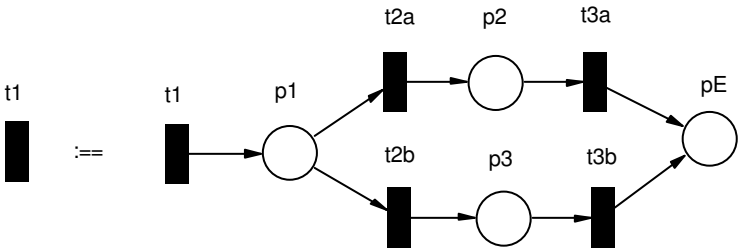
b) Transformación de paralelismo sin TD: AND Split & AND Join.



c) Transformación de paralelismo con TD: AND Split & AND Join.



d) Transformación de selección implícita: X-OR Split & X-OR Join.



e) Transformación de selección explícita:= X-OR & X-OR Join.

Figura 4.6 Transformaciones para el proceso de construcción.

La diferencia entre el patrón de selección explícito con el implícito de la figura 4.6 es que en el primero se puede saber el estado cuando ocurre la selección, es decir, se tiene un estado para indicar cuál rama se esta ejecutando en el momento actual posterior a la selección.

Aspecto dinámico

El aspecto dinámico de la construcción consiste en actualizar el estado de cada red transformada de tal manera que permanezca siempre correcta en su marcaje. La construcción dinámica implica que cada inserción sólo se aplica a las transiciones activas (una transición que no esta activa, no representa un aspecto dinámico de la red).

El proceso general de construcción de una red WF correcta en otra red WF correcta consiste en dos pasos:

1. Expansión: inserción de un patrón WF correcto serial/paralela/selección a la última transición activa de la red.
2. Actualización del estado.

Este proceso general permite obtener la transformación de una red WF correcta al insertar un patrón. La expansión de una red mediante la inserción de otra se considera un tipo de cambio y si se encuentra en ejecución, el cambio se considera dinámico. A continuación se propone la siguiente definición que describe el proceso general de construcción mediante la unión de dos redes WF:

Definición 15 (Proceso general de construcción dinámica). Sea PN1 = (P1, T1, F1) y PN2 = (P2, T2, F2) dos redes WF tal que $T1 \cap T2 = 0$, $P1 \cap P2 = \{pE\}$, $pI \in P1 \neq pI2 \in P2$ y $t+ \in T1$, dónde $(t+)\bullet = pE$. PN3 se obtiene uniendo en t+ de PN1 el patrón PN2:

$$P3 = P1 \cup P2$$

$$T3 = T1 \cup T2$$

$$F3 = \{ (x, y) \in F1 \mid \{x, y\} \cap \{pI, pE\} = 0\} \cup \{ (x, y) \in F2 \mid \{x, y\} \cap \{pI2, pE\} = 0\} \cup$$
$$\{ (x, y) \in P1 \times T1 \mid (pI, y) \in F1 \wedge (x, t+) \in F1 \} \cup$$
$$\{ (x, y) \in T1 \times P2 \mid (t+, pI2)\} \cup$$
$$\{ (x, y) \in T2 \times P2 \mid (x, pE) \in F2 \}$$

Prueba

Sean (PN1, pI) segura y PN1, PN2 *sound*. Primero se necesita probar que (PN3, i) es viva y limitada (*sound*). La sub red en PN3 que corresponde a $PN3 \cap PN1$ es segura y por lo tanto las transiciones $T3 \cap T1$ son vivas (t+ esta viva) y las plazas $P3 \cap P1$ son limitadas. Si la transición t+ se dispara entonces se alcanza el estado (PN2, pI2) seguro y de aquí que las transiciones $T3 \cap T2$ también son vivas y las plazas $P3 \cap P2$ son limitadas, por lo tanto (PN3, pI) es segura y *sound* (correcta).

La prueba de la definición 15 valida los aspectos sintácticos y semánticos del proceso general de construcción dinámica y demuestra que la unión de dos redes WF correctas generan una red WF resultante correcta. La figura 4.7 muestra gráficamente el proceso general de construcción dinámica de dos redes PN1 y PN2. La figura 4.8 muestra un ejemplo particular del proceso al insertar una secuencia a una red WF.

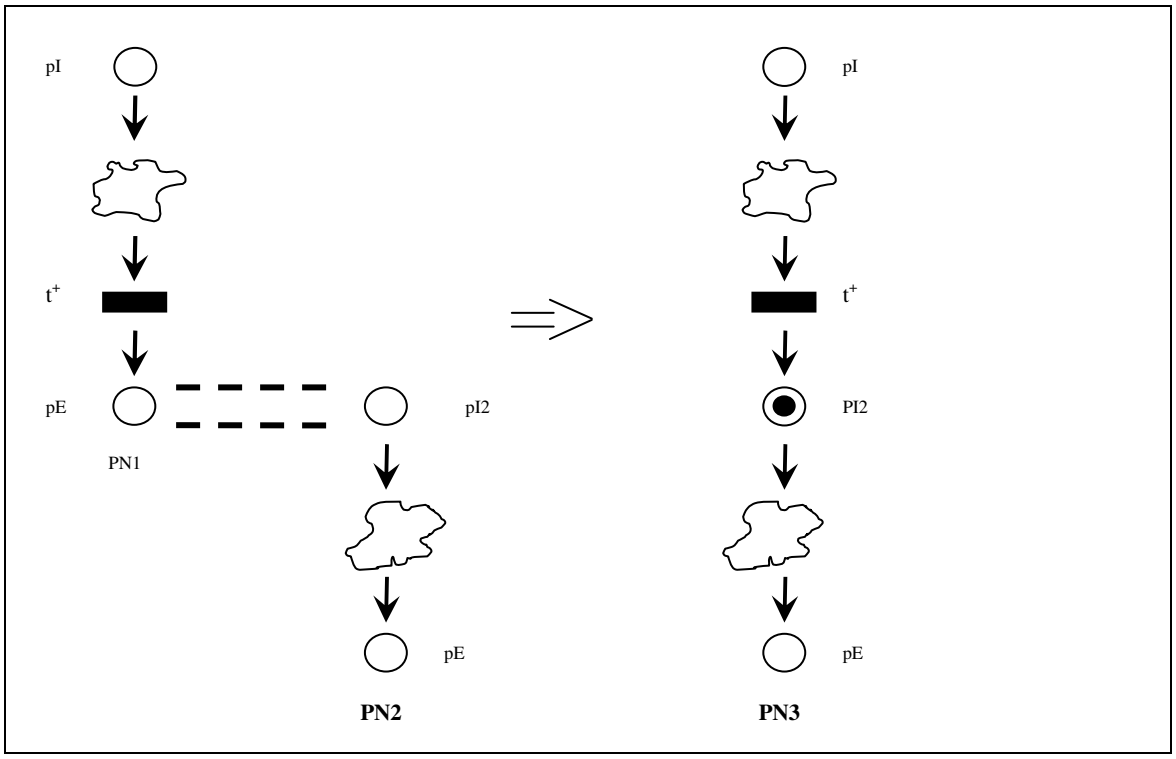


Figura 4.7 Proceso general de construcción dinámica.

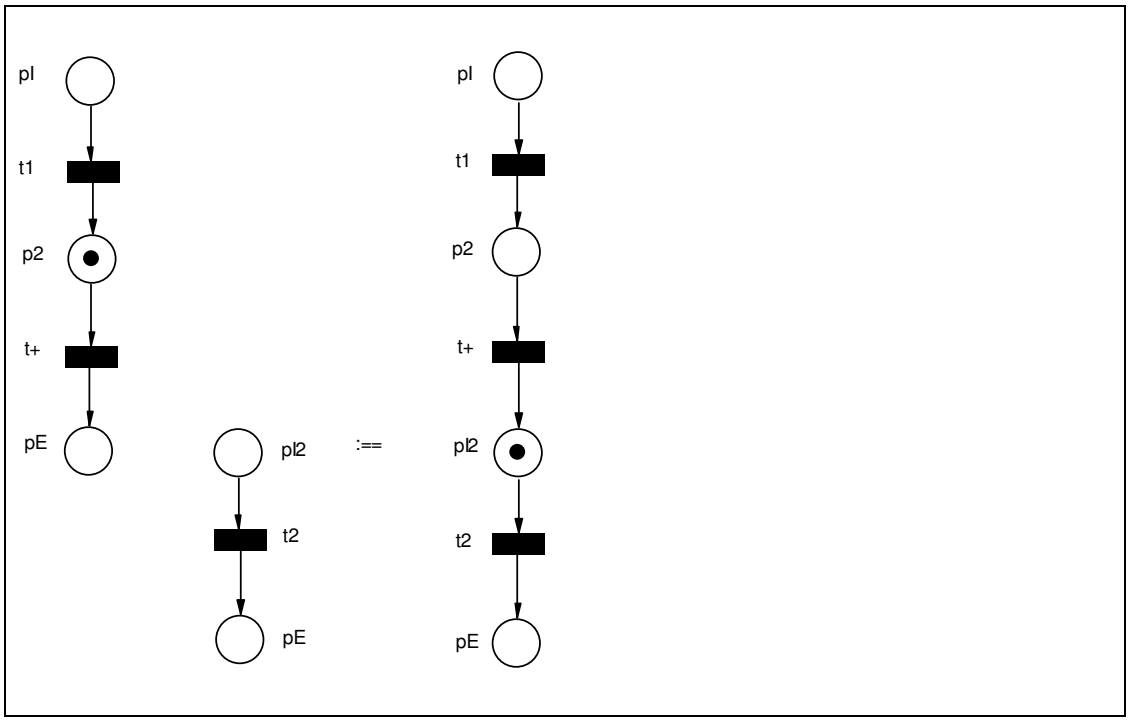


Figura 4.8 Ejemplo de la inserción de una secuencia.

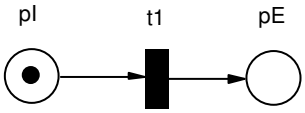
En la unión de las dos redes la plaza inicial del patrón *pI2* se inserta en la última transición de la red WF original. La plaza *pI* de *PN1* y la plaza *pE* de *PN1* y *PN2* se mantienen como plaza inicial y plaza final respectivamente.

Aplicando el proceso general de construcción dinámica

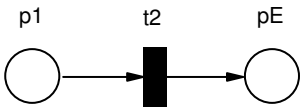
La figura 4.9 muestra una secuencia inicial a partir de la cuál se pueden obtener las cuatro construcciones dinámicas mostradas en las figuras 4.10-4.13. Para diferenciar la plaza inicial de la red WF original y con respecto a la de los patrones a insertar se denomina como p1 la plaza inicial de cada patrón. En cada figura primero se muestran la red WF y el patrón a insertar, posteriormente la red resultante al aplicar el proceso general de construcción dinámica. En cada caso de la modelación visual se incluye su equivalente en modelación textual mediante BNF.

La figura 4.10 muestra la red resultante al insertar en una secuencia un patrón tipo secuencia. La figura 4.11 muestra dos variantes para la red resultante al insertar en una secuencia un patrón paralelo, la figura 4.11a) muestra la inserción de una red WF completa tal como lo muestra la transformación de la figura 4.6c), la figura 4.11b) muestra una red resultante utilizando la transformación de la figura 4.6b). El proceso general de construcción une dos redes WF correctas, esto implica que la construcción secuencia-red paralela correcta es la mostrada en la figura 4.11a), ya que la 4.11b no muestra una construcción, si no muestra el reemplazo de la transición de sincronización paralela por la transición t1, violando la definición 15. La figura 4.12 muestra la red resultante al insertar en una secuencia un patrón de selección implícito, la figura 4.13 muestra la inserción de un patrón de selección explícito.

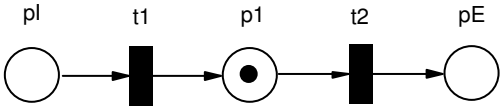
Figura 4.9 Red inicial.



Red WF original



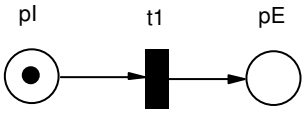
Patrón a insertar: secuencia



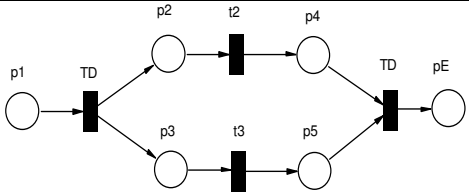
Red WF resultante

WF → SEQ
→ PLC TRANS PLC
→ pI TRANS PLC
→ pI TRANS pE
→ pI TRANS PLC TRANS pE
→ pI t1 p1 TRANS pE
→ pI t1 p1 t2 pE

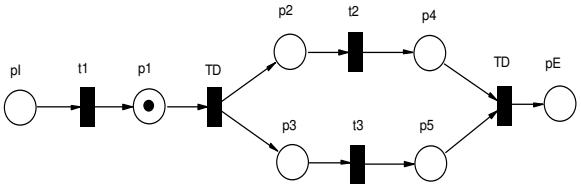
Figura 4.10 Construcción secuencia-secuencia.



Red WF original

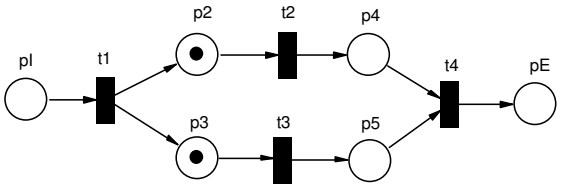


Patrón a insertar: paralelismo



a) Red WF paralela resultante usando TD

WF → SEQ
→ PLC TRANS PLC
→ pI TRANS PLC
→ pI TRANS pE
→ pI TRANS CONCSEQ TRANS pE
→ pI TRANS PLC TRANS [SEQ ↕ SEQ] TRANS pE
→ pI t1 p1 TD [PLC TRANS PLC ↕ PLC TRANS PLC] TD pE
→ pI t1 p1 TD [p2 TRANS p4 ↕ p3 TRANS p5] TD pE
→ pI t1 p1 TD [p2 t2 p4 ↕ p3 t3 p5] TD pE

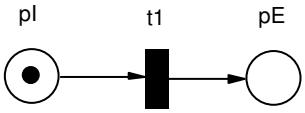


b) Red WF paralela resultante sin usar TD

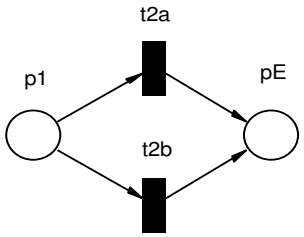
WF → SEQ
→ PLC TRANS PLC
→ pI TRANS PLC
→ pI TRANS pE
→ pI TRANS CONCSEQ TRANS pE
→ pI TRANS [SEQ ↕ SEQ] TRANS pE
→ pI t1 [PLC TRANS PLC ↕ PLC TRANS PLC] t4 pE
→ pI t1 [p2 t2 p4 ↕ p3 t3 p5] t4 pE
→ pI t1 [p2 t2 p4 ↕ p3 t3 p5] t4 pE

Red WF BNF resultante

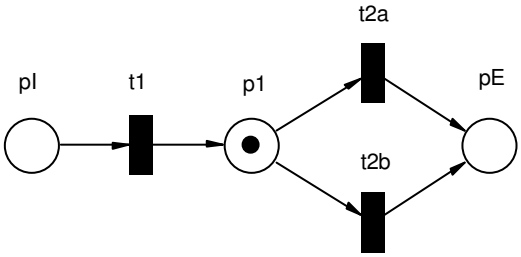
Figura 4.11 Construcción secuencia-red paralela.



Red WF original



Patrón a insertar: X-OR implícito

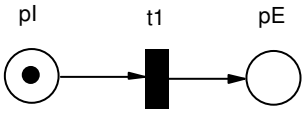


Red WF resultante con t2a o con t2b

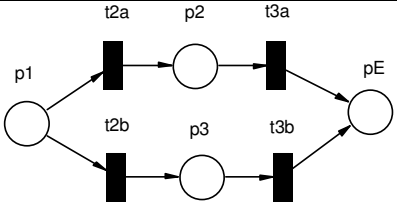
WF → SEQ
→ PLC TRANS PLC
→ pI TRANS pE
→ pI TRANS PLC TRANS pE
→ pI t1 p1 [TRANS ↕ TRANS] pE
→ pI t1 p1 [t2a ↕ T2b] pE

Red WF BNF resultante

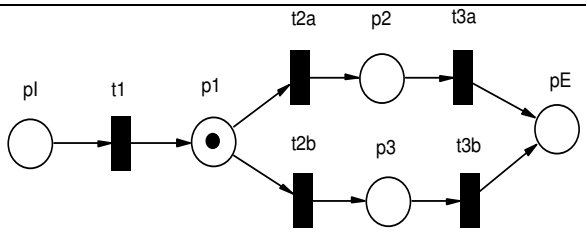
Figura 4.12 Construcción secuencia-selección implícita.



Red WF original



Patrón a insertar: X-OR explícito



Red WF resultante con t2a y t3a ó t2b y t3b

WF → SEQ
→ PLC TRANS PLC
→ pI TRANS pE
→ pI TRANS PLC TRANS pE
→ pI T1 P1 [TRANS ↕ TRANS] pE
→ pI T1 P1 [TRANS PLC TRANS ↕ TRANS PLC TRANS] pE
→ pI t1 p1 [t2a p2 t3a ↕ t2b p3 t3b] pE

Red WF BNF resultante

Figura 4.13 Construcción secuencia-selección explícita.

4.4.4 Propiedades para la construcción dinámica del WF

Sea una red WF = (PWF, TWF, FWF, MoWF) especificada por la definición 7, dónde cualquier transición $t_i \in$ TWF representa una actividad o un servicio. Las

siguientes propiedades se pueden derivar de la definición 15 para que el estado de la construcción dinámica de una red WF en particular sea correcta:

Propiedad DCW1

Si dos transiciones t_i, t_j están relacionadas causalmente

$$t_i \rightarrow t_j \; \forall \; ti, tj \in TWF \mid ti \neq tj, ti < tj, 1 \leq i, j \leq n.$$

Si ti es seleccionada entonces en un momento futuro podrá seleccionarse t_j. De manera inversa, para que t_j sea seleccionada, t_i debió de haber sido ejecutada (figura 4.14).

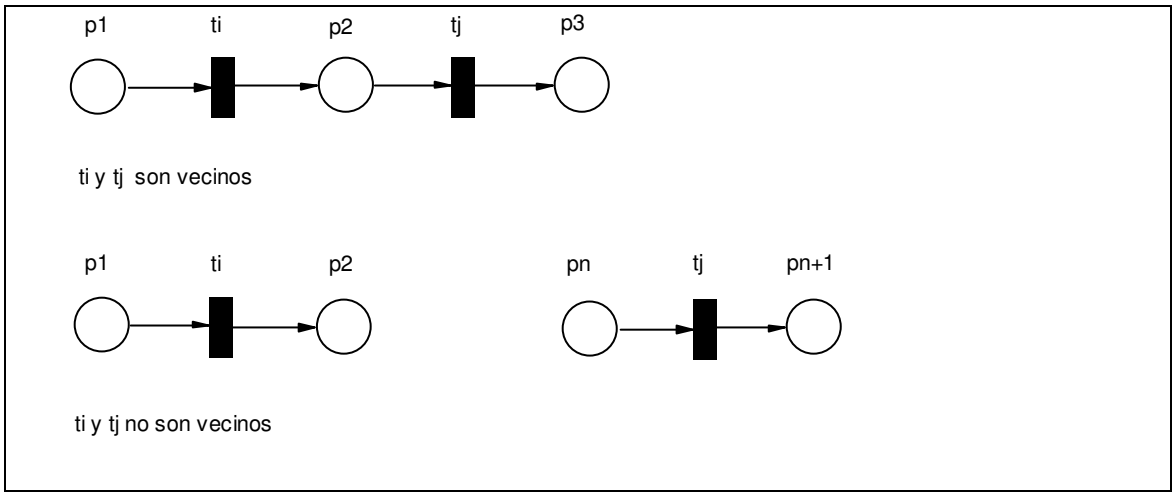


Figura 4.14 Propiedad 1: ti $\rightarrow$ tj.

Esta propiedad se aplica cuando dos actividades son dependientes y la activación de una requiere que la otra se haya ejecutado.

Propiedad DCW2

Toda transición t que no tiene relación con ninguna otra finalmente será seleccionada para cada caso. Toda transición que sea parte de la red será seleccionada y aplicada en cualquier orden

$$\forall t \in TWF \; \exists \; M, M' \mid Mpl \xrightarrow{*} M \xrightarrow{t} M'$$

Esta propiedad implica que no debe haber actividades que nunca se ejecuten, es decir, transiciones muertas.

Propiedad DCW3

Si ti y tj son dos transiciones que no están relacionadas, entonces su orden de ejecución es indistinto y puede realizarse en paralelo, figura 4.15

$$\forall\, t_i \updownarrow t_j \in TWF \mid t_i \neq t_j \;\exists\, M, M' \mid (Mpl \xrightarrow{*} M \xrightarrow{t_i} M(\bullet t_i) \xrightarrow{t_j} M(\bullet t_j) \rightarrow M') = (Mpl \xrightarrow{*} M \xrightarrow{t_j} M(\bullet t_j) \xrightarrow{t_i} M(\bullet t_i) \rightarrow M')$$

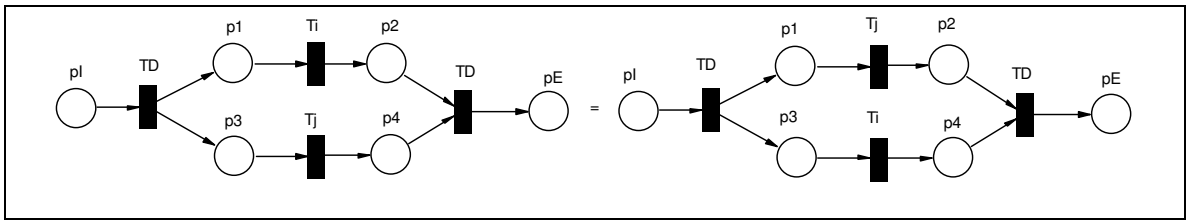


Figura 4.15 Propiedad 3: ti $\updownarrow$ tj.

Esta propiedad implica que si tenemos dos actividades que no son dependientes entre ellas, pueden ejecutarse en paralelo en caso de que lo permita la finalización exitosa del caso.

Propiedad DCW4

Dada una WF-Net que representa un caso, W= (P, T, F, M_0), las transiciones son únicas en la red si

$\forall$ ti $\in$ T | ti $\neq$ tj, 1 $\leq$ i, j $\leq$ n.

Esta propiedad implica que no debe haber en la red WF actividades iguales. También implica que no deben repetirse, es decir, no debe haber ciclos en la red WF.

Aplicando las propiedades de construcción

La aplicación de las propiedades de construcción facilita la actualización del estado de la nueva red WF. Las propiedades no suplantan el proceso general de construcción dinámica, proporcionan de manera simple la actualización del estado de la nueva red. Por ejemplo, la figura 4.16 muestra la inserción de una secuencia figura 4.16b) en una red WF figura 4.16a) y los tres estados que podrían resultar al construir la red WF. La aplicación del proceso general de construcción dinámica junto con las propiedades de construcción dinámica facilita un WF correcto tanto en su estructura como en su estado. Por ejemplo, la red de la figura 4.16c) es la construcción correcta cumpliendo con las cuatro propiedades, la red de la figura 4.16d) representa la inserción de una nueva transición T2 sin haber ejecutado la actual T1 violando DCW1 y, la figura 4.16e) representa haberse saltado una transición violando DCW2. Para que la construcción sea correcta todas las transformaciones de redes WF deben generarse cumpliendo con las cuatro propiedades de construcción.

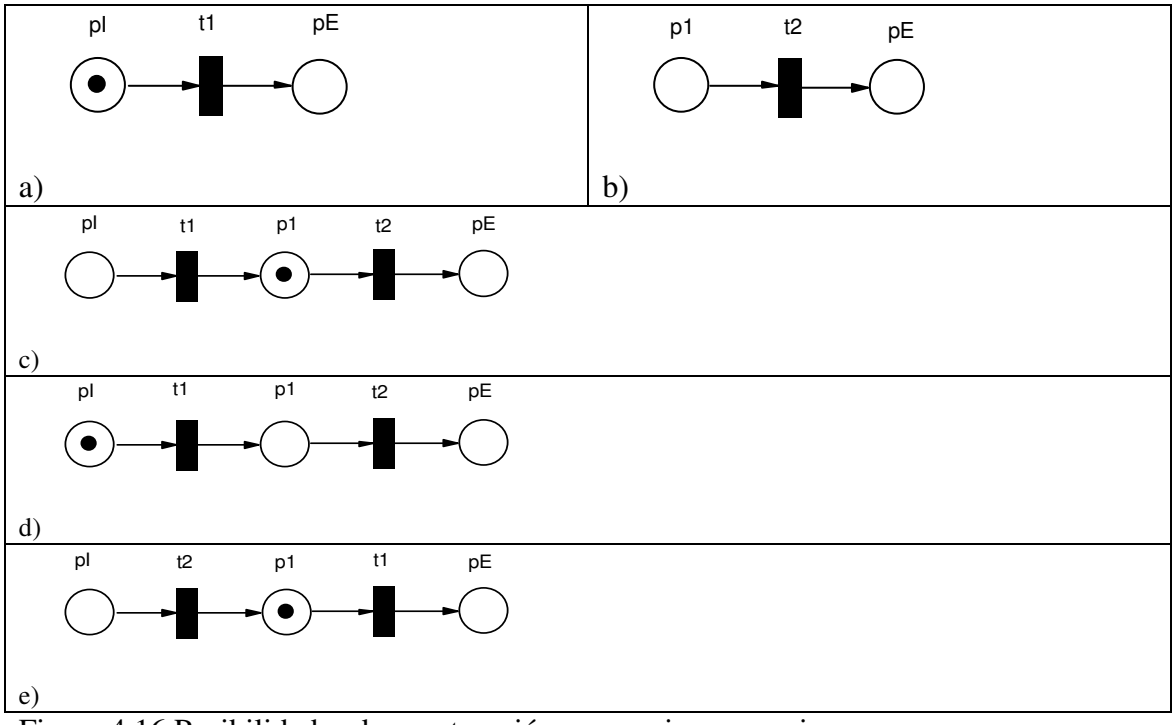


Figura 4.16 Posibilidades de construcción secuencia-secuencia.

4.5 Definiendo acciones semánticas en redes WF BNF

El proceso de construcción dinámica de WF consiste en cambiar la red en el vuelo, en el caso de BNF se realiza expandiendo las producciones y en el caso de la red WF mediante la inserción de una nueva red WF en la última transición de la actual. Note que la última plaza en las dos redes WF debe ser pE y que en la parte de BNF todas las derivaciones pueden reducirse a la secuencia inicial. Posteriormente se debe de actualizar el estado, las figuras 4.10-4.13 también muestran cómo debe quedar el nuevo estado en redes de Petri para las nuevas redes WF.

En el caso de BNF las figuras 4.10-4.13 no muestran la actualización del estado, esto es debido a que la gramática y las producciones descritas para redes WF requieren de mecanismos para representar el aspecto dinámico. Dichos mecanismos deben actualizar el estado de la red WF BNF y requiere de la especificación de acciones semánticas en las producciones (Aho et al. 1998). Las acciones semánticas son fragmentos de código encerrados entre llaves en el lado derecho de las producciones y se ejecutan cuando se reduce la producción asociada. Cualquier componente de una producción puede accederse desde la acción semántica, usando el apuntador $$ que se refiere al valor del atributo asociado con el no terminal del lado izquierdo y un apuntador $k se refiere al valor del atributo asociado con el k-ésimo símbolo gramatical, terminal o no terminal, del lado derecho. Normalmente la acción semántica calcula un valor para $$ en función de los $k. Para poder representar el aspecto sintáctico y el dinámico en las redes WF BNF la nueva gramática con acciones semánticas en las producciones se define de la siguiente forma:

WFK ::== SEQ (5.8)

SEQ ::== PLC TRANS PLC (5.9)

```
[if token($1) and execute($2){
    FS = computeFollowSet($2);
    if(FS != 0) {  $2 =  FS; }
} else if (reduced($2)){
   $$ = $2; }]
```

TRANS ::== TRANS PLC TRANS (5.10)

TRANS ::== [TRANS ↕ TRANS] (5.11)

```
[ NFS1 = computeFollowSet($1);
  NFS2 = computeFollowSet($2);
  if (NFS1== Ø or NFS2== Ø) {
$$= 0; } else {
$$=NFS1 +  NFS2}]
```

TRANS ::== TRANS CONCSEQ TRANS (5.12)

CONCSEQ ::== [SEQ ↕ SEQ] (5.13)

[NFS1 = computeFollowSet($1);
NFS2 = computeFollowSet($2);
if (NFS1== Ø or NFS2== Ø) {
$$= 0; } else {
$$=NFS1 + NFS2}]

CONCSEQ ::== [SEQ ↕ CONCQSEQ] (5.14)

[NFS1 = computeFollowSet($1);
NFS2 = computeFollowSet($2);
if (NFS1== Ø or NFS2== Ø) {
$$= 0; } else {
$$=NFS1 + NFS2}]

La producción (5.9) realiza la construcción cuando se ejecuta la transición, el segundo condicional se dispara cuando *TRANS* es reducida. La ejecución de la transición hará que la función *computeFollowSet()* calcule el nuevo patrón WF a insertar y la una con ella, incluyéndola en la derivación y además, transferirá el token a la plaza p1 de la subred insertada (figuras 4.10-4.13), activándose la primera transición de cada producción y actualizando el estado de la red. Esto puede realizarse considerando los elementos plazas y transiciones como no terminales, de este modo, cada tipo tiene sus propias acciones semánticas en el cuál se almacenan estados y condiciones, por ejemplo, se puede diferenciar cuándo una transición está activa o ejecutada. Esta diferencia de estados en una transición no se puede realizar con redes de Petri, por lo que, BNF añade valor agregado a la formalización del modelo.

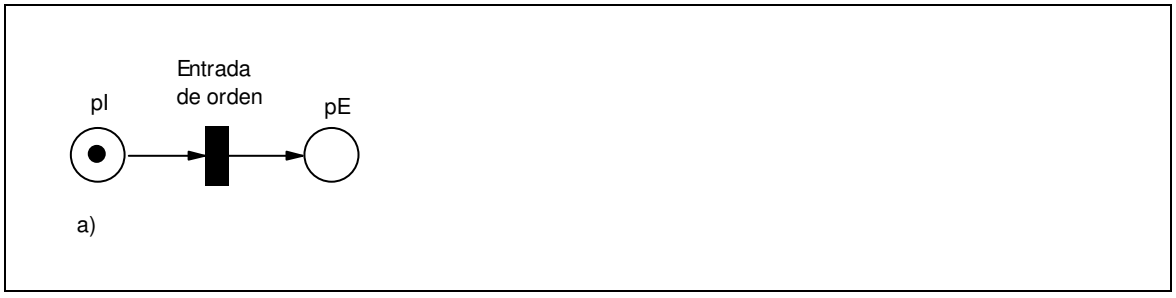
Las figuras 4.10-4-13 muestran una ventaja importante de BNF si se aplican acciones semánticas al modelar la construcción dinámica de WF. Dicha ventaja es que permite que los patrones sean genéricos, es decir, no se tiene el problema de nombrar de manera diferente a cada uno dado que cada actividad y cada plaza se representan por no-terminales. Por ejemplo, si a una secuencia se le añade otra secuencia la red resultante sería la mostrada por la figura 4.10. Si a esta red se le añade otra secuencia de modo de que queden tres transiciones en total en la red, no se tiene el problema de que haya dos plazas nombradas como p1, ya que como están representadas por no-terminales al hacer la derivación toma el nombre del proceso de negocio que se esta modelando y el estado de cada nodo (plaza/transición) puede accederse directamente y diferenciarse de los demás mediante acciones semánticas (ver figura 4.17).

Con la especificación de la gramática anterior queda completo el aspecto sintáctico y semántico de todo el proceso de construcción, mediante el formalismo visual

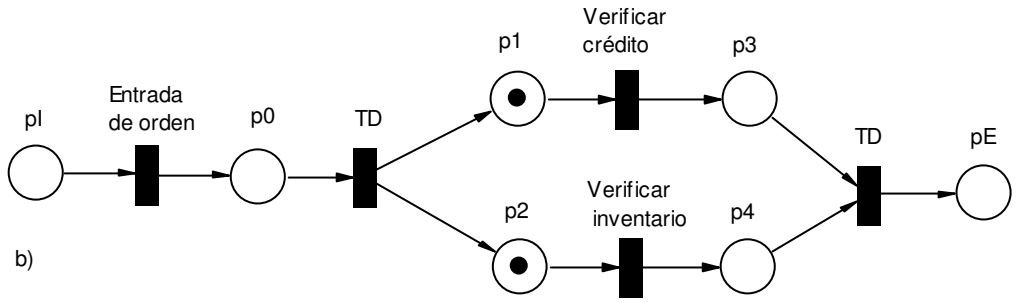
en redes de Petri y el textual con BNF. La siguiente sección muestra un ejemplo de construcción dinámica que incluye los dos formalismos de modelación.

4.6 Ejemplo de construcción dinámica

El proceso de construcción dinámica resuelve el problema de ir construyendo redes WF que incluyan casos en ejecución. Cada red generada puede expandirse nuevamente con la inserción de un nuevo patrón, mediante el proceso general de construcción dinámica y aplicando las propiedades de construcción. La figura 4.17 muestra un ejemplo de construcción dinámica para un proceso de órdenes de compra de una compañía típica. Cada inciso esta dividido en tres partes, la primera parte muestra la red WF y su estado (p.ej. 4.17a)), la segunda parte muestra su representación en BNF (p.ej. 4.17a.1)) y la tercera describe qué actividad representa cada transición en BNF (p.ej. 4.17a.2)). Las figuras 4.17a) a 4.17e) muestran toda la secuencia del proceso de construcción y su respectiva derivación en BNF con el objetivo de construir la red representada por la figura 4.17g). La figura 4.17a) muestra la secuencia inicial, cuando arriba un caso. La ejecución de la actividad *Entrada de orden* genera la creación de dos actividades en paralelo *Verificar crédito* y *Verificar inventario* sincronizadas por las transiciones *TD* (figura 4.17b)). La ejecución de las dos actividades permite generar la actividad *Evaluación*, la cuál se activa (figura 4.17c)). Cuando se ejecuta *Evaluación* se crea una red de selección en la cuál la ejecución seguirá la trayectoria de un solo camino, en este caso el de la actividad *Aprobar* (figura 4.17d)). La ejecución de la actividad *Aprobar* genera dos actividades en paralelo *Enviar* y *Cobrar* nuevamente sincronizadas por las transiciones *TD* (figura 4.17 e)) y que al ejecutarse estas dos actividades en paralelo generan la actividad *Registrar* (figura 4.17f)). Por último, la ejecución de la actividad *Registrar* finaliza el caso (figura 4.17g)). Note que la red siempre es correcta.

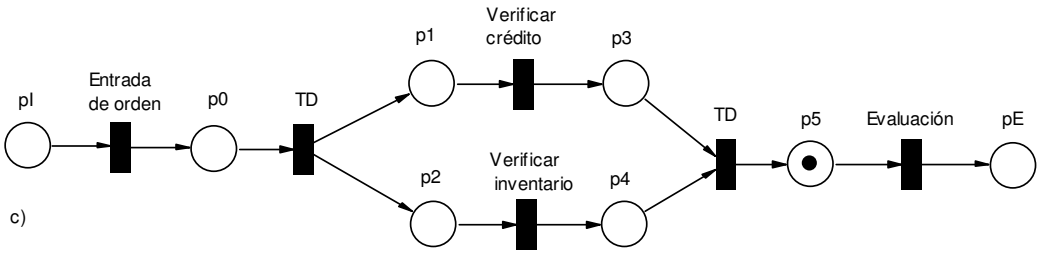


WF → SEQ → PLC TRANS PLC → pI TRANS PLC → pI t1 pE *[$1 = 1]* a.1) WF BNF	t1 = Entrada de orden a.2) Actividades



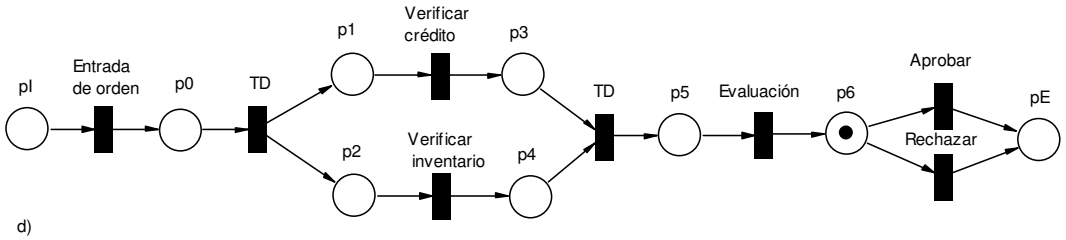
b)

WF → pI TRANS pE → pI TRANS CONCSEQ TRANS pE → pI TRANS PLC TRANS [SEQ ↕ SEQ] TRANS pE → pI t1 p0 TD[p1 t2 p3 ↕ p2 t3 p4] TD pE *[$5 = 1 and $8 = 1]* b.1) WF BNF	t1 = Entrada de orden t2 = Verificar crédito t3 = Verificar inventario b.2) Actividades



c)

WF → pI TRANS pE → pI TRANS CONCSEQ TRANS pE → pI TRANS PLC TRANS [SEQ ↕ SEQ] TRANS pE → pI TRANS PLC TRANS [SEQ ↕ SEQ] TRANS PLC TRANS pE → pI t1 p0 TD [p1 t2 p3 ↕ p2 t3 p4] TD p5 t4 pE *[$12 = 1]* c.1) WF BNF	t1 = Entrada de orden t2 = Verificar crédito t3 = Verificar inventario t4 = Evaluación c.2) Actividades



d)

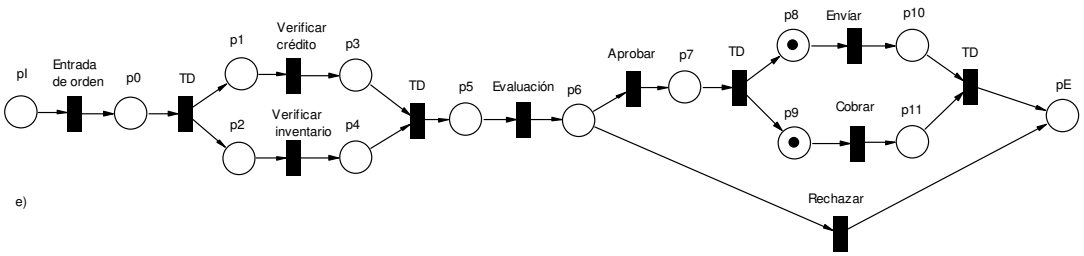
WF → pI TRANS pE
→ pI TRANS CONCSEQ TRANS pE
→ pI TRANS PLC TRANS [SEQ ↕ SEQ] TRANS pE
→ pI TRANS PLC TRANS [SEQ ↕ SEQ] TRANS PLC TRANS pE
→ pI t1 p0 TD [p1 t2 p3 ↕ p2 t3 p4] TD p5 TRANS pE
→ pI t1 p0 TD [p1 t2 p3 ↕ p2 t3 p4] TD p5 TRANS PLC TRANS pE
→ pI t1 p0 TD [p1 t2 p3 ↕ p2 t3 p4] TD p5 t4 p6 [t5 ↕ t6] pE

[$14 = 1]

d.1) WF BNF

t1 = Entrada de orden
t2 = Verificar crédito
t3 = Verificar inventario
t4 = Evaluación
t5 = Aprobar
t6 = Rechazar

d.2) Actividades



e)

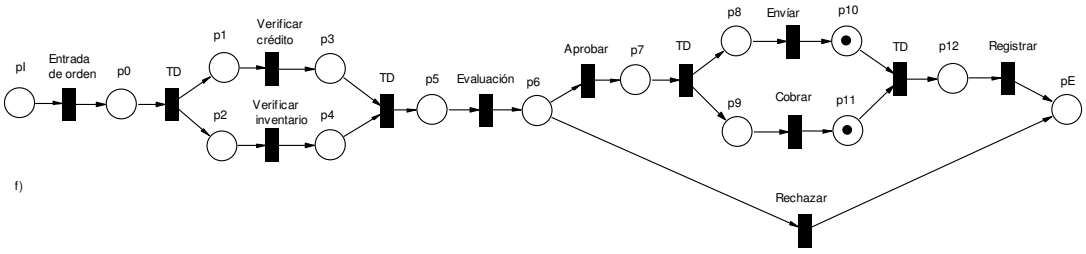
WF → pI t1 p0 TD [p1 t2 p3 ↕ p2 t3 p4] TD p5 TRANS PLC TRANS pE
→ pI t1 p0 TD [p1 t2 p3 ↕ p2 t3 p4] TD p5 t4 p6 [TRANS ↕ t6] pE
→ pI t1 p0 TD [p1 t2 p3 ↕ p2 t3 p4] TD p5 t4 p6 [TRANS CONCSEQ TRANS ↕ t6] pE
→ pI t1 p0 TD [p1 t2 p3 ↕ p2 t3 p4] TD p5 t4 p6 [TRANS [SEQ ↕ SEQ] TD ↕ t6] pE
→ pI t1 p0 TD [p1 t2 p3 ↕ p2 t3 p4] TD p5 t4 p6 [TRANS PLC TRANS [PLC TRANS PLC ↕ PLC TRANS PLC]TD ↕ t6] pE
→ pI t1 p0 TD [p1 t2 p3 ↕ p2 t3 p4] TD p5 t4 p6 [t5 p7 TD[p8 t7 p10 ↕ p9 t8 p11] TD ↕ t6] pE

[$18 = 1 and $21 = 1]

e.1) WF BNF

t1 = Entrada de orden
t2 = Verificar crédito
t3 = Verificar inventario
t4 = Evaluación
t5 = Aprobar
t6 = Rechazar
t7 = Enviar
t8 = Cobrar

e.2) Actividades



f)

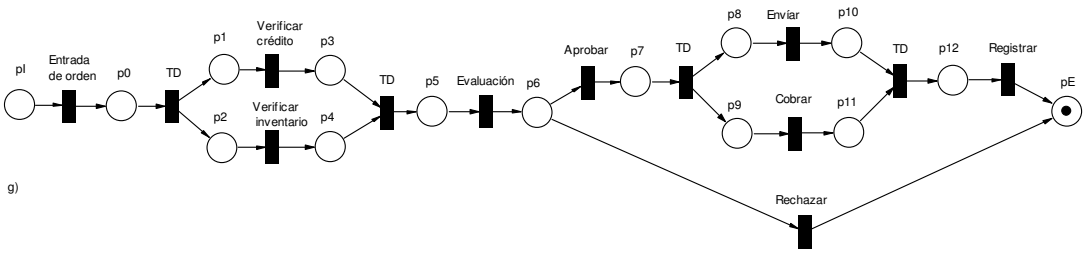
WF → pI t1 p0 TD [p1 t2 p3 ↕ p2 t3 p4] TD p5 t4 p6 [t5 p7 TD [p8 t7 p10 ↕ p9 t8 p11] TRANS ↕ t6] pE
→ pI t1 p0 TD [p1 t2 p3 ↕ p2 t3 p4] TD p5 t4 p6 [t5 p7 TD [p8 t7 p10 ↕ p9 t8 p11] TRANS PLC TRANS ↕ t6] pE
→ pI t1 p0 TD [p1 t2 p3 ↕ p2 t3 p4] TD p5 t4 p6 [t5 p7 TD [p8 t7 p10 ↕ p9 t8 p11] TD p12 t9 ↕ t6] pE

[$20 = 1 and $23 = 1]

f.1) WF BNF

t1 = Entrada de orden
t2 = Verificar crédito
t3 = Verificar inventario
t4 = Evaluación
t5 = Aprobar
t6 = Rechazar
t7 = Enviar
t8 = Cobrar
t9 = Registrar

f.2) Actividades



g)

Figura 4.17 Construcción dinámica de una red WF.

En la figura 4.18 se muestra el algoritmo que engloba la construcción dinámica de una red WF de DCW. Recibe como entrada el estado actual del WF y al entrar al ciclo va ejecutando y construyendo el WF. El objetivo del ciclo es que si ocurre una falla el estado permanecerá igual y nuevamente se ejecutará el ciclo. La función **computeFollowSet()** tiene dos tareas principales: la primera es calcular la siguiente(s) transiciones(s) que se debe(n) realizar con respecto a todas las anteriores ejecutadas y la segunda es detectar cuándo se detecta el estado final, terminando la construcción del WF. El cálculo del siguiente patrón se realiza con base en la función.

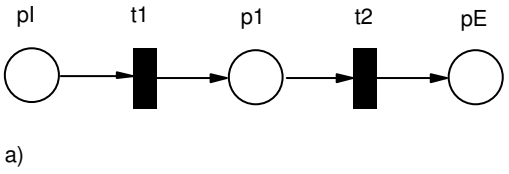
```
Algoritmo para la construcción dinámica WF de DCW
    Input: Si                                   // Estado actual, secuencia actual
    S <- Si
    while  estado_final no sea alcanzado
            Di = Wait (ejecución Ti actual)     // Ejecución transición actual
            TT = computeFollowSet(S, Di)        // Calcular siguiente(s) patrón(s)
            Insertar Transformaciones(S, TT)    // Serial, paralela o selección
            Actualizar S                        // Regla de transformación
            S <- Si+1
    end while
```

Figura 4.18 Algoritmo para la construcción dinámica del WF.

4.7 Utilizando WF BNF para la Composición de redes WF

Una ventaja de la gramática textual BNF definida es que se puede aplicar en la composición de redes WF que consiste en poder reemplazar cualquier transición por una red WF correcta, esto puede realizarse, porque una transición consume un token de su plaza de entrada y después de un tiempo, produce un token en su plaza de salida de la misma forma que una red WF correcta. La composición de redes WF difiere de la composición WF de servicios Web actual, en que en el primero se inserta una red mediante el reemplazo de una transición por dicha red y el segundo consiste en generar WF complejos mediante módulos WF simples almacenados en un repositorio WF (capítulo 2). La composición de redes WF en términos de un proceso es reemplazar una actividad por otra(s) u otro subproceso y en términos de WF se trata también de un cambio en el WF (figura 4.19). El resultado del reemplazo, tal como en la construcción dinámica, genera un WF correcto tanto en su estructura como en su semántica y está basado en la propiedad *sound* definida y demostrada por (Aalst 2000):

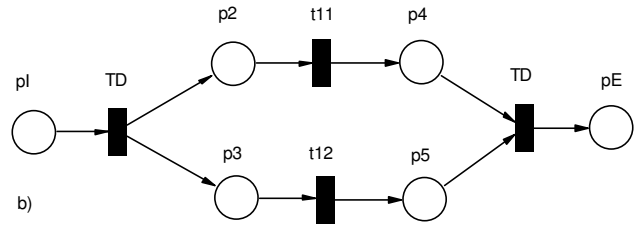
Definición 16 (Composición de redes WF). Si se tienen dos redes WF segura y sound V y W y una tarea t en V que tiene sólo una plaza de entrada y sólo una plaza de salida, entonces se puede reemplazar la tarea t en V por W y la red WF resultante será safe y sound.

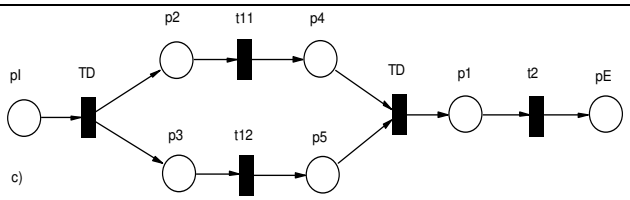


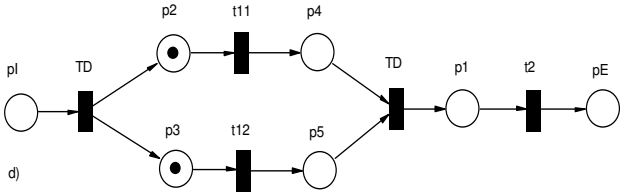
a)

Red WF original: t1 se va a reemplazar

```
WF → SEQ
     → PLC TRANS PLC
     → pI TRANS PLC
     → pI TRANS pE
     → pI TRANS PLC TRANS pE
     → pI t1 p1 t2  pE
```

<table>
<tr>
<td> 

Patrón de reemplazo</td>
<td>WF → SEQ
→ PLC TRANS PLC
→ pI TRANS PLC
→ pI TRANS pE
→ pI TRANS CONCSEQ TRANS pE
→ pI TRANS [SEQ ↕ SEQ] TRANS pE
→ pI TD [PLC TRANS PLC ↕ PLC TRANS PLC] TD pE
→ pI TD [p2 TRANS p4 ↕ p3 TRANS p5] TD pE
→ pI TD [p2 t11 p4 ↕ p3 t12 p5] TD pE</td>
</tr>
<tr>
<td> 

Red WF resultante sin marcaje</td>
<td>WF → SEQ
→ PLC TRANS PLC
→ pI TRANS PLC TRANS pE
→ pI TRANS p1 t2 pE
→ pI TRANS CONCSEQ TRANS p1 t2 pE
→ pI TD [SEQ ↕ SEQ] TD p1 t2 pE
→ pI TD [PLC TRANS PLC ↕ PLC TRANS PLC] TD p1 t2 pE
→ pI TD [p2 t11 p4 ↕ p3 t12 p5] TD p1 t2 pE</td>
</tr>
<tr>
<td> 

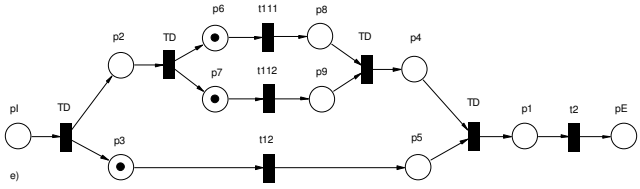
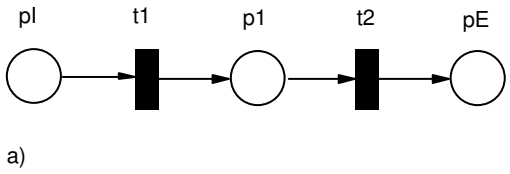
Red WF resultante con marcaje</td>
<td>WF → SEQ
→ PLC TRANS PLC
→ pI TRANS PLC TRANS pE
→ pI TRANS p1 t2 pE
→ pI TRANS CONCSEQ TRANS p1 t2 pE
→ pI TD [SEQ ↕ SEQ] TD p1 t2 pE
→ pI TD [PLC TRANS PLC ↕ PLC TRANS PLC] TD p1 t2 pE
→ pI TD [p2 t11 p4 ↕ p3 t12 p5] TD p1 t2 pE
[$3 = 1 and $6 = 1]</td>
</tr>
<tr>
<td> 
</td>
<td>WF → SEQ
→ PLC TRANS PLC
→ pI TRANS PLC TRANS pE
→ pI TRANS p1 t2 pE
→ pI TD [SEQ ↕ SEQ]TD p1 t2 pE
→ pI TD [PLC TRANS PLC ↕ PLC TRANS PLC] pE
→ pI TD [PLC TRANS CONCSEQ TRANS PLC ↕ p3 t12 p5]TD p1 t2 pE
→ pI TD [p2 TD [SEQ ↕ SEQ] TD p4 ↕ p3 t12 p5] TD p1 t2 pE
→ pI TD [p2 TD [PLC TRANS PLC ↕ PLC TRANS PLC] TD p4 ↕ p3 t12 p5] TD p1 t2 pE
→pI TD [p2 TD [p6 t111 p8 ↕ p7 t112 p9] TD p4 ↕ p3 t12 p5] TD p1 t2 pE
[$5 = 1 and $8 = 1 and $13 = 1]</td>
</tr>
</table>

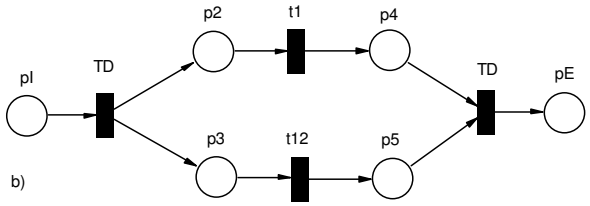
Figura 4.19 Composición de una red WF: cambio dinámico.

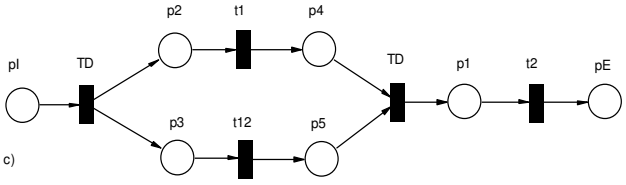
De la figura 4.19 se puede describir lo siguiente, el reemplazo de cualquier transición de la red WF correcta, figura 4.19a) por otra red WF correcta, figura 4.19b), producirá una red WF resultante correcta, figura 4.19c). Al proceso de reemplazo de una transición, tal como se muestra en la figura 4.19, de un WF correcto por otro WF correcto se denomina composición de redes WF y ha sido propuesto y demostrado en (Aalst 2000), dicha demostración sólo lo enfocan en la estructura, no consideran redes con casos y por lo mismo no muestran actualizaciones de estado en las redes WF.

El reemplazo de una transición por una red no implica que la transición se pierda, si se desea conservar simplemente se puede añadir en la nueva red W (comparar figura 4.20 con 4.19). En la figura 4.20 se muestra que la red que va a reemplazar la transición t1 contiene una actividad que realiza lo mismo. Por lo tanto la actividad se conserva en el reemplazo.

Las propiedades de construcción pueden aplicarse en la composición también, por ejemplo, en el caso de que la transición a reemplazar (t1) esté activa, figura 4.19a) se debe actualizar el estado como se muestra en la figura 4.19d). La figura 4.19e) muestra nuevamente el reemplazo de una transición t11 por un patrón paralelo y su actualización de estado, esta red resultante no aplica en la construcción dinámica ya que el patrón que se inserta debe ser el último en la red, en este caso, esta demás la plaza p1 y la transición t2. Note que el reemplazo se representa en BNF mediante la derivación mostrada a la derecha de la figura 4.19c). De esta representación se pueden enfatizar tres puntos importantes, el primero es que la gramática y las acciones semánticas definidas no cambian por hacer un reemplazo, es decir, se pueden utilizar para reemplazar y para insertar, esto se debe a que el nuevo cambio en la red lo decide la función computeFollowSet() teniendo como parámetro la transición donde ocurrirá el cambio. El segundo punto es que se presenta una nueva forma de composición de redes WF usando BNF y complementando la composición propuesta por Aalst (2000), por último, el tercer punto, es que pueden aplicarse las propiedades de construcción para actualizar el estado de la red.

<table>
<tr>
<td> 

Red WF original: t1 se va a reemplazar</td>
<td>WF → SEQ
→ PLC TRANS PLC
→ pI TRANS PLC
→ pI TRANS pE
→ pI TRANS PLC TRANS pE
→ pI t1 p1 t2 pE</td>
</tr>
<tr>
<td> 

Patrón de reemplazo</td>
<td>WF → SEQ
→ PLC TRANS PLC
→ pI TRANS PLC
→ pI TRANS pE
→ pI TRANS CONCSEQ TRANS pE
→ pI TRANS [SEQ ↕ SEQ] TRANS pE
→ pI TD [PLC TRANS PLC ↕ PLC TRANS PLC] TD pE
→ pI TD [p2 TRANS p4 ↕ p3 TRANS p5] TD pE
→ pI TD [p2 t11 p4 ↕ p3 t12 p5] TD pE</td>
</tr>
<tr>
<td> 

Red WF resultante</td>
<td>WF → SEQ
→ PLC TRANS PLC
→ pI TRANS PLC TRANS pE
→ pI TRANS p1 t2 pE
→ pI TRANS CONCSEQ TRANS p1 t2 pE
→ pI TD [SEQ ↕ SEQ] TD p1 t2 pE
→ pI TD [PLC TRANS PLC ↕ PLC TRANS PLC] TD p1 t2 pE
→ pI TD [p2 t1 p4 ↕ p3 t12 p5] TD p1 t2 pE</td>
</tr>
</table>

Figura 4.20 Composición de una red WF: t1 permanece.

4.8 Construcción dinámica y composición de redes WF

La construcción dinámica crea redes a partir de los cuatro patrones básicos y aplicando la definición 15 puede además, unir dos redes WF de cualquier tipo mientras sean correctas. Los cuatro patrones básicos pueden formar casi todo tipo de redes WF correctas, y este "casi", consiste en la limitación del patrón paralelo de sólo poder insertar dos actividades en paralelo. ¿Qué pasaría si se requieren insertar tres o más actividades en paralelo?, la construcción dinámica por sí sola no lo permite ya que requeriría un patrón de tres, un patrón de cuatro, un patrón de *n* actividades lo cuál esta fuera de la realidad. La solución puede observarse en las figuras 4.19d) y 4.19e) en el reemplazo de la secuencia que contiene t11 por una paralela. Si en un patrón básico paralelo se aplica la composición en una de sus ramas puede generarse otro patrón paralelo en la rama dando como resultado tres ramas en paralelo, si nuevamente se aplica la composición en una de las ramas se genera un patrón de cuatro ramas y así sucesivamente hasta el número de

ramas que sea necesario. Esto es posible en la construcción siempre y cuando el estado sea anterior a dónde se va a aplicar el reemplazo, es decir, $\bullet T_{reemplazada} = 1$, de esta manera la construcción permanece correcta y la definición 15 se sigue aplicando (figura 4.19e) sin p1 y t2: $\bullet pE = TD$). El reemplazo de una rama secuencial por una paralela es considerado como un cambio lo que implica el WF construido sea un WF dinámico.

En conclusión, la respuesta a la pregunta anterior es el uso de la construcción dinámica para la creación de WF complementada con la composición de redes WF para la formación de patrones paralelos de *n* ramas, lo que implica poder representar cualquier red WF que se forme con dichos patrones. Las aplicación de las dos propuestas esta sustentada por las definiciones 15 y 16 lo que hacen que los WF permanezcan siempre correctos.

Con respecto al BNF, la gramática no cambia y se sigue aplicando tal y como se muestra en la sección 4.7 y sigue complementando la modelación visual para ambas propuestas.

4.9 Composición WF de servicios Web

El concepto de composición de servicios Web consiste en dos aspectos fundamentales (Cardoso and Seth 2002):

1. El descubrimiento de servicios Web
2. Interoperabilidad al integrarlo

DCW facilita la composición permitiendo abordar los dos aspectos anteriores. En el primer caso la función *computeFollowSet()* puede ser implantada para el descubrimiento de los servicios Web, existen muchos métodos para realizarlo ver sección 2.3.2. El segundo aspecto, la interoperabilidad de los servicios a la hora de integrarlos, puede ser mantenida de forma correcta por la construcción dinámica, no importa si un servicio Web representa una transición, una patrón básico o un subproceso, en cualquiera de los tres casos la integración de los servicios se mantiene correcta al crear la red WF que los represente. La ventaja de DCW es que permite separar correctamente el aspecto del

descubrimiento con el de interoperabilidad, por lo mismo es indistinto el método usado para descubrir los servicios, la interoperabilidad siempre estará garantizada.

4.10 Validación del modelo

Para la validación del modelo se debe demostrar que la construcción dinámica garantiza que todo WF que se construya representando un proceso será correcto en su estructura y en su semántica, debido a que siempre preservará las propiedades de *red segura* y *sound*. La definición 15 valida todo el proceso de construcción dinámica cuando se une una red WF con un patrón de inserción. Dicha validación sólo requiere que los patrones de inserción sean redes WF correctas. A continuación se describe el proceso de validación de los patrones

Patrones WF

Las redes WF usadas para la construcción dinámica son redes *free-choice* (definición 6) y pueden representar secuencias, paralelismo y selección. En Murata 1989 se muestra que las redes *free-choice* son una generalización de las máquinas de estado y los grafos marcados.

Una máquina de estado es una red de Petri tal que cada transición t tiene exactamente una plaza de entrada y una plaza de salida:

$| \bullet t | = |t \bullet| = 1$ para todo $t \in T$.

Los patrones usados que corresponden a una máquina de estado son la secuencia y la selección (X-OR split y X-OR join).

Como un disparo de una transición de una máquina de estado mueve sólo un token de una plaza a otra, los siguientes teoremas de (Murata, 1989) pueden probarse con las definiciones 9, 10 y 11:

Teorema 1.- Una máquina de estado (PN, Mo) está viva si y sólo si es fuertemente conectada y m(pI) = 1.

Teorema 2.- Una máquina de estado (PN, Mo) es safe si y sólo si m(pI) tiene a lo más un token. Una máquina de estado (N, Mo) es safe si y sólo si m(pI) = 1.

Con base en lo anterior la secuencia y la selección son redes correctas.

Un grafo marcado es una red de Petri tal que cada plaza *p* tiene exactamente una transición de entrada y una transición de salida:

$$|\bullet p| = |p\bullet| = 1 \qquad \text{para todo } p \in P.$$

El patrón usado que corresponde a grafos marcados es la estructura paralela (AND split y AND join). En este caso, también el disparo de cualquier transición mueve exactamente un token de una plaza a otra, lo que permite que toda transición se active. Dado un estado inicial (PN, MpI) cualquiera de las dos secuencias de disparo del patrón ver figura 4.11a) α = {{t1TDt2t3tD}, {t1TDt3t2tD}}guiará del estado inicial al estado final (PN, MpE)

$$MpI \xrightarrow{\alpha} MpE$$

cumpliendo con las definiciones 2, 3 y demostrándose con las definiciones 9, 10 y 11.

Es fácil ver que los patrones son redes WF correctas y que la creación de redes WF con dichos patrones forman redes WF correctas. El proceso general de construcción dinámica no sólo permite formalmente la unión de una red WF y un patrón básico, también permite formalmente la unión de dos redes WF cualquiera que soporte los patrones, es decir, permite la unión de dos redes WF correctas. Esto permite construir redes WF a partir de redes WF correctas complejas, por ejemplo, con el proceso general de construcción dinámica se pueden construir una red WF que conste de las redes mostradas en la figura 4.17g) y 4.19e).

4.11 DCW con respecto a las cinco modificaciones

A continuación se muestra la comparación de DCW con respecto a las cinco modificaciones con que se analizaron los enfoques estudiados en el estado del arte:

1. Cambios en el pasado
    a. Permite cambios puros en el pasado pero no resuelve el problema de inconsistencia de datos.
    b. Permite la compartición de datos con redes WF, falta la definición en BNF.

2. Tolerancia a ciclos
    a. El modelo no utiliza ciclos, el problema es prevenido.

3. Estado de actividades
    a. Distingue los estados de las actividades entre activas y en ejecución mediante las acciones semánticas en BNF.

4. Cambio de orden
    a. Las propiedades de actualización de estado no permiten el cambio de orden.

5. Inserción paralela
    a. Permite la inserción paralela.

El objetivo de DCW es la construcción dinámica de WF y la adaptabilidad de las instancias, si se le añade que resuelve cuatro de las cinco modificaciones se considera como un modelo robusto y flexible para la representación y administración de procesos de negocio.

4.12 Conclusiones del capítulo

DCW permite la modelación correcta de todo proceso que pueda construirse con los patrones y su proceso de construcción dinámica. Garantiza la construcción de la red y la ejecución de cada caso de forma correcta gracias a sus dos formalismos de modelación:

redes de Petri y BNF. La modelación visual con redes de Petri es fácil de aplicar y permite ilustrar completamente los procesos de negocio y el estado en que se encuentran permitiendo su monitoreo en todo momento.

La modelación textual con BNF complementa la visual y genera cinco ventajas importantes:

1. La primera es la modelación textual propuesta para modelar WF denominada en esta tesis WF BNF.
2. La segunda permite conocer el estado de un proceso, mediante sus acciones semánticas, y diferenciar los estados de una transición, por ejemplo, si se encuentra en estado activo o de ejecución.
3. La tercera ventaja permite que los patrones sean genéricos al utilizar no-terminales para representarlos y poder acceder sus plazas/transiciones mediante acciones semánticas.
4. La cuarta ventaja es la combinación de las tres anteriores, y consiste en la creación de una gramática con acciones semánticas que permite modelar correctamente WF y sobretodo ser la base para la creación de una herramienta que analice y verifique WF tal como lo hacen los compiladores con los lenguajes de programación.
5. Por último, la última ventaja es que la gramática esta definida de tal manera que puede representar la construcción dinámica, la composición de redes WF y la combinación de ambos.

La parte central de DCW son sus dos formalismos de modelación y su proceso general de construcción dinámica que no sólo permite la inserción de patrones básicos WF sino también la unión correcta de dos WF correctos. La combinación del proceso general de construcción, la composición de redes WF y la gramática definida para representar WF BNF crea un modelo robusto, completo y formal para la especificación y construcción dinámica de WF dinámicos.

Capítulo 5. Implantación del modelo

5.1 Introducción

En este capítulo se presenta la aplicación con la que se implanta el modelo DCW. El objetivo de la implantación es cómo el modelo puede representar una aplicación con un dominio de conocimiento específico. Dicha aplicación se insertará en un sistema experto y es de capital importancia la modelación del sistema y la correctes con la que se ejecuta mediante DCW. La tesis no se centra en el estudio de sistemas expertos, por lo mismo, no se profundiza en esta área de conocimiento. La aplicación se desarrolló con una herramienta para el desarrollo de sistemas expertos denominada *JESS*. La validez del modelo desarrollado en el capítulo anterior garantiza que la aplicación se ejecutará de manera correcta sin redundancias ni saltos que generalmente se presentan en este tipo de aplicaciones.

5.2 DIACNE

DIACNE es una aplicación que ayuda en la toma de decisiones dentro del ámbito de los negocios electrónicos a las pequeñas y medianas empresas (Espinosa et al. 2005) (Espinosa et al. 2004b). Fue desarrollado por un grupo de Investigación del Tecnológico de Monterrey Campus Ciudad de México en la cátedra de "Explotación de la Información para la Toma de Decisiones". DIACNE consta de dos módulos principales DICNE y ACNE, los cuales interactúan con un repositorio de información (Espinosa et al. 2004b). Los módulos fueron escritos en *Java* utilizando las tecnologías *jsp* y *servlets*; y probados como aplicaciones Web en un servidor de aplicaciones *Tomcat*.

DICNE es un módulo de Software disponible electrónicamente en la Web y consiste en (Espinosa et al. 2005a) 1) Llenado de datos generales de la empresa 2) Despliegue de la encuesta que debe contestar el empresario 3) Presentación de un resumen del estado actual de la empresa, de acuerdo a las respuestas obtenidas.

ACNE es otro módulo utilizado para la ayuda de Toma de Decisiones basado en la arquitectura básica de un Sistema Tutor Inteligente. ACNE ayuda a los empresarios en el uso de DICNE, con el fin de evitar que por desconocimientos de los conceptos solicitados en las preguntas de la encuesta, proporcionen información errónea o la

abandonen antes de concluirla. En (Espinosa et al. 2005b) se detalla completamente la arquitectura de ACNE.

Funcionamiento de DIACNE

El empresario ingresa y se registra en DIACNE a través de la Web, utilizando un navegador de Internet. DICNE inicia preguntando las características generales de la empresa las cuales utiliza para clasificarla; consulta su repositorio de preguntas y respuestas, selecciona la siguiente, se la presenta al empresario, éste selecciona la respuesta y la envía a DICNE, la respuesta se almacena en la base de datos y es presentada la siguiente pregunta al empresario.

A medida que DICNE va presentando la encuesta, ACNE se encuentra a la escucha de lo que se pregunta y se contesta. El propósito de ACNE es aparecer en puntos específicos durante la sesión del cuestionario e interrumpir momentáneamente la encuesta cuando:

- Explica un concepto posiblemente confuso para el usuario (Espinosa et al. 2004b).
- Presenta advertencias cuando las respuestas del usuario sean inconsistentes entre ellas.

Cuestionario de DIACNE

El cuestionario de DIACNE esta diseñado con base en siete factores de éxito que se utilizan como métrica para la evaluación del estado de una empresa (Espinosa et al. 2004b). Cada factor a su vez esta dividido en subfactores con la finalidad de que la evaluación de cada factor sea más precisa. El cuestionario esta diseñado de manera que incluye preguntas para cada subfactor y por lo mismo de cada factor. La tabla 5.1 muestra los siete factores con sus respectivos subfactores, para el significado detallado de cada uno ver (Espinosa et al. 2004a y 2004b).

FACTORES CRITICOS DE ÉXITO	
Factor	Subfactores
1. Actitud	1. Liderazgo
	2. Organización
2. Beneficios	1. Aumento de Ingresos
	2. Expansión de Mercados
	3. Acceso a la Información del Cliente
	4. Reducción de Costos
3. Tecnologías de Información	1. Equipamiento
	2. Internet
	3. ERP
	4. Cadena de Suministro
	5. CRM
4. Finanzas	1. Estructura de costos
	2. Ingresos
5. Cultura Informática	1. Sistemas Administrativos
	2. Manejo de Información
6. Recursos Humanos	1. Capacitación de personal
	2. Desarrollo de Competencias Laborales
7. Implantación	1. Orientación Estratégica
	2. Documentación y Actualización
	3. Capacitación y Soporte

Tabla 5.1 Factores y subfactores de DIACNE.

La relación entre factores está definida por el mapa semántico mostrada en la figura 5.1. Dicha red forma la base de conocimientos del sistema actual de DIACNE (Espinosa et al. 2005b).

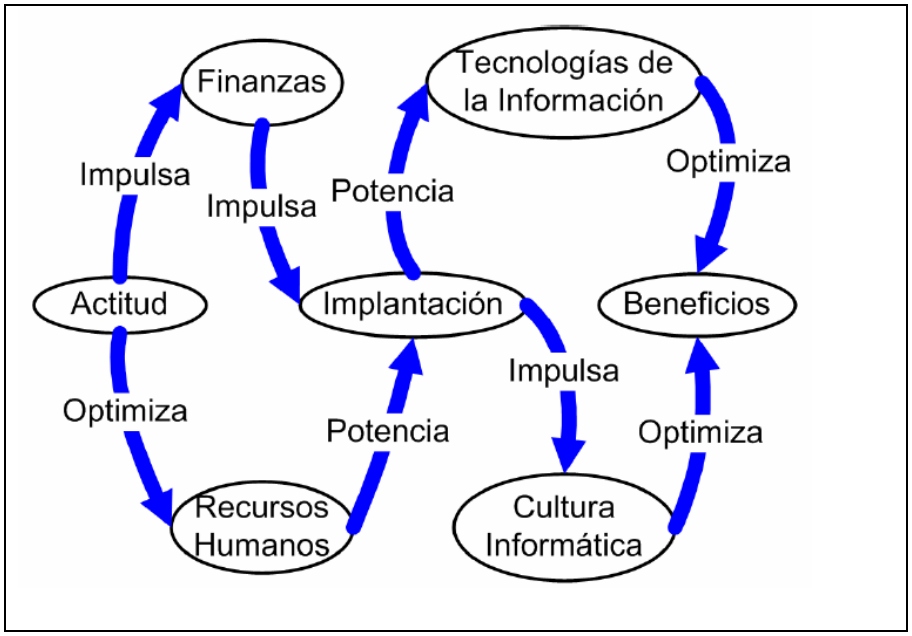


Figura 5.1 Mapa semántico de los factores de DIACNE.

La relación entre preguntas y respuestas del cuestionario de DIACNE se basa en el mapa semántico y se describe a través de una tabla relacional. La figura 5.2 muestra los primeros diez registros de la tabla (Espinosa et al. 2005b).

IdRegla	IdSector	IdFactor	IdSubFactor	IdPregunta	IdRespuesta	TipoRegla	Regla	IdReglaAnt
1	CO	1	1	1		Inicio		0
2	CO	1	1	1	a	Pregunta	CO.1.1.2	0
3	CO	1	1	1	b	Pregunta	CO.1.1.4	0
4	CO	1	1	2		Pregunta	CO.1.1.3	0
5	CO	1	1	3	a	Pregunta	CO.1.1.4	0
6	CO	1	1	3	b	Pregunta	CO.1.1.4	0
7	CO	1	1	3	c	Pregunta	CO.1.1.4	0
8	CO	1	1	3	d	Pregunta	CO.1.1.4	0
9	CO	1	1	4	a	Pregunta	CO.1.1.5	0
10	CO	1	1	4	b	Pregunta	CO.1.1.5	0
11	CO	1	1	4	c	Pregunta	CO.1.1.5	0
12	CO	1	1	5	a	Pregunta	CO.1.1.6	0
13	CO	1	1	5	b	Pregunta	CO.1.1.6	0
14	CO	1	1	5	c	Pregunta	CO.1.1.6	0
15	CO	1	1	5	d	Pregunta	CO.1.1.6	0
16	CO	1	1	6	a	Pregunta	CO.1.2.1	0
17	CO	1	1	6	b	Pregunta	CO.1.2.1	0
18	CO	1	1	6	c	Pregunta	CO.1.2.1	0
19	CO	1	1	6	d	Pregunta	CO.1.2.1	0
20	CO	1	1	6	e	Pregunta	CO.1.2.1	0

Figura 5.2 Encabezado de la tabla relacional del cuestionario de DIACNE.

Cada registro esta dividido en nueve campos mostrados en la figura 5.2 y cada campo significa lo siguiente:

IdRegla

Es el número de pregunta.

Id Sector

Indica el sector: CO (comercio), IN (industria) y SE (servicios). La tabla administra tres sectores y cada sector contiene el mismo cuestionario.

IdFactor

Indica el factor al que pertenece cada pregunta.

IdSubFactor

Indica el subfactor al que pertenece cada pregunta.

IdPregunta

Indica el número de pregunta con base en el factor y el subfactor.

IdRespuesta

Indica el inciso de cada respuesta. Si una pregunta tiene cinco incisos entonces habrá cinco registros cada uno con una respuesta por inciso.

TipoRegla

Contiene una cadena de caracteres, el primer campo contiene la cadena *Inicio* y todos los demás contienen la cadena *Pregunta*.

Regla

Indica cuál pregunta sigue con respecto al campo *Id Respuesta*. Por ejemplo, en la figura 5.2, la pregunta dos indica que si contestan el inciso a) como respuesta la siguiente pregunta deberá ser la 1.1.2.

IdReglaAnt

Campo sin información para uso futuro.

Análisis de DIACNE

Actualmente DIACNE tienes tres sectores y cada sector contiene el mismo cuestionario, la tabla relacional de preguntas consta de 1500 registros, donde cada pregunta y cada opción de respuestas es un registro, es decir, si una pregunta contiene cinco opciones de respuesta habrá cinco registros para esa pregunta uno por respuesta. El sistema no contiene herramientas manejadoras de bases de datos para su automatización y cada pregunta esta ligada a una regla si-entonces implícita que con base en la respuesta se selecciona la siguiente pregunta, por lo mismo el cuestionario es un árbol estático complejo, difícil de modificar ya que para ello se deben cambiar en forma manual todas las reglas posteriores a la modificación. Además del problema de la modificación, se tiene el problema de un crecimiento exponencial del árbol a medida que se requiera añadir preguntas. Otra desventaja, es que no existen propiedades que verifiquen la

secuencia correcta de las preguntas, por ejemplo, que se salte una(s), que haya redundancias o ciclos repetitivos de ellas que generen una operación infinita, es decir, que nunca termine el cuestionario.

Todo lo anterior implica la necesidad de cambiar la tabla relacional con sus reglas en un sistema portable, fácil de expresar, de modificar y de añadir preguntas. La combinación de DCW con un sistema experto permite crear el sistema ideal para resolver las desventajas anteriores. Con DCW se garantiza la correctes de la secuencia del cuestionario y el sistema experto permite adminstrar la relación de preguntas de manera eficiente gracias a su motor de inferencias. También, se resuelve el problema complejo que se tiene actualmente de añadir más preguntas o modificar el cuestionario. La combinación de DCW con el sistema experto permite las siguientes ventajas sobre el sistema DIACNE actual:

- Construcción dinámica genérica para la estructura del cuestionario
- Formato de reglas portable sin la necesidad de recompilar el sistema.
- Integración del dominio del conocimiento en DCW mediante un sistema experto.
- Verificación correcta de la secuencia del cuestionario mediante DCW evitando redundancias, ciclos y saltos de preguntas.
- Uso de un motor de inferencias para la generación de la siguiente pregunta.

5.3 Sistemas expertos basados en reglas

Los sistemas expertos son una rama de la inteligencia artificial que hace un uso amplio del conocimiento especializado que intenta resolver problemas como un especialista humano (Friedman-Hill 2003). Esto es, el especialista tiene conocimientos o habilidades especiales que la mayoría no conoce o dispone y puede resolver problemas que la mayoría no podría o los resuelve con mayor eficiencia.

El conocimiento de los sistemas expertos puede obtenerse por experiencia o consulta de libros, revistas, encuestas validadas y de personas capacitadas. El término sistema experto también se denomina sistema basado en conocimiento o sistema experto basado en conocimiento. Actualmente existen dos tipos de sistemas expertos según la naturaleza del problema:

- Deterministas
- Estocásticos

Los problemas de tipo determinista pueden ser formulados usando un conjunto de reglas que relacionen objetos bien definidos. El estado actual depende del estado anterior y las acciones sobre el entorno. Se denominan sistemas expertos basados en reglas y usan un mecanismo de razonamiento lógico para obtener conclusiones.

En los problemas de tipo estocástico existen situaciones inciertas por lo que es necesario introducir algunos medios para tratar la incertidumbre. La incertidumbre puede considerarse como la falta de información adecuada para tomar una decisión y genera un problema porque puede evitar que se tome la mejor e incluso una equivocada. Se han desarrollado varias teorías para manejar la incertidumbre y pueden clasificarse entre las que utilizan la probabilidad y las que utilizan otras teorías tales como Inteligencia Artificial (Giarratano 2001):

- Las teorías que utilizan la probabilidad se denominan de incertidumbre reproducibles. Por ejemplo, teorías que utilizan la probabilidad clásica, condicional, bayesiana, de razonamiento temporal y las cadenas de Markov.
- Las teorías no probabilísticas que utilizan factores de incertidumbre y se consideran de razonamiento inexacto, por ejemplo, la teoría de Dempster y Shafer (Dempster 1967) (Shafer 1976), y la teoría de la lógica difusa (Zadeh 1965).

Actualmente los sistemas basados en reglas intentan reemplazar el conocimiento experto humano y se enfocan en automatizar o codificar prácticas de negocio u otras actividades que son parte fundamental de toda organización. Muchos servidores de aplicaciones comerciales incorporan un motor de reglas (motor basado en reglas) y otros explícitamente o implícitamente ofrecen capacidades de integración con ellos.

La modelación de DIACNE implica transformar el cuestionario a términos WF, representarlo en la base de conocimientos y definir un conjunto de reglas de inferencia para las respuestas del usuario a la hora de contestar el cuestionario. El motor de

inferencias del sistema experto con la aplicación de estas reglas obtendrá conclusiones y seleccionará la siguiente pregunta. Esto implica que el cálculo de la siguiente pregunta depende del estado anterior, es decir, de las respuestas del usuario y de la aplicación de las reglas. La modelación de DIACNE requiere de un sistema experto basado en reglas porque la base de conocimiento contiene el conjunto de reglas que definen el problema y no se tiene incertidumbre.

5.3.1 Arquitectura de un sistema basado en reglas

Un sistema basado en reglas es un programa de cómputo que usa reglas para llegar a conclusiones a partir de un conjunto de premisas. Sus raíces históricas incluyen sistemas de producción y en la actualidad tiene un amplio rango de aplicaciones que incluye desde sistemas de control integrados de tiempo real hasta la planeación de recursos empresariales para corporaciones multinacionales. Los sistemas basados en reglas requieren un enfoque diferente de programación en el cuál un sistema de ejecución en tiempo real se usa para las decisiones de flujo de control y de planificación.

La figura 5.3 muestra la arquitectura típica de un sistema basado en reglas (Friedman-Hill 2003) (Giarratano and Riley 2001). El sistema trabaja de la siguiente manera: el igualador de patrones aplica las reglas de la base de conocimiento a los hechos en la memoria activa para construir la agenda. El motor de ejecución dispara las reglas de la agenda, cambia el contenido de la memoria activa y el proceso entero es repetido. A continuación se describen los componentes:

Motor o mecanismo de inferencia (*inference engine*).

- Hace inferencias al decidir cuáles reglas satisfacen los hechos u objetos. Usualmente trabaja en ciclos discretos y un ciclo consiste:
    1. Todas las reglas de la base de conocimiento se comparan con respecto a los hechos por el igualador de patrones y decide cuáles deben activarse. La lista generada de reglas activas en el ciclo y por las de ciclos anteriores se denomina conjunto de conflictos (conflict set).
    2. El conjunto de conflictos ordenado forma la agenda que consiste de la lista de reglas cuyo lado derecho será ejecutado o disparado. El proceso de

ordenamiento de la agenda se denomina resolución de conflictos, por ejemplo, por prioridad.

3. Para terminar la ejecución del ciclo la primera regla se dispara con el posible resultado de un cambio en la memoria activa y se repite el proceso para la ejecución de un nuevo ciclo.

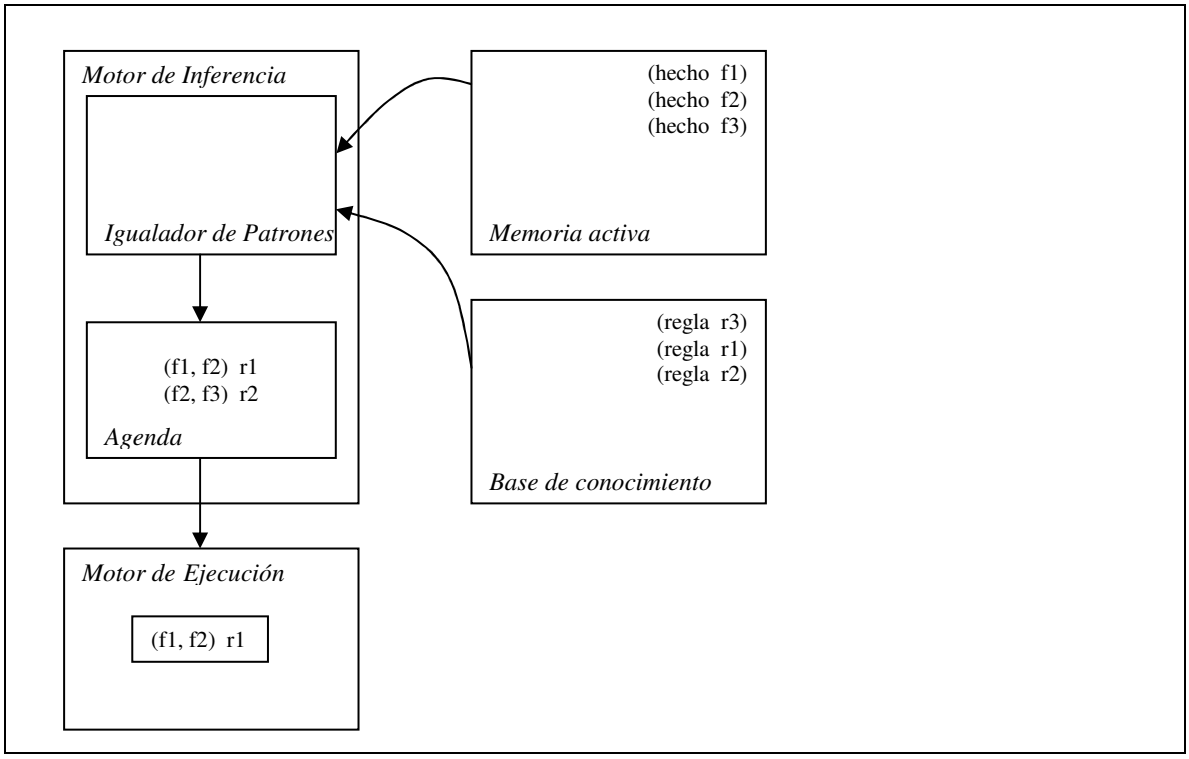


Figura 5.3 Arquitectura típica de un sistema basado en reglas.

Base de conocimiento de reglas (*rule base*).

- Contiene el conocimiento del dominio necesario para resolver los problemas codificados en forma de reglas. Las reglas son un paradigma para representar conocimiento.
- Las reglas pueden almacenarse como cadenas de caracteres y a menudo un compilador de reglas las procesa de tal manera que el motor de inferencias trabaje con mayor eficiencia. Actualmente los sistemas expertos compilan las reglas para construir una estructura compleja de datos indexada denominada red *Rete*, la cuál facilita el procesamiento de reglas.

- Algunos motores de reglas permiten almacenar las reglas en una base de datos relacional externa y otros la tienen integrada.

Memoria activa o de trabajo (*working memory*).

- Es una base de datos de los hechos usados por las reglas y es denominada como base de hechos.
- Puede contener las premisas y las conclusiones de las reglas.

Igualador de patrones (*pattern matcher*)

- Su propósito es decidir cuáles reglas se aplican con base en el contenido actual de la memoria activa. El proceso de igualación de patrones siempre se mantiene actualizado y se lleva a cabo independientemente de si los hechos se afirman antes o después que se ha definido una regla. Típicamente esto es un proceso que requiere de mucho procesamiento, por ejemplo, si la memoria activa contiene miles de hechos y cada regla contiene dos o tres premisas, el igualador de patrones necesita buscar a través de millones de combinaciones de hechos para encontrar aquellas que satisfagan las reglas. Para este proceso, los sistemas actuales utilizan el algoritmo *Rete* desarrollado en 1979 Charles L. Forgy. El algoritmo es un rápido igualador de patrones que en lugar de tener que igualar los hechos con todas las reglas en cualquier ciclo, sólo busca los cambios en las correspondencias de cada ciclo (Giarratano and Riley 2001).

Agenda

- Es el lugar dónde se almacenan las reglas que potencialmente se pueden disparar. Es responsable de usar una estrategia de conflictos para decidir de todas las reglas que se aplican cuál tiene la mayor prioridad y debe dispararse primero. En otras palabras, es una lista con prioridades asignadas a las reglas, creada por el mecanismo de inferencia, cuyos patrones satisfacen los hechos de la memoria activa.

- La estrategia de conflictos comúnmente toma en cuenta la especificidad o complejidad de cada regla y el tiempo relativo de la premisa en la memoria de trabajo. Cada motor de inferencia tiene su enfoque en particular.

Motor de ejecución

- Es el componente del motor de reglas que dispara las reglas. Los sistemas actuales ofrecen un lenguaje de programación completo que puede usarse para definir qué pasa cuando una regla se dispara, por lo mismo, el motor de ejecución representa el ambiente en el cuál el lenguaje de programación se ejecuta.

5.4 Jess: Java Expert System Shell

Jess es un motor de reglas y un lenguaje de interpretación desarrollado en el lenguaje *Java* por Sandia Nacional Laboratories en Livermore, California a finales de 1990 (Friedman-Hill 2003). En la actualidad *Jess* se ha usado para desarrollar un amplio rango de herramientas de *SW* comercial tales como:

- Sistemas expertos
- Agentes de predicción de la bolsa de valores
- Auditores de seguridad y detección de intrusos en redes
- Interruptores inteligentes de redes para telecomunicaciones
- Servidores para ejecutar reglas de negocios
- Sitios de comercio electrónico inteligente
- Juegos

La programación con *Jess* puede ser de dos maneras. La primera consiste en usar a *Jess* como un motor de reglas, un programa basado en reglas puede aplicar miles de reglas a los datos. Generalmente las reglas representan el conocimiento heurístico de un experto humano en algún dominio de conocimiento y la base de dicho conocimiento representa el estado de una situación cambiante, por ejemplo, una entrevista o una emergencia. En este caso se dice que las reglas constituyen un sistema experto y recientes aplicaciones de dichos sistemas incluyen parte del razonamientos de agentes inteligentes

para sistemas ERP y uso y validación en comercio electrónico. La segunda manera de usar *Jess* es como un lenguaje de programación de propósito general que puede acceder directamente clases y bibliotecas del lenguaje de programación *Java*. Por esta razón *Jess* frecuentemente es usado como ambiente de desarrollo de aplicaciones. Una ventaja sobre *Java* es que *Jess* no requiere el proceso de compilación para poder ejecutar el código, es decir, permite codificarlo y ejecutarlo inmediatamente después de ser tecleado, lo que permite experimentar con APIs de *Java* de manera interactiva y construir aplicaciones de manera incremental.

Su uso como lenguaje de programación y como motor de reglas le da a *Jess* lo mejor de esos dos mundos, permitiendo extender el lenguaje con nuevos comandos y ser configurado para aplicaciones específicas, esto le da una amplia ventaja que si se usara de manera independiente como un motor o un lenguaje de programación. La arquitectura de *Jess* consiste de los componentes mostrados en la figura 5.3.

Dada su flexibilidad, *Jess* puede usarse en aplicaciones de líneas de comando, aplicaciones *GUI, servlets y applets*. *Jess* puede proporcionar la función *main()* para un programa en *Java* o puede escribirlo el programador en *Java*. Se pueden desarrollar aplicaciones *Java* con Jess con interfase o sin interfase gráfica sin compilar una sola línea de código en *Java*. Se pueden escribir aplicaciones en *Jess* que sean controladas enteramente por código en *Java* con un mínimo de código en *Jess*. Las aplicaciones que pueden desarrollarse pueden ser (Friedman-Hill 2003):

1. En puro lenguaje *Jess* sin código en *Java*
2. En puro lenguaje *Jess* con acceso al API de *Java*
3. En puro lenguaje *Jess* y nuevos comandos en Jess escritos con *Java*
4. En lenguaje *Jess* y código en *Java* con la función principal escrita en *Jess* o *Java*.
5. En código en *Java* llamando código en *Jess* en la ejecución.

La ejecución de *Jess* es rápida, el algoritmo usado para la igualación de patrones es una versión muy eficiente y mejorada del algoritmo *Rete*. Este algoritmo sacrifica espacio de disco por velocidad, de tal forma que *Jess* utiliza bastante memoria. Esto no es

un problema en las computadoras actuales pero el rendimiento es sensible al comportamiento del colector de basura de *Java*.

5.5 Implantando DCW

El proceso de implantación consiste en modelar y ejecutar DIACNE con DCW. Dicha implantación consiste en modelar la aplicación con DCW y diseñar el sistema experto que calculará la siguiente pregunta con base en las anteriores, siendo el sistema experto una aplicación desarrollada de tipo cinco descrita en la sección anterior.

En términos de DCW el sistema experto realiza las operaciones de la función *computeFollowSet()* mostrado en el algoritmo de ejecución descrito en la figura 4.18 del capítulo anterior. El proceso de implantación de DCW incluye la migración de DIACNE a un sistema basado en reglas y consta de los siguientes pasos ver (Capítulo 4) y (Kim et al. 2007):

1. Modelación de DIACNE con DCW.
2. Análisis de la tabla relacional de DIACNE.
3. Definición de tipo de documento DTD para la verificación de correctes del archivo XML.
4. Transformación de los registros de la tabla relacional a reglas en XML.
5. Transformación de las reglas XML a reglas en Jess.
6. Construcción de la red de Petri de DIACNE mediante la ejecución del sistema basado en reglas.

1. Modelación de DIACNE con DCW

La modelación de DIACNE implica primero asociar la aplicación en términos de redes WF:

- Cada cliente que contesta el cuestionario se considera un caso.
- Una tarea es una pregunta y se representa por una transición.

- Las condiciones se representan por plazas y se construyen al terminar de contestar una pregunta.
- Una actividad es el proceso de contestar una pregunta.

Las cuatro propiedades generales de correctes se transforman en términos del cuestionario de la siguiente manera:

DCW1

Si dos preguntas Q_i , Q_j están relacionadas causalmente $Q_i \rightarrow Q_j$ y si Qi es seleccionada entonces en un momento futuro podrá seleccionarse Q_j. De manera inversa, para que Q_j sea seleccionada, Q_i debió de haber sido contestada.

DCW2

Toda pregunta que no tiene relación con ninguna otra finalmente será seleccionada para cada caso. Las preguntas correspondientes a diferentes factores de la encuesta pueden ser aplicadas en cualquier orden.

DCW3

Si Qi y Qj son dos preguntas que no están relacionadas, entonces su presentación al usuario es indistinta y pueden realizarse en paralelo.

DCW4

En cada sesión a la hora de contestar el cuestionario no pueden repetirse las preguntas, es decir, una pregunta sólo puede seleccionarse una vez. Las preguntas representadas por transiciones son únicas en la red si $\forall$ ti $\in$ T | ti $\neq$ tj, $1 \leq i, j \leq n$. En el caso de que una pregunta esté relacionada con otra, el orden de las mismas debe ser respetado.

2.- Análisis de la tabla relacional de DIACNE

El esquema de la tabla relacional de DIACNE es el siguiente:

Reglas
IdRegla
IdSector
IdFactor
IdSubFactor
IdPregunta
IdRespuesta
TipoRegla
Regla
IdReglaAnt

De los nueve campos de la tabla existen cuatro campos que no aportan información y por lo mismo no se necesitan:

1. *IdRegla*, cada pregunta puede numerarse con base en el factor y sub factor por lo que este campo no se requiere.
2. *IdSector*, siempre es el mismo cuestionario para cada sector por lo que no es necesario.
3. *TipoRegla*, sólo es una cadena redundante de caracteres.
4. *IdReglaAnt*, es un campo sin utilizar.

3. Definición de tipo de documento (DTD) para la verificación de correctes del archivo XML.

Para la representación de las relaciones entre preguntas mostradas en la tabla relacional por los registros, se usa un formato de texto XML fácil de entender y manipular. Se utiliza un DTD para la representación y validación de cada regla, dicho formato se muestra en la figura 5.4

```
<!ELEMENT rules (rule+)>
<!ELEMENT rule (and*, or*, condition*, function)>
<!ATTLIST rule name CDATA #REQUIRED>
<!ELEMENT and (condition+, or*, and*)>
<!ELEMENT or (condition+, or*, and*)>
<!ELEMENT condition (question)>
<!ELEMENT question(#PCDATA)>
<!ATTLIST question number CDATA #REQUIRED answer CDATA #REQUIRED)>
<!ELEMENT function(#PCDATA)>
```

Figura 5.4 DTD para la representación y validación de reglas en XML.

Y un ejemplo de una regla en XML se muestra a continuación:

```
<rules>
  <rule name="question_1.1.1-a">
    <and>
      <condition>
        <question number="1.1.1" answer="a" />
      </condition>
      <condition>
        <question number="0.0.0" answer="a" />
      </condition>
    </and>
    <function>(fire-question 1.1.2)</function>
  </rule>
</rules>
```

Toda regla tiene un nombre, para facilidad de lectura se establece la palabra "*question*" seguida de un guión bajo y el numero de pregunta, después se enlista la secuencia de la regla según su pregunta, esto es debido a que para cada respuesta de una pregunta existe una regla. Cada regla puede tener condiciones agrupadas en operadores *and* u *or*, dentro de cada condición existe una pregunta, que tiene las propiedades de número de pregunta y la respuesta dada para esta pregunta. Después de la declaración de condiciones y preguntas, viene el llamado a una función, que se va a ejecutar en cuanto se cumpla.

4. Transformación de los registros de la tabla relacional a reglas en XML

Expresar las reglas de la tabla en un archivo XML va a permitir modificar y/o añadir más reglas sin recompilar el sistema basado en reglas. La transformación de un registro a una regla en XML queda de la siguiente manera:

Dado el siguiente registro:

IdRegla	IdSector	IdFactor	IdSubFactor	IdPregunta	IdRespuesta	TipoRegla	Regla	IdReglaAnt
5	CO	1	1	3	a	Pregunta	CO.1.1.4	0

La regla resultante es:

```
<rules>
  <rule name="question_1.1.3-a">
   <condition>
       <question number="1.1.3" answer="a" />
   </condition>
   <function>(fire-question 1.1.4)
   </function>
  </rule>
</rules>
```

Los cinco campos que quedan del registro se interpretan de la siguiente manera:

- El *IdFactor*, *IdSubFactor* e *IdPregunta* se juntan en una cadena de caracteres separados por puntos, por ejemplo: “1.1.3”.
- *IdRespuesta* permanece intacta.
- *Regla* es la pregunta a disparar cuando se cumpla la regla.

5. Transformación de las reglas XML a reglas en Jess.

Jess es un motor de reglas principalmente enfocado en el desarrollo de sistemas expertos, facilita a los desarrolladores incorporarlo en sistemas desarrollados en *Java* y el acceso a todas sus bibliotecas. Para representar las reglas en *Jess* simplemente se definen las preguntas contestadas como hechos, a continuación se muestra el proceso.

Reglas

La representación de reglas mediante *Jess* se realiza de la siguiente forma:

```
(defrule regla
        (a)
        =>
        (llamada_función parámetros) )
```

Dónde:

defrule	Define una nueva regla
regla	Es el nombre de la regla
=>	Símbolo que separa la premisa (parte izquierda) de la acción/conclusión (parte derecha)
(a)	Son los hechos
Llamada_función parámetros	Funciones que se ejecutan cuando la reglas se cumplan.

Para representar los hechos se define una plantilla para almacenar las preguntas y las respuestas, cada ranura (*slot*) de la plantilla es un espacio de memoria dónde se almacenan datos:

```
(deftemplate questionTemplate
(slot numQuestion)
(slot answer))
```

el primer *slot* se utiliza para almacenar el número de pregunta y el segundo para almacenar la respuesta.

Combinando la definición de reglas y la plantilla se definen las reglas que se usan para las preguntas y respuestas, por ejemplo:

```
(defrule question_1.1.1-a
(and
(questionTemplate (numQuestion 1.1.1) (answer a))
(questionTemplate (numQuestion 0.0.0) (answer a)))
=>
(fire-question 1.1.2))
```

dónde:

- Cada regla tiene el nombre de una pregunta, no puede haber dos o más reglas con el mismo nombre.
- En el ejemplo la regla contiene un *and* que encapsula dos preguntas y respuestas que se deben de dar al sistema para que la regla se active.
- La conclusión de la regla consiste en llamar a la función fire-question y el número de pregunta que se debe desplegar.

Hechos

Para agregar más hechos a la memoria activa de *Jess* se utiliza la función (assert (hecho)). Cuando los hechos cumplen con las reglas, éstas se disparan cuando se ejecuta el motor de reglas. Un hecho se agrega al sistema ejecutando la siguiente instrucción en el motor de reglas:

```
(assert
        (questionTemplate
                (numQuestion 1.1.1)
                (answer a)
         )
)
```

dónde:

assert	añade hechos a la memoria activa
questionTemplate	plantilla para encapsular el hecho.

Funciones de usuario

Son aquellas funciones que al implantarlas permiten extender la funcionalidad de *Jess*. Dichas funciones se denominan así porque son creadas por el usuario y/o desarrollador. Para la implantación de DIACNE se definieron tres funciones de usuario:

(answer-question)
(fire-question)
(show-error-message)

dónde:

answer-question	se ejecuta cuando el usuario contesta una pregunta.
fire-question	se ejecuta cuando se cumple una regla e indica la siguiente pregunta.
Show-error-message	se ejecuta cuando se detecta una inconsistencia en las respuestas dadas por el usuario.

6. Construcción de la red de Petri de DIACNE mediante la ejecución del sistema basado en reglas.

La construcción inicia a partir de una secuencia básica que representa el perfil del usuario y DCW va generando en tiempo real una red WF conforme el usuario contesta la aplicación. Los patrones básicos que se utilizan en la modelación de DIACNE son la secuencia, selección y paralelismo, debido a que en un cuestionario se realizan preguntas en forma secuencial cuando son dependientes (DCW1), de selección cuando hay una salto de las mismas o se realizan al mismo tiempo cuando son independientes (DCW4).

La parte del sistema experto consiste de un API de tres métodos entre el servidor y *Jess* al momento que el usuario ingresa:

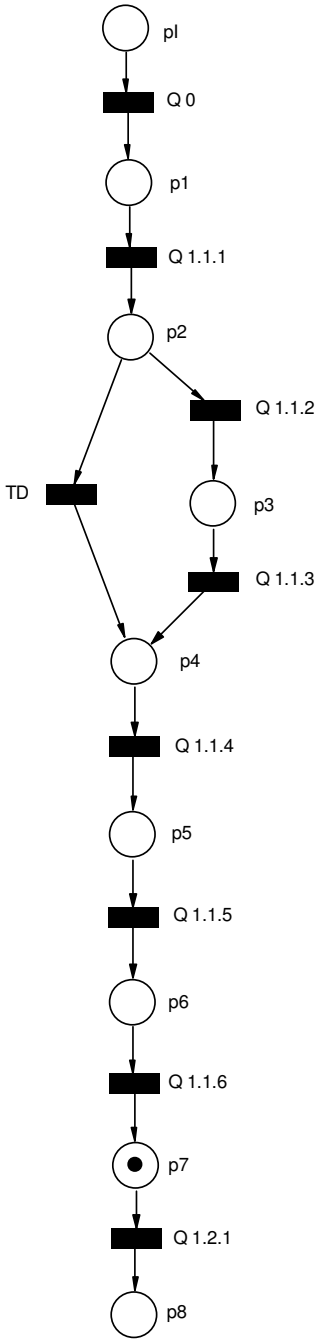
- El primer método lee el archivo en XML de las reglas y las transforma en reglas en Jess, una réplica por cada usuario que ingresa. Posteriormente se ejecutan los siguientes pasos en un ciclo:

1. Al pedir la siguiente pregunta se llama a una función *fire-question* que dispara las reglas de *Jess* y genera la siguiente pregunta.
2. Por cada respuesta de usuario se agrega un hecho en su memoria activa (*answer-question*). Cada usuario tiene su propia definición de reglas y de hechos.

En el momento de la ejecución el paso uno dispara las reglas y el paso dos va generando un hecho por cada respuesta. Si un usuario que ingresa al sistema no termina el cuestionario, se almacenan sus hechos de tal manera que cuando vuelve a ingresar al sistema, se agregan todos los hechos por cada respuesta que había contestado la primera vez.

La figura 5.5 muestra la ejecución del algoritmo de DCW mostrado en la figura 4.18. La figura muestra el inicio y ejecución de las primeras seis preguntas del cuestionario, pertenecientes al subfactor *Liderazgo* del factor *Actitud* ver (Espinosa et al. 2004b) para una consulta del cuestionario completo. La figura 5.5a) muestra la red de Petri generada, la figura 5.5b) muestra su correspondiente notación en BNF y la figura 5.5c) muestra el texto de cada pregunta y sus opciones de respuesta. La red de Petri contiene un patrón básico de selección con base a la respuesta de la pregunta Q 1.1.1. La primera opción de respuesta envía el token a la secuencia de preguntas Q 1.1.2, Q 1.1.3 y Q 1.1.4, en el otro caso se envía directamente a la pregunta Q 1.1.4 mediante una transición de sincronización TD. La respuesta a cada pregunta implica generar la siguiente pregunta, la función *computeFollowSet()* del algoritmo de la figura 4.18 consiste en la ejecución del sistema experto y disparar la siguiente pregunta con respecto a las respuestas consideradas como hechos. DCW garantiza la ejecución correcta del cuestionario. La ejecución construyó la red WF y la red WF BNF de la figura 5.5.

Las reglas en XML para la red de Petri se muestran en la figura 5.6, en la primera columna se muestran las reglas para las preguntas Q 1.1.1 a Q 1.1.3 y en la segunda columna se muestra la continuación hasta la pregunta Q 1.1.6. Las reglas mostradas son parte del archivo que contiene todas las reglas en XML para las 105 preguntas del cuestionario ver (Kim et al. 2007) para un listado completo.



a)

WF → pI Q0 p1 Q111 p2 [Q112 p3 Q113 ↕ TD] p4 Q114 p5 Q115 p6 Q116 p7 Q121 p8

b)

Q0 → Inicio

Q 1.1.1 ¿Se realizan planes en la empresa?

a) Si

b) No

Q 1.1.2 ¿Quiénes participan en la elaboración de los planes en la empresa?

a) a

Q 1.1.3 ¿Se comunican los planes a todos lo empleados de la empresa?

a) Sí

b) No

c) Solo Directores

d) Directores y mandos intermedios

Q 1.1.4. ¿La dirección participa en la propuesta, definición y seguimiento de proyectos tecnológicos en la empresa?

a) Siempre

b) A veces, dependiendo del proyecto

c) Nunca

Q 1.1.5. ¿Se comunican a todos los empleados los resultados operativos de la empresa?

a) Frecuentemente

b) De vez en cuando

c) Rara vez

d) Nunca

Q 1.1.6. ¿En que porcentaje el personal conoce las responsabilidades y funciones de cada departamento de la empresa?

a) No las conoce

b) Menos del 30 %

c) 50 %

d) Entre el 50 % y el 75 %

e) Más del 75 %

Q 1.2.1. ¿Se tienen las descripciones de puesto con las funciones y responsabilidades actuales para cada miembro de la organización?

a) No

b) Menos del 30 %

c) 50 %

d) Entre el 50 % y el 75 %

e) Más del 75 %

c)

Figura 5.5 Ejecución del cuestionario del subfactor *Liderazgo* del factor *Actitud.*

```
<rules>
  <rule name="question_1.1.1-a">
    <and>
      <condition>
        <question number="0.0.0" answer="a" />
      </condition>
      <condition>
        <question number="1.1.1" answer="a" />
      </condition>
    </and>
    <function>(fire-question 1.1.2)</function>
  </rule>
  <rule name=" question _1.1.1-b">
      <or>
        <condition>
          <question number="0.0.0" answer="a" />
        </condition>
        <condition>
          <question number="1.1.1" answer="b" />
        </condition>
      </or>
    <function>(fire-question 1.1.4)</function>
  </rule>
  <rule name=" question _1.1.2">
        <condition>
          <question number="1.1.2" answer="a" />
        </condition>
    <function>(fire-question 1.1.3)</function>
  </rule>
  <rule name=" question _1.1.3">
      <or>
        <condition>
          <question number="1.1.3" answer="a" />
        </condition>
        <condition>
          <question number="1.1.3" answer="b" />
        </condition>
        <condition>
          <question number="1.1.3" answer="c" />
        </condition>
        <condition>
          <question number="1.1.3" answer="d" />
        </condition>
      </or>
    <function>(fire-question 1.1.4)</function>
  </rule>
```

```
  <rule name=" question _1.1.4">
      <or>
        <condition>
          <question number="1.1.4" answer="a" />
        </condition>
        <condition>
          <question number="1.1.4" answer="b" />
        </condition>
        <condition>
          <question number="1.1.4" answer="c" />
        </condition>
      </or>
    <function>(fire-question 1.1.5)</function>
  </rule>
  <rule name=" question _1.1.5">
      <or>
        <condition>
          <question number="1.1.5" answer="a" />
        </condition>
        <condition>
          <question number="1.1.5" answer="b" />
        </condition>
        <condition>
          <question number="1.1.5" answer="c" />
        </condition>
        <condition>
          <question number="1.1.5" answer="d" />
        </condition>
      </or>
    <function>(fire-question 1.1.6)</function>
  </rule>
  <rule name=" question _1.1.6">
      <or>
        <condition>
          <question number="1.1.6" answer="a" />
        </condition>
        <condition>
          <question number="1.1.6" answer="b" />
        </condition>
        <condition>
          <question number="1.1.6" answer="c" />
        </condition>
        <condition>
          <question number="1.1.6" answer="d" />
        </condition>
        <condition>
          <question number="1.1.6" answer="d" />
        </condition>
      </or>
    <function>(fire-question 1.2.1)</function>
  </rule>
…
</rules>
```

Figura 5.6 Reglas en XML del cuestionario para el subfactor liderazgo.

Las reglas en *Jess* para la red mostrada en la figura 5.5 se muestran en la figura 5.7. La pregunta 0.0.0 se incluyo en XML y *Jess* como hecho inicial y es una condición que facilita el manejo de condiciones para la regla inicial, para las demás reglas existen condiciones que deben ser cumplidas excepto para esta regla inicial, ver (Kim et al. 2007) para un listado completo.

```
(defrule question 1.1.1-a ?answer <-
  (and
  (questionTemplate
  (numQuestion 0.0.0)(answer a))
  (questionTemplate
  (numQuestion 1.1.1)(answer a))) =>
  (fire-question 1.1.2))

(defrule question 1.1.1-b ?answer <-
  (or
  (questionTemplate
  (numQuestion 0.0.0)(answer a))
  (questionTemplate
  (numQuestion 1.1.1)(answer b))) =>
  (fire-question 1.1.4))

(defrule question 1.1.2 ?answer <-
  (questionTemplate
  (numQuestion 1.1.2)(answer a)) =>
  (fire-question 1.1.3))

(defrule question 1.1.3 ?answer <-
  (or
  (questionTemplate
  (numQuestion 1.1.3)(answer a))
  (questionTemplate
  (numQuestion 1.1.3)(answer b))
  (questionTemplate
  (numQuestion 1.1.3)(answer c))
  (questionTemplate
  (numQuestion 1.1.3)(answer d))) =>
  (fire-question 1.1.4))
```

```
(defrule question 1.1.4 ?answer <-
  (or
  (questionTemplate
  (numQuestion 1.1.4)(answer a))
  (questionTemplate
  (numQuestion 1.1.4)(answer b))
  (questionTemplate
  (numQuestion 1.1.4)(answer c))) =>
  (fire-question 1.1.5))

(defrule question 1.1.5 ?answer <-
  (or
  (questionTemplate
  (numQuestion 1.1.5)(answer a))
  (questionTemplate
  (numQuestion 1.1.5)(answer b))
  (questionTemplate
  (numQuestion 1.1.5)(answer c))
  (questionTemplate
  (numQuestion 1.1.5)(answer d))) =>
  (fire-question 1.1.6))

(defrule question 1.1.6 ?answer <-
  (or
  (questionTemplate
  (numQuestion 1.1.6)(answer a))
  (questionTemplate
  (numQuestion 1.1.6)(answer b))
  (questionTemplate
  (numQuestion 1.1.6)(answer c))
  (questionTemplate
  (numQuestion 1.1.6)(answer d))
  (questionTemplate
  (numQuestion 1.1.6)(answer e))) =>
  (fire-question 1.2.1))
```

Figura 5.7 Reglas en *Jess* del cuestionario para el subfactor liderazgo.

Los hechos se generan al momento de la ejecución por lo mismo no se muestran en una figura, pero la figura 5.7 muestra las reglas que disparan los hechos que se generaron.

5.6 Conclusiones del capítulo

La aplicación usada para la implantación se seleccionó para resolver dos problemas al mismo tiempo: primero migrar el sistema DIACNE a un sistema experto de modo de flexibilizar el sistema permitiendo la modificación de su cuestionario sin tantos problemas y segundo, la implantación de DCW.

DCW puede construir WF correctos de cualquier proceso que soporte los patrones de construcción básicos. La administración de todo proceso requiere del conocimiento de negocio del mismo, por lo mismo, DCW añade un motor de reglas para que la construcción y administración del WF sea completa en el contexto del proceso.

Por último, dependiendo del contexto del proceso DCW puede representar diferentes procesos transformando el significado de cada transición:

- La transición como actividad
    - Modelación y construcción de procesos de negocio
    - Composición de redes WF
- La transición como servicio Web
    - Composición WF de servicios Web.
- La transición como pregunta
    - Cuestionarios en línea

El conjunto de aplicaciones que puede modelar tiene como alcance cualquier aplicación que puede representarse de manera formal con las construcciones básicas presentadas.

Capítulo 6 Conclusiones

El objetivo principal de todo sistema de administración WF es crear una infraestructura estratégica para la administración de información y soporte de sus procesos de negocio. En la actualidad, la única constante es el cambio en el ambiente y por lo mismo es de capital importancia el uso de WF dinámicos que puedan seguir cumpliendo con dicho objetivo.

La modelación WF actual tiene como objetivo tener un control automático de todo proceso de negocio sobre todo en ambientes dinámicos. Por lo mismo se requieren herramientas que complementen al trabajo que existe actualmente y sean parte hacia solución general del problema del cambio dinámico.

La construcción dinámica de WF es la motivación principal del presente trabajo. Por lo mismo, no es comparable en el rendimiento de su ejecución con un WF ya construido. DCW es un modelo que consiste en cinco patrones básicos de construcción, un proceso general de construcción dinámica y una gramática definida con acciones semánticas para representar dinámicamente redes WF y redes WF BNF. El diseño y la aplicación de DCW generan las siguientes aportaciones de capital importancia en el área de procesos y administración WF:

1. Un modelo para la construcción dinámica de WF que representa WF con un lenguaje de modelación visual en redes de Petri y un lenguaje de modelación textual en BNF. Los dos formalismos permiten el análisis matemático de los WF de manera de verificar que estén correctos en todo momento.

2. La representación visual permite el monitoreo gráfico del comportamiento de los WF. La representación textual permite detectar el estado de una transición, por ejemplo activa o en ejecución, esto no se puede con redes de Petri, añadiendo valor agregado a la modelación.

3. El proceso general de construcción dinámica de WF puede crear WF a partir de una secuencia básica mediante la inserción de patrones. También puede unir correctamente redes WF completas creando sistemas que representen procesos intra e inter organizacionales.

4. La adaptabilidad de las instancias en las redes WF al momento de la construcción es de capital importancia ya que permite que los casos se administren correctamente.

5. La construcción dinámica permite crear WF correctos que hacen flexible la composición de procesos y servicios Web, con la aportación del aspecto sintáctico, la unión de los WF y el aspecto semántico, la actualización del estado.

6. El modelo puede ayudar a modelar negocios orientados a servicios mediante una correspondencia correcta entre actividad y servicio. La construcción dinámica presenta ventajas de flexibilidad y robustez en la construcción de redes permitiendo ser usada en plataformas SOA, en este caso, la función *computeFollowSet()* para la búsqueda del siguiente patrón cumpliría la función de búsqueda de servicios Web.

7. La composición de redes WF complementa la construcción dinámica permitiendo la representación de cualquier patrón en paralelo de *n* ramas. Además, la combinación no representa ningún problema para el modelo creando WF dinámicos correctos tanto en su estructura como en su estado.

8. La gramática definida en BNF con acciones semánticas para la construcción de WF dinámicos, es innovadora así como la correspondencia entre BNF y redes de Petri, lo que permite tener una herramienta de modelación robusta utilizada para definir lenguajes de programación, y además, ser la base para la creación de una herramienta que analice y verifique cualquier red WF tal como los hacen los compiladores con los lenguajes de programación. En la actualidad sólo existe una

herramienta para la verificación de WF ya construidos (Aalst et al. 1997b) denominada *WOFLAN* y es un analizador de Worflows basado en redes de Petri.

9. La representación de redes WF mediante BNF y acciones semánticas permite que los patrones de construcción sean genéricos y puedan accederse las plazas/transiciones gracias a su representación de símbolos no-terminales y a las acciones semánticas.

10. La gramática textual definida en BNF es tan robusta que permite representar la construcción dinámica, la composición de redes WF y la combinación de ambos para construir dinámicamente WF dinámicos.

DCW representa el conocimiento de negocios de una empresa gracias a que integra sus formalismos de modelación de procesos con un motor de reglas. Los formalismos modelan cualquier proceso que incluya patrones de construcción tales como secuencia, selección y paralelismo. El motor de reglas permite añadir el conocimiento de negocios de la empresa mediante la inserción de sus reglas de negocio. Por lo mismo, se considera que DCW es una herramienta completa para representar y construir en forma dinámica WF que representen los procesos de negocio de una empresa.

Con respecto al diseño del sistema experto basado en reglas, se requirió migrar una tabla relacional a un entorno para el diseño de sistemas expertos. La mejora del sistema se dio en dos aspectos fundamentales:

- La definición en XML de la tabla de tal manera que se pueda modificar el cuestionario ya sea eliminando y/o añadiendo preguntas sin mayor problema.
- El uso del motor de inferencias del sistema experto que permite rapidez y eficiencia a la hora de seleccionar la siguiente pregunta del cuestionario.

Se usó el entorno de Jess para el desarrollo del sistema experto debido principalmente a las siguientes ventajas:

- La familiaridad y la solidez del lenguaje Java con el que esta desarrollado
- El desarrollo de aplicaciones en Java que pueden extender el entorno y la funcionalidad de Jess.
- Todas las aplicaciones en Java pueden ejecutarse en Jess sin necesidad de compilarse.
- Jess puede manejar incertidumbre, una línea de investigación futura de DCW.

En resumen, el presente trabajo plantea un modelo novedoso y formal que cumple con los objetivos planteados en el capítulo 1 y con las aportaciones mencionadas en el capítulo 3.

Líneas de investigación y trabajo futuro

Los resultados de esta investigación generaron un modelo formal que sienta las bases para la realización de las siguientes líneas de investigación futuras:

1. Añadir al modelo más patrones de construcción, con el objeto de generalizarlo, para representar WF más complejos, dos ejemplos serían la incorporación del patrón OR y el patrón de repetición. Esto es un área de investigación fuerte ya que el patrón OR puede hacer que un WF no tenga la propiedad *sound*. En el caso de la repetición, podría ser necesario añadir más producciones a la gramática en BNF para su representación. En ambos casos tiene que comprobarse totalmente la correctes de los WF construidos.

2. La incorporación de patrones implica que tengan que ser definidos en el sistema experto.

3. Representar procesos y conocimiento de negocios con mayor complejidad y que requieran el manejo de incertidumbre en el sistema experto. Esto generaría un

modelo mucho más completo y sólo requiere cambios en la parte del diseño del sistema experto, el proceso general de construcción dinámica no cambia.

4. Con la gramática definida en BNF para representar WF es posible crear una herramienta automática de verificación de Workflows. Dicha herramienta extendería el proceso de construcción permitiendo verificar la correctes de WF construidos por otras herramientas y que se necesiten utilizar en la construcción de nuevos WF.

5. En el caso de la modelación de DIACNE, por las características del cuestionario que requieren que cada usuario tenga una instancia se construye un WF por usuario. La generación de los WF tiende a uno que cumpla con el modelo de muchos usuarios lo cuál podrá caracterizar en forma estadística el perfil mayoritario de las empresas que existen en México. En este caso la modelación de un WF por usuario genera una ventaja. El problema es que no todos los procesos pueden tener instancias por caso ya que puede generar un gasto de memoria y administración sin precedentes. Este problema esta considerado como trabajo futuro y una vía de solución en la que se está trabajando son las redes de Petri coloreadas combinadas con estándares de BPM, lo que permite conservar el proceso general de construcción.

Bibliografía

W.M.P. van der Aalst (2004). Business Process Management Demystified: A tutorial on Models, Systems and Standards for Workflow Management. In J. Desel, W. Reisig, and G. Rozenberg, editors, Lectures on Concurrency and Petri Nets, volume 3098 of Lecture Notes in Computer Science, pages 1-65. Springer-Verlag, Berlin.

W.M.P. van der Aalst (2003a). Don't go with the flow: Web services composition standards exposed. IEEE Intelligent Systems, 18(1):72-76.

W.M.P. van der Aalst (2003b). Inheritance of Business Processes: A Journey Visiting Tour Notorious Problems. In H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber, editors, Petri Net Technology for Communication Based Systems, volume 2472 of Lecture Notes in Computer Science, pages 383-408. Springer-Verlag, Berlin, 2003.

W.M.P. van der Aalst, Weske, Mathias and Wirtz, Guido (2003a). Advanced Topics in Workflow Management: Issues, Requirements and Solutions. 2003 Society for Design and Process Science.

W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede (2003b). Web Service Composition Languages: Old Wine in New Bottles? . In G. Chroust and C. Hofer, editors, Proceeding of the 29th EUROMICRO Conference: New Waves in System Architecture, pages 298-305. IEEE Computer Society, Los Alamitos, CA.

W.M.P. van der Aalst, Hofstede, A.H.M. ter, Kiepuszewski, B. and Barros, A.P (2003c). Workflow Patterns. Distributed and Parallel Databases, 14(1):5-51.

W.M.P. van der Aalst and Kees van Hee (2002a). Workflow Management Models, Methods and Systems. MIT. 2002.

W.M.P. van der Aalst and T. Basten (2002b). Inheritance of Workflows: An Approach to Tackling Problems Related to Change. Theoretical Computer Science, 270(1-2):125-203.

W.M.P. van der Aalst (2000). Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques. In Business Process Management: Models, Techniques, and Empirical Studies, volume 1806 of Lecture Notes in Computer Science, pages 161-183. Springer-Verlag, Berlin, 2000

W.M.P. van der Aalst, Basten, T., Verbeek, H.M.W., Verkoulen, P.A.C. and Voorhoeve, M. (1999). Adaptive Workflow. On the interplay between flexibility and support. In J. 3 and J. Cordeiro, editors, Proceedings of the first International Conference on Enterprise Information Systems, volume 2, pages 353-360, Setúbal, Portugal, March 1999.

W.M.P. van der Aalst (1997a). Verification of Workflow Nets. In P. Azéma and G. Balbo, editors, Application and Theory of Petri Nets 1997, volume 1248 of Lecture Notes in Computer Science, pages 407-426. Springer-Verlag, Berlin.

W.M.P. van der Aalst, Hauschildt D. and Verbeek H.M.W. (1997b). WOFLAN: a Petri-net-based Workflow Analyzer. Computing Science Reports 97/12, Eindhoven University of Technology, Eindhoven, 1997.

W.M.P. va der Aalst-URL (2006) http://is.tm.tue.nl/research/patterns/standards.htm.

Aho A.V., Sethi R. and Ullman (1998). Compiladores. Principios, Técnicas y Herramientas. Pearson

.

Agostini, A. and DeMichelis G. (2000a) Improving Flexibility of Workflow Management Systems. BPM'00, LNCS, vol. 1806, 2000.

Agostini, A. and DeMichelis G. (2000b) A Light Workflow Management System Using Simple Process Models. Int. Journal of Collaborative Computing. 16.

Albert P., Henocque L. and Kleiner M. (2005). Configuration Based Workflow Composition. Internationa Conference on Web Services IEEE.

BEA 2006.
http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/weblogic/

Buhler P. and Vidal J.M. (2005). Toward Afaptive Workflow Enactment Using Multiagent Systems. Information Technology and Management 6, 61-87, 2005.

Buhler, P.; Vidal, J. M.; and Verhagen, H. (2003). Adaptive Workflow = web services + agents. In Proceedings of the International Conference on Web Services, 131--137. CSREA Press.

Cao, Jian, Zhang, Shensheng, Li, Minglu, Wang, Jie. (2004). Verification of Dynamic Process Model Change to Support the Adaptive Workflow. IEEE International Conference on Services Computing(SCC'04).

Cardoso J. and Sheth A. (2002). Semantic w-Workflow Composition. Technical Report. 2002.

Casati Fabio, Castano Silvana, Fugini Mariagrazia, Mirbel Isabelle and Pernici Barbara. (2000). Using Patterns to Design Rules in Workflows. IEEE Transactions On software Engineering, Vol. 26, No. 8, Agosto 2000.

Casati F., Ceri S., Pernici B. and Pozzi G. Workflow Evolution. Proc. of the 15th ER'96 international conference. Lecture Notes in Computer Science, 1996.

Chisholm Malcolm (2004). How to Build a Business Rules Engine. Morgan Kaufmann Publishers. Elsevier.

De Carolis, Bernardina, De Rosis, Fiorella. (1994). Modelling Adaptive Interaction of OPADE by Petri Nets. SIGCHI Bulleting, Vol 26, No. 2.

Debevoise T. (2005). Business Process Management with a Business Rules Approach. Business Knowledge Architects.

Dehnert, Juliane, Aalst, W.M.P. van der. (2004). Bridging the Gap Between Business Models and Workflow Specifications. International Journal of Cooperative Information Systems. World Scientific Publishing Company.

Dempster A.P. (1967). Upper and Lower Probabilities Induced by Multivalued Mappings. Annals of Math Stat. 28 pp 325-329. 1967.

Deng, ShuiGuang, Yu, Zhen, Wu, ZhaoHui, Huang, LiCan. (2004). Enhancement of workflow flexibility by composing activities at run-time. ACM Symposium on Applied Computing.

Doshi P., Goodwin R. and Akkiraju R. (2004). Dynamic Workflow Composition using Markov Decision Processes. International Conference on Web Services IEEE.

Ellis C. and Maltzahn C. (1997). The Chautauqua Workflow System. Proceedings of the 30th Hawaii International Conference on System Sciences: Information Systems Track—Internet and the Digital Economy - Volume 4 Page 427.

Ellis, C. and Rozenberg, G. (1995). Dynamic Change within Workflow Systems. COOCS 95. Milpitas CA USA. ACM.

Erl Thomas (2005). Service-Oriented Architecture. Concepts, Technology, and Design. Prentice-Hall.

Eshuis, Rik, Wieringa, Roel. (2002). Verification support for workflow design with UML activity graphs. ICSE.

Espinosa E., Molina M., Pacheco E., Cervantes M. and Rubio E. (2005a). Training Entrepeneurs During Corporate Assesment. Encuentro Nacional de Ciencias de la Computación 2005 (ENC2005). September 2005. Puebla, México.

Espinosa E., Molina M., Pacheco E., Cervantes M., Rubio E. y Cadena F. (2005b). Asesor de Conceptos de Negocios Electrónicos. IEEE 3º Congreso Internacional en Innovación y Desarrollo. Tecnológico de Monterrey, Campus Cuernavaca. Septiembre 2005.

Espinosa E.D., Junco A., Ramirez J. and Ramos F. (2004a). Companion Agents for Enabling B2B culture in SMB's. ITREE 2004

Espinosa E.D., Junco A., Ramirez J., Ramos F., Rubio E., Vazquez M. Cárdenas R. (2004b). Diagnosing Key Needs for Emergent B2B in SMB's. CEC2004.

Friedman-Hill E. (2003). Jess in Action. Manning Publications Co.

Giarratano J. and Riley G. (2001). Sistemas Expertos. Principios y Programación. Thomson.

Girault C. and Valk R. (2003). Petri Nets for Systems Engineering. Springer. 2003.

Glance N.S., Pagani D.S. and Pareschi R. (1996). Generalized Process Structure Grammars (GPSG) for Flexible Representations of Work. Computer Supported Cooperative Work '96. Cambridge MA USA.

Hamadi R. and Benatallah B. (2003). A Petri Net-Based Model for Web Service Composition. ADC2003. Australian Computer Society.

Havey M. (2005). Essential Business Process Modeling. O'Reilly Media Inc.

Janssen, Will, Jonkers, Henk, Verhoosel, Jack (1997). What makes business processes special? An evaluation framework form modeling languages and tools in Business Process Redesign.

Jensen, K, (1997): Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use, EATCS monographs on Theoretical Computer Science, Springer, Berlin.

Jorgensen H.D. (2001). Interaction as a Framework for Flexible Workflow Modeling. GROUP2001. Boulder, Colorado. USA.

Juan, Eric Y.T., Tsai, Jefrey J.P., Murata, Tadao. (1998). Compositional Verification of Concurrent Systems Using Petri-Net-Based Condensation Rules. ACM Transactions on Programming Languages and Systems, Vol. 20, No 5.

Kalakota R. and Robinson M. (2003). Services Blueprint: Roadmap to Execution. Addison-Wesley. Boston 2003.

Kalakota R. and Robinson (1999). e-Business: Roadmap for Success. Addison Wesley.

Kassam F. and Kontogiannis K. (2004). Quality and Constraint Driven Workflow Composition. STEP'04 Proceedings. IEEE.

Kiepuszewski B., Hofstede ter A.H.M. and Bussler C.J. (2000). On Structured Workflow Modelling. CAiSE'00. LNCS, vol. 1789, 2000, pp. 431-445.

Kim Alfonso, Molina Martín and Muñoz Jesús. Aplicación de un sistema experto para el modelado de procesos de negocio en PyMEs. Manual de Investigación. Departamento de computación, ITESM-CCM. Mayo 2007.

Kim J., Spraragen M. and Gil Y. (2004). An Intelligent Assistant for Interactive Workflow Composition. IUI'04. ACM.

Kindler, Ekkart, Aalst, Will van der (1999). Liveness, Fairness and Recurrence in Petri Nets. Information Processing Letters, Vol. 70, No. 6, pages 269-274. June 1999.

Kiritsis, D., Xirouchakis, P., Gunther, C. (1998) Petri net representation for the process specification language – Part 1: Manufacture Process Planning. Proceedings of the 1st International Workshop on Intelligent Manufacturing Systems IMS-EUROPE, EPFL, Lausanne; April.

Kradolfer Markus and Geppert Andreas (1999). Dynamic Workflow Schema Evolution Based on Workflow Type Versioning and Workflow Migration. TRAMS.

Krammer P.J. (2000). Techniques for Supporting Dynamic and Adaptive Workflow. Computer Supported Cooperative Work. Volumne 9, Issue 3-4.

Kwan M.M. and Balasubramanian P.R. (1997). Dynamic Workflow Management: A Framework for Modeling Workflows. Proceedings of the Thietieth Annual Hawwaii International Conference. IEEE.

Laguna Manuel and Marklund Johan (2004). Business Process Modeling, Simulation and Design. Pearson.

Laukkanen M. and Helin H. (2003). Composing Workflows of Semantic Web Services. WSABE.

Liu, Dongsheng, Wang, Jianmin, Chan, Stephen C.F., Sun Jiaguang, Zhang, Li. (2002). Modeling workflow processes with colored Petri nets. Elsevier Science B. V.

Mangan, Peter, Sadiq, Shazia. (2002). On Building Workflows Models for Flexible Processes. Australian Computer Society, Inc.

Marinescu, D.C. (2002). Internet-Based Workflow Management. Wiley-Intersciense.

Marks Eric and Bell Michael (2006). A Planning and Implementation Guide for Business and Technology. Wiley.

Mauch J.E. and Birch J. (1993). Guide to the Successful Thesis and Dissertation. A Handbook for Students and Faculty. Dekker.

Microsoft 2007. http://www.microsoft.com/biztalk/default.mspx.

Mnaouer, Adel Ben, Shekhar, Anand, Liang, Zhao Yi. (2004). A Generic Framework for Rapid Application Development of Mobile Web services with Dynamic Workflow Management. Proceedings of the 2004 IEEE International Conference on Services Computing.

Momotko M. and Subieta K. (2002). Dynamic Changes in Workflow Particpant Assigment. ADBIS 2002, pp 175-184.

Morel, Jean-Yves, Bourcerie, Marc. (2004). On the analysis and Synthesis of Colored Petri Nets. IEEE International Conference on Systems, Man and Cybernetics.

Muller, R. and Rahm E. (1999). Rule-Based Dynamic Modification of Workflows in a Medical Domain. Buchman (Editor). Proc. Of BTW99. Freiburg im Breisgau, 1-3. Springer, Berlin 1999.

Murata, Tadao. (1989). Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, Vol. 77, No. 4.

Narendra N.C. (2000). Adaptive Workflow Management – An Integrated Approach and System Architecture. SAC2000 March 19-21, Como, Italy.

OASIS 2007. http://www.oasis-open.org

OMG 2007. http://www.omg.org

Oracle 2007. http://www.oracle.com/technology/products/ias/bpel/index.html.

Pancerz, Krzysztof, Suraj, Zbigniew. (2003). Modelling Concurrent Systems Specified by Dynamic Information Systems. A Rough Set Approach. Elsevier Science B.V. Vol. 82.

Pankratius V. and Stucky W. (2005). A Formal Foundation for Workflow Composition, Workflow View Definition, and Workflow Normalization based on Petri Nets. APCCM 2005.

Poole John, Chang Dan, Tolbert Douglas and Mellor David (2003). Common Warehouse Metamodel. Developer's Guide. Wiley.

Reichert M. and Dadam P. (1998). ADEPTflex- Supporting Dynamic Changes of Workflows without Losing Control. JIIS 10 (2) 93-129.

Reichert Manfred, Rinderle Stefanie, Kreher Ulrich and Dadam Peter. (2005). Adaptative Process Management with ADEPT2. Proceedings of the 21st International Conference on Data Engineering(ICDE 2005).

Rinderle Stefanie, Reichert Manfred and Dadam Peter. (2004). Correctness Criteria for Dynamic Changes in Workflow Systems- a survey. Data & Knowledge Engineering 50, 9-34. Elsevier.

Ross Ronald G. (2003). Principles of the Business Rule Approach. Addison-Wesley Information Technology Series.

Sadiq Shazia W. (2000). Handling Dynamic Schema in Process Models. Database Conference, 2000. ADC 2000. Proceedings. 11th Australasian. 31 Jan.-3 Feb. 2000 Page(s):120 - 126

Sadiq Shazia W., Marjanovic Olivera and Orlowska M. E. (2000). Managing Change and Time in Dynamic Workflow Processes. International Journal of Cooperative Information Systems, Vol. 9, Nos. 1 & 2. World Scientific Publishing Company.

SAP 2007. http://www.sap.com

Secretaría de Economía 2006 (SE 2006). http://www.contactopyme.gob.mx.

Shafer G. (1976). A Mathematical Theory of Evidence. Princeton University Press. 1976.

Shafiq B., Samuel A. and Ghafoor H. (2005). A GTRBAC Based System for Dynamic Workflow Composition and Management. Proceedings of ISORC 2005. IEEE.

Skogan D., Gronmo R. and Soldheim I. (2004). Web Service Composition in UML. Proceedings of EDOC 2004. IEEE.

Sykara K., Paolucci M., Ankolekar A. and Srinivasan N. (2003). Automated Discovery, Interaction and Composition of Semantic Web Services, Journal of Web Semantics,. Volume 1, Issue 1.

Thomson and Strickland. (2004). Administración Estratégica. McGraw-Hill.

Tsai Hui-Liang (2003). Information Technology and Business Process Reengineering. New Perspectives and Strategies. Praeger. British Library.

Wang J., Rosca D., Tepfenhart W., and Milewski A. (2005). An Intuitive Formal Approach to Dynamic Workflow Modeling and Analysis. BPM 2005, LNCS 3649, pp. 137-152, 2005.

Weske, M. (2001). Formal Foundation and Conceptual Design of Dynamic Adaptations in a Workflow Management System. HICSS-34.

W3C 2007. World Wide Web Consortium. http://www.w3.org/

Workflow Management Coalition 2006. (WfMC-URL 2007). URL http://www.wfmc.org

Yamaguchi S., Mishima A., GE Qi-Wei and Tanaka Minoru. (2005). A Flexible and Efficient Workflow Change Type: Selective Shift. IEICE Transactions Fundamentals, Vol. E88-A. No. 6 June 2005.

Zadeh L.A. (1965). Fuzzy Sets. Information and Control 8 pp 338-353 June 1965.

Zhuge H. (2003). Component-Based workflow systems development. Decision Support Systems 35 (2003) 517-536. Elsevier Science B.V.

Anexo A. Análisis de los enfoques descritos en el estado del arte

En este anexo, se presenta un análisis detallado de los nueve enfoques estudiados para el estado del arte. El análisis consiste en tres partes: primero se da la descripción del enfoque, después se describe el criterio de correctes y por último se compara dicho criterio con respecto a las cinco modificaciones que pueden generar problemas en el cambio dinámico.

I. WF Nets

(Aalst and Basten 2002b)

1. Esquema conceptual basado en redes de Petri.
2. La idea central del enfoque es como sigue: una instancia I en el esquema S, representado por una red marcada, cumple (*compliant*) con el esquema modificado S' si S y S' están relacionadas bajo relaciones de herencia.
3. Un esquema S es una subclase de otro esquema S' si no podemos distinguir el comportamiento de S y S' cuando (S > S').
4. Proporcionan operaciones de cambio especiales que automáticamente preservan una de las cuatro relaciones de herencia presentadas entre el esquema original y el cambiado.
5. Esas operaciones de cambio comprenden la inserción y el borrado de estructuras de ciclos, secuencias, ramas paralelas y de selección.
6. Las relaciones de herencia se alcanzan combinando ocultamiento y bloqueo de tareas.
7. Sea *x* y *y* dos redes, dónde *x* es una subclase de *y*, con respecto a las nuevas tareas (en x pero no en y) existen dos mecanismos que pueden ser usados
    - El primero deshabilita la ejecución de cualquier nueva tarea y compara el comportamiento resultante de x con respecto a y. Dicho mecanismo guía a la siguiente noción de herencia:
        - *Si no es posible distinguir el comportamiento de x y y cuando sólo las tareas de x que están en y son ejecutadas entonces x es una subclase de y.*
        - *Esta definición implica bloquear las tareas nuevas en x y se denomina herencia de protocolo.*
    - El segundo permite la ejecución de las nuevas tareas pero considera sólo los efectos de las viejas.
        - *Si no es posible distinguir el comportamiento de x y y cuando cualquier tarea de x se ejecuta, pero cuando sólo los efectos de las tareas que están presentes en y son consideradas, entonces x es una subclase de y.*
        - *Esta noción de herencia implica ocultar las tareas (hacer las transiciones sin efecto) y se denomina herencia de proyección.*
8. Administra cuatro relaciones de herencia:
    - De protocolo
    - De proyección

- De protocolo/proyección
- De ciclo de vida bloqueando algunas y escondiendo otras.

La red de la figura 1(1) muestra una red original *N0* y su red derivada *N1*. La diferencia entre relación de herencia de protocolo y proyección, puede observarse en N1. Si *verificar* es bloqueada ocurre un interbloqueo entre actividades y no sería una subclase de N. Si *verificar* es ocultada no hay diferencia en los comportamientos de *N0* y *N1*.

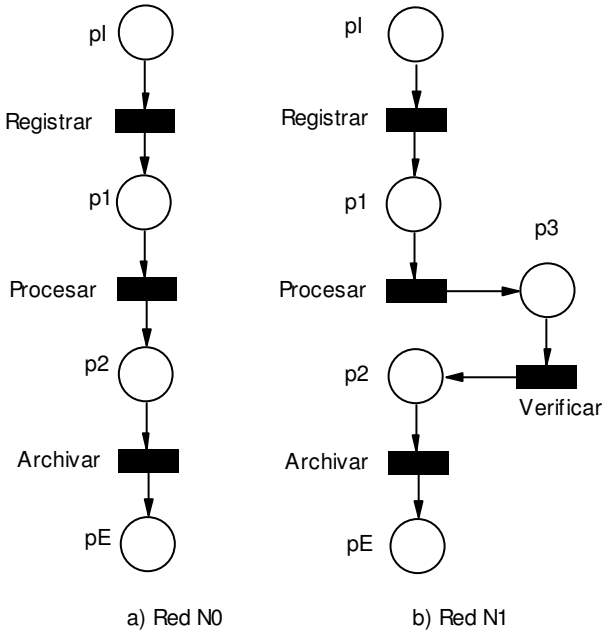


Figura 1(1). a) Red WF N0. b) Red WF N1.

Criterio de correctes

Sea S un esquema que es correctamente transformado en otro esquema S'. Entonces la instancia I en S cumple con S' = S + Δ si y sólo si S y S' están relacionadas bajo herencia: S es una subclase de S' o viceversa.

Para verificar si un cambio Δ es un cambio que preserva la herencia y de este modo S y S' están relacionadas bajo herencia los autores definen condiciones precisas con respecto a S y a S'. Utiliza una herramienta denominada *Woflan* para que automáticamente decida las reglas de herencia para dos esquemas dados. Las reglas de herencia restringen el conjunto de cambios aplicables a añadir o eliminar. Pospone la transferencia de casos cuando se encuentran en estados inválidos.

Utiliza reglas de transferencia para migrar los casos y asegurar la propiedad *sound*. Una regla de transferencia es aceptable y válida si y sólo si cada transferencia de un caso resulta en un estado en la nueva definición WF el cuál es también aceptable por los nuevos casos iniciándose, no debe generar interbloqueos u otras anomalías.

Utiliza cuatro reglas de transformación de preservación de herencia en dos direcciones para restringir los cambios en las definiciones de los WF de tal manera que la

nueva definición herede ciertas propiedades de la definición anterior. Las cuatro reglas de transformación combinadas con las 10 reglas de transferencia habilitan el cambio dinámico.

Modificaciones

Cambios en el pasado

Bajo el criterio 1 es posible cambiar regiones de WF pasadas. Todo cambio en el pasado cumple las relaciones de herencia. La figura 1(2) muestra un ejemplo de cambio puro en el pasado.

Ejemplo 1: insertar una actividad

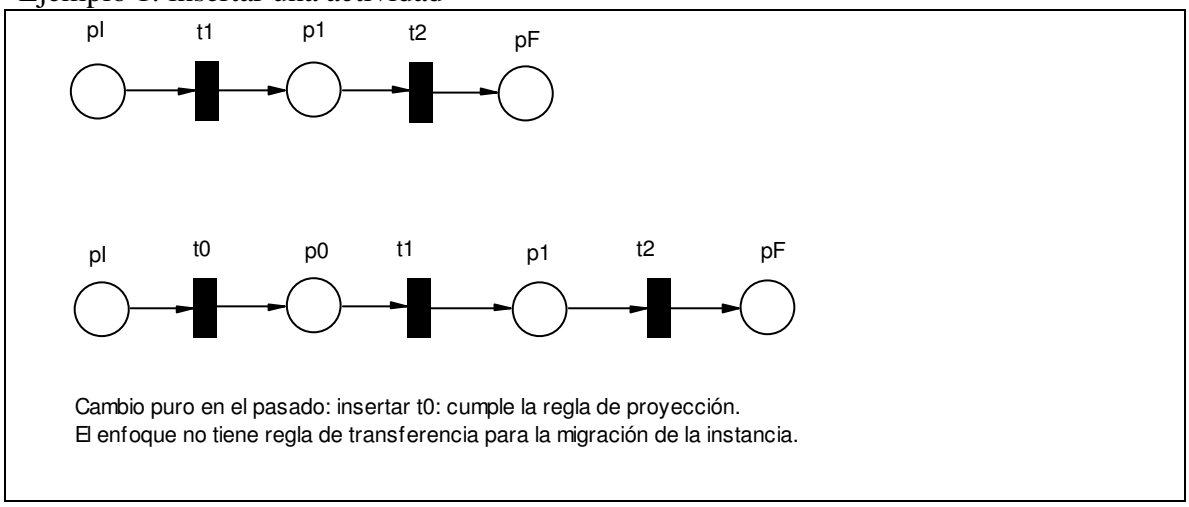


Figura 1(2). Cambio puro en el pasado.

Ejemplo 2: Permite cambios en el pasado con compartición de datos al insertar una actividad antes y después del caso. Se aplica la regla de transferencia, ver figura 1(3).

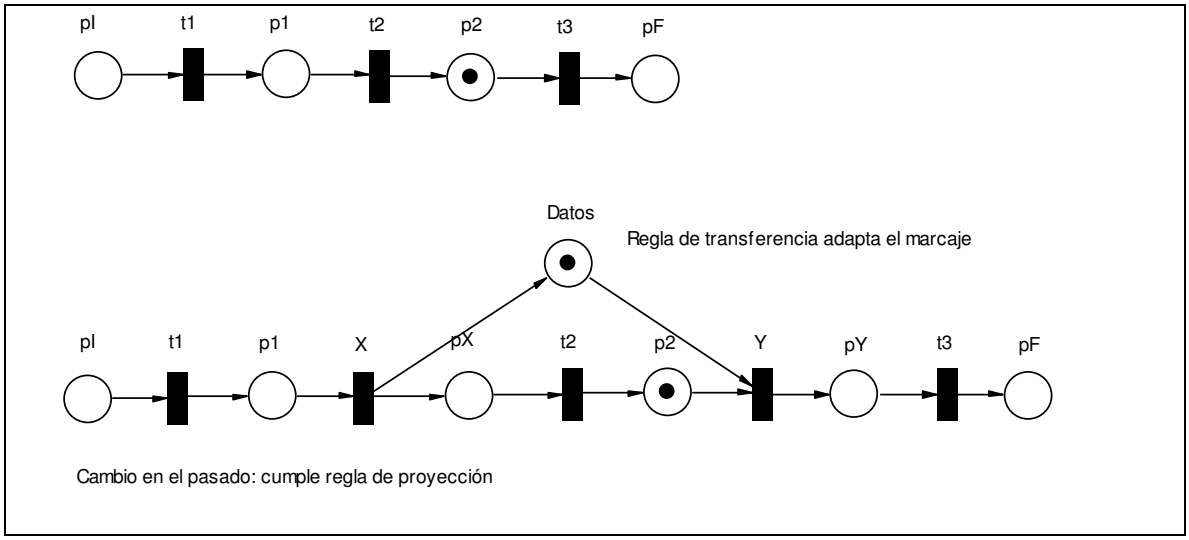


Figura 1(3). Cambio en el pasado con compartición de datos.

La figura muestra el cambio en el pasado con compartición de datos en el futuro. La red nueva cumple la regla de proyección y cumple con la regla de transferencia de proyección para actualizar la instancia en la nueva red.

Tolerancia a ciclos

Cumple con la herencia de proyección/protocolo si las actividades de la rama insertada para el ciclo son ocultadas (transiciones tX, tY sin efecto) o bloqueadas, ver figura 1(4).

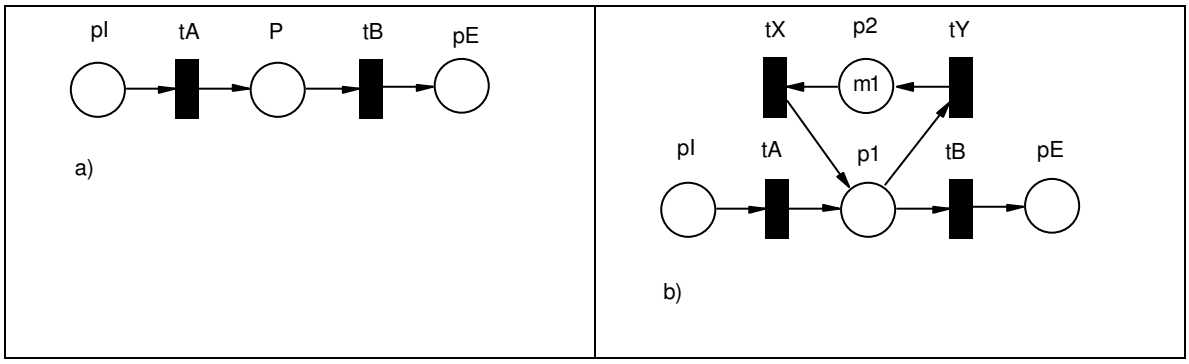


Figura 1(4). Tolerancia y cambio en ciclos.

Identificación de los estados de las actividades

Problema inherente al modelo, las redes de Petri no diferencian entre actividades activas y en ejecución.

Cambio de orden

Las operaciones de cambio de orden son explícitamente excluidas por este enfoque. El cambio de orden no cumple con las relaciones de herencia.

Inserción paralela

Permitido: cumple con la herencia de protocolo si la rama insertada es bloqueada. La figura 1(5) muestra que se deben insertar *tokens* para evitar bloqueos entre instancias, en este caso *pX2* debe tener un token para que la transición *t3* pueda activarse y posteriormente ejecutarse.

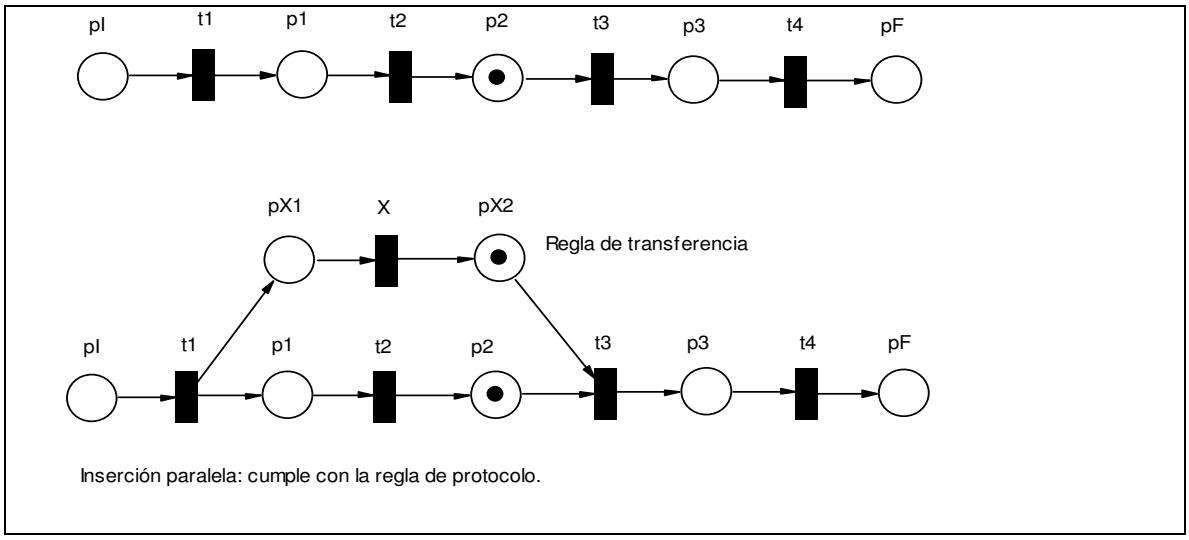


Figura 1(5). Inserción paralela.

II. Milano Nets

(Agostini and DeMichelis 2000a, 2000b)

1. Modelo desarrollado en un prototipo con redes de Petri clásicas.
2. Permite cambios locales y excepciones en la red, mediante la creación de trayectorias denominadas saltos.
3. Utiliza una especificación crítica mínima para cambios e influye en su criterio de correctes de estados seguros.

Criterio de correctes

Sea S un esquema y la I una instancia en S. Sea Δ un cambio que transforma S en otro esquema correcto S'. Entonces: I cumple con S' si I no esta en un estado inseguro en S con respecto a S'. Un estado de S es inseguro con respecto a S' si no esta presente en S'.

En términos prácticos, una instancia en ejecución esta en un estado seguro si hay una imagen determinada de su estado dentro del nuevo modelo. Moviendo esa instancia al nuevo esquema el WF terminará correctamente. Se considera incorrecto que algunas actividades se reejecuten. Permite tres tipos de cambio dinámico: paralelización, secuencialización e intercambio. La figura 2(1) muestra los tres tipos.

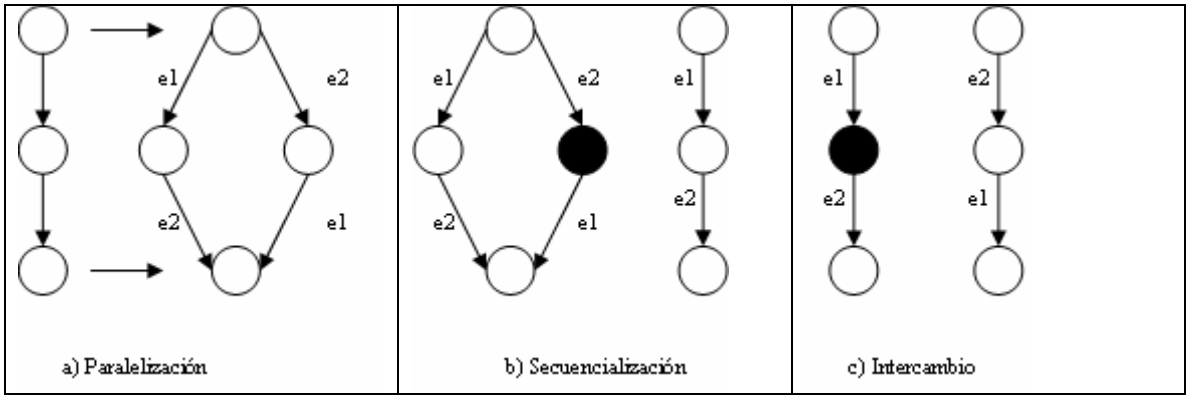


Figura 2(1). Los tres tipos de cambio de Milano.

a) Paralelización: hacer dos acciones secuenciales concurrentes
b) Secuencialización: crear una secuencia con dos acciones concurrentes.
c) Intercambio: invertir el orden de dos acciones secuenciales.

Los estados sombreados representan estados inseguros. Ejemplos de los tres tipos de cambio en redes de Petri se muestran en la figura 2(2). Los *tokens* en las figuras b) y c) indican estados inseguros y significa que el cambio no puede realizarse cuando se encuentren en esa posición.

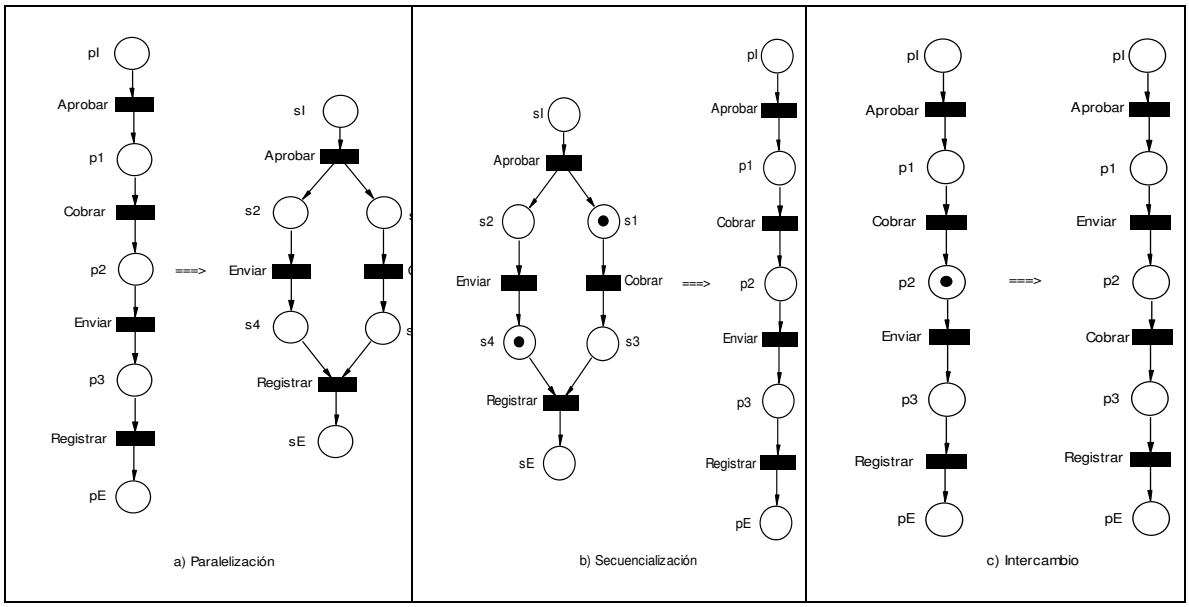


Figura 2(2). Ejemplo de los tres tipos de cambio de Milano.

Cambios en el pasado.

No permite la inserción de actividades en el pasado. Un ejemplo de cambio en el pasado sería que dos actividades se intercambien como lo muestra la figura 2(3).

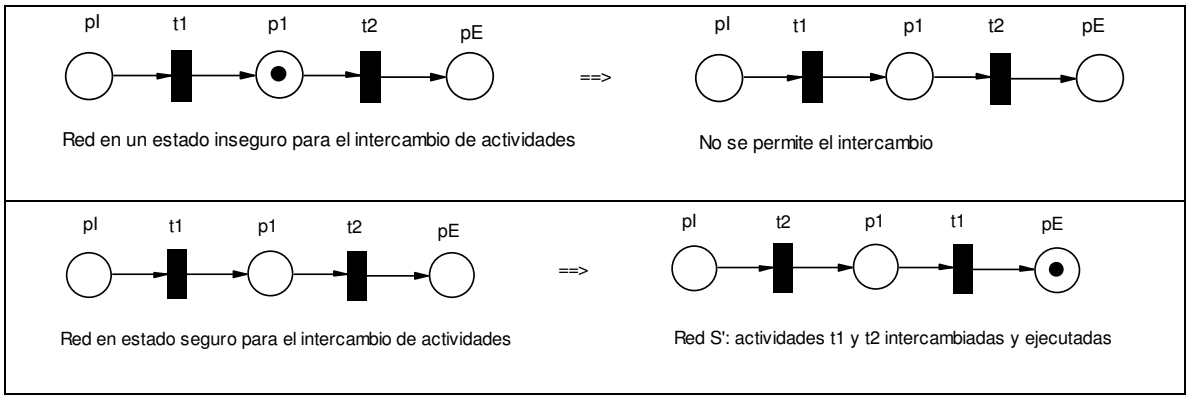


Figura 2(3). Cambio en el pasado con intercambio de actividades.

Tolerancia a ciclos

Puede simular ciclos si se contemplan en las excepciones y son simulados con saltos. No permite explícitamente ciclos utiliza redes acíclicas.

Identificación de los estados de las actividades

Problema inherente al modelo, las redes de Petri no diferencian entre actividades activas y en ejecución.

Cambio de orden

Proporciona operaciones de cambio de orden tal como lo muestra la figura 2(1): paralelización, secuencialización e intercambio de actividades. No proporciona cómo actualiza las instancias después del cambio pero da ejemplos de estados inseguros.

Inserción paralela

No proporciona operaciones de inserción, ver figura 2(1).

III. Flow Nets

(Ellis and Rozenberg 1995)

1. El modelo se implantó en un sistema denominado Chautaqua (Ellis and Maltzahn 1997), pero no se dan los detalles de cómo lo implantaron.
2. Primer trabajo publicado que definió el cambio dinámico.
3. Utiliza redes de Petri de alto nivel (más de un token en las plazas).
4. Presenta dos tipos de operaciones de cambio y una clase cambio especial
    a. La primera es abortar.
    b. La segunda es *flush*: las instancias actuales en el viejo esquema, las nuevas en el nuevo
    c. La clase especial la denominan *Synthetic Cut-Over Change* (SCOC): no hay reemplazo de la región N1 por la N2. SCOC mantiene la región N1 junto con N2 en el nuevo esquema S'.
5. Administra dos escenarios de cambio *Upsizing* y *Downsizing*
    a. *Upsizing* N2 puede hacer más que N1, N2 > N1. (serie a paralelo)
    b. *Downsizing* N2 hace menos que N1, N2 < N1. (paralelo serie) manteniendo la correctes.
    c. La identificación de las regiones es manual.
6. Define una secuencia de ejecución hasta antes del cambio para su criterio de correctes.

Secuencia antes del cambio (pre-change)

- Sea w la secuencia de todas las actividades que se ejecutaron antes del cambio.
- FSS de un esquema se define como: sea m y m' dos marcajes en S, entonces FSS(S, m, m') es el conjunto de todas las secuencias de disparo que guían de m a m' en S.
- $w \in FSS$

Criterio de correctes (Secuencia de disparo antes del cambio)

Sea I una instancia de S con marcaje m y sea $w \in FSS(S, MpI, m)$. Sea Δ un cambio que transforma S en S' y sea m' el marcaje resultante de I en S'. Entonces I cumple con S' si y sólo si

1. $FSS(S, m, MpE) \neq 0 \Rightarrow FSS(S', m', MpE) \neq 0$.
2. $\forall w' \in FSS(S', m', MpE) \Rightarrow w' \in FSS(S, m, MpE) \vee ww' \in FSS(S', MpI, MpE)$

MpI es el marcaje inicial con m(pI) = 1.
MpE es el marcaje final con m(pE) = 1.

La primera implicación de 2 significa:

- Permite cambios puros en el pasado. El marcaje w' de m' a MpE debe existir en el esquema anterior y debe ser igual que de m a MpE.

La segunda implicación de 2 significa:

- Prohibe cambios que afecten ambas regiones, pasado y futuro.

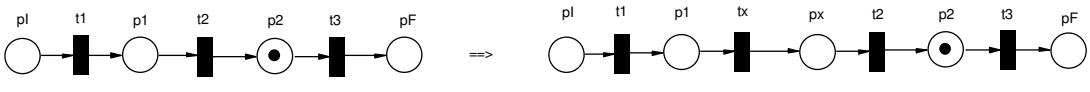
- La secuencia de disparo w que guía a m en S puede ser continuada por w' en S'. El criterio presupone que el marcaje resultante de la migración de la instancia I a S' es conocido.

Cambios en el pasado

Permitido.

Ejemplo 1:

La primera implicación del criterio permite cambios en el pasado tal como insertar una actividad y se muestra en la figura 3(1).

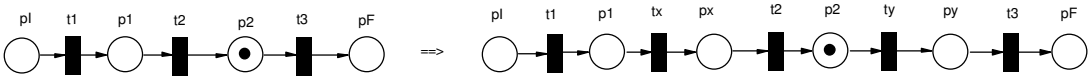


W = <t1, t2> w' = <t3> ∈ FSS(S, m, MpE)

Figura 3(1). Cambio puro en el pasado.

Ejemplo 2:

La segunda implicación del criterio prohíbe cambios en el pasado con compartición de datos, ver figura 3(2). Indica que w' no existe de m a MpE y que no existe w antes del cambio ligado a w' después del cambio: ww'.



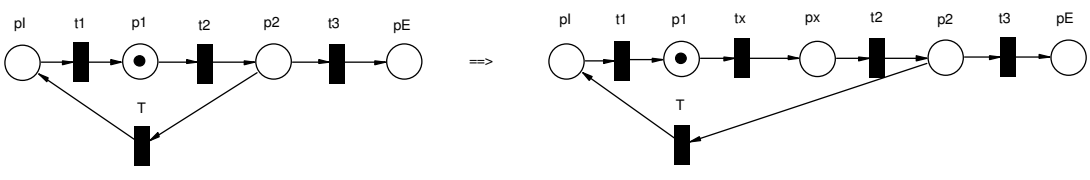
w = <t1, t2> w' = <ty, t3> != FSS(S, m, MpE) ∧
ww' = <t1, t2, ty, t3> != FSS(S', MpI, MpE)

No es permitido

Figura 3(2). Cambio en el pasado con compartición de datos.

Tolerancia a ciclos

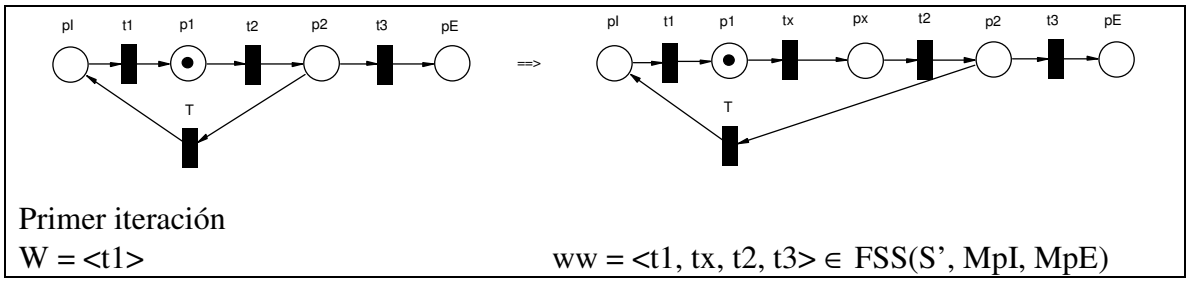
Cambios en los ciclos depende de cómo esté la red antes del cambio. Por ejemplo en una segunda iteración el cambio no cumple con el criterio, ver figura 3(3).



Segunda iteración w' = <tx, t2, t3> != FSS(S, m, MpE)
W = <t1, t2, T, t1> ww' = <t1, t2, T, t1, tx, t2, t3> != FSS(S', MpI, MpE)

Figura 3(3). Tolerancia y cambio en ciclos.

Permite cambios sólo en la primera iteración, ver figura 3(4).



Primer iteración

W = <t1> ww = <t1, tx, t2, t3> ∈ FSS(S', MpI, MpE)

Figura 3(4). Tolerancia y cambio en ciclos en la primera iteración.

Identificación de los estados de las actividades

Problema inherente al modelo, las redes de Petri no diferencian entre actividades activas y en ejecución.

Cambio de orden

Mediante la operación de cambio SCOC, consistente de la región nueva y la anterior, en un escenario de cambio *Upsizing* el cambio de orden es permitido y se mantiene correcto. Mantiene las dos regiones N1 y N2, dónde los casos actuales se ejecutan en la región anterior N1 y los casos nuevos en la nueva región N2 tal como lo muestra la figura 3(5). La identificación de las regiones es manual.

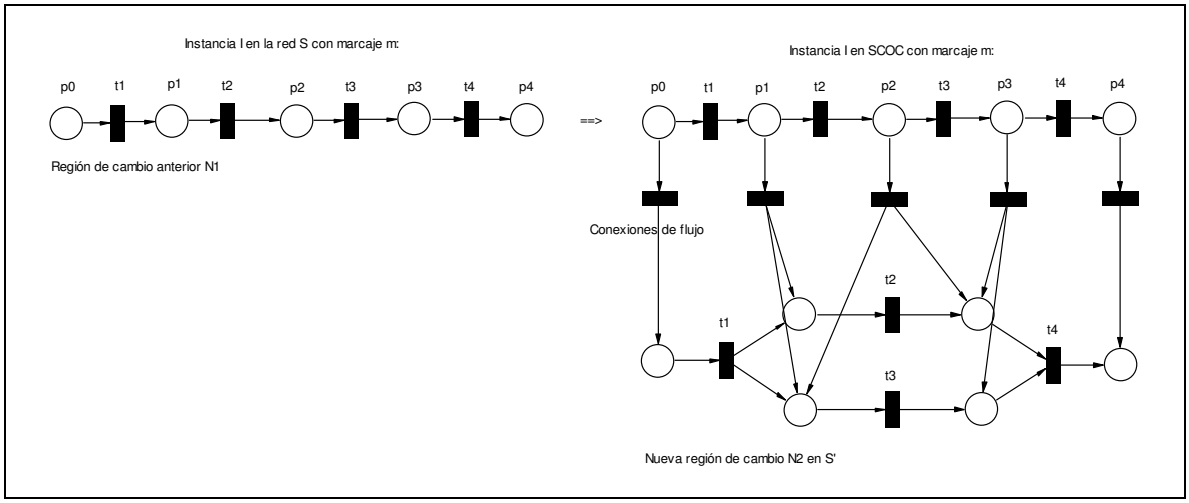


Figura 3(5). Operación de cambio SCOC.

Inserción paralela

Como el criterio presupone que el marcaje debe ser conocido no se puede definir el marcaje resultante en una inserción paralela. La implicación 2 del criterio de correctes lo puede permitir pero no define el marcaje.

IV. Selective shift

(Yamagushi and Mishima 2005)

1. Modelo conceptual basado en redes de Petri de alto nivel.
2. Presenta el concepto y la definición de un tipo de cambio denominado *Selective shift* (SS).
3. No presenta un entorno WF pero la operación de cambio la compara con los enfoques SCOC de *Flow Nets* y *Migrate de Breeze*.
4. SS es una operación de cambio más flexible comparada con SCOC de Flow Net. En SCOC todos los casos en ejecución se quedan en la región anterior y los nuevos se procesan de acuerdo a la nueva región. El mérito de SCOC es la administración en la región anterior los casos que estaban y en la nueva región los casos que llegan. En *Migrate* los casos en ejecución se migran al nuevo esquema. El mérito de *Migrate* es el beneficio del nuevo WF. Los que no se pueden se abortan.
5. En SS algunos casos en ejecución se quedan en la región anterior y otros casos se transfieren a la nueva de acuerdo a una regla de migración. Para la selección de casos de forma separada propone un algoritmo y una regla de migración. No especifica como administra las dos regiones.
6. Sólo se concreta al uso de dos regiones, no presenta explícitamente operaciones de cambio cuando no se requieran las dos regiones, por ejemplo: *Flush*, *Abort*, SCO y *Migrate*. Pero los describe en trabajos anteriores.

Selective Shift tiene la propiedad de

- Comportarse como SCO de *Flow Nets* cuando el conjunto de casos no puede migrarse a la nueva región. $C_{anterior} = C_{total}$ todos los casos en la región anterior.
- Comportarse a *Migrate* de *Breeze* cuando el conjunto de casos totales se pueden ejecutar en la nueva región.

Selective shift utiliza redes WF extendidas y acíclicas. Una plaza puede tener más de un token y cada red tiene una función L(T) de etiquetas para las transiciones, dónde cada transición tiene una etiqueta única diferente a todas las demás.

Criterio de correctes

La regla de migración para las instancias al nuevo esquema es:

- Para cualquier actividad t de T y t' de T', si t y t' tienen el mismo nombre L(t) = L(t') decimos que t y t' tienen el mismo rol.
- Cuando un caso *c* por un conjunto de actividades Tc en N_{old}, decimos que el caso ha sido procesado por un conjunto de actividades T'c en N_{new} si y sólo si $\{L'(t')|t' \in T'c\} \subseteq \{L(t)|t \in Tc\}$.

Cambios en el pasado

No proporciona operaciones de inserción o de borrado. Aunque aparentemente la regla de migración no lo permite, el uso de dos regiones podría soportarlo con restricciones de operaciones de cambio.

Tolerancia a ciclos

Problema evitado, inherente al modelo: acíclico.

Identificación de los estados de las actividades

Problema inherente al modelo, las redes de Petri no diferencian entre actividades activas y en ejecución

Cambio de orden

Administra el cambio de orden mediante dos regiones N1 y N2 repartiendo los casos nuevos y en ejecución entre las dos regiones mediante su regla de migración. La figura 4(1) muestra un ejemplo de cambio de orden. La red de la figura 4(1)a) muestra una red WF completa que requiere un cambio dinámico que consiste en hacer en paralelo las actividades t1 y t2, y también t3 y t4. La figura 4(1)b) muestra la red transicional resultante al aplicar *Selective shift* para realizar el cambio dinámico. Por la regla de migración los casos 1 y 2 permanecen en la región anterior N1 y los casos 3 y 4 se migraron a la nueva región N2 obteniendo sus beneficios. La plaza $P_{recursos}$ representa los recursos compartidos por las actividades, P_{test} y T_{test} se añaden para que los casos que lleguen a p4 sean en orden de primero entrar primero en salir.

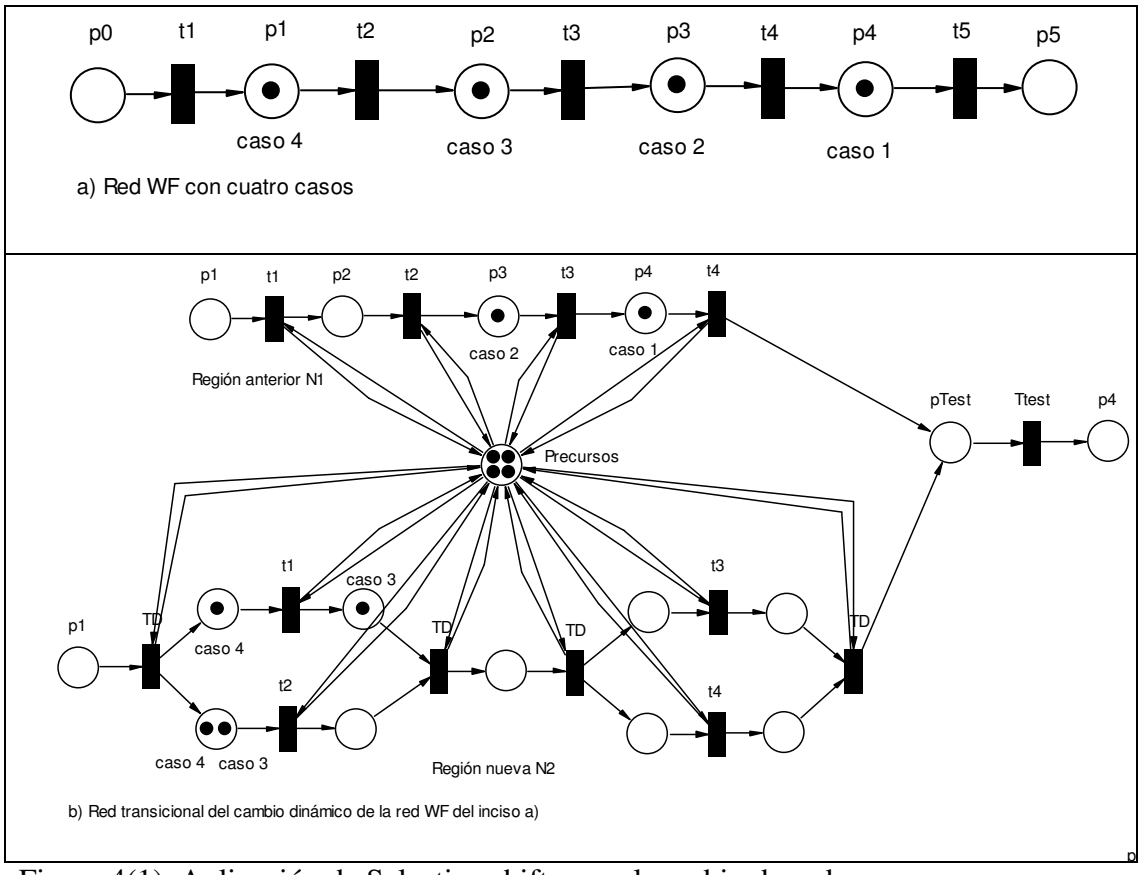


Figura 4(1). Aplicación de Selective shift para el cambio de orden.

Inserción paralela

Aunque no especifica una operación de inserción paralela y la actualización de los casos, puede administrarse mediante las dos regiones de cambio en ciertos casos.

V.- WIDE

(Casati et al. 2000)

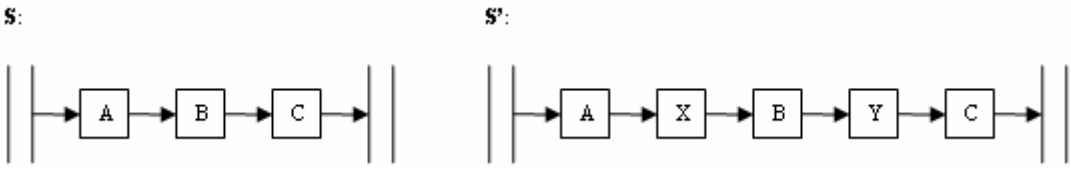
1. Metamodelo conceptual basado diagramas de actividad, tipo diagramas de flujo.
2. Define un conjunto mínimo de primitivas que permiten la modificación correcta de esquemas WF, por ejemplo, primitivas de flujo de control. El cambio en los esquemas se limita a sus primitivas de tal manera que sea correcto.
3. Introduce una taxonomía de políticas para el cambio: *abort*, *flush* y *progressive*.
4. *Progressive* implica el uso de distintas versiones por caso.
5. Permite la modelación de la secuencia, paralelismo, selección y ciclos.
6. Cada esquema S es asociado con un conjunto de variables globales de procesos cuyos valores pueden leerse o modificarse por las instancias en ejecución.
7. Una instancia I es descrita por su esquema S y su historia de ejecución.
8. Los autores introdujeron el criterio de cumplimiento o conformidad (*compliance*).
9. El cambio de un esquema S puede propagarse a la instancia I en S si y sólo si la ejecución de I puede ser simulada en el esquema S'.
10. Como el esquema trabaja con un modelo basado en la historia de ejecución, el cumplimiento esta basado en tratar de reproducir la historia de ejecución de la instancia I en el esquema cambiado S'
11. La actualización del estado es automático al reproducir la historia de ejecución. Puede haber un bajo rendimiento debido al volumen de datos mantenidos en memoria.

Criterio de correctes

Sea S un esquema e I una instancia S con una historia de ejecución E(S, I). Sea Δ una operación de cambio que transforme S en S', entonces I cumple con S' si E(S, I) puede reproducirse también en S' = S + Δ. Por ejemplo todos los eventos almacenados en E(S, I) pueden registrarse por una instancia en S' en el mismo orden.

Cambios en el pasado

Su criterio no permite cambios en el pasado, no puede reproducirse, figura 5(1).

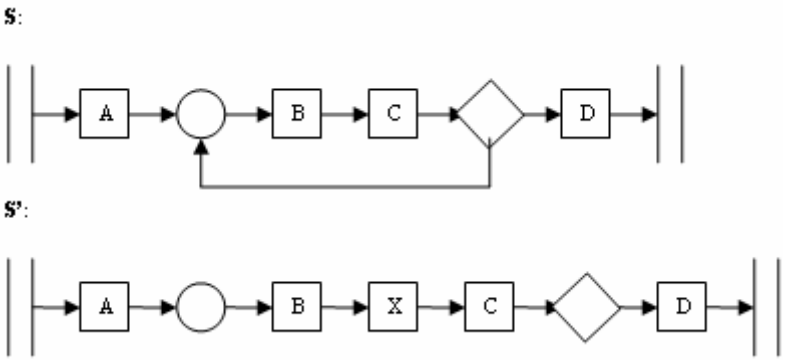


I en S: E(S, I) = ((E_{start}), (E_A), (E_B)) no puede reproducirse en S'.

Figura 5(1). Cambios en el pasado.

Tolerancia a ciclos

Permite ciclos pero no permite modificaciones en los ciclos. La figura 5(2) muestra que no pueden reproducirse.



I en S: E(S, I) = ((E_{start}), (E_A), (E_B), (E_C), (E_B)) no puede reproducirse en S'.

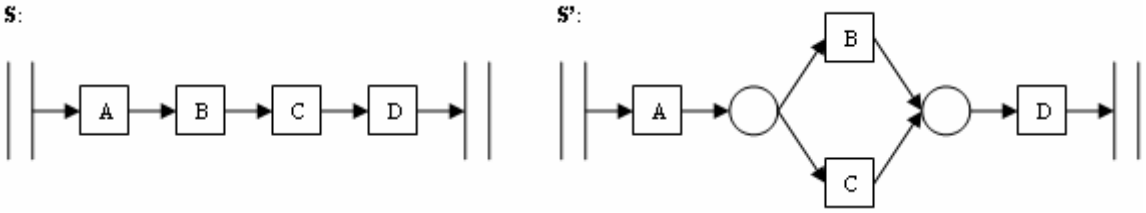
Figura 5(2). Tolerancia y cambio en ciclos.

Identificación de los estados de las actividades

No distingue. El histórico generado por este enfoque contiene sólo actividades terminadas.

Cambio de orden

Permite el cambio de serial a paralelo, ver figura 5(3).

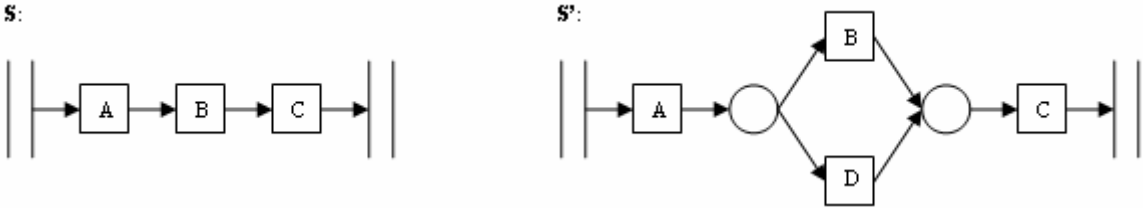


I en S: E(S, I) = ((E_{inicio}), (E_A), E_B, (E_C)) I en S': E(S', I) = (E_{inicio}), (E_A), (E_B), (E_C)

Figura 5(3). Cambio de orden.

Inserción paralela

Permitido: no existe el problema. La figura 5(4) muestra un ejemplo de inserción paralela.



I en S: E = (S, I) = ((E_{inicio}), (E_A), (E_B)) I en S': E(S', I) = ((E_{inicio}), (E_A), (E_B), (E_D))

Figura 5(4). Inserción paralela.

VI.- TRAMS

(Kradolfer and Geppert 1999)

1. Esquema declarativo desarrollado en un prototipo.
2. Administra la evolución de un esquema mediante versiones de tipo WF. La colección de los tipos definidos en cierto punto en el tiempo forman el esquema WF.
3. Definen las condiciones bajo las cuáles la migración de las instancias WF es permitida en nuevos esquemas o continúan en el esquema anterior.
4. En lugar de actualizar los tipos WF ellos son versionados.
5. Las modificaciones a un esquema guían a una nueva versión de tipo y la versión anterior permanece en el esquema. De este modo las instancias se migran a nuevos tipos de un esquema o continúan la ejecución en el anterior.
6. Permite registrar la historia de la evolución de los tipos WF.
7. Un tipo WF puede ser un WF complejo o un tipo actividad. Cada tipo WF tiene un nombre único y un árbol de versiones de tipo. Un tipo puede incluir una o más versiones de tipo. Una versión puede derivarse de otra versión y pueden formar árboles de versiones
8. En contraste a los enfoques basados en actividad, el flujo de control no es realizado por arcos (conectores) de control. En su lugar, es descrito por una forma declarativa usando condiciones de inicio y fin de actividades.
9. El esquema de correctes es preservado por invariantes, es decir, las condiciones relacionadas al esquema que deben reunirse. El flujo de datos es explícitamente especificado conectando parámetros de entrada y salida de actividades subsecuentes. Distingue los estados de las actividades y registra cambios de estado en la historia de ejecución de la instancia respectiva.
10. La propiedad más importante de una instancia es sí es una instancia válida de su versión. Esta propiedad se expresa usando la noción de historia de ejecución.
11. La historia de ejecución registra todos los cambios de estado de ejecución de los WF. Estos se registran como historias de eventos.
12. Una versión tipo WF define un conjunto de eventos históricos válidos y cualquier instancia cuya historia de ejecución es miembro de este conjunto es una instancia válida de la versión.
13. Debido a la definición declarativa del flujo de control, la inserción de actividades es un cambio complejo: primero se inserta el nodo de actividad y entonces se añade el flujo de control estableciendo las condiciones de inicio de la actividad de forma declarativa (arcos de entrada) y los sucesores (arcos de salida) mediante métodos con parámetros.

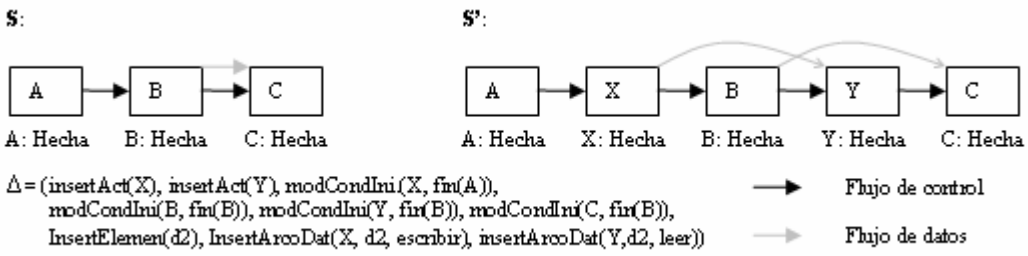
Criterio de correctes

Sea S un esquema e I una instancia S con una historia de ejecución E(S, I). Sea Δ una operación de cambio que transforme S en S', entonces I cumple con S' si E(S, I) puede reproducirse también en S' = S + Δ. Por ejemplo todos los eventos almacenados en E(S, I) pueden registrarse por una instancia en S' en el mismo orden.

- Cada WF w es ejecutado como es preescrito por su versión. En caso de que w sea un subworflow obedece su correspondiente declaración de tipo subworkflow.

Cambios en el pasado

Su criterio no permite cambios en el pasado, no puede reproducirse en el nuevo esquema, figura 6(1).



I en S: E(S, I) = ((S_A), (E_A), (S_B), (E_B))

S_X indica inicio de la actividad X
E_X indica fin de la actividad X

Condiciones de migración: (IniB ∈ E(S, I) && Hecho(X)) = Falso.

Figura 6(1). Cambios en el pasado.

Tolerancia a ciclos

Problema evitado, inherente al modelo: acíclico.

Identificación de los estados de las actividades

Distingue las actividades: inicio, ejecución y terminación.

Cambio de orden

Permite el cambio de serial a paralelo, la instancia en ejecución en la red anterior puede reproducirse en la red nueva, ver figura 6(2).

I en S, E(S, I): (S_A, E_A, S_B, E_B, S_C, E_C) I en S' ∈ E(S, I): (S_A, E_A, S_B, E_B, S_C, E_C)
Las actividades B, C y D requieren que la predecesora se haya ejecutado y es explícito en su estado como se muestra en la figura 7(1).

Figura 6(2). Cambio de orden.

Inserción paralela

Permitido: no existe el problema, ver figura 6(3).

I en S, E(S, I): (S_A, E_A, S_B)

I en S' ∈ E(S, I): (S_A, E_A, S_B, E_B, S_D)

Figura 6(3). Inserción paralela.

VII.- ADEPT

(Reichert and Dadam 1998)

1. Esquema implantado basado en una semántica operacional bien definida, es decir, el estado de un instancia es definida por el estado de sus nodos, arcos, los valores almacenados de sus datos y su historia de ejecución.
2. Soporta tres tipos de ramas: paralelo, de selección y una combinación de entrada paralela con sincronización de selección (Al inicio se ejecutan varias actividades en paralelo y la rama que se termine primero es la única que se selecciona en la salida).
3. No considera cambios a nivel de esquema y la propagación de las instancias, sólo considera cambios a nivel de instancia.
4. Define reglas de ejecución que describen las condiciones bajo las cuáles un nodo puede activarse.
5. El modelo formal define un conjunto de operaciones de cambio para la instancias mientras mantiene la correctes antes y después del cambio.
6. El conjunto de operaciones de cambio permite a usuarios autorizados añadir tareas, eliminar tareas, saltar la ejecución de tareas, salto hacia adelante a regiones que no se han activado, serializar tareas paralelas y ejecutar operaciones para restaurar estados sobre un cambio temporal.
7. Es otro enfoque con marcaje inherente al modelo consistente de redes marcadas bien estructuradas denominadas *WSM-Nets*.
8. Esta basado en un criterio de correctes que también trabaja con cambios en ciclos y en el flujo de datos.
9. Utiliza el criterio de cumplimiento como el enfoque WIDE pero lo modifica para que sea tolerante a ciclos. La solución clave es diferenciar entre iteraciones previas y actuales/futuras, por ejemplo, las consideraciones son restringidas a las partes relevantes de la historia de ejecución.

Definición (Historia de ejecución reducida).

Sea I una instancia con historia de ejecución Π. La historia de ejecución reducida Π_{red} se obtiene:

- En la ausencia de ciclos Π es idéntica a Π_{red}.
- De otro modo, se deriva de Π descartando todas las entradas (registros) históricas relacionadas a iteraciones del último ciclo o de la actual iteración.

Una instancia I en un esquema S cumple con un esquema cambiado S' si y sólo si la historia de ejecución reducida de I puede producirse en S'.

Criterio de correctes

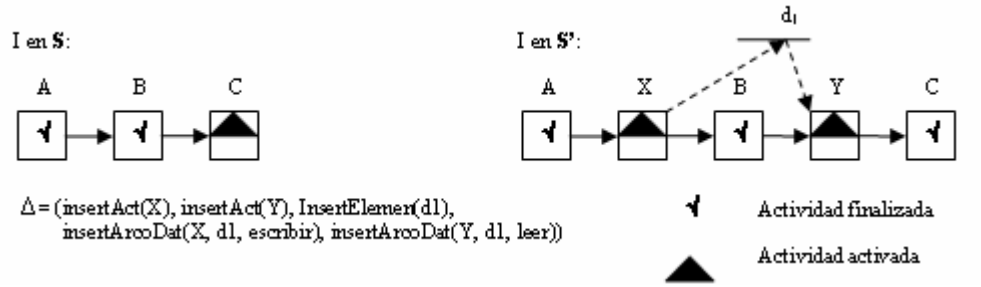
Sea I una instancia en el esquema S con historia de ejecución Π e historia de ejecución Π_{red}. Asumiendo que el cambio Δ transforma S en el esquema correcto S'. Entonces:

- I cumple con S' si y sólo si puede ser reproducido en S'.
- En caso de que se cumpla, el estado resultante de I en S' es correcto.

ADEPT utiliza por cada tipo de cambio condiciones de actualización del estado de las instancias, es decir, usa condiciones para las operaciones de cambio.

Cambios en el pasado

Su criterio no permite cambios puros en el pasado ni con compartición de datos, no pueden reproducirse, ver figura 7(1).

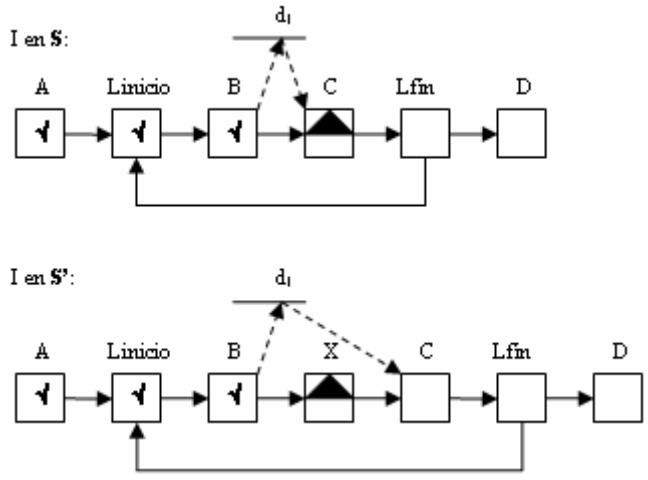


I en S: La historia de ejecución no puede reproducirse en S'.

Figura 7(1). Cambios en el pasado.

Tolerancia a ciclos

Soporta ciclos y cambios en ciclos, ver figura 7(2).



La historia de ejecución reducida en la segunda iteración se obtiene descartando la primera iteración:

$\Pi red = ((S_A), (E_A), (S_{Linicio}, it = 2), (E_{Linicio}), (S_B), (E_B, <d1, 0>))$ ➔ I en S = I en S'

S_X indica inicio de la actividad X
E_X indica fin de la actividad X

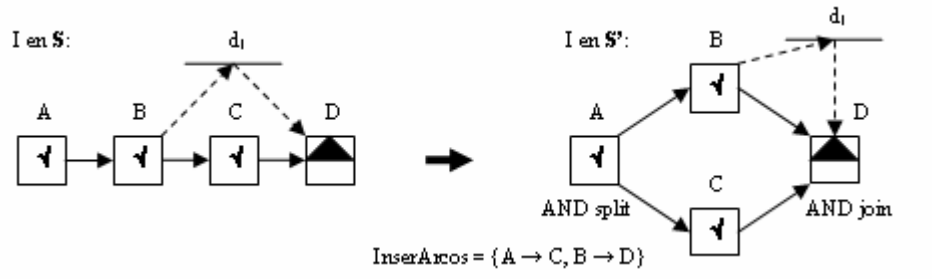
Figura 7(2). Tolerancia y cambio en ciclos.

Identificación de los estados de las actividades

Distingue las actividades: no activas, activas, ejecutándose, terminada, fallada y saltada.

Cambio de orden

El cambio de orden es permitido y es administrado por las operaciones de cambio y las condiciones de migración, ver figura 7(3).

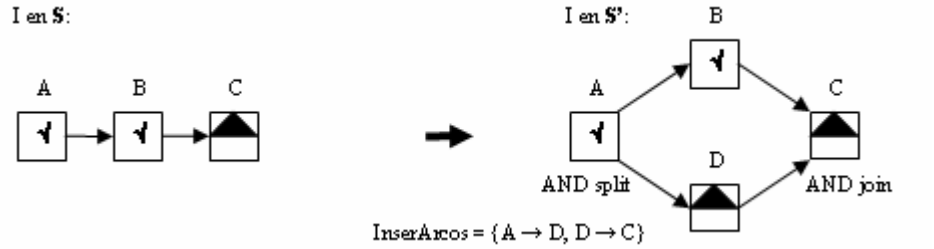


Historia de ejecución reducida:
$\Pi_I = \Pi_{I\,red} = (S_A, E_A, S_B, E_B(d_1), S_C, E_C)$
La historia de ejecución de la instancia I es reproducible en el nuevo esquema S'.

Figura 7(3). Cambio de orden.

Inserción paralela

Permitido: no existe el problema, la figura 7(4) muestra un ejemplo.



Π_I en S = Π_I en S'

Figura 7(4). Inserción paralela.

VIII.- Breeze

(Sadiq 2000)

1. Modelo conceptual basado en redes de actividad.
2. Presenta un análisis y una clasificación de cambios WF en relación al cambio en el proceso de negocio. Asume que los cambios deben ser conocidos y deben verificarse de acuerdo con las propiedades de correctes.
3. Utilizan una herramienta para la verificación de correctes de acuerdo a su enfoque.
4. Sus redes utilizan un marcaje implícito entre nodos.
5. Su esquema es descrito por un grafo acíclico dirigido W(N, F) tal que N es un conjunto finito de nodos y F la relación de flujo $F \subseteq N \times N$.
6. Modela secuencia, selección y paralelismo.
7. Un esquema es correcto si existe sólo un nodo inicial n_i y sólo un nodo final n_f y para todo $n \in N$ existe una trayectoria de n_i a n_f vía n.
8. Los datos los administra como un conjunto de variables globales y pueden leerse y modificarse por las tareas.
9. Define cuatro operaciones de cambio: insertar una nueva tarea, removerla, modificar propiedades de tareas y modificar el orden de ejecución de las tareas. La figura 8(1) muestra un ejemplo de inserción de tareas.

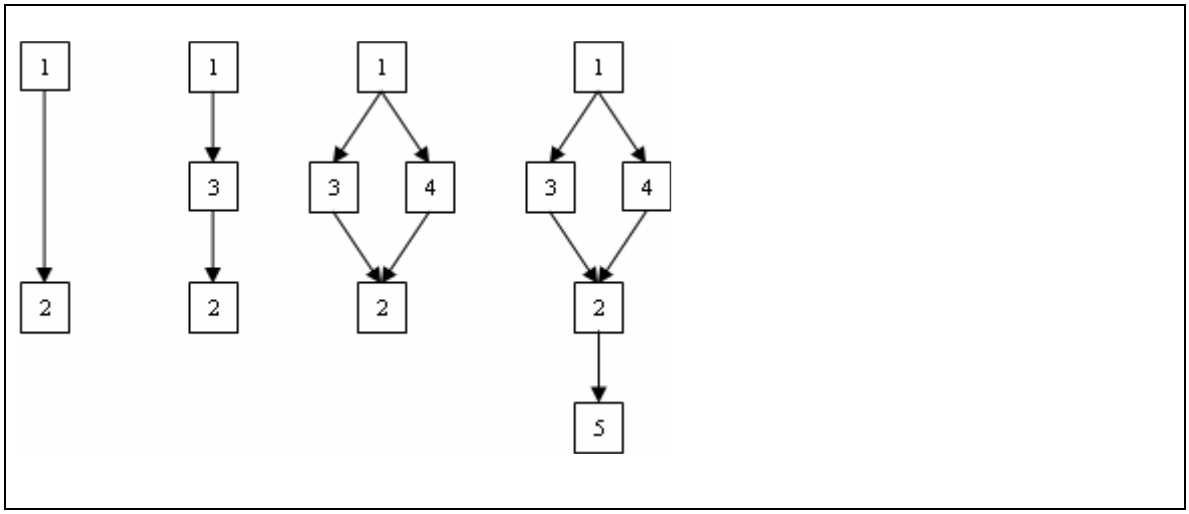


Figura 8(1). Ejemplos de inserción de tareas.

Introduce un esquema de modificación de tres fases para el cambio de un esquema S a S'. La primera consiste en la definición del cambio y consiste de una secuencia de operaciones preescritas que cambian de S a S'. La segunda fase consiste en clasificar las instancias en conformidad con el esquema cambiado de acuerdo al criterio de correctes. Para las instancias que no cumplen con el criterio se construye un grafo de compensación de actividades que sirve para poder migrar dichas instancias al esquema cambiado, si existen instancias que no pueden compensarse, se abortan. El grafo denominado de cumplimiento consiste parcialmente del modelo anterior, el grafo con las actividades de compensación y parcialmente el modelo nuevo. Es como un puente entre el modelo anterior y el nuevo. La tercera fase administra la migración. No presentan

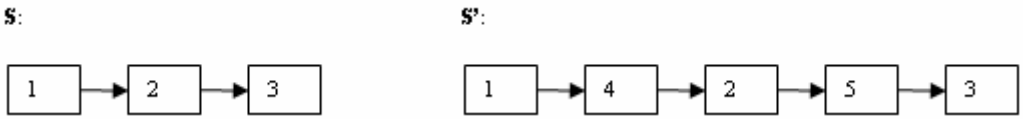
explícitamente como verifican que las instancias cumplan con la nueva red. Las modificaciones se hacen primero sin instancias y después se verifica que las instancias cumplan con la nueva red.

Criterio de correctes

Sea S un esquema e I una instancia S con una historia de ejecución E(S, I). Sea Δ una operación de cambio que transforme S en S', entonces I cumple con S' si E(S, I) puede reproducirse también en S' = S + Δ. Por ejemplo todos los eventos almacenados en E(S, I) pueden registrarse por una instancia en S' en el mismo orden.

Cambios en el pasado

Permite cambios en el pasado en redes sin instancias, ver figura 8(1). El criterio correctes no permite los cambios en instancias en ejecución ni con compartición de datos como muestra la figura 8(2).



I en S: = (S_1, E_1, S_2, E_2) no puede reproducirse en S'

Figura 8(2). Cambios en el pasado.

Tolerancia a ciclos

Problema evitado, inherente al modelo: acíclico.

Identificación de los estados de las actividades

Distingue entre actividades planificadas y en ejecución. La actividad en ejecución es denominada activa.

Cambio de orden

Permite el cambio de serial a paralelo de instancias en ejecución. La reproducción de la bitácora puede realizarse, ver figura 8(3)a) y 8(3)b):

I en S: = (S_1, E_1, S_2, E_2) puede reproducirse en S' figura 8.3b).

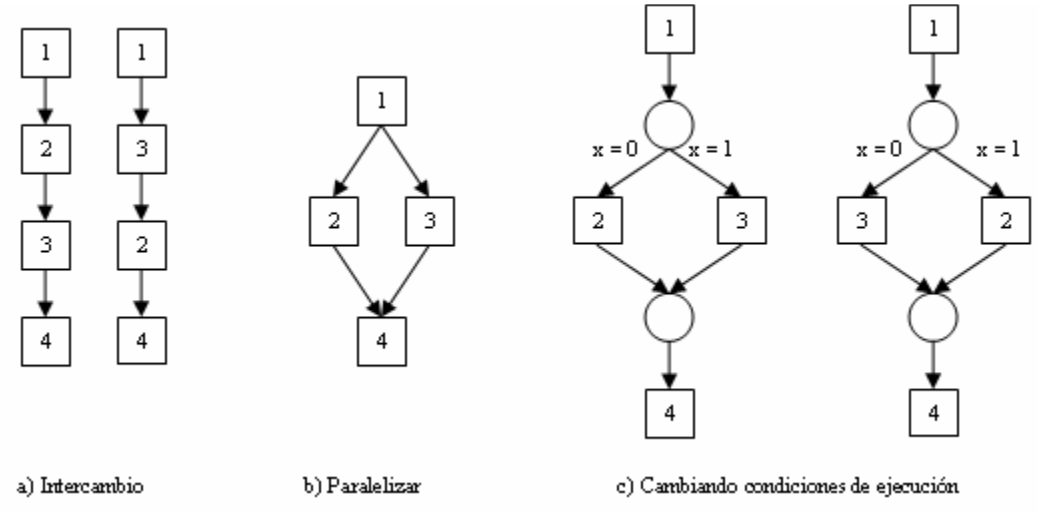


Figura 8(3). Ejemplos de cambios de orden de las tareas.

Inserción paralela

Permitido: no existe el problema, la figura 8(4) muestra la modificación.

I en S = (S_1, E_1, S_2, E_2) I en S' = (S_1, E_1, S_2, E_2, S_4)

Figura 8(4). Inserción paralela.

XIX.- WASA

(Weske 2001)

1. Metamodelo basado en nodos de actividades desarrollado en un prototipo.
2. Utiliza esquemas WF e instancias WF que cumplan con el esquema y son acíclicos por definición.
3. La correctes de las instancias en la presencia de adaptaciones dinámicas esta caracterizadas con respecto a los esquemas.
4. Las instancias se describen por grafos cuyo estado es denotado por un marcaje de actividades inherente al modelo: se representa el estado de las instancias, entradas y salidas de control y de datos.
5. Existe una relación entre instancia y esquema mediante un método denominado *SchemaOf()*.
6. Una instancia WF compleja I es adaptable a un esquema S' si y sólo si hay una continuación de I tal que I cumple con S'. Esto indica que no puede haber cambios en el pasado.
7. Su criterio de correctes se basa en el concepto de mapeo válido y grafos de instancia purgados. Un grafo de instancia purgado es derivado del esquema original eliminando todas las actividades que no han sido iniciadas y eliminando los conectores de control y datos asociados.
8. Un mapeo m: VI a VS' entre la instancia I y el esquema S' asigna a todo nodo n de la instancia un único nodo m(n) del esquema S'. Es decir por cada nodo en I debe haber un nodo en S'

Criterio de correctes

Sea I un grafo de instancia purgado derivado de S. Sea Δ un cambio que transforma correctamente S en S'. Entonces I cumple con S' si y sólo si existe un mapeo válido de I a S'.

- Por cada conector de datos en la instancia I, hay un conector de datos en el esquema S'.
- Por cada conector de control en S' hay un conector en la instancia.

$(\forall i', j' \in V_{S'}$ con $\exists(i', j') \in C_{S'}, \exists i, j \in V_I: i' = m(i), j' = m(j) \wedge (i, j) \in C_I)$ y viceversa $\wedge$

$(\forall k', l' \in V_{S'}$ con $\exists(k', l') \in D_{S'}, \exists k, l \in V_I: k' = m(k), l' = m(l) \wedge (k, l) \in D_I)$

Una instancia puede migrarse a un esquema S' si cada actividad terminada de I esta también contenida en S' y todas las dependencias de control y de datos que existen en I tienen sus contrapartes en S':

$V_I \subseteq V_{S'}, C_I \subseteq C_{S'}, D_I \subseteq D_{S'}$ y $\forall(p, q) \in (C_{S'} - C_I) \cup (D_{S'} - D_I): q \notin V_I$

Cambios en el pasado

No puede haber cambios en el pasado. La figura 9(1) muestra un ejemplo de un cambio dinámico de un esquema S a S', figuras 9(1)a) y 9(1)b). La figura 9(1)c) muestra que la instancia I no puede adaptarse porque se esta ejecutando la actividad cuatro, esto implica que no existe un mapeo válido debido a que no pueden reproducirse los mismos conectores de control y de datos. La figura 9(1)d) muestra una instancia que puede adaptarse ya que existe un mapeo válido.

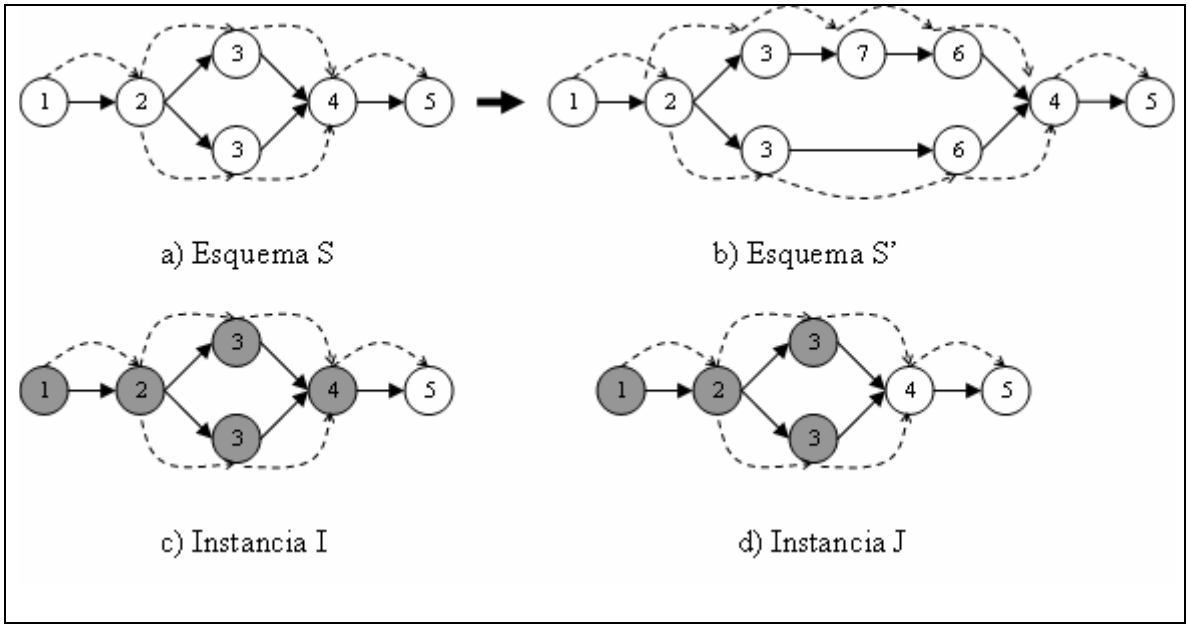


Figura 9(1). Ejemplo del mapeo de dos instancias: I, J.

Tolerancia a ciclos

Problema evitado, inherente al modelo: acíclico.

Identificación de los estados de las actividades

No distingue entre actividades iniciadas y no iniciadas.

Cambio de orden

Es muy restringido a los que cumplen con el mapeo, por ejemplo el cambio de serie a paralelo no es posible, algo que la mayoría cumple, ver figura 9(2).

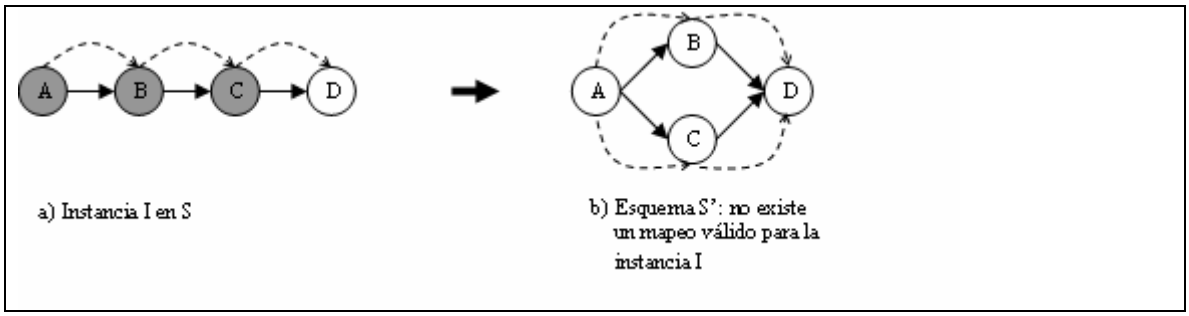


Figura 9(2). Cambio de orden.

Inserción paralela

Permitido: no existe el problema, la figura 9(3) muestra la modificación.

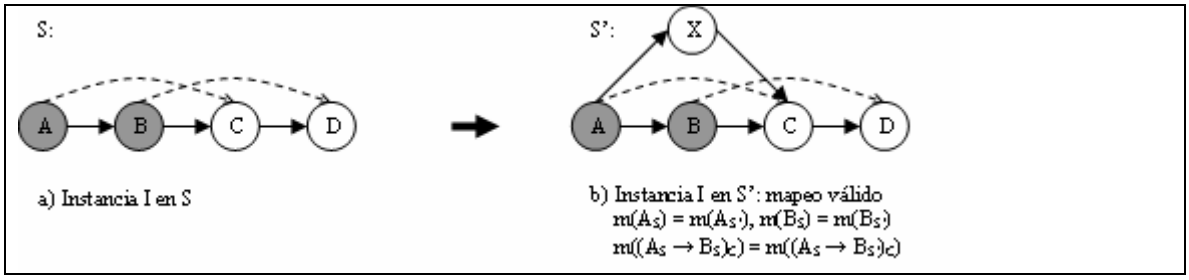


Figura 9(3). Inserción paralela.

Anexo B. Resumen del artículo publicado como resultado de parte de la tesis[2]

Modeling Business Diagnosis with Dynamic Workflow Construction

Espinosa, E. Bueno, A. Molina, M. Munoz, J.
Monterrey Inst. of Technol., Mexico City
This paper appears in: **Service Operations and Logistics, and Informatics, 2006. SOLI '06. IEEE International Conference on**
Publication Date: June 2006
On page(s): 197 - 202
Number of Pages: 197 - 202
Location: Shanghai
INSPEC Accession Number:9165619
Digital Object Identifier: 10.1109/SOLI.2006.329062
Posted online: 2007-03-12 12:44:06.0

Abstract
Mapping workflow (WF) modeling with real-life business process management has become key problem on today's service-based economy. Such real time delivery would give them a competitive lead, and is considered an asset on most scenarios. For the most part, small and medium sized businesses lack the resources to achieve this goal. We propose that a first step to achieving the desired realism is to build global knowledge on businesses. This may be achieved through interactive diagnosing of small and medium sized companies. Such diagnosis must be equivalent to the business it wants to model. For this purpose, we propose a model for dynamic construction of a workflow (DCW), describes the diagnosis process as a workflow that is built dynamically as an entrepreneur answers an online questionnaire. We propose that semantic actions within context-free grammars will enable the dynamic construction process. The model serves as the foundation for true interdisciplinary work

[2] Tal como aparece en el sitio de la IEEE:
http://0-ieeexplore.ieee.org.millenium.itesm.mx/Xplore/dynhome.jsp